МЦК КТИТС
Межрегиональный центр компетенций
Казанский техникум информационных технологий и связи

МЕТОДИЧЕСКОЕ ПОСОБИЕ
ПО ПРОГРАММЕ

«Практика и методика реализации образовательных программ среднего профессионального образования с учетом спецификации стандартов Ворлдскиллс по компетенции «Программные решения для бизнеса»

академия

Оглавление

Движение WorldSkills International и Ворлдскиллс Россия 2

1.1. История и современное состояние движения WSI. История и современное состояние движения Ворлдскиллс Россия («Молодые профессионалы»). Роль движения Ворлдскиллс Россия («Молодые профессионалы») в развитии профессиональных сообществ и систем подготовки кадров 2

1.2. Современные технологии в профессиональной сфере 8

1.3. Стандарты компетенции WSI 09 IT Software Solutions for Business (Программные решения для бизнеса) 17

Культура безопасного труда 31

Особенности обучения в соответствии со стандартами Ворлдскиллс и спецификацией стандартов Ворлдскиллс по компетенции 45

Выбор варианта реализации адаптированной образовательной программы СПО 65

МОДУЛЬ КОМПЕТЕНЦИИ 1: «ПРОЕКТИРОВАНИЕ» 70

Инструментальные средства для анализа и проектирования программных решений. 70

Платформы для проектирования и реализации баз данных 172

МОДУЛЬ КОМПЕТЕНЦИИ 2: «РАЗРАБОТКА ОКОННЫХ ПРИЛОЖЕНИЙ» 194

Платформы для разработки программных решений 194

МОДУЛЬ КОМПЕТЕНЦИИ 3: «ТЕСТИРОВАНИЕ ПРОГРАММНЫХ РЕШЕНИЙ» 253

Тестирование программных решений 253

ОРГАНИЗАЦИЯ И ПРОВЕДЕНИЕ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА ПО СТАНДАРТАМ ВОРЛДСКИЛЛС РОССИЯ. ОЦЕНКА КВАЛИФИКАЦИИ СТУДЕНТА (ВЫПУСКНИКА) В ХОДЕ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА. ЗАСТРОЙКА ПЛОЩАДКИ ПРОВЕДЕНИЯ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА 277

Методика организации и проведения демонстрационного экзамена по компетенции «Программные решения для бизнеса» 325

Движение WorldSkills International и Ворлдскиллс Россия

1.1. История и современное состояние движения WSI. История и современное состояние движения Ворлдскиллс Россия («Молодые профессионалы»). Роль движения Ворлдскиллс Россия («Молодые профессионалы») в развитии профессиональных сообществ и систем подготовки кадров

WorldSkills International (WSI) — международная некоммерческая ассоциация, целью которой является повышение статуса и стандартов профессиональной подготовки и квалификации по всему миру, популяризация рабочих профессий через проведение международных соревнований по всему миру. Основана в 1953 году. На сегодняшний день в деятельности организации принимают участие 77 стран.

Своей миссией WSI называет привлечение внимания к рабочим профессиям и создание условий для развития высоких профессиональных стандартов. Её основная деятельность — организация и проведение профессиональных соревнований различного уровня для молодых людей в возрасте до 22 лет. Раз в два года проходит мировой чемпионат рабочих профессий WorldSkills, который также называют «Олимпиадой для рабочих рук». В настоящее время это крупнейшее соревнование подобного рода.

История

В 1947 году в Испании впервые прошел национальный конкурс по профессионально-технической подготовке. Он был призван поднять популярность рабочих специальностей и способствовать созданию эффективной системы профессионального образования, так как в стране, восстанавливающейся после Гражданской войны, существовала острая нехватка квалифицированных рабочих. Автором данной идеи был генеральный директор Испанской молодёжной организации Хосе Антонио Элола Оласо.

Первой эту инициативу поддержала Португалия. В результате в 1950 году прошли первые международные Пиренейские соревнования, в которых приняли участие 12 представителей обеих стран. Три года спустя к соревнованиям присоединились конкурсанты из Германии, Великобритании, Франции, Марокко и Швейцарии. Таким образом, в 1953 году была сформирована организация по проведению конкурсов профессионального мастерства — International Vocational Training

Organisation (IVTO).

Впервые за пределами Испании соревнования были проведены в 1958 году в рамках Всемирной выставки в Брюсселе, а в 1970 году они первый раз прошли в другой части света — в Токио. В начале 2000-х годов IVTO изменила название и символику, и с тех пор ведет свою деятельность под именем WorldSkills International. Сегодня под эгидой WSI проводится множество мероприятий, включая региональные и национальные соревнования, континентальные первенства и, раз в два года, мировой чемпионат.

Направления, по которым идут соревнования

В структуру чемпионата WorldSkills входят 45 профессиональных компетенций, разделенных на шесть магистральных направлений.

Сервис на воздушном транспорте

Строительные технологии

Изготовление архитектурного камня, Каменщик, Производство корпусной мебели, Плотник, Электрик, Столяр, Ландшафтный дизайн, Маляр, Отделочник штукатур, Сантехника и отопление, Холодильная техника и системы кондиционирования воздуха, Облицовка плиткой

Творчество и дизайн

Дизайн одежды, Флористика, Графический дизайн, Ювелир, Оформитель витрин

Информационные и коммуникационные технологии

Информационные кабельные сети, ИТ Сетевое администрирование, ИТ Решения для бизнеса, Полиграфия, Веб-дизайн

Производственные и инженерные технологии

Фрезеровщик на станках с ЧПУ, Токарь на станках с ЧПУ, Изготовление конструкций из металла, Электроника, Автоматизированные системы контроля и управления в производстве, Производственная сборка изделий, Графический CAD дизайн, Мехатроника, Мобильная робототехника, Изготовление изделий из пластика, Полимеханика/Автоматизация, Создание прототипов, Технология обработки листового металла, Сварка

Специалисты в сфере услуг

Косметология, Кондитер, Повар, Парикмахер, Социальный работник, Официант

Спасательные работы

Обслуживание гражданского транспорта

Обслуживание авиационной техники, Кузовной ремонт, Автомеханик, Автопокраска

Соревнования в области робототехники проводятся с помощью

специальной платформы Robotino

На сегодняшний день на соревнованиях WorldSkills International тысячи молодых профессионалов демонстрируют свои знания и навыки, представляя более чем 70 стран.

За 70 лет масштабы чемпионатов профессионального мастерства выросли: в 1950 г. было всего 12 конкурсантов, в 2017 г. на 44-м чемпионате WorldSkills в Абу-Даби - уже 1300. На международном первенстве в Казани в 2019 г. ожидается уже более 1500 участников.

Роль движения Ворлдскиллс Россия («Молодые профессионалы») в развитии профессиональных сообществ и систем подготовки кадров

Хотя основная миссия движения осталась прежней, изменения на рынке труда и развитие технологий сформировали новые вызовы. Автоматизация и переход к цифровой экономике создали потребность в рабочих кадрах нового типа, способности которых измеряются компетенциями, а не дипломами и грамотами.

Теперь WorldSkills необходимо готовить не просто молодого конкурентоспособного профессионала, а адаптированного к современным реалиям специалиста, готового работать бок о бок с умными аппаратами и робототехникой, постоянно расширять свои знания.

Россия присоединилась к движению WorldSkills в 2012 году. В тот период техникумы и колледжи, несмотря на попытки реформ, по-прежнему казались многим низшей образовательной ступенью, которую проходят только те, кому не удалось поступить в вуз и кто не рискнул сдавать ЕГЭ.

Реформирование системы среднего профессионального образования стало первой задачей, которая встала перед Союзом «Молодые профессионалы (Ворлдскиллс Россия)». В этом заключается важное отличие российской модели движения от аналогов в других странах.

Если условные SwissSkills («Ворлдскиллс Швейцария») или WorldSkills France выступают в первую очередь центром привлечения молодых профессионалов и их подготовки к чемпионатам, то российское подразделение WorldSkills стремится реформировать всю систему образования.

Учебные заведения в стране пока еще с трудом адаптируются к реалиям современного рынка труда. Устаревшие учебные программы, годами не менявшиеся принципы подготовки преподавателей,

противоречивые стандарты и вышедшее из употребления оборудование привело к закономерному дефициту кадров.

В 1990-е многие образовательные учреждения утратили связь с предприятиями. Выпускники приобретали абстрактные навыки для абстрактного будущего, часто - из давно ушедшего прошлого. В результате, поступив на работу, вынуждены были переучиваться.

За 7 лет существования WorldSkills в России движение поддержали десятки партнёров. Среди них есть крупные государственные корпорации, такие как Ростех, Роскосмос и Росатом. Они не скрывают, что стремятся выйти на международный рынок и остро нуждаются в кадрах, которые отвечали бы мировым стандартам.

Таких специалистов удаётся найти среди участников региональных, национальных и международных чемпионатов WorldSkills. Также госкорпорации готовят специалистов изнутри, полагаясь на мировые стандарты профподготовки.

Другая часть партнеров – это российские представительства зарубежных компаний, например, производители робототехники Kuka и Festo. Они не только нуждаются в кадрах, но и пытаются популяризовать относительно новые отрасли среди школьников и студентов.

Для этого совместно с Ворлдскиллс Россия компании открывают робототехнические лаборатории в регионах. Четыре года назад российское представительство Kuka первым ввело компетенцию «Промышленная робототехника» в линейку WorldSkills.

Союз «Молодые профессионалы (Ворлдскиллс Россия)» пытается наладить двусторонний контакт между учебными заведениями и компаниями.

Подготовка молодых профессионалов к реалиям цифровой экономики – ещё одна задача, которая пока не решена. Совместно с консалтинговой компанией Boston Consulting Group в WorldSkills Russia изучили ситуацию на рынке труда в современной России. Как показал опрос, не менее 66% предприятий опасаются, что не смогут развиваться из-за нехватки квалифицированных специалистов.

По оценкам аналитиков, уже к 2025 году Россия столкнётся с дефицитом кадров в 10 миллионов человек.

Тенденция в принципе характерна для большинства учебных заведений среднего профессионального образования в мире. Исследование McKinsey выявило парадоксальную ситуацию: 50% молодых людей считают, что следующая после окончания школы ступень образования повышает их шансы на трудоустройство.

Более того, большинство образовательных учреждений (72%)

считают, что их выпускники готовы к реальной работе. Однако только 43% работодателей находят специалистов с нужным уровнем квалификации.

Демонстрационный экзамен

В 2014 г. требование привести российские колледжи к мировым стандартам подготовки специалистов оформилось уже на правительственном уровне.

А спустя два года в России решили испытать формат демонстрационного экзамена по методике WorldSkills - новой системы государственной итоговой аттестации, которая позволяет проверить навыки выпускников в реальных производственных условиях.

Министерство образования и науки уже намекает, что скоро методику могут официально утвердить в качестве основной формы ГИА. Демонстрационный экзамен учитывает реальные требования рынка труда, а не условные стандарты.

Задания для демонстрационного экзамена действительно трудно сравнить с ответами на билеты. В течение нескольких дней студенты выполняют такие же задачи, что и участники международных чемпионатов WorldSkills. Теория сведена к минимуму - она лишь подкрепляет действия, которые нужно выполнять в реальных условиях на реальном оборудовании.

В настоящий момент главная задача учебных заведений – формирование новых образовательных условий для повышения качества образования. И демонстарционный экзамен в данном случае – инструмент (линейка), позволяющий оценить это качество подготовки специалистов, востребованных на рынке труда.

В этих условиях выигрывают все. Учебные заведения совершенствуют систему подготовки, студенты отрабатывают навыки для реального, а не абстрактного сектора.

Причём компетенции подтверждаются не дипломом с оценками по предметам, а Skills-паспортом с перечислением конкретных навыков - ещё одним проектом Ворлдскиллс Россия. В нём обозначены модули, которые выполнил студент. Результаты участника отражаются в графике, в котором также указаны минимальные и максимальные допустимые результаты по стандарту.

Так, на примере компетенции «Программные решения для бизнеса», четко прослеживаются измения за последние 3 года: большинство колледжей России прошли лицензирование по образовательным программам ТОП-50, выстраивается новое содержание образования – колледжи уходят от устаревших языков

программирования, работают с современными программными продуктами, все больше колледжей внедряют демонстрационный экзамен не только в рамках ГИА, но и в рамках промежуточной аттестации, что все больше подтверждает действенность этого инструмента.

Около 40 компаний, в том числе Росатом, «Р-Фарм» и IEK признали Skills-паспорта инструментом независимой оценки подготовки студентов.

Классификация компетенций – ещё одна проблема российской системы профподготовки. Представители многих техникумов и вузов признают, что сегодня специальности уже не имеют таких чётких границ, как раньше. Условный оператор ЧПУ-станка может сегодня управлять сразу несколькими станками на заводе и фактически руководить производственным процессом. Новый технологический уклад меняет сам формат профессий, причём большинство из них получают цифровое измерение.

Параллельно с направлением Hi-Tech Ворлдскиллс Россия создаёт и другие кластеры компетенций.

Сверхзадача - избежать появление класса лишних людей, чьи способности окажутся невостребованными в условиях цифровой экономики.

Для этого WSR запустил проект FutureSkills, который пытается установить стандарты оценки специалистов по геномной инженерии, эксплуатации беспилотных авиационных систем, лазерным технологиям, промышленной робототехнике и другим компетенциям будущего.

Экспертов для обсуждения будущего рынка труда отбирают по обширной сетке контактов, в том числе среди потенциальных работодателей, ИТ-компаний и поставщиков технологического оборудования.

Один из главных трендов - это увеличение скорости изменений, которая прослеживается во всех отраслях. Именно поэтому странам необходимо готовить специалистов с опережением. Об этом говорит и Роберт Уразов, который признаёт, что России нужно перестать быть догоняющей страной.

В среднем стандарты WorldSkills меняются на 30% как минимум каждые два года.

На мировом чемпионате WorldSkills 2019 в Казани компетенциям будущего будет отведена отдельная зона, площадью 10 тысяч кв. м.

Всё больше профессиональных сфер нуждаются в «движке»,

который помог бы перестроиться и адаптироваться к условиям новой экономики.

И роль такого «движка» выполняет движение Ворлдскиллс Россия.

1.2. Современные технологии в профессиональной сфере

Каждый человек хоть раз написавший какую-либо программу, достаточно хорошо может представить себе, как разработать «небольшую» программу, решающую обычно одну конкретную несложную задачу и предназначенную, чаще всего, для использования одним человеком или узкой группой людей.

Сложные или «большие» программы, называемые также программными системами, программными комплексами, программными продуктами, отличаются от «небольших» не столько по размерам (хотя обычно они значительно больше), сколько по наличию дополнительных факторов, связанных с их востребованностью и готовностью пользователей платить деньги как за приобретение самой программы, так и за ее сопровождение и даже за специальное обучение работе с ней. Обычно сложная программа обладает следующими свойствами.

• Она решает одну или несколько связанных задач, зачастую сначала не имеющих четкой постановки, настолько важных для каких-либо лиц или организаций, что те приобретают значимые выгоды от ее использования.

• Существенно, чтобы она была удобной в использовании. В частности, она должна включать достаточно полную и понятную пользователям документацию, возможно, также 12 специальную документацию для администраторов, а также набор документов для обучения работе с программой.

• Ее низкая производительность на реальных данных приводит к значимым потерям для пользователей.

• Ее неправильная работа наносит ощутимый ущерб пользователям и другим организациям и лицам, даже если сбои происходят не слишком часто.

• Для выполнения своих задач она должна взаимодействовать с другими программами и программно-аппаратными системами, работать на разных платформах.

• Пользователи, работающие с ней, приобретают дополнительные выгоды от того, что программа развивается, в нее вносятся новые

функции и устраняются ошибки. Необходимо наличие проектной документации, позволяющей развивать ее, возможно, вовсе не тем разработчикам, которые ее создавали, без больших затрат на обратную разработку (реинжиниринг).

• В ее разработку вовлечено значительное количество людей (более 5-ти человек). «Большую» программу практически невозможно написать с первой попытки, с небольшими усилиями и в одиночку.

• Намного больше количество ее возможных пользователей, и еще больше тех лиц, деятельность которых будет так или иначе затронута ее работой и результатами.

Строго говоря, ни одно из указанных свойств не является обязательным для того, чтобы программу можно было считать «большой», но при наличии двух-трех из них достаточно уверенно можно утверждать, что она «большая». На основании некоторых из перечисленных свойств можно сделать вывод, что «большая» программа или программная система чаще всего представляет собой не просто код или исполняемый файл, а включает еще и набор проектной и пользовательской документации.

Изучением организационных, инженерных и технических аспектов создания ПО, включая методы разработки, занимается дисциплина, называемая программной инженерией.

Большая часть трудностей при разработке программных систем связана с организацией экономически эффективной совместной работы

многих людей, приводящей к практически полезному результату. Это требует рассмотрения следующих аспектов.

• Над программой обычно работает много людей, иногда географически удаленных друг от друга и из различных организаций. Их работа должна быть организована так, чтобы затраты на разработку были бы покрыты доходами от продаж и предоставления услуг, связанных с полученной программой. В затраты входят зарплаты разработчиков, затраты на закупленное оборудование и программные инструменты разработки, на приобретение лицензий и патентование собственных решений, часто еще и затраты на исследование потребностей клиентов, проведение рекламы и другой маркетинговой деятельности.

• Значимые доходы могут быть получены, только если программа будет предоставлять пользователям в реальных условиях их работы такие возможности, что они готовы будут заплатить за это деньги (которым, заметим, без труда можно найти другие полезные применения).

Для этого нужно учесть множество аспектов. Доходы от продаж значительно снизятся, если многие из пользователей не смогут воспользоваться программой только потому, что в их компьютерах процессоры слишком медленные или мало оперативной памяти, или потому что данные к системе часто поступают в искаженном виде и она не может их обработать, или потому что они привыкли работать с графическим интерфейсом, а система требует ввода из командной строки, и т.п. Важно отметить, что практически полезная сложная программная система не обязательно является «правильной».

Большинство опытных разработчиков и исследователей считают, что практически значимые программные системы всегда содержат ошибки. При переходе от «небольших» программ к «большим» понятие «правильной» программы становится практически бессмысленным. Про программную систему, в отличие от приведенной выше программы вычисления числа π, нельзя утверждать, что она «правильная», т.е. всегда правильно решает все поставленные перед ней задачи. Этот факт связан как с практической невозможностью полного доказательства или проверки этого, так и с тем, что смысл существования программной системы — удовлетворение потребностей и запросов большого количества различных заинтересованных лиц. А эти потребности не только нечетко определены, различны для разных групп пользователей и иногда противоречивы, но и значительно изменяются с течением времени.

Стоимость исправления ошибки в ходе ЖЦ ПС

В связи с этим, вместо рассмотрения «правильных» и «неправильных» программных систем, в силу практического отсутствия первых, рассматривают «достаточно качественные» и «недостаточно качественные». Поэтому и основные проблемы разработки сложных программных систем связаны с нахождением разумного компромисса между затратами на разработку и качеством ее результата. В затраты входят все виды используемых ресурсов, из которых наиболее важны затрачиваемое время, бюджет проекта и используемый персонал. Удовлетворение пользователей от работы с программой (а, следовательно, доходы от ее продаж и предоставления дополнительных

услуг) и удовлетворение разработчиков от ее создания определяются качеством программы, которое включает в себя такие аспекты, как набор предоставляемых возможностей, надежность, удобство использования, гибкость, удобство внесения изменений и исправления ошибок.

Принципы работы со сложными системами

Помимо методических рекомендаций, при конструировании больших систем часто используются прагматические принципы работы со сложными системами вообще. Они играют значительную роль в выработке качественных технических решений в достаточно широком контексте. Эти принципы позволяют распределять работы между участвующими в проектах людьми с меньшими затратами на обеспечение их взаимодействия и акцентировать внимание каждого из участников на наиболее существенных для его части работы характеристиках системы. К таким принципам относятся использование абстракции и уточнения, модульная разработка и переиспользование.

Абстракция (abstraction) и уточнение (refinement). Абстракция является универсальным подходом к рассмотрению сложных вещей. Интеллект одного человека достаточно ограничен и просто не в силах иметь дело сразу со всеми элементами и свойствами систем большой сложности. Известно, что человеку крайне тяжело держать в голове одновременно десяток-полтора различных мыслей, а в современных системах число различных существенных аспектов доходит до сотен. Для того чтобы как-то все-таки работать с такими системами, мы пользуемся своей возможностью абстрагироваться, т.е. отвлекаться от всего, что несущественно для достижения поставленной в данной момент частной цели и не влияет на те аспекты рассматриваемого предмета, которые для этой цели важны. Чтобы перейти от абстрактного представления к более конкретному, используется обратный процесс последовательного уточнения. Рассмотрев систему в каждом аспекте в отдельности, мы пытаемся объединить результаты анализа, добавляя аспекты по одному и обращая при этом внимание на возможные взаимные влияния и возникающие связи между элементами, выявленными при анализе отдельных аспектов. Абстракция и уточнение используются, прежде всего, для получения работоспособных решений, гарантирующих нужные свойства результирующей системы.

Пример абстракции и уточнения. Систему хранения идентификаторов пользователей Интернет-магазина можно представить как множество целых чисел, забыв о том, что эти числа — идентификаторы пользователей, и о том, что все это как-то связано с Интернет-магазином. Затем описанную модель системы хранения

идентификаторов пользователей Интернетмагазина можно уточнить, определив конкретную реализацию множества чисел, например, на основе сбалансированных красно-черных деревьев. Другой пример. Рассматривая задачу передачи данных по сети, можно временно абстрагироваться от большинства проблем организации связи и заниматься только одним аспектом — организацией надежной передачи данных в нужной последовательности.

При этом можно предполагать, что мы как-то умеем передавать данные между двумя компьютерами в сети, хотя, быть может, и с потерями и с нарушением порядка их прибытия по сравнению с порядком отправки. Установленные ограничения выделяют достаточно узкий набор задач. Любое их решение представляет собой некоторый протокол передачи данных транспортного уровня, т.е. нацеленный именно на надежную упорядоченную доставку данных. Выбирая такой протокол из уже существующих, например, TCP, или разрабатывая новый, мы производим уточнение исходной общей задачи передачи данных. Другой способ уточнения — перейти к рассмотрению протоколов, обеспечивающих исходные условия для нашей первой абстракции, т.е. возможность вообще что-то передавать по сети.

При этом возникают протоколы нижележащих уровней — сетевого (отвечают за организацию связи между не соединенными непосредственно компьютерами при наличии между ними цепи машин, соединенных напрямую), канального (такие протоколы отвечают за определение формата передаваемых данных и надежность передачи отдельных элементов информации между двумя физически соединенными компьютерами) и физического (отвечают за определение физического носителя передаваемого сигнала и правильность интерпретации таких сигналов обеими машинами, в частности, за конкретный способ передачи битов с помощью электрических сигналов или радиоволн).

Модульность (modularity). Модульность — принцип организации больших систем в виде наборов подсистем, модулей или компонентов. Этот принцип предписывает организовывать сложную систему в виде набора более простых систем — модулей, взаимодействующих друг с другом через четко определенные интерфейсы. При этом каждая задача, решаемая всей системой, разбивается на более простые, решаемые отдельными модулями подзадачи, решение которых, будучи скомбинировано определенным образом, дает в итоге решение исходной задачи. После этого можно отдельно рассматривать каждую подзадачу и модуль, ответственный за ее решение, и отдельно — вопросы интеграции

полученного набора модулей в целостную систему, способную решать исходные задачи. Выделение четких интерфейсов для взаимодействия упрощает интеграцию, позволяя проводить ее на основе явно очерченных возможностей этих интерфейсов, без обращения к многочисленным внутренним элементам модулей, что привело бы к росту сложности.

Пример. Примером разбиения на модули может служить структура пакетов и классов библиотеки JDK. Классы, связанные с основными сущностями языка Java и виртуальной машины, собраны в пакете java.lang.

Интерфейсом класса служат его общедоступные методы, а интерфейсом пакета — его общедоступные классы.

Другой пример модульности — принятый способ организации протоколов передачи данных. Мы уже видели, что удобно выделять несколько уровней протоколов, чтобы на каждом решать свои задачи. При этом надо определить, как информация передается от машины к машине при помощи всего этого многоуровневого механизма. Обычное решение таково: для каждого уровня определяется способ передачи информации с или на верхний уровень — предоставляемые данным уровнем службы. Точно так же определяется, в каких службах нижнего уровня нуждается верхний, т.е. как передать данные на нижний уровень и получить их оттуда. После этого каждый протокол на данном уровне может быть сформулирован в терминах обращений к нижнему уровню и должен реализовать операции- службы, необходимые верхнему. Это позволяет заменять протокол-модуль на одном уровне без внесения изменений в другие. Хорошее разбиение системы на модули — непростая задача. При ее выполнении привлекаются следующие дополнительные принципы. о Выделение интерфейсов и сокрытие информации. Модули должны взаимодействовать друг с другом через четко определенные интерфейсы и скрывать друг от друга внутреннюю информацию — внутренние данные, детали реализации интерфейсных операций. При этом интерфейс модуля обычно значительно меньше, чем набор всех операций и данных в нем.

Адекватность, полнота, минимальность и простота интерфейсов. Этот принцип объединяет ряд свойств, которыми должны обладать хорошо спроектированные интерфейсы.

Адекватность интерфейса означает, что интерфейс модуля дает возможность решать именно те задачи, которые нужны пользователям этого модуля. Например, добавление в интерфейс очереди метода, позволяющего получить любой ее элемент по его номеру в очереди,

сделало бы этот интерфейс не вполне адекватным — он превратился бы почти в интерфейс списка, который используется для решения других задач. Очереди же используются там, где полная функциональность списка не нужна, а реализация очереди может быть сделана более эффективной.

Полнота интерфейса означает, что интерфейс позволяет решать все значимые задачи в рамках функциональности модуля.

Минимальность интерфейса означает, что предоставляемые интерфейсом операции решают различные по смыслу задачи и ни одну из них нельзя реализовать с помощью всех остальных (или же такая реализация довольно сложна и неэффективна). Большое значение минимальности интерфейса уделяют, если размер модулей оказывает сильное влияние на производительность программы. Например, при проектировании модулей операционной системы — чем меньше она занимает места в памяти, тем больше его останется для приложений, непосредственно необходимых пользователям. При проектировании библиотек более высокого уровня имеет смысл не делать интерфейс минимальным, давая пользователям этих библиотек возможности для повышения производительности и понятности их программ. Например, часто бывает полезно реализовать действия «проверить, что элемент не принадлежит множеству, и, если нет, добавить его» в одном методе, не заставляя пользователей каждый раз сначала проверять принадлежность элемента множеству, а затем уже добавлять его.

Простота интерфейса означает, что интерфейсные операции достаточно элементарны и не представимы в виде композиций некоторых более простых операций на том же уровне абстракции, при том же понимании функциональности модуля.

Разделение ответственности. Основной принцип выделения модулей — создание отдельных модулей под каждую задачу, решаемую системой или необходимую в качестве составляющей для решения ее основных задач.

Принцип разделения ответственности имеет несколько важных частных случаев.

Разделение политик и алгоритмов. Этот принцип используется для отделения постоянных, неизменяемых алгоритмов обработки данных от изменяющихся их частей и для выделения этих частей, называемых политиками, в параметры общего алгоритма.

Стоимость товара для клиента может зависеть от привилегированности клиента, размера партии, которую он покупает и сезонных скидок. Все перечисленные элементы можно выделить в виде

политик, являющихся, вместе с базовой ценой товара, входными данными для алгоритма вычисления итоговой стоимости.

Разделение интерфейса и реализации. Этот принцип используется при отделении внешне видимой структуры модуля, описания задач, которые он решает, от способов решения этих задач.

Слабая связность (coupling) модулей и сильное сродство (cohesion) функций в одном модуле. Оба эти принципа используются для выделения модулей в большой системе и тесно связаны с разделением ответственности между модулями. Первый требует, чтобы зависимостей между модулями было как можно меньше. Модуль, зависящий от большинства остальных модулей в системе, скорее всего, надо перепроектировать — это означает, что он решает слишком много задач. И наоборот, «сродство» функций, выполняемых одним модулем, должно быть как можно выше. Хотя на уровне кода причины этого «сродства» могут быть разными — работа с одними и теми же данными, зависимость от работы друг друга, необходимость синхронизации при параллельном выполнении и пр. — цена их разделения должна быть достаточно высокой. Наиболее существенно то, что эти функции решают тесно связанные друг с другом задачи.

Переиспользование. Этот принцип требует избегать повторений описаний одних и тех же знаний — в виде структур данных, действий, алгоритмов, одного и того же кода — в разных частях системы. Вместо этого в хорошо спроектированной системе выделяется один источник, одно место фиксации для каждого элемента знаний и организуется его переиспользование во всех местах, где нужно использовать этот элемент знаний. Такая организация позволяет при необходимости (например, при исправлении ошибки или расширении имеющихся возможностей) удобным образом модифицировать код и документы системы в соответствии с новым содержанием элементов знаний, поскольку каждый из них зафиксирован ровно в одном месте.

ЗАДАНИЕ: Разработайте схему основных принципов работы со сложными системами на основе предосталенного материала.

1.3. Стандарты компетенции WSI 09 IT Software Solutions for Business (Программные решения для бизнеса)

Стремительные темпы глобализации за последние десять лет были в основном вызваны разработками в области информационных и коммуникационных технологий (ИКТ). Спрос на ИТ-специалистов растет в целом ряде отраслей, одной из которых является предоставление программных решений для бизнеса.

Разработка программных решений для повышения производительности бизнеса охватывает многочисленные компетенции и дисциплины. Ключом к ним является осознание динамичной природы отрасли и способность идти в ногу с постоянными переменами.

Профессионалы в области программных решений всегда тесно сотрудничают с клиентами для модификации существующих или создания новых систем. Они могут модифицировать готовое программное обеспечение и интегрировать его в существующие системы. Они часто работают в составе команды профессиональных студент или специалистов, отвечающих за спецификацию требований, системный анализ и проектирование, построение, испытание, обучение и развертывание, а также техническое обслуживание коммерческих программных систем.

Задачи, выполняемые профессионалами в области программных решений, в числе прочего, включают следующее:

- анализ существующей системы и представление идей по усовершенствованию, включая анализ затрат-выгод;
- анализ и уточнение требований пользователя;
- составление детальных спецификаций для новых систем или для модификаций существующих систем;
- разработку систем программного обеспечения и тщательное тестирование программных решений;
- подготовку обучающих материалов для пользователей, обучение пользователей и представление программного решения пользователям;
- установку, развертывание и обслуживание программной системы.

Профессионалы в области программных решений могут быть приняты на работу в крупные, средние и малые предприятия в качестве

разработчиков ПО, в консультационные фирмы в качестве консультантов и в компании, выпускающие ПО, в качестве подрядчиков.

Они могут выполнять самые разнообразные роли: роль разработчика для индивидуальной разработки или персонализации программных решений, вспомогательную роль для управления системами, роль бизнес-аналитика для предоставления решений, упрощающих и автоматизирующих рутинные офисные и бизнес-процессы, а также обучающую роль для обучения пользователей применению прикладных программ.

WSSS определяет знание, понимание и конкретные компетенции, которые лежат в основе лучших международных практик технического и профессионального уровня выполнения работы. Она должна отражать коллективное общее понимание того, что соответствующая рабочая специальность или профессия представляет для промышленности и бизнеса.

Целью соревнования по компетенции является демонстрация лучших международных практик, как описано в WSSS и в той степени, в которой они могут быть реализованы. Таким образом, WSSS является руководством по необходимому обучению и подготовке для соревнований по компетенции.

В соревнованиях по компетенции проверка знаний и понимания осуществляется посредством оценки выполнения практической работы. Отдельных теоретических тестов на знание и понимание не предусмотрено.

WSSS разделена на четкие разделы с номерами и заголовками.

Каждому разделу назначен процент относительной важности в рамках WSSS. Сумма всех процентов относительной важности составляет 100.

В схеме выставления оценок и конкурсном задании оцениваются только те компетенции, которые изложены в WSSS. Они должны отражать WSSS настолько всесторонне, насколько допускают ограничения соревнования по компетенции.

Схема выставления оценок и конкурсное задание будут отражать распределение оценок в рамках WSSS в максимально возможной степени. Допускаются колебания в пределах 5% при условии, что они не исказят весовые коэффициенты, заданные условиями WSSS.

Раздел		Важность (%)
1	**Организация и управление работой**	10
	Специалист должен знать и понимать: • принципы и методы, обеспечивающие продуктивную работу в команде; • как взять на себя инициативу и быть предприимчивыми с целью выявления, анализа и оценки информации из различных источников; • как создать корректную последовательность операций разрабатываемой системы с обеспечением необходимых уведомлений; • как подготовить соответствующую документацию об использовании разрабатываемой системы; • как правильно подготовить перечень требований со стороны клиента и выполнить полную поставку системы; • как применять в системе внутрифирменный стандарт (руководство по стилю).	
	Специалист должен уметь: • планировать производственный график на каждый день в соответствии с доступным временем и принимать во внимание временные ограничения и сроки сдачи работы; • применять исследовательские навыки и методики, чтобы поддерживать уровень собственной осведомлённости в актуальных отраслевых руководствах; • анализировать результаты собственной деятельности в сравнении с ожиданиями и потребностями клиента и организации; • создавать корректную последовательность операций разрабатываемой системы, с необходимыми уведомлениями; • готовить необходимую системную документацию по использованию, установке и запуску системы;	

	• осуществлять подготовку разработанной системы к поставке в соответствии с требованиями клиента; • подготавливать и реализовывать руководство по стилю для всей поставляемой системы; • внедрять внутрифирменный стандарт (руководство по стилю) для всей системы.	
2	**Компетенции общения и межличностных отношений**	**5**
	Специалист должен знать и понимать: • важность умения слушать; • необходимость осмотрительности и конфиденциальности при общении с заказчиками; • важность разрешения недопонимания и конфликтных ситуаций; • важность установления и поддержания доверия заказчика и продуктивных рабочих отношений; • важность навыков письменной и устной коммуникации; • как обеспечить правильную и понятную документацию по программному решению; • как подготовить доступный отчет и сообщить о результатах, задачах и других проблемах на протяжении всего процесса разработки и внедрения системы.	
	Специалист должен уметь: <u>Использовать навыки грамотности для:</u> • следования задокументированным инструкциям в предоставленном руководстве; • понимания инструкции по организации рабочего места и другой технической документации; • интерпретации и понимания системных спецификаций; • поддержания уровня собственной осведомлённости в актуальных отраслевых руководствах. <u>Использовать навыки устного общения для:</u> • обсуждения и выдвижения предложений относительно спецификации системы;	

	• регулярного уведомления клиента о ходе работы над системой; • ведения переговоров с клиентом относительно бюджета и сроков выполнения проекта; • сбора и подтверждения требований клиента; • презентации предлагаемого и итогового программного решения. <u>Использовать навыки письменного общения для:</u> • документирования программной системы (например, составления технических документов, руководств пользователя); • регулярного уведомления клиента о ходе работы над системой; • подтверждения, что созданное приложение соответствует исходным спецификациям, и утверждения пользователем готовой системы. <u>Использовать коммуникационные навыки при работе в команде для:</u> • сотрудничества с другими специалистами для получения желаемых результатов; • успешной работы над групповым решением проблем. <u>Использовать навыки управления проектами в:</u> • расстановке приоритетов и формировании графика выполнения задач; • распределении ресурсов между задачами.	
3	**Решение проблем, инновации, креативность**	**5**
	Специалист должен знать и понимать: • общие типы проблем и требований, которые могут возникнуть при разработке программного обеспечения; • общие типы проблем и требований, которые могут возникнуть в коммерческой организации; • диагностические подходы и подходящие к решению проблем системы или программные решения; • тенденции и разработки в отрасли, включая новые платформы, языки, условные обозначения и технические навыки;	

	• как использовать новейшие технологии, которые будут применяться в сценарии программного решения, которое требуется для наглядного сложного бизнес-решения проблемы; • как настроить, разработать и интегрировать в разработанное решение новейшие технологии и оборудование, которые будут способствовать лучшему бизнес-решению.	
	Специалист должен уметь: <u>Использовать аналитические навыки для:</u> • синтезирования сложной или неоднородной информации; • определения функциональных и нефункциональных требований спецификации. <u>Использовать навыки исследования и обучения для:</u> • понимания пользовательских требований (например, результатов опросов, анкет, поиска и анализа документов, объединенной разработки приложений и наблюдений); • независимого исследования возникших проблем. <u>Использовать навыки решения проблем для:</u> • своевременной идентификации и решения проблем; • грамотного сбора и анализа информации; • разработки альтернативы для использования новейших технологий для поддержки лучшего бизнес-решения; • выбора наиболее подходящей альтернативы для получения требуемого решения. Некоторые технологии могут использовать для решения аппаратные средства.	
4	**Анализ и проектирование программных решений**	**25**
	Специалист должен знать и понимать: • важность рассмотрения всех возможных вариантов и выбора лучшего решения на основе взвешенного аналитического суждения и интересов клиента; • важность использования системного анализа и методологий проектирования (например,	

	унифицированного языка моделирования (Unified Modelling Language), программной платформы MVC (Model-View-Control), фреймворков, шаблонов проектирования); • необходимость быть в курсе новых технологий и принимать решение о целесообразности их применения; • важность оптимизации архитектуры системы с учетом модульности и повторного использования; • принципы построения хранилищ данных, необходимых для бизнес-аналитики / отчетов о состоянии выполненных работ; • принципы построения интерфейсов и структур для мобильных решений.	
	Специалист должен уметь: <u>Анализировать системы с помощью:</u> • моделирования и анализа вариантов использования (например, диаграммы прецедентов, описания прецедентов, описания действующих субъектов (актеров), диаграммы пакетов вариантов использования); • структурного моделирования и анализа (например, объекты, классы, диаграммы классов предметной области); • динамического моделирования и анализа (например, диаграммы последовательностей, диаграммы взаимодействия, диаграммы состояний, диаграммы деятельности); • инструментов и методов моделирования (например, диаграмма сущностей и связей, нормализация, словарь данных). <u>Проектировать системы на основе:</u> • диаграммы классов, диаграммы последовательностей, диаграммы состояний, диаграммы деятельности; • описания объектов и пакетов; • схемы реляционной или объектной базы данных и диаграмм потоков данных;	

	• структуры человеко-машинного интерфейса / механизма взаимодействия с пользователем; • средств безопасности и контроля; • структуры многозвенного приложения.	
5	**Разработка программных решений**	**50**
	Специалист должен знать и понимать: • важность рассмотрения всех возможных вариантов и выбора лучшего решения для удовлетворения требований пользователя и интересов клиента; • важность использования методологий разработки системы (например, объектно-ориентированные технологии); • важность рассмотрения всех нормальных и ненормальных сценариев и обработки исключений; • важность соблюдения стандартов (например, соглашения по формату кода, руководства по стилю, дизайна пользовательского интерфейса, управления каталогами и файлами); • важность точного и постоянного контроля версий; • важность использования существующего кода в качестве основы для анализа и модификации; • важность выбора наиболее подходящих средств разработки из предложенных вариантов.	
	Специалист должен уметь: • использовать системы управления базами данных для построения, хранения и управления структурами и наборами данных для требуемой системы; • использовать подходящие версии программного обеспечения, среды разработки и инструменты, предназначенные для изменения существующего и написания нового исходного кода клиент-серверного программного обеспечения; • использовать новейшие средства разработки программного обеспечения и среды для создания или изменения мобильных решений с использованием физических мобильных	

	устройств в соответствии с требованиями клиента. • использовать подходящие версии программного обеспечения, среды разработки и инструменты, предназначенные для изменения существующего и написания нового исходного кода для системной интеграции с использованием веб-решений, веб-сервисов или единой подписки (например, с использованием службы каталогов) или API; • определять и интегрировать соответствующие библиотеки и фреймворки в программные решения; • строить и обслуживать многоуровневые приложения.	
6	**Тестирование программных решений**	**5**
	Специалист должен знать и понимать: • принципы устранения распространенных проблем программных решений; • важность отладки программных решений; • важность тщательного тестирования программных решений.	
	Специалист должен уметь: • осуществлять отладку программных решений; • разрабатывать тест-кейсы и проверять результаты тест-кейсов; • устранять и исправлять ошибки в программных решениях.	
	Всего	**100**

ОЦЕНОЧНАЯ СТРАТЕГИЯ И ТЕХНИЧЕСКИЕ ОСОБЕННОСТИ ОЦЕНКИ

Стратегия устанавливает принципы и методы, которым должны соответствовать оценка и начисление баллов WSR.

Экспертная оценка лежит в основе соревнований WSR. По этой причине она является предметом постоянного профессионального совершенствования и тщательного исследования. Накопленный опыт в оценке будет определять будущее использование и направление развития основных инструментов оценки, применяемых на

соревнованиях WSR: схема выставления оценки, конкурсное задание и информационная система чемпионата (CIS).

Оценка на соревнованиях WSR попадает в одну из двух категорий: измерение и судейское решение. Для обеих категорий оценки использование точных эталонов для сравнения, по которым оценивается каждый аспект, является существенным для гарантии качества.

Схема выставления оценки должна соответствовать процентным показателям в WSSS. Конкурсное задание является средством оценки для соревнования по компетенции, и оно также должно соответствовать WSSS. Информационная система чемпионата (CIS) обеспечивает своевременную и точную запись оценок, что способствует надлежащей организации соревнований.

Схема выставления оценки в общих чертах является определяющим фактором для процесса разработки Конкурсного задания. В процессе дальнейшей разработки Схема выставления оценки и Конкурсное задание будут разрабатываться и развиваться посредством итеративного процесса для того, чтобы совместно оптимизировать взаимосвязи в рамках WSSS и Стратегии оценки. Они представляются на утверждение Менеджеру компетенции вместе, чтобы демонстрировать их качество и соответствие WSSS.

Схема выставления оценки и Конкурсное задание могут разрабатываться одним человеком, группой экспертов или сторонним разработчиком. Подробная и окончательная Схема выставления оценки и Конкурсное задание, должны быть утверждены Менеджером компетенции.

Кроме того, всем экспертам предлагается представлять свои предложения по разработке Схем выставления оценки и Конкурсных заданий на форум экспертов для дальнейшего их рассмотрения Менеджером компетенции.

Во всех случаях полная и утвержденная Менеджером компетенции Схема выставления оценки должна быть введена в информационную систему соревнований (CIS) не менее чем за два дня до начала соревнований, с использованием стандартной электронной таблицы CIS или других согласованных способов. Главный эксперт является ответственным за данный процесс.

Основные заголовки Схемы выставления оценки являются критериями оценки. В некоторых соревнованиях по компетенции критерии оценки могут совпадать с заголовками разделов в WSSS; в других они могут полностью отличаться. Как правило, бывает от пяти до девяти критериев оценки, при этом количество критериев оценки должно

быть не менее трёх. Независимо от того, совпадают ли они с заголовками, Схема выставления оценки должна отражать долевые соотношения, указанные в WSSS.

Критерии оценки создаются лицом (группой лиц), разрабатывающим Схему выставления оценки, которое может по своему усмотрению определять критерии, которые оно сочтет наиболее подходящими для оценки выполнения Конкурсного задания.

Сводная ведомость оценок, генерируемая CIS, включает перечень критериев оценки.

Количество баллов, назначаемых по каждому критерию, рассчитывается CIS. Это будет общая сумма баллов, присужденных по каждому аспекту в рамках данного критерия оценки.

Каждый критерий оценки разделяется на один или более субкритериев. Каждый субкритерий становится заголовком Схемы выставления оценок.

В каждой ведомости оценок (субкритериев) указан конкретный день, в который она будет заполняться.

Каждая ведомость оценок (субкритериев) содержит оцениваемые аспекты, подлежащие оценке. Для каждого вида оценки имеется специальная ведомость оценок.

Каждый аспект подробно описывает один из оцениваемых показателей, а также возможные оценки или инструкции по выставлению оценок.

В ведомости оценок подробно перечисляется каждый аспект, по которому выставляется отметка, вместе с назначенным для его оценки количеством баллов.

При принятии решения используется шкала 0–3. Для четкого и последовательного применения шкалы судейское решение должно приниматься с учетом:

- эталонов для сравнения (критериев) для подробного руководства по каждому аспекту
- шкалы 0–3, где:
  - 0: исполнение не соответствует отраслевому стандарту;
  - 1: исполнение соответствует отраслевому стандарту;
  - 2: исполнение соответствует отраслевому стандарту и в некоторых отношениях превосходит его;
  - 3: исполнение полностью превосходит отраслевой стандарт и оценивается как отличное

Каждый аспект оценивают три эксперта, каждый эксперт должен произвести оценку, после чего происходит сравнение выставленных

оценок. В случае расхождения оценок экспертов более чем на 1 балл, экспертам необходимо вынести оценку данного аспекта на обсуждение и устранить расхождение.

При измеримых аспектах оценка каждого аспекта осуществляется тремя экспертами. Если не указано иное, будет присуждена только максимальная оценка или ноль баллов. Если в рамках какого-либо аспекта возможно присуждение оценок ниже максимальной, это описывается в Схеме оценки с указанием измеримых параметров.

Главный эксперт и Заместитель Главного эксперта обсуждают и распределяют Экспертов по группам (состав группы не менее трех человек) для выставления оценок. Каждая группа должна включать в себя как минимум одного опытного эксперта. Эксперт не оценивает участника из своей организации.

Каждый эксперт выступает в качестве члена команды оценки тестового проекта.

Эксперты будут разделены на команды оценки при максимально возможном равенстве в количестве оценки критериев.

Состав команд оценки будет определять Главный эксперт и Заместитель главного эксперта с целью достижения баланса между новыми и опытными экспертами в каждой из команд.

Судейские оценки не должны превышать 20%.

КОНКУРСНОЕ ЗАДАНИЕ

Продолжительность Конкурсного задания не должна быть менее 15 и более 22 часов.

Возрастной ценз участников для выполнения Конкурсного задания от 14 до 22 лет.

Вне зависимости от количества модулей, КЗ должно включать оценку по каждому из разделов WSSS.

Конкурсное задание не должно выходить за пределы WSSS.

Оценка знаний участника должна проводиться исключительно через практическое выполнение Конкурсного задания.

При выполнении Конкурсного задания не оценивается знание правил и норм WSR.

Конкурсное задание содержит 5 модулей:

1. Модуль 1. Системный анализ и проектирование.
2. Модуль 2. Разработка программного обеспечения.
3. Модуль 3. Стандарты разработки.
4. Модуль 4. Документирование.
5. Модуль 5. Оформление решения.

Набор модулей разрабатывается в зависимости от Конкурсного задания.

Общие файлы данных могут быть предоставлены на русском и английском языке и только английские версии программного обеспечения.

Участникам разрешен выход в интернет в зоне соревнования. Интернет будет доступен на обозначенных компьютерах в пределах 15 минут на участника на сессию. Это время включается во соревновательное время конкурса.

В течении соревновательного времени может быть объявлен «Overdrive» – неожиданное независимое испытание на скорость. Привычным запросом в данной области является, что чья-то работа может быть прервана по просьбе. В какой-то момент в каждый из дней проведения конкурса участникам может быть поставлена задача, которая должна быть решена в течение 30 минут. Это будет задача визуального характера, которая будут привлекать зрителей к территории соревнований. Задача должна быть одна, и иметь быстрое решение.

Все предконкурсные обсуждения проходят на особом форуме (http://forum.worldskills.ru). Решения по развитию компетенции должны приниматься только после предварительного обсуждения на форуме. Также на форуме должно происходить информирование о всех важных событиях в рамке компетенции. Модератором данного форума являются Международный эксперт и (или) Менеджер компетенции (или Эксперт, назначенный ими).

Информация для конкурсантов публикуется в соответствии с регламентом проводимого чемпионата. Информация может включать:

- Техническое описание;
- Конкурсные задания;
- Обобщённая ведомость оценки;
- Инфраструктурный лист;
- Инструкция по охране труда и технике безопасности;
- Дополнительная информация.

ОСОБЫЕ ПРАВИЛА ВОЗРАСТНОЙ ГРУППЫ 14-16 ЛЕТ

Время на выполнения задания не должны превышать 4 часов в день.

При разработке Конкурсного задания и Схемы оценки необходимо учитывать специфику и ограничения применяемой техники безопасности и охраны труда для данной возрастной группы. Так же необходимо учитывать антропометрические, психофизиологические и психологические особенности данной возрастной группы. Тем самым

Конкурсное задание и Схема оценки может затрагивать не все блоки и поля WSSS в зависимости от специфики компетенции.

Вопросы для самоконтроля

1. Что такое компетенция в терминах Ворлдскиллс Россия?

1. Это уровень профессиональных навыков конкурсанта
2. Это набор знаний и навыков в определенной профессиональной области
3. Это название площадки на чемпионате

2.Что такое Skill Management Plan (SMP)?

1. План работы на площадке компетенции
2. План застройки площадки
3. План развития навыков конкурсантов

3.Что указывается в Плане застройки площадки?

1. Планировка конкурсных участков
2. Расположение инфраструктуры на площадке (розетки, выводы сжатого воздуха, вода и т.п.)
3. Список инструмента, который может привезти с собой участник

4.Что должно указываться в Инфраструктурном листе?

1. Расположение инфраструктуры на площадке (розетки, выводы сжатого воздуха, вода и т.п.)
2. Список всего необходимого оборудования, инструмента, расходных материалов, офисного оснащения и принадлежностей, необходимых для работы площадки, предоставляемых организатором
3. Параметры, как освещенность, напряжение, давление и т.п.

5.Сколько часов, как правило, отводится на выполнение конкурсного задания?

1. от 3 до 15
2. от 15 до 22
3. 24

6.Какой из документов устанавливает "рамки компетенции"?

1. Техническое описание компетенции
2. Конкурсное задание
3. Регламент чемпионата

7.Какова элементарная позиция Критериев оценки?

1. Критерий
2. Субкритерий
3. Аспект

Культура безопасного труда

Культура безопасного труда — основа современного производства.

В современном мире, где наивысшей ценностью считается здоровье и жизнь каждого человека, требования к организации производства становятся все жестче.

На любом предприятии внешние и внутренние надзорные органы следят за безопасностью технологии, оборудования, материалов, микроклимата.

Сейчас это целая концепция, затрагивающая очень много аспектов и факторов. Культура её состоит в обеспечении приемлемых условий работы, а также в создании и соблюдении техники безопасности.

Культура безопасного труда – это сложная система, требующая специальных знаний и навыков. Создание её эффективной основы дело не быстрое и не простое. Если теории сейчас вполне достаточно и в бумажных книгах, и в электронных изданиях, то с практикой гораздо труднее.

На создание функционирующей культуры безопасного труда может уйти не один год – все зависит от размеров предприятия и количества работников. Что будет в результате:

1. Принципиальное уменьшение несчастных случаев (прежде всего, тяжелых) и развития профессиональных заболеваний.

2. Производство станет более комфортным и более производительным. Прежде всего, оно становится удобным для исполнителя.

Сейчас практикующим специалистам стало очевидно, что простое разделение режима и условий труда на безопасные и опасные не позволяет далеко продвинуться в предотвращении несчастных случаев. Преимущество должно быть отдано системному подходу. Осознание того, что люди, поставленные ими производственные задачи, их оборудование и окружающая среда представляют собой динамическую систему, позволяет создать значительно более эффективную технику безопасности.

Одной из новых ступеней в управлении безопасностью является понятие культуры безопасного труда. Это абстрактное понятие, относящееся к определенной организации или обществу. Не существует прямых способов ее изменения. Тем не менее, понятие культуры безопасности является решающим, если речь идет о возможности принятия превентивных мер. Фактически, когда существует культура безопасности как правильно выстроенная система, организация вряд ли

нуждается в соответствующих программах безопасности, поскольку последняя рассматривается в этом случае как обязательная часть процесса управления.

Поведение работников определяется тем, как они понимают культуру организации безопасности, и потому именно культура является самым важным элементом, от которого зависит, будет или нет программа безопасности труда эффективной.

Культура безопасного труда и безопасное поведение

Культура устанавливается не с помощью сформулированной на бумаге политики, а руководителями компании. Это достигается ежедневными действиями и решениями, а также посредством систем, обеспечивающих деятельность по организации безопасности со стороны руководителей всех звеньев (менеджеров, супервайзеров) и рабочих.

Культура может быть надлежащим образом оценена путем различных исследований. Впоследствии, когда в организации будут осознаны цели, к которым необходимо стремиться, она может быть усовершенствована.

В Российской Федерации вопросами общественного здравоохранения, включая координацию вопросов охраны труда, занимаются Министерство здравоохранения и Министерство труда и социальной защиты. Различные формы энергии — механическая, тепловая, радиационная, химическая и электрическая — рассматриваются как «болезнетворное начало», лежащее в основе телесного повреждения или травмы, причем механизм его распространения аналогичен микроорганизмам, вызывающим инфекционные заболевания. Специалисты и врачи, работающие по многим научно-техническим направлениям, особенно в области эпидемиологии, техники, эргономики, биомеханики, психологии поведения, техники безопасности и гигиены труда, занимаются изучением факторов, воздействующих:

- на работника (организм-хозяин);
- окружающую среду (производственную среду);
- источники ЭНЕРГИИ (болезнетворное начало);
- а также на приспособления, машины и конкретные производственные задачи (носители или передатчики «инфекции»), которые и обусловливают производственный травматизм.

По статистике около 96% всех травм и происшествий, происходящих из-за тактических ошибок, связано с опасными действиями — ПОВЕДЕНИЕМ самого работника, и только 4% происшествий — с условиями труда на рабочем месте.

Таким образом, деятельность преподавателя или мастера производственного обучения должна быть направлена на формирование культуры безопасного труда студентами СПО (будующими специалистами предприятий).

Чем так важны охрана труда и соблюдение правил техники безопасности?

В первую очередь потому, что самой высокой ценностью всегда является человек, его жизнь и здоровье. Ни размер заработной платы, ни уровень рентабельности предприятия, ни ценность производимого продукта не могут служить основанием для пренебрежения правилами безопасности и оправданием существующих угроз жизни или здоровью работников. Кроме того, в данном случае речь также идет о ценности конкретного человека как сотрудника с присущими ему знаниями, навыками и опытом.

Во-вторых, правильно организованная работа по обеспечению безопасности труда повышает дисциплинированность работников, что, в свою очередь, ведет к повышению производительности труда, снижению количества несчастных случаев, поломок оборудования и иных нештатных ситуаций, то есть повышает в конечном итоге эффективность производства.

В-третьих, охрана труда подразумевает не только обеспечение безопасности работников во время исполнения ими служебных обязанностей. На самом деле сюда также относятся самые разные мероприятия: например, профилактика профессиональных заболеваний, организация полноценного отдыха и питания работников во время рабочих перерывов, обеспечение их необходимой спецодеждой и гигиеническими средствами и даже выполнение социальных льгот и гарантий. Правильный подход к организации охраны труда на предприятии, грамотное использование различных нематериальных способов стимулирования работников дают последним необходимое чувство надежности, стабильности и заинтересованности руководства в своих сотрудниках. Таким образом, благодаря налаженной охране труда снижается также текучесть кадров, что тоже благотворно влияет на стабильность всего предприятия.

Характеристика опасных и вредных производственных факторов для студентов ИТ специальностей и сотрудников ИТ отделов

Требования к размещению рабочих мест с ПК и ПЭВМ относительно световых проемов.

Правильное расположение мест относительно световых проёмов способствует рациональному освещению рабочей поверхности, оказывает благоприятное психофизиологическое воздействие на работников, безопасности труда, снижению утомления и травматизма.

Недостаточное освещение рабочего места снижает эффективность длительной работы, повышает утомление, способствует развитию близорукости. Плохое освещение и длительное нахождение человека в таких условиях сопровождается снижением интенсивности обмена веществ в организме и ослаблением его реактивности. Излишне яркий свет снижает зрительные функции, перевозбуждает нервную систему, уменьшает работоспособность. В крайних случаях вызывает фотоожоги глаз, катаракты и другие нарушения.

Рабочие столы следует размещать таким образом, чтобы видео дисплейные терминалы были ориентированы боковой стороной к световым проёмам, что бы естественный свет падал преимущественно слева. На окна рекомендуется установить вертикальные или горизонтальные жалюзи.

Создание искусственного освещения в помещении кабинета обязательно, так как естественного для нормальных санитарно-гигиенических условий труда недостаточно.

Согласно СанПиН 2.2.2./2.1.1.1278-03 «Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий» освещенность должна составлять 400 Люкс.

Выбираем для освещения помещения люминесцентные лампы типа ЛБ – лампы белого света (мощность 40 Вт, номинальный световой поток Ф = 2600 лм) [ГОСТ 6825-91].

Проведем расчет необходимого количества светильников так, чтобы в помещении была

Расчет искусственного освещения выполняется методом коэффициента использования светового потока:

$$\Phi = \frac{E_H \cdot S \cdot z \cdot k}{}$$

где ЕН – освещенность рабочих поверхностей по нормали, лк;

Ф – световой поток одной лампы, лм;

S – площадь освещаемого помещения, м2;

z – поправочный коэффициент светильника, z = 1,15;

k – коэффициент запаса, учитывающий изменение освещенности при эксплуатации, k = 1,4;

m – число люминесцентных ламп в светильнике, шт.;

n – число светильников, шт.;

u – коэффициент использования светового потока, зависящий от индекса помещения, отраженности и т.д.

Индекс помещения рассчитывается по формуле:

$$i = \frac{S}{h \cdot (A + B)}$$

где А, В – соответственно длина и глубина помещения, м;

h – высота подвеса ламп над рабочей поверхностью, м.

Зная индекс помещения, и приняв коэффициенты отраженности от потолка, стен и рабочей поверхности соответственно светильники типа ЛДОР 50%, 30% и 10%, по справочнику выбираем коэффициент использования светового потока u=0,44.

При заданной освещенности и выбранном световом потоке конкретной лампы можно определить количество светильников n:

$$n = \frac{E_H \cdot S \cdot z \cdot k}{}$$

Расстояния между рабочими столами.

При расположении в помещении нескольких рабочих мест с ПЭВМ следует располагать их следующим образом:

Расстояние от задней поверхности видеомонитора до переднего края следующего стола не менее двух метров.

Расстояние между боковыми поверхностями видеомониторов не менее 1,2 метра.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 60-70 см, но не ближе 50 см с учетом размеров алфавитно-цифровых знаков и символов.

Размещение экрана видеомонитора.

Видеомонитор должен быть расположен в месте рабочей зоны, обеспечивающем удобство зрительного наблюдения в вертикальной плоскости под углом ±30 градусов от прямой линии взгляда студент или специалиста, а также комфортное использования ПЭВМ и возможность поворота видеомонитора относительно горизонтальной и вертикальной осей. Клавиатура располагается на поверхности стола на расстоянии 10 – 30 см от края, ближнего к пользователю или на регулируемой, отдельной от основной столешницы, поверхности. Конструкция стола должна обеспечить оптимальное и эргономичное размещение используемых периферийных устройств.

Конструкция рабочего стола и стула.

Для комфортной работы за ПЭВМ, стол должен соответствовать следующим условиям конструкции:

Высота стола для взрослого пользователя должна регулироваться в пределах 68-80 см, при отсутствии такой возможности высота должна быть 72,5 см.

Размеры рабочей поверхности стола для ПЭВМ, на основании которых производятся расчеты конструктивных размеров, считать:

Ширину – 800, 1000, 1200, 1400 мм (при высоте 72,5 см);

Глубину – 800 и 1000мм (при высоте 72,5 см).

Пространство для ног высотой от 60 см и шириной не менее 50 см. Глубина на уровне колен от 45 см и на уровне вытянутых ног от 65 см.

Кресло должно иметь подъёмно-поворотное и регулироваться по высоте и углам наклона сиденья и спинки. Также конструкция кресла должна обеспечиваться:

Ширина и глубина поверхности сиденья не меньше 40 см;

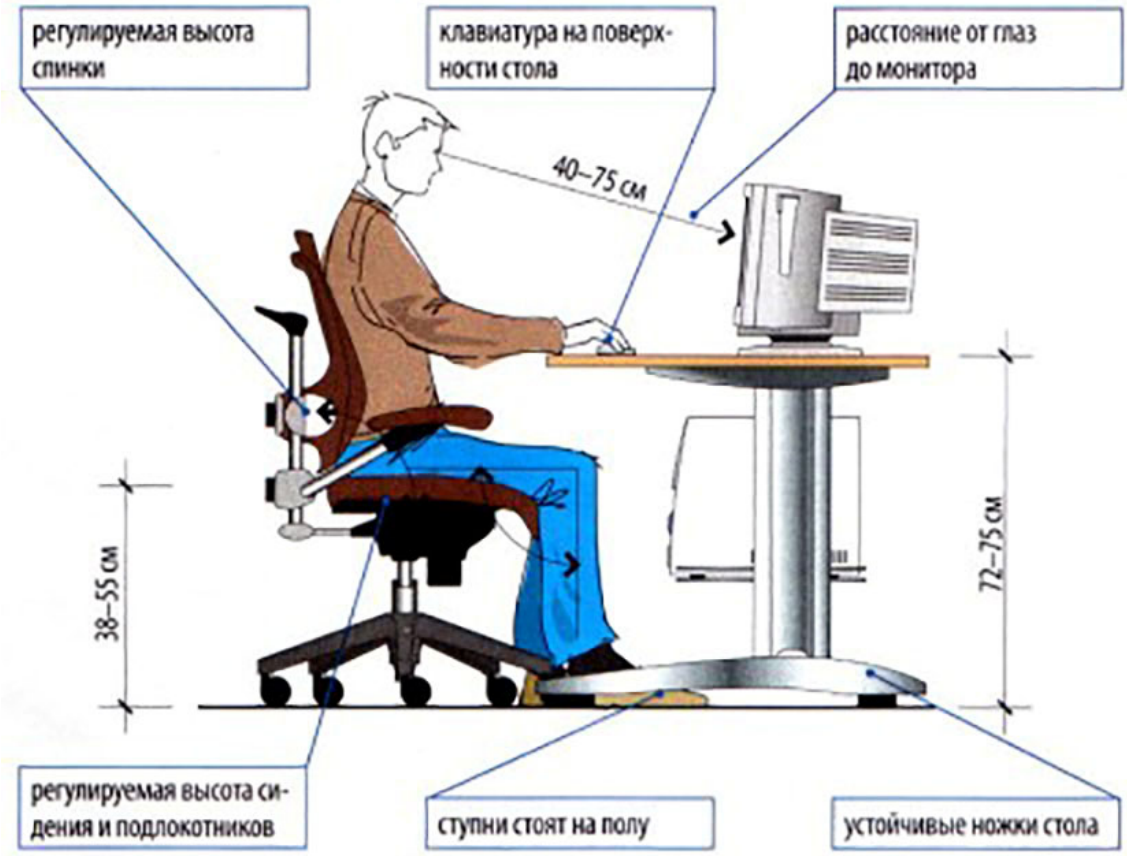
Закруглённым передним краем сиденья;

Регуляторами высоты в пределах 40 – 55 см и углом наклона вперед до 15 градусов и до 5 градусов назад;

Высотой опорной поверхности спинки 30 ± 2 см, шириной от 38 см и радиусом кривизны горизонтальной плоскости 40 см;

Регулятор наклона спинки в вертикальной плоскости ± 30 градусов;

Съёмные или стационарные подлокотники длиной от 25 см и шириной 5 -7 см, с возможностью регулировки высоты над сиденьем в пределах 23 ± 3 см и внутреннего расстояния в пределах 35 - 50 см.



Требования к окрашиванию стен, потолка и пола

Окрашивание помещений должно создавать благоприятные условия зрительного восприятия. Источники света в помещении могут давать отражение от экрана видеомонитора, значительно ухудшая точность знаков, что влечет за собой значительное напряжение, особенно при долгой работе. Отражения, в том числе и от второстепенных источников света, должно быть сведено к минимуму. Для этой цели могут использоваться шторы, жалюзи, экраны. студент или специалист рабочий труд производственный

В помещении, в котором располагается ПЭВМ, необходимо обеспечить следующие величины коэффициента отражения:

Для потолка: 60 – 70%;

Для стен: 40 – 50%;

Для пола: ~30%;

Для прочих поверхностей: 30 – 40%.

Пожарная безопасность

Понятие «пожарная безопасность» означает состояние объекта, при котором исключается возможность пожара, а в случае его возникновения

предотвращается воздействие на людей опасных факторов пожара и обеспечивается защита материальных ценностей.

Пожарная безопасность подразумевает разработку политики предприятия по недопущению возникновения и развития пожара, направленную на решение следующего круга задач:

Реализацию комплекса мероприятий, направленных на ограничение распространения пожара.

Обеспечение объектов средствами пожарного контроля, оповещения сотрудников предприятия о возникновении нештатной ситуации и непосредственного пожаротушения.

Принятие организационных мер, направленных на контроль над соблюдением сотрудниками нормативных требования ПБ.

Повышение уровня информированности работников и должностных лиц о мерах по обеспечению пожарной безопасности.

Организацию и проведение производственного контроля.

Обеспечение пожарной безопасности неразрывно связано с соблюдением основных нормативных требований в сфере техники безопасности и принятием инструкции по пожарной безопасности, действующей в рамках предприятия.

Основной причиной возникновения пожаров в здании может являться неисправность электропроводки, не правильная эксплуатация электротехники, использование открытого огня в не приспособленных для этого местах.

Определение пожароопасной категории помещения осуществляется путем сравнения максимального значения удельной временной пожарной нагрузки на любом из участков с величиной удельной пожарной нагрузки, приведенной в таблице.

Категорирование помещений В1 – В4

Категория помещения	Удельная пожарная нагрузка g на участке, МДж•м-2	Способ размещения
В1	Более 2200	Не нормируется
В2	1401 — 2200	См. примечание
В3	181 — 1400	См. примечание
В4	1 — 180	На любом участке пола помещения площадью 10 м2 .

Приведем пример расчета пожарной нагрузки (для обычного кабинета на 15 РМ)

Перечень материалов

Оборудование	Вещество/материал	Масса оборудования, кг	Количество оборудования	Суммарная масса оборудования, кг
Стол	Древесина	25	22	550
Стул	Древесина	5	30	150
Шкаф	Древесина	60	4	240
ПЭВМ	Пластмасса	10	11	110
Сканер	Пластмасса	4	1	4
Принтер	Пластмасса	5	1	5
Линолеум	Линолеум на тканной основе	50	1	50
Жалюзи	Пластмасса	5	3	15
Доска	Пластмасса	25	1	25
Учебники	Бумага	1	100	100

Согласно таблице, материал пожарной нагрузки, имеющийся в помещении: изделия из пластмассы – 159кг, бумага – 100 кг, древесина – 940 кг, линолеум – 50 кг.

Определяем пожарную нагрузку Q (МДж) из соотношения:

$$Q = \sum_{i=1}^{n} G_i \cdot Q_{ni}^{p} \quad (12)$$

где - количество i – го материала пожарной нагрузки, кг;

Q_{ni}^{p} - низшая теплота сгорания i – го материала пожарной нагрузки, МДж/кг.

$Q = 159 * 45,67 + 100 * 13,4 + 940 * 13,8 + 50 * 20,292 = 22588,13$ Мдж

Определяем удельную пожарную нагрузку g (МДж/$м^2$).

$$g = \frac{Q}{S}$$

где S – площадь размещения пожарной нагрузки, $м^2$.

$$g = \frac{Q}{S} = \frac{22588,13}{45} = 501,96$$ МДж/$м^2$

Здесь g т = 2200 МДж/м2 при 1401 МДж/м2 2200 МДж/м2 и g т = 1400 МДж/м2 при 181 МДж/м2 1400 МДж/м2 .

Данное помещение относится к категории В2 в соответствии с нормами пожарной безопасности.

При возникновении пожара необходимо эвакуировать персонал согласно плану эвакуации.

Пути эвакуации из здания не загромождены, дверные проемы по пути следования оснащены сигнальным табло с подсвечиваемой надписью «выход». Здание оборудовано пожарной сигнализацией, включающей в себя сеть дымоулавливающих датчиков и устройств сигнализации. В случае возникновения пожара сигнализацию можно включить принудительно, нажав на кнопку пожарной тревоги.

Типовая инструкция по охране труда для студентов/специалистов ИТ отделов

1. ОБЩИЕ ТРЕБОВАНИЯ БЕЗОПАСНОСТИ

1.1. Настоящая инструкция по охране труда студента или специалиста, занятого эксплуатацией персональных электронно-вычислительных машин (ПЭВМ) и видеодисплейных терминалов (ВДТ), разработана с учетом условий его работы в конкретной организации.

1.2. На студента или специалиста могут воздействовать опасные и вредные производственные факторы:

а) физические:

- повышенные уровни электромагнитного излучения;
- повышенные уровни рентгеновского излучения;
- повышенные уровни ультрафиолетового излучения;
- повышенный уровень инфракрасного излучения;
- повышенный уровень статического электричества;
- повышенные уровни запыленности воздуха рабочей зоны;
- повышенное содержание положительных аэроионов в воздухе рабочей зоны;
- пониженное содержание отрицательных аэроионов в воздухе рабочей зоны;
- пониженная или повышенная влажность воздуха рабочей зоны;
- пониженная или повышенная подвижность воздуха рабочей зоны;
- повышенный уровень шума;
- повышенный или пониженный уровень освещенности;
- повышенный уровень прямой блесткости;
- повышенный уровень отраженной блесткости;
- повышенный уровень ослепленности;
- неравномерность распределения яркости в поле зрения;
- повышенная яркость светового изображения;
- повышенный уровень пульсации светового потока;

- повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;

б) химические:

- повышенное содержание в воздухе рабочей зоны двуокиси углерода, озона, аммиака, фенола, формальдегида и полихлорированных бифенилов;

в) психофизиологические:

- напряжение зрения;
- напряжение внимания;
- интеллектуальные нагрузки;
- эмоциональные нагрузки;
- длительные статические нагрузки;
- монотонность труда;
- большой объем информации, обрабатываемой в единицу времени;
- нерациональная организация рабочего места;

г) биологические:

- повышенное содержание в воздухе рабочей зоны микроорганизмов.

1.3. К работам допускаются:

- лица не моложе 18 лет, прошедшие обязательный при приеме на работу и ежегодные медицинские освидетельствования на предмет пригодности для работы;
- прошедшие вводный инструктаж по охране труда;
- прошедшие обучение безопасным приемам и методам труда по программе, утвержденной руководителем предприятия (работодателем), разработанной на основе Типовой программы, и прошедшие проверку знаний, в том числе по электробезопасности;
- прошедшие курс обучения на персональном компьютере с использованием конкретного программного обеспечения;
- прошедшие инструктаж по охране труда на конкретном рабочем месте по данной инструкции.

1.4. Студент или специалист должен быть обеспечен СИЗ в соответствии с Межотраслевыми правилами обеспечения работников специальной одеждой, специальной обувью и другими средствами индивидуальной защиты, утвержденными Приказом Минздравсоцразвития России от 01.06.2009 N 290н; выдаваемые работникам средства индивидуальной защиты должны соответствовать характеру и условиям работы и обеспечивать безопасность труда. Не допускаются приобретение и выдача работникам средств индивидуальной защиты без сертификата соответствия.

Характеристика выданных СИЗ (номенклатура, срок выдачи и нормы соответствия) устанавливается из личных карточек работников, занятых на определенном рабочем месте.

Нормативные номенклатура и сроки выдачи СИЗ определяются согласно Типовым отраслевым нормам бесплатной выдачи рабочим и служащим специальной одежды, специальной обуви и других СИЗ.

2. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ ПЕРЕД НАЧАЛОМ РАБОТЫ

2.1. Перед началом работы студент или специалист обязан:

- осмотреть и привести в порядок рабочее место;
- отрегулировать освещенность на рабочем месте, убедиться в достаточности освещенности, отсутствии отражений на экране, отсутствии встречного светового потока;
- проверить правильность подключения оборудования в электросеть;
- протереть специальной салфеткой поверхность экрана;
- убедиться в отсутствии дискет в дисководах процессора персонального компьютера;
- проверить правильность установки стола, стула, подставки для ног, пюпитра, положения оборудования, угла наклона экрана, положение клавиатуры и, при необходимости, произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.

2.2. При включении компьютера соблюдать правила электробезопасности.

2.3. Студенту или специалисту запрещается приступать к работе при:

- отсутствии на ВДТ гигиенического сертификата, включающего оценку визуальных параметров;
- отсутствии информации о результатах аттестации условий труда на данном рабочем месте или при наличии информации о несоответствии параметров данного оборудования требованиям санитарных норм;
- отключенном заземляющем проводнике защитного фильтра;
- обнаружении неисправности оборудования;
- отсутствии защитного заземления устройств ПЭВМ и ВДТ;
- отсутствии углекислотного или порошкового огнетушителя и аптечки первой помощи;

- нарушении гигиенических норм размещения ВДТ (при однорядном расположении менее 1 м от стен, при расположении рабочих мест в колонну на расстоянии менее 1,5 м, при размещении на площади менее 6 кв. м на одно рабочее место, при рядном размещении дисплеев экранами друг к другу).

3. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ ВО ВРЕМЯ РАБОТЫ

3.1. Студент или специалист во время работы обязан:

- выполнять только ту работу, которая ему была поручена и по которой он был проинструктирован;
- в течение всего рабочего дня содержать в порядке и чистоте рабочее место;
- держать открытыми все вентиляционные отверстия устройств;
- при необходимости прекращения работы на некоторое время корректно закрыть все активные задачи;
- выполнять санитарные нормы и соблюдать режимы работы и отдыха;
- соблюдать правила эксплуатации вычислительной техники в соответствии с инструкциями по эксплуатации;
- соблюдать установленные режимом рабочего времени регламентированные перерывы в работе и выполнять в физкультпаузах и физкультминутках рекомендованные упражнения для глаз, шеи, рук, туловища, ног;
- соблюдать расстояние от глаз до экрана в пределах 60 - 80 см.

3.2. Студенту или специалисту во время работы запрещается: прикасаться к задней панели системного блока при включенном питании; переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании; загромождать верхние панели устройств бумагами и посторонними предметами; допускать захламленность рабочего места бумагой - в целях недопущения накапливания органической пыли; производить отключение питания во время выполнения активной задачи; производить частые переключения питания; допускать попадание влаги на поверхность системного блока, монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств; включать сильно охлажденное (принесенное с улицы в зимнее время) оборудование; производить самостоятельно вскрытие и ремонт оборудования.

4. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ В АВАРИЙНЫХ СИТУАЦИЯХ

4.1. Студент или специалист обязан:

- во всех случаях обнаружения обрыва проводов питания, неисправности заземления и других повреждений электрооборудования, появления запаха гари немедленно отключить питание и сообщить об аварийной ситуации руководителю и дежурному электрику;
- при обнаружении человека, попавшего под напряжение, немедленно освободить его от действия тока путем отключения электропитания и до прибытия врача оказать потерпевшему первую медицинскую помощь;
- при любых случаях сбоя в работе технического оборудования или программного обеспечения немедленно вызвать представителя инженерно-технической службы эксплуатации вычислительной техники;
- в случае появления рези в глазах, при резком ухудшении видимости - невозможности сфокусировать взгляд или навести его на резкость, появлении боли в пальцах и кистях рук, усилении сердцебиения немедленно покинуть рабочее место, сообщить о происшедшем руководителю работ и обратиться к врачу;
- при возгорании оборудования отключить питание и принять меры к тушению очага пожара при помощи углекислотного или порошкового огнетушителя, вызвать пожарную команду и сообщить о происшествии руководителю работ.

5. ТРЕБОВАНИЯ БЕЗОПАСНОСТИ ПОСЛЕ ОКОНЧАНИЯ РАБОТЫ

5.1. По окончании работ студент или специалист обязан соблюдать следующую последовательность выключения вычислительной техники:

- произвести закрытие всех активных задач;
- убедиться, что в разъемах нет внешних накопителей;
- выключить питание системного блока;
- выключить питание всех периферийных устройств;
- отключить блок питания.

5.2. По окончании работ студент или специалист обязан осмотреть и привести в порядок рабочее место, вымыть с мылом руки и лицо.

Особенности обучения в соответствии со стандартами Ворлдскиллс и спецификацией стандартов Ворлдскиллс по компетенции

Современный этап развития образования ставит перед будущим учителем ряд важных задач, связанных с его готовностью работать в новых условиях перестройки содержания обучения. Говоря о технологическом обеспечении, можно выделить следующие направления применительно к образовательному процессу в колледже.

Во–первых необходимо совершенствовать и оптимизировать процесс передачи возрастающего объема информации в учебных дисциплинах. В связи с этим в системе профессионального образования расширяются возможности применения педагогических технологий, как проблемного, программирования, интенсивного, модульного и интегративного обучения, технологии игрового моделирования, новых информационных технологий.

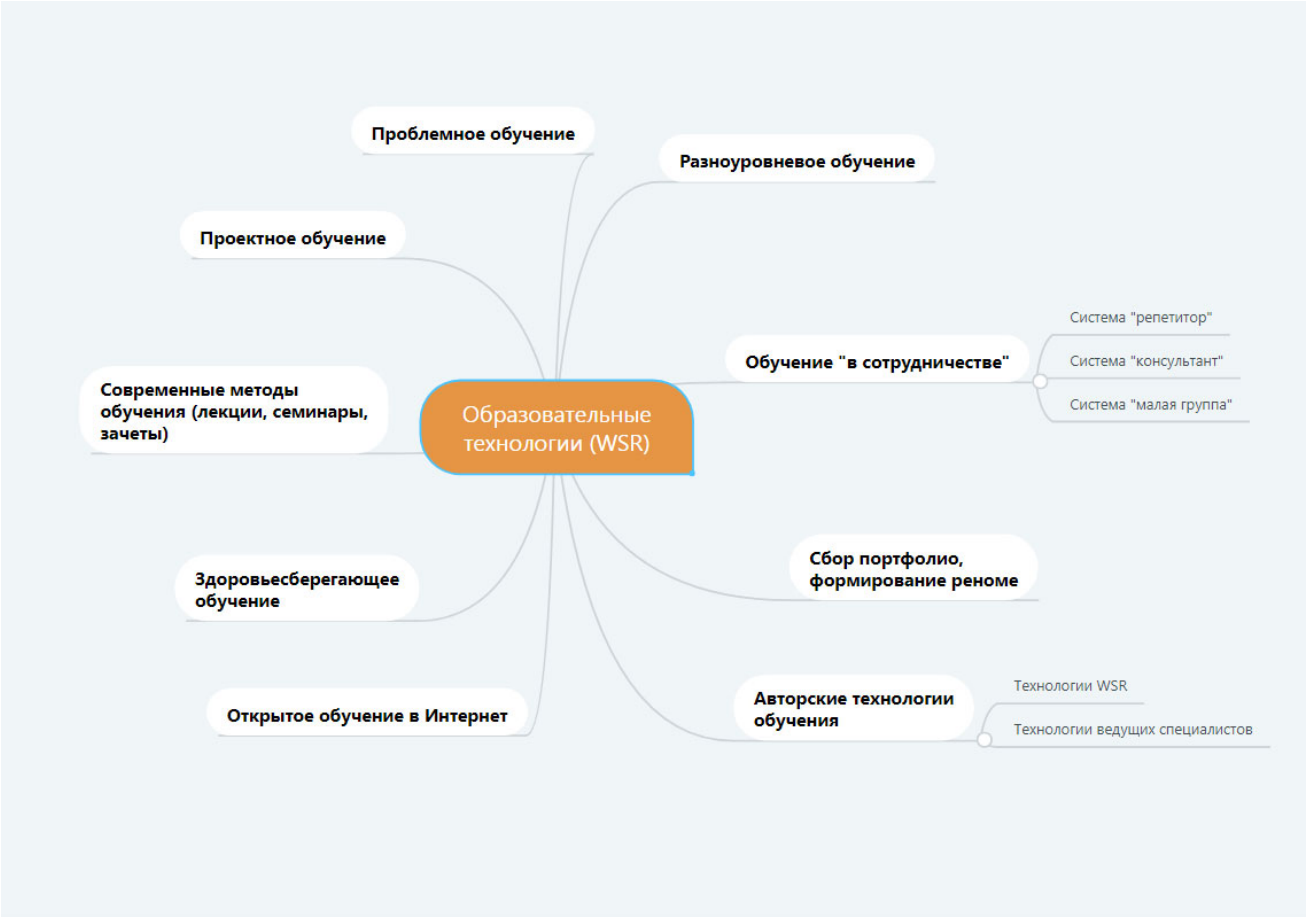
Во–вторых, по прежнему актуальность имеют образовательные технологии предметного обучения. При технологизации предметного обучения важно учитывать ценностно-деятельностный аспект всех этапов изучения предмета и его профессиональную специализацию.

Образовательные технологии, используемые в колледже будут более результативны, если они не только инструментально обоснованы, но и обеспечивают профессионально-личностный рост будущего специалиста, а также учитывают механизмы включения студентов в исследовательскую и творческую деятельность.

В–третьих, сфера образования требует новой научно-обоснованной методики подготовки «новых» специалистов, а также определение степени влияния на эти результаты новых учебных планов, образовательных программ и технологий.

Для образования проблема педагогических технологий стала столь актуальной ещё и в условиях экономических изменений в стране.

Внедрение стандартов Ворлдскиллс Россия предполагает не только изменения в содержании образования, но и новые технологические приемы в организации образовательного процесса.



Важное значение имеют новые образовательные технологии.

В понимании сущности «образовательная технология» можно выделить следующие подходы:

Технико-инструментальный. Он характеризуется тем, что понятие «педагогическая технология» по содержанию сводится в основном к техническим средствам обучения и его инструментам.

Первоначально под «педагогическая технологиями» понимали средства, используемые педагогами для решения образовательных задач. Особое внимание было уделено таким устройствам, которые не только дополняли образовательный процесс (средства звукозаписи, учебное кино, телевидение и др.), но и могли бы влиять на ход обучения, брать на себя отдельные функции учителя.

Таким образом, сторонники технико-инструментального подхода ориентировались на выбор средств, обеспечивающих эффективность образовательного процесса. Вопрос о соотношении метода, приёмов и технологий обучения активно дискутируются психологами, педагогами и методистами.

Второй подход функционально-процессуальный. По сути, есть технологический подход к построению образовательного процесса, в котором способы, средства и условия могут выполнять разные функции при достижении образовательных задач.

В рамках данного подхода «педагогическая технология» характеризуется как процесс выбора и использования определенного набора средств, для достижения педагогической цели в определенной логике, эффективность которых будет достигаться только при определенно заданных условиях.

Технологический подход к построению образовательного процесса ориентируется на выделении таких процедур как: компонент педагогического процесса в виде системы действий; циклический алгоритм действий учителя и учащихся; возможность построения педагогической системы на основе определенного набора дидактических педагогических приемов (2).

Своеобразие технологического подхода проявляется также и в том, что он даёт не описательную, а конструктивную, предписывающую схему образовательного процесса. В технологии образовательного процесса особое место уделяется развитию технологических средств обучения, использованию их возможностей в практике работы преподавателей вуза для большего охвата семинарских занятий, индивидуализации каналов подачи учебного материала и др.

Если рассматривать педагогическую технологию с этой позиции, то их применение в большей мере направлено на создание своего рода технологической среды в обучающем процессе независимо от использования технических — организационных средств.

Совершенствование подготовки специалистов связано и с созданием условий для осознанного восхождения личности к более высокому уровню компетентности, а затем и к профессионализму.

Курс «Базы данных» является одним из краеугольных камней профессионального цикла учебного плана направления подготовки ФГОС 09.02.07 любого профиля. Согласно стандарта, профессиональная деятельность выпускника этого направления включает, в частности, «внедрение, адаптацию, настройку и интеграцию проектных решений по созданию информационных систем; сопровождение и эксплуатацию информационных систем». При этом вполне очевидно, что любая информационная система имеет в своей основе некую базу данных и содержит в качестве одной из составных частей систему управления базой банных (СУБД).

Во всех технических ссузах РФ преподается дисциплина, так или иначе, связанная с проектированием баз данных, построением запросов на выборку, вставку, обновления или удаления.

В более продвинутых ссузах помимо перечисленного требуется глубоко понимать, как происходит индексирование таблиц с данными,

как производится нормализация реляционных таблиц, а также как правильно провести оптимизацию сложных запросов.

У большинства студентов не получается с первого раза понять квинтэссенцию правильной организации связанных таблиц, они путаются в нормализации (напомню, что существует всего 5 форм нормализации), не понимают, как правильно составлять связанные запросы, которые объединяют несколько таблиц одновременно.

В конце учебного курса, как правило, требуется провести реализацию курсовой работы по программированию. Пользовательский интерфейс и код программы создаются на одном из современных языках программирования, а данные требуется хранить в какой-либо **базе данных**.

Вот тут и начинаются фундаментальные проблемы. Мало того, что студент практически не владеет базовыми конструкциями выбранного языка программирования, так нужно еще и уметь «доставать» данные из реляционных таблиц, причем эти таблицы должны иметь правильную структуру.

Таким образом, будущая профессиональная деятельность обучащихся обязательно будет связана с базами данных и, следовательно, качество их подготовки напрямую зависит от полноты освоения именно этого курса.

При планировании курса мы ставим следующие три цели:

1. Познакомить студентов с теоретическими основами баз данных.
2. Обучить студентов методам проектирования баз данных и создания приложений для них.
3. Познакомить учащихся с примерами работы конкретных СУБД.

Большинство опытных преподавателей согласятся с тем, что оптимальным и наиболее логичным способом достижения этих целей будет работа над ними именно в перечисленном порядке: сначала нужно изложить теоретические основы, потом, на их базе, перейти к проектированию и программированию, и уже в качестве завершающих штрихов устроить обзорное знакомство с некоторыми СУБД.

Такой подход позволил бы сократить время на прохождение второго и третьего этапа за счет того, что к началу второго этапа студенты уже хорошо знакомы с теорией, и проектирование становится для них просто примером ее применения; а к началу третьего – учащиеся уже попробовали самостоятельно разработать базу данных и приложения для нее, поэтому в состоянии относиться к различным СУБД просто как к частным случаям применения приобретенных навыков.

Таким образом, описанный выше абсолютно логичный подход оказывается неприменим в нашей ситуации. Поэтому нам представляется оптимальным спиралевидное построение учебной программы: практически каждый элемент курса изучается дважды, но на разных по сложности уровнях.

Начинать изучение курса лучше с изучения общих принципов реляционной модели. Сразу же, в качестве иллюстрации, рассматриваем общие принципы проектирования баз данных и переходим к изучению стандарта языка SQL. Язык рассматриваем достаточно подробно, и параллельно на лабораторных занятиях идет изучение возможностей MSSQL или MySQL, что позволяет знакомиться с SQL не только в теории, но и на практике.

После довольно детального знакомства с SQL возвращаемся к реляционной модели: теперь уже студенты хорошо понимают, о чем будет идти речь, и это позволит компенсировать некоторую нехватку математической базы. Конечно, в изложение этого материала все равно придется вносить некоторые коррективы с учетом уровня учащихся, но такое построение курса все же позволит осветить теоретические основы весьма тщательно, что обеспечит глубокое освоение предмета. Здесь же активно используются для иллюстрации теоретических моментов примеры на языке SQL или описывается, в чем язык отклоняется от идеалов реляционной модели.

Завершающим этапом курса становится изучение методологий проектирования баз данных с описанием всевозможных нормальных форм, которые накладывают ограничения на процесс проектирования. Нетрудно заметить, что этот материал– довольно сухой, теоретический. Поэтому содержание лабораторных работ в это время отклоняется от лекционного курса, что позволяет дать студентам навыки разработки собственных приложений для работы с базами данных – к этим темам преподаватель лабораторного практикума переходит сразу после завершения изучения MSSQL или MySQL и может самостоятельно выбрать СУБД и язык программирования без оглядки на содержание лекций.

Главное – реализация идеи, что учащиеся должны получить начальные навыки создания приложений для работы с базами данных.

Помимо этого, также в рамках практических занятий, необходимо дать представление об инструментах проектирования баз данных.

Изучение **баз данных** сопоставимо по сложности с изучением какого-либо языка программирования.

Предлагаем следующую последовательность изучения курса по укрупненным темам в соответствие с ПООП 09.02.07.

Введение в SQL

- Что такое язык структурированных запросов SQL
- Создание баз данных. Реляционные базы данных
- Создание таблицы
- Модификация существующей таблицы
- Удаление существующей таблицы
- Анализ потребностей хранения данных
- Разделение данных с точки зрения логики
- Правильный выбор типов данных
- Использование первичных ключей

Заполнение таблиц данными

- Вставка данных с помощью оператора INSERT
- Обновление данных с помощью оператора UPDATE
- Директива WHERE
- Логические операции AND и OR
- Удаление данных с помощью оператора DELETE

Извлечение данных

- Оператор на выборку данных SELECT
- Ключевое слово DISTINCT
- Использование псевдонимов
- Фильтрация с использованием директивы WHERE
- Приоритет операций
- Логическая операция NOT
- Операция BETWEEN
- Групповые символы и операция LIKE
- Операция IN
- Сортировка результатов с помощью директивы ORDER BY
- Объединение столбцов
- Выбор данных из нескольких столбцов
- Данные типа NULL

Проектирование баз данных

- Нормализация
- Первая нормальная форма

- Вторая нормальная форма
- Третья нормальная форма
- Использование ограничений
- Ограничение NOT NULL
- Ограничение UNIQUE
- Ограничение CHECK
- Первичный ключ PRIMARY KEY
- Внешний ключ FOREIGN KEY
- Индексы - специальные таблицы поиска

Обработка данных

- Арифметика SQL
- Общие математические функции: ABS(), POWER(), SQRT(), RAND(), CEILING(), FLOOR(), ROUND()
- Функции для обработки строк: SUBSTRING(), UPPER(), LOWER(), REVERSE(), TRIM(), LENGTH(), SOUNDEX(), DIFFERENCE()
- Функции даты: DAY(), MONTH(), YEAR()
- Преобразование типов данных
- Значение NULL и математические функции
- Значение NULL и строковые данные
- Комбинация INSERT INTO и SELECT

Группировка данных и вычисление итогов

- Предложение GROUP BY
- Суммирование данных и вычисление итоговых значений: SUM()
- Усреднение результатов: AVG()
- Максимальное и минимальное значение: MAX(), MIN()
- Комбинация предложений HAVING и GROUP BY

Извлечение данных из нескольких таблиц

- Внутреннее соединение
- Соединения по эквивалентности и по неэквивалентности
- Множественные соеднинения и множественные условия
- Перекрестное соединение
- Автосоединения
- Внешине соединения
- Левое внешнее соединение

- Правое внешнее соединение
- Полное внешнее соединение
- Операция UNION

Подзапросы

- Подзапросы в списке SELECT
- Подзапросы в предложениях WHERE
- Операции в подзапросах
- Использование операций ANY, SOME, ALL
- Использование операции EXISTS
- Использование HAVING с подзапросами
- Связанные подзапросы
- Подзапросы в других операторах
- Подзапросы с оператором UPDATE
- Подзапросы с оператором DELETE FROM

Сложные запросы

- Анализ данных на выборку
- Выбор столбцов для списка SELECT
- Создание предложения FROM
- Рекомендации по созданию сложных запросов

Представления

- Базовые понятия о представлениях
- Создание представлений
- Типы представлений
- Базовое представление
- Представления строк
- Представления полей
- Фильтруемые окна представлений
- Итоговые представления
- Обновление данных представлений
- Ограничения обновлений
- Опция CHECK
- Удаление представлений

Транзакции

- Использование транзакций

- Свойства транзакций: атомарность, согласованность, изолированность, устойчивость
- Модель ANSI
- Оператор COMMIT
- Оператор ROLLBACK
- Transact-SQL: BEGIN TRANSACTION, COMMIT TRANSACTION, SAVE TRANSACTION, ROLLBACK TRANSACTION
- Журналы транзакций
- Блокировки и их типы
- Эскалация множества блокировок
- Уровни локализации: SET TRANSACTION, SERIALIZABLE, REPEATABLE READ, READ COMMITED, READ UNCOMMITED
- Версии данных
- Потерянные обновления, незафиксированные данные, несогласованные данные, фантомная вставка

Безопасность SQL

- Основные понятия безопасности
- Идентификаторы пользователей и учетные записи
- Создание учетных записей
- Изменение учетных записей
- Удаление учетных записей
- Группы пользователей (роли)
- Объекты и привилегии
- Опция WITH GRANT OPTION
- Оператор REVOKE
- Опция CASCADE и RESTRICT

Настройка базы данных

- Настройка аппаратных средств
- Рабочие станции
- Файлы базы данных
- Процессоры
- Разделенные сети
- Кэши: жесткий диск, процессор, база данных
- Настройка программного кода
- Индексы и сканирование таблиц

Примерная модульная схема для изучения курса по базам данных

Модуль БД

1	Введение в SQL Понятие БД. Понятие СУБД. Реляционная БД. XML. Структура и таблицы БД. Понятие SQL. Типы данных. Создание переменных. Типы данных в T-SQL. Создание БД и таблицы. Заполнение таблицы.
2	Запросы Data Manipulation Language. Выборкаданных. Запрос SELECT. Выборка данных с использованием конструкции WHERE, HAVING. Конструкция сортировки ORDER BY. Агрегированные функции SUM, COUNT, AVG, MIN, MAX. Оператор изменение данных в таблице INSERT ,UPDATE. Оператор удаления данных из таблицы DELETE. INNER JOIN, OUTER JOIN, LEFT JOIN, RIGHT JOIN, UNION, EXCEPT, INTERSECT
3	Основы DDL Удаление, создание и изменение БД. Создание таблицы - CREATE и удаление таблицы – DROP. Изменение таблицы - ALTER TABLE. Создание и работа с первичным ключом. Создание и работа с внешним ключом. Ограничение ссылочной целостности. Типы связи
4	Проектирование БД Нормализация (1НФ, 2НФ, 3НФ). Денормализация. Словарь данных
5	Вложенные запросы Подзапрос. Связанные запросы. Вложенные запросы. Связанный вложенный запрос – EXISTS, ALL, ANY, IN. Временные таблицы.
6	*Индексирование* *Организация памяти БД, понятие «Страница». Создание таблицы без индексов и поиск по ней. Понятие «Индекс». Кластеризованный индекс. Некластеризованный индекс.*
7	*Хранимые процедуры* *Понятие хранимой процедуры. Условная конструкция IF … ELSE. Операция EXISTS. Оператор CASE. Оператор WAITFOR. Блоки TRY и CATCH. Активизация сообщения об ошибке вручную. Создание, вызов и удаление хранимой процедуры. Создание хранимой процедуры с оператором RETURN. Создания процедуры регистрации ошибок*
8	*Транзакции. Триггеры*

	Понятие транзакции. Понятие триггера. Работа с транзакциями. Точка сохранение транзакции. Создание процедуры записи данных о пользователе. Создание и проверка работы триггеров.
9	Практика: Создание БД. Добавление и удаление таблиц в существующую БД. Импорт данных. Запросы. Подзапросы.

Одними из основных разделов современных курсов информационной направленности, преподаваемых в большинстве российских колледжей, являются разделы, связанные с обучением программированию, формированием у обучаемых алгоритмического мышления, подготовке к оперированию с важнейшими инструментальными системами и средствами. Вместе с тем в ранее опубликованных работах неоднократно подчеркивалась необходимость совершенствования методических систем обучения программированию в связи с потребностью подготовки специалистов, владеющих процедурными, объектно-ориентированными, логическими и функциональными подходами к разработке алгоритмов и программированию. Подробные подходы принято называть парадигмами программирования. Таким образом, становятся актуальными вопросы изучения существующих подходов к организации обучения.

Если говорить более точно, для подготовки студентов колледжа в области информатики и вычислительной техники необходима система курсов, основанная на интеграции парадигм программирования, которая строится в соответствии с понятием информатики как научной дисциплины.

Сопоставляя определения предмета информатики и понятие программирования, можно сделать вывод, что программирование занимает одну из важнейших частей информатики. Поэтому при подготовке специалиста в этой области программированию должна быть отведена адекватная часть его доли, занимаемой в информатике как науке. В программировании концентрируются инженерные вопросы реализации алгоритма при заданных пространственно-времен-ных ограничениях, средствами конкретного языка программирования с учетом всего жизненного цикла программного продукта.

Изучение курса предполагает получение фундаментальных знаний в области информатики. Введение нескольких языков, а, тем более, парадигм программирования позволяет адаптировать полученные знания к быстро меняющейся обстановке в сфере новых

информационных технологий, что, в свою очередь, позволяет на новом качественном уровне использовать информационные технологии в учебном процессе, предоставляет возможность реализовать требуемую модель подготовки студентов.

Содержание информационной подготовки студентов отражается в двух ее структурных составляющих: компоненте образования и компоненте обучения. Причем компонента образования предназначена для формирования общих знаний об основных принципах информатики и обобщенных способах построения, функционирования и использования информационных технологий. Компонента образования составляет теоретическую часть содержания информационной подготовки. Компонента обучения должна формировать умения и навыки работы в конкретных условиях применения современных информационных технологий. Такая компонента содержит практическую часть информационной подготовки.

Курс программирования на основе изучения определенной методологии разработки алгоритмов отвечает, с одной стороны, требованиям, заложенным как в компоненте образования, так и в компоненте обучения. С другой стороны, он призван дать необходимые знания о языке программирования, который лежит в основе построения информационных технологий на современном этапе развития информатики.

Отбор содержания системы курсов программирования, основанных на интеграции парадигм программирования, должен осуществляться согласно специальных методических принципов, основные из которых перечисляются далее.

1. *Научная строгость и последовательность курса*, которая предполагает непротиворечивость и логическую последовательность изложения материала. Для практической реализации данного критерия отбора содержания определены критерии научной строгости и последовательности учебного материала: каждая тема должна быть изложена логически непротиворечиво, реализация каждой темы должна отвечать оценке научного уровня и характеристикам логической строгости.

2. *Системность научных знаний*. Основные положения этого критерия сводятся к тому, что каждое основное понятие должно иметь четко определенное место в системе понятий всего раздела, изложение основных идей и понятий должно быть произведено с использованием достаточного набора соответствующих факторов, методы, используемые

в системе курсов программирования, должны обеспечивать рациональное решение практических задач.

3. *Принцип доступности* обеспечивается постепенностью перехода от простого к сложному, посильностью и целесообразностью терминологии и символики, соответствием имеющемуся запасу знаний, умений и навыков.

4. *Принцип практической направленности теоретического материала*. Данный принцип заключается в том, что должна существовать четкая связь теоретического материала с практикой, причем не только в качестве его использования при решении учебных задач, но и с практикой, как видом человеческой деятельности. Кроме того, содержание должно обеспечивать приобретение у обучаемых практических навыков использования полученных знаний в области программирования и алгоритмизации.

5. *Принцип соответствия целям обучения*, опирающийся на то, что каждое понятие или метод, входящие в содержание обучения информатике, должны соответствовать определенным целям, которых необходимо достичь в процессе обучения, а также быть ориентированными на приобщение обучаемых к программированию с использованием всех возможных парадигм.

6. *Изучение материала в единстве теории, технологии и техники,* что подразумевает использование взаимосвязи между различными аспектами информатики (теоретическим, технологическим и техническим), использование триады «модель–алгоритм–программа», которая лежит в основе применения методологии информатики в различных сферах человеческой деятельности.

Отметим, что при определении содержания обучения информационным технологиям необходимо учитывать сложную структуру соотношений между знаниями умениями и навыками, которые в учебной деятельности студента выступают в диалектическом единстве и характеризуют процесс формирования понятий. Содержание любого учебного предмета – это всегда определенная информация о явлениях или методах деятельности, характерных для данной области.

Существенных исследований требует методика обучения информатике в случае базирования соответствующей методической системы на интеграции различных парадигм программирования. При этом под методом обучения в вузе понимаются упорядоченные способы взаимосвязанной деятельности преподавателя и студента, направленные на достижение поставленных целей обучения конкретной научной дисциплине.

Классификации методов обучения отличаются друг от друга критерием, положенным в основу каждой из них. Рассмотрим основные классификации методов обучения с точки, зрения применения этих методов при обучении курсу программирования.

По способу передачи информации от преподавателя к студенту различают вербальные, наглядные и практические методы обучения. При обучении курсам и разделам программирования используются вербальные (при изложении лекционного материала) и практические (выполнение лабораторных работ, практикумов, решение задач) методы, причем основной акцент делается на практические методы, в процессе применения которых студенты не только получают новые знания, но и приобретают практические навыки. Преподаватель при этом инструктирует, указывает цели работы, направляет и проверяет ход ее исполнения. В деятельности студентов преобладает практическая работа (вещественные и умственные действия), в ходе которой особую роль играет самостоятельный мыслительный процесс, позволяющий осуществить поиск данных и парадигмы решения задачи.

По основным видам дидактических проблем, решаемых на занятии, можно выделить методы приобретения знаний, формирования умений, применения знаний, методы творческой деятельности и методы проверки знаний, умений и навыков.

Отметим, что все перечисленные методы приемлемы для использования при обучении системе курсов программирования на базе интеграции вышеупомянутых парадигм.

Следует отметить, что очень часто методика учебной деятельности представляет собой итерационный поступательный процесс. Такие выводы позволяют предложить метод, применение которого целесообразно при обучении описываемой системе курсов программирования. Речь идет об итерационном методе обучения.

Рассматривая итерацию как пошаговое приближение к определенной цели, можно применить метод итерации как при изложении лекционного материала, так и в процессе выполнения лабораторных работ по информатике. Тем более, что специфика заданий, предназначенных для выполнения на лабораторном практикуме, вполне соответствует поступательному итерационному процессу, который выражается в построении ряда алгоритмов и программ решения задачи, причем каждый следующий алгоритм является уточнением или расширением предыдущего. Таким образом, построение итоговой программы с применением одной из парадигм программирования представляет собой итерационный процесс, на каждом шаге которого

происходят некоторые изменения, что и позволяет нам применить итерационный метод обучения.

Согласно отмеченным положениям, последовательность изложения лекционного материала зависит от порядка практических и лабораторных работ. Учитывая их итерационный характер, изложение лекционного курса также имеет смысл строить на основе итерационного метода.

В случае практического применения подобной методики обучение реализуется не на основе постепенного изучения новых структур и операторов одной из возможных парадигм программирования, а с помощью поступательного итерационного процесса уточнения и расширения возможностей программной реализации моделируемой системы. Причем введение новых структур данных и возможностей языка программирования обосновывается с точки зрения их необходимости для решения новой задачи.

Для профориентации и обучения школьников в рамках дополнительного образования, а так же обучения студентов первой профессии с учетом стандартов компетенции «программные решения для бизнеса» предлагаем использовать модульных подход следующей программы:

Модуль 2

1	Общие сведения о C# Обзор среды разработки VisualStudio. Алгоритм создания программы на языке C#. Консольные приложения. Оконные приложения. Web приложения.
2	Основы программирования в C# Понятие переменных. Понятие типов данных. ООП. Классы. Объекты. Примеры создания переменных. Целые типы. Вещественные типы. Десятичный, логический, символьный, строковый типы. Значения по умолчанию. Применение типов float, double, decimal. Тип char в 16-ричном формате и формате unicode. Типы данных, допускающие значения NULL
3	Переменные и типы данных Переменные. Ключевые слова. Константы. Преобразование значений типов (кастинг). Арифметические операторы. Области видимости. Использование ключевых слов, как идентификаторов. Проверка и запрет проверки переполнения. Сцепление строк. Форматированный вывод. Флаги форматирования строк. Неявно типизированные переменные. Сравнение значений разных типов.
4	Условные конструкции

	Понятие и виды условных конструкций. Тернарный (третичный) оператор. Условная конструкция switch – case.
5	Логические операции Понятие логических операций. Конъюнкция. Дизъюнкция. Исключающее или. Отрицание. Битовые логические операции. Побитовое «И». Побитовое «ИЛИ». Побитовое «Исключающее ИЛИ». Побитовое отрицание. Двоичная арифметика. Примеры использования логических операций.
6	Циклические конструкции Схема работы циклов. Цикл с предусловием (while). Использование циклической конструкции while. Цикл с постусловием (do-while). Примеры использования цикла do-while. Цикл со счетчиком (for). Использование циклической конструкции for. Вложенный цикл for. Бесконечные циклы. Операторы прерывания цикла: continue, break, return
7	Методы Понятие метода. Создание методов. Функции и процедуры. Пример возврата из метода. Методы с изменяемыми параметрами. Методы с выходными параметрами. Перегрузка методов. Аргументы(параметры). Использование именованных параметров. Рекурсия.
8	Массивы Понятие массива. Индекс массива. Использование одномерных массивов. Создание одномерных массивов. Двумерные массивы. Использование двумерных массивов. Массивы из 1 элемента. Трехмерные массивы. Коллекции и цикл foreach

Модуль 3.

1	Введение в WPF Особенности WPF, новшества технологии WPF. Независимое разрешение в WPF. Структура WPF приложения, Page и Frame. XAML.
2	Компоновка WPF Понятие и правила компоновки WPF. Grid, StackPanel, WrapPanelи примеры их использования. Ознакомление с Margin и Padding. Canvas, Z-index и примеры их использования. Свойства компоновки элементов.
3	Элементы управления Обзор элементов управления и их свойств. Элементы управления содержимым. Кнопки. Класс Textblock, TextBox, CheckBox.

	Класс Radio Button, ToolTip, PopUp. ListView, Hyperlink, UserControl WPF. SelectionChanged. Прокрутка (ScrollViewer). CheckBoxList (событие SelectionChanged). RadioButtonList (RadioButon, GroupBox). Transparent (свойство Opacity, прозрачность). Класс ToolTip (всплывающая подсказка). Класс Popup. DragAndDrop (перетаскивание контролов мышью). Создание вкладок и TabControl. Меню. ToolBar, TreeView, DataGrid, Progress Bar и Slider. Работа с датами: Calendar и DatePicker. Работа с изображениями: Image и InkCanvas
4	Стили в WPF Создание стиля. Настройка дизайна с помощью ресурсов. Наследование стилей. Свойства стилей. Задание фона кнопки с помощью стиля.
5	Содержимое Свойство Content. Иерархия элементов управления содержимым. Элементы управления содержимым. Пример использования свойства Content. Динамический контент. Различный контент в элементах управления. Пример использования класса ScrollViewer. Пример использования класса GroupBox. Пример использования класса TabControl. Помещение картинки в заголовок вкладки.
6	Понятие событий в WPF События. Пример работы с событиями. Пример работы со свойствами зависимостей. Создание свойств зависимостей.
7	Работа с ресурсами в WPF Типы ресурсов. Статические и динамические ресурсы. Системные ресурсы. Пример работы с ресурсами сборки. Пример работы с ресурсами объекта. Ресурсы приложения. Локализация приложения WPF.
8	Триггеры в WPF Триггеры. Пример работы с триггерами, EventTrigger.
9	Application Класс Application, его события. Жизненный цикл приложения. Метод Main. Отслеживание окон в приложении. Многопоточность (Dispatcher, BackgroundWorker). Ресурсы приложения.
10	Взаимодействие с базой данных Создание базы данных. Подключение БД. Работа с Entity Framework. Привязка данных (Binding). Лямбда-выражения. Язык интегрированных запросов LINQ

1	Понятие окна в WPF
1	Класс Window. Основные виды и типы окон. События окна. Создание модальных и немодальных окон. Обработка закрытия окна. Позиционирование окна. Взаимодействие между окнами. Использование главных и дочерних окон. Использование OpenFileDialog и SaveFileDialog. Создание окна определенной формы.

Модуль 4.

1	Качество ПО Тестирование и отладка ПО. Виды тестирования. Функциональное тестирование. Интеграционное тестирование. Оптимальное тестовое покрытие. Тестовая документация.
	Презентация ПП Правила подготовки презентаций. Интерактивные презентации.
	Стандартизация разработки ПО. Оформление программного кода.

Одной из самых сложных тем в подготовке студент или специалистов является тема «Проектирование интерфейсов программного обеспечения». Тема сложна в силу малого количества рускоязычной литературы и в силу несформированности понятийного аппаратна научной отрасли проектирования интерфейсов. Предлагаем использовать элементы рабочей программы:

Обучающийся по изучению темы

знает:

- подходы и технологии разработки графического интерфейса;
- пользователя промышленных информационных систем;

умеет:

- разрабатывать графический интерфейса пользователя;
- промышленных информационных систем;

владеет:

- современными программными средствами и технологиями разработки графического интерфейса пользователя промышленных информационных систем.

Раздел 1. Основные концепции проектирования интерфейса пользователя

Введение. Предмет и задачи дисциплины. Основные понятия, термины и определения

Тема 1. Процесс проектирования программных систем

Тема 2. Сбор требований к интерфейсу пользователя

Тема 3. Прототипирование интерфейса пользователя

Тема 4. Персонажи и сценарии. Юзабилити. Уровни дизайна

Тема 5. User Centered Design. Ментальные модели. Метод персонажей

Тема 6. Сценарии пользователей

Тема 7. Проектирование человеко-компьютерного интерфейса

Раздел 2. Элементы интерфейса

Тема 8. Общие правила организации элементов интерфейса

Тема 9. Законы композиции при проектировании пользовательского интерфейса

Тема 10. Использование законов цвета при проектировании интерфейса

Тема 11. Шрифты. Основы типографики. Компьютерные шрифты. Использование шрифтов

Тема 12. Базовые элементы интерфейса пользователя

Раздел 3. Типовые интерфейсные решения

Тема 13. Общая организация экранного пространства

Тема 14. Типовые решения пользовательского интерфейса

Тема 15. Особенности реализации типовых интерфейсов с помощью WEB- технологий

Темы лабораторных работ:

1. Выполнение предварительного и высокоуровневого проектирования при разработке пользовательского интерфейса
2. Разработка требований к графическому интерфейсу пользователя информационной системы.
3. Инструменты быстрого прототипирования графического интерфейса пользователя.
4. Разработка главного меню в среде разработки приложения с анализом и обоснованием его различных состояний.
5. Разработка интерфейса пользователя для различных ролей информационной системы. Создание ключевых сценариев.
6. Определение функциональных групп и иерархических связей между ними.
7. Usability тестирование тестовой версии ПИ по набору раннее определенных показателей.
8. Подготовка пользовательской документации и разработка программы обучения.

9. Разработка интерфейса пользователя информационной web-системы.

В рамках реализации ФГОС 09.02.07 к разработке программных продуктов студентами колледжей предъявляются новые требования – целостность программного продукта, готовность к использованию пользователем. Приведем пример задания для учебной практики по одному из междисциплинарных курсов на примере комплексного задания:

Адаптированная образовательная программа СПО - программа подготовки квалифицированных рабочих, служащих (ППКРС) или программа подготовки специалистов среднего звена (ППССУ), адаптированная для обучения инвалидов и лиц с ограниченными возможностями здоровья с учетом особенностей их психофизического развития, индивидуальных возможностей и при необходимости обеспечивающая коррекцию нарушений развития и социальную адаптацию указанных лиц.

Формы адаптации образовательных программ в РФ

θ выбор мест прохождения практики с учетом требований их доступности – 65,9% образовательных организаций, обучающих инвалидов;

θ проведение текущего контроля успеваемости, промежуточной и государственной итоговой аттестации обучающихся с учетом ограничений их здоровья – 59,4%;

θ использование методов обучения, исходя из их доступности для обучающихся инвалидов и обучающихся с ОВЗ – 56,6%;

θ использование практико-ориентированного (дуального) обучения – 26,2%;

θ разработка индивидуальных учебных планов и индивидуальных графиков обучающихся инвалидов и обучающихся с ОВЗ– 23,4%;

θ обеспечение обучающихся инвалидов и лиц с ОВЗ печатными и электронными образовательными ресурсами в формах, адаптированных к ограничениям их здоровья – 18,9%;

θ включение в вариативную часть образовательных программ среднего профессионального образования адаптационных дисциплин – 5,6%

Методическая основа разработки адаптированной ОП

ϖ Требования к организации образовательного процесса для обучения инвалидов и лиц с ограниченными возможностями здоровья в профессиональных образовательных организациях, в том числе

оснащенности образовательного процесса (письмо Департамента подготовки рабочих кадров и ДПО Министерства образования и науки Российской Федерации 18 марта 2014 г. № 06-281).

ϖ «О направлении доработанных рекомендаций по организации получения среднего общего образования в пределах освоения образовательных программ среднего профессионального образования на базе основного общего образования с учетом требований ФГОС и получаемой профессии или специальности СПО» (Письмо Минобрнауки РФ от 17 марта 2015 г. №06-259).

ϖ Методические рекомендации по разработке и реализации адаптированных образовательных программ СПО (утв. директором Департамента подготовки рабочих кадров и ДПО от 20.04.2015 № 06-830вн) в целях обеспечения права инвалидов и лиц с ОВЗ на получение СПО, а также реализации специальных условий для обучения данной категории обучающихся.

Выбор варианта реализации адаптированной образовательной программы СПО

Вариант	Особенности
Обучение в инклюзивной группе, предполагает изучение то же самого набора дисциплин и в те же сроки обучения, что и остальные обучающиеся.	**Адаптированная ОП направлена на создание специальных условий для реализации его особых образовательных потребностей**
Обучение в отдельной группе в те же сроки обучения, что и остальные обучающиеся, или увеличенные сроки обучения.	**В адаптированную ОП вводятся адаптационные дисциплины, а также обеспечиваются специальные условия для реализации их особых образовательных потребностей**
Обучение по индивидуальному учебному плану, в том числе с использованием дистанционных образовательных технологий.	**Возможно освоение ОП в увеличенные сроки обучения и введение в адаптированную ОП адаптационных дисциплин, предусматриваются специальные условия для реализации особых образовательных потребностей обучающихся**

Требования, направленные на обеспечение инклюзии в ФГОС СПО и других нормативных документах

θ Возможность увеличения срока получения образования инвалидами и лицами с ОВЗ, но не более чем на 6 месяцев по профессиям СПО, на 10 месяцев по специальностям СПО.

θ При обучении лиц с ограниченными возможностями здоровья электронное обучение и дистанционные образовательные технологии должны предусматривать возможность приема - передачи информации в доступных для них формах.

θ Для обучающихся инвалидов и лиц с ограниченными возможностями здоровья образовательная организация устанавливает особый порядок освоения дисциплины «Физическая культура» с учетом состояния их здоровья.

θ При формировании образовательной программы образовательная организация должна предусматривать включение адаптационных дисциплин, обеспечивающих коррекцию нарушений развития и социальную адаптацию обучающихся инвалидов и лиц с ОВЗ.

θ Для обучающихся инвалидов и лиц с ОВЗ выбор мест практик должен учитывать состояние здоровья и требования по доступности.

θ Обучающиеся из числа лиц с ОВЗ должны быть обеспечены печатными и (или) электронными образовательными ресурсами в формах, адаптированных к ограничениям их здоровья.

Инклюзивное (франц. inclusif – включающий в себя, от лат. include – заключаю, включаю) или **включенное** образование– термин, используемый для описания процесса обучения детей с особыми потребностями в общеобразовательных (массовых) школах.

В основу инклюзивного образования положена идеология, которая исключает любую дискриминацию детей, которая обеспечивает равное отношение ко всем людям, но создает особые условия для детей, имеющих особые образовательные потребности.

Инклюзивное образование – процесс развития общего образования, который подразумевает доступность образования для всех, в плане приспособления к различным нуждам всех детей, что обеспечивает доступ к образованию для детей с особыми потребностями.

Для **ИКТ в инклюзивном образовании** отведены **три главные роли**:

- **компенсаторная** — техническая помощь для облегчения традиционных для образования видов деятельности: чтения и письма;
- **дидактическая** — процесс использования ИКТ в целом и изменение в связи с этим подходов к обучению. Существует много возможностей использования ИКТ в качестве дидактического инструмента для создания подходящей учебной среды;

- **коммуникационная** — для коммуникационных технологий — часто относящаяся к использованию систем поддерживающей альтернативной коммуникации.

Основными типами средств ИКТ, используемых для обучения инвалидов и способных выполнять указанные функции, являются следующие:

- **стандартные технологии** — например, компьютеры, имеющие встроенные функции настройки для лиц с ОВЗ;
- **доступные форматы данных**, известные также как альтернативные форматы — например, доступный HTML, говорящие книги системы DAISY (Digital Accessibility Information System — электронная доступная информационная система); а также «низкотехнологичные» форматы, такие как система Брайля;
- **вспомогательные технологии:** слуховые аппараты, устройства для чтения с экрана, клавиатуры со специальными возможностями, и т.д. Вспомогательные технологии (ВТ) — это «устройства, продукты, оборудование, программное обеспечение или услуги, направленные на усиление, поддержку или улучшение функциональных возможностей людей с ограниченными возможностями здоровья».

Речевые тренажеры Go Talk («Гоу Ток»)

Устройство предназначено для усвоения, развития или восстановления речевых навыков с помощью педагога и самостоятельно. Оно выполняет функции речевого тренажера и средства для элементарной речевой коммуникации. Основные функции устройства состоят в наличии диктофона, с помощью которого можно записать или воспроизвести заранее записанные на диктофон звуки, слоги, слова, предложения. Усвоенное можно удалить или дополнить новыми звуками, словами.

Устройство может использоваться для обучения как нормально развивающихся детей, так и детей с нарушением интеллекта и речи (афазия, алалия, дислалия, дизартрия), для реабилитации и облегчения коммуникации и общения взрослых после перенесенного заболевания (черепно-мозговой травмы, инсульта), а также как элементарное коммуникативное устройство для лиц с нарушенной речью.

Duxbury BrailleTranslator **(DBT)** – это программа, которая осуществляет двунаправленный перевод: обыкновенный шрифт переводится в азбуку Брайля и обратно. Также DBT — это полнофункциональный текстовый редактор, при помощи которого

можно подготовить любой документ к печати по брайлю на нескольких десятках языков, в самых разнообразных кодировках.

Особенности Duxbury BrailleTranslator:

- Позволяет импортировать файлы в формате MS Word, WordPerfect, HTML.
- Также текст можно создавать непосредственно в редакторе DBT.
- Ввод текста как обычным способом, так и азбукой Брайля. Во втором случае клавиши основного ряда клавиатуры работают как клавиши брайлевской печатной машинки.
- Программа содержит большое количество «ключей форматирования» — встроенных команд, позволяющих задать необходимый формат документов.
- Комбинации ключей форматирования позволяют создавать «стили», облегчающие работу с текстом.
- В комплект поставки входят основные стили, но пользователь имеет возможность создания новых.
- Совокупность стилей, ключей форматирования и текста можно сохранить в качестве шаблона и использовать в дальнейшем для создания других документов.
- Включает в себя орфографический словарь на 300000 слов.
- Функция «Quick Find Misspelling» позволяет быстро обнаружить орфографические ошибки и устранить их.

Программа DBT поддерживает практически все существующие модели брайлевских принтеров.

К категории ВТ относятся **индивидуальные средства** — например, устройства для облегчения передвижения (инвалидные кресла), **системы поддерживающей альтернативной коммуникации**, а также **оборудование и программное обеспечение** (ПО), облегчающее доступ к компьютеру (например, специальная клавиатура, устройство для чтения с экрана). Высокотехнологичные ВТ, возникшие за два последних десятилетия, в корне изменили доступность образования.

Другие средства ИКТ для обучения включают в себя **обучающее ПО и Виртуальные обучающие среды**. Эти ИКТ могут применяться всеми учениками. В связи с этим крайне важно, чтобы образовательные структуры обеспечивали **универсальный дизайн используемых технологий** и их соответствие требованиям Конвенции ООН «О правах инвалидов».

Мета-исследование эффективности использования доступных ИКТ в образовании выявило, что для всех заинтересованных категорий пользователей существует **целый ряд преимуществ** — от облегчения участия в учебном процессе и общении в общеобразовательных классах до достижения автономности в обучении и возможности создавать индивидуальные задания с учетом возможностей и способностей каждого конкретного ученика.

МОДУЛЬ КОМПЕТЕНЦИИ 1: «ПРОЕКТИРОВАНИЕ»

Инструментальные средства для анализа и проектирования программных решений.

Для того, чтобы разработать программную систему, приносящую реальные выгоды определенным пользователям, необходимо сначала выяснить, какие же задачи она должна решать для этих людей и какими свойствами обладать.

Требования к ПО определяют, какие свойства и характеристики оно должно иметь для удовлетворения потребностей пользователей и других заинтересованных лиц. Однако сформулировать требования к сложной системе не так легко. В большинстве случаев будущие пользователи могут перечислить набор свойств, который они хотели бы видеть, но никто не даст гарантий, что это — исчерпывающий список. Кроме того, часто сама формулировка этих свойств будет непонятна большинству программистов: могут прозвучать фразы типа «должно использоваться и частотное, и временное уплотнение каналов», «передача клиента должна быть мягкой», «для обычных швов отмечайте бригаду, а для доверительных — конкретных сварщиков», и это еще не самые тяжелые для понимания примеры.

Чтобы ПО было действительно полезным, важно, чтобы оно удовлетворяло реальные потребности людей и организаций, которые часто отличаются от непосредственно выражаемых пользователями желаний. Для выявления этих потребностей, а также для выяснения смысла высказанных требований приходится проводить достаточно большую дополнительную работу, которая называется анализом предметной области или бизнес-моделированием, если речь идет о потребностях коммерческой организации. В результате этой деятельности разработчики должны научиться понимать язык, на котором говорят пользователи и заказчики, выявить цели их деятельности, определить набор задач, решаемых ими. В дополнение стоит выяснить, какие вообще задачи нужно уметь решать для достижения этих целей, выяснить свойства результатов, которые хотелось бы получить, а также определить набор сущностей, с которыми приходится иметь дело при решении этих задач. Кроме того, анализ предметной области позволяет выявить места возможных улучшений и оценить последствия принимаемых решений о реализации тех или иных функций.

После этого можно определять область ответственности будущей программной системы — какие именно из выявленных задач будут ею решаться, при решении каких задач она может оказать существенную помощь и чем именно. Определив эти задачи в рамках общей системы задач и деятельностей пользователей, можно уже более точно сформулировать требования к ПО.

Анализом предметной области занимаются системные аналитики или бизнес-аналитики, которые передают полученные ими знания другим членам проектной команды, сформулировав их на более понятном разработчикам языке. Для передачи этих знаний обычно служит некоторый набор моделей, в виде графических схем и текстовых документов.

Часто для описания поведения сложных систем и деятельности крупных организаций используются диаграммы потоков данных (data flow diagrams). Эти диаграммы содержат 4 вида графических элементов: процессы, представляющие собой любые трансформации данных в рамках описываемой системы, хранилища данных, внешние по отношению к системе сущности и потоки данных между элементами трех предыдущих видов.

Процессы на диаграммах потоков данных могут уточняться: если некоторый процесс устроен достаточно сложно, для него можно нарисовать отдельную диаграмму, описывающую потоки данных внутри этого процесса. На ней показываются те элементы, с которыми этот процесс связан потоками данных, и составляющие его более мелкие процессы и хранилища. Таким образом, возникает иерархическая структура процессов. Обычно на самом верхнем уровне находится один процесс, представляющий собой систему в целом, и набор внешних сущностей, с которыми она взаимодействует.

Диаграммы потоков данных появились как один из первых инструментов представления деятельности сложных систем при использовании структурного анализа. Для представления структуры данных в этом подходе используются диаграммы сущностей и связей (entity- relationship diagrams, ER diagrams), изображающие набор сущностей предметной области и связей между ними. И сущности, и связи на таких диаграммах могут иметь атрибуты.

Хотя методы структурного анализа могут значительно помочь при анализе систем и организаций, дальнейшая разработка системы, поддерживающей их деятельность, с использованием объектно-ориентированного подхода часто требует дополнительной работы по

переводу полученной информации в объектно-ориентированные модели.

Методы объектно-ориентированного анализа предназначены для обеспечения более удобной передачи информации между моделями анализируемых систем и моделями разрабатываемого ПО. В качестве графических моделей в этих методах вместо диаграмм потоков данных используются рассматривавшиеся при обсуждении RUP диаграммы вариантов использования, а вместо диаграмм сущностей и связей — диаграммы классов.

Однако диаграммы вариантов использования несут несколько меньше информации по сравнению с соответствующими диаграммами потоков данных: на них процессы и хранилища в соответствии с принципом объединения данных и методов работы с ними объединяются в варианты использования, и остаются только связи между вариантами использования и действующими лицами (аналогом внешних сущностей). Для представления остальной информации каждый вариант использования может дополняться набором разнообразных диаграмм UML — диаграммами деятельностей, диаграммами сценариев, и пр.

Выделение и анализ требований

После получения общего представления о деятельности и целях организаций, в которых будет работать будущая программная система, и о ее предметной области, можно определить более четко, какие именно задачи система будет решать. Кроме того, важно понимать, какие из задач стоят наиболее остро и обязательно должны быть поддержаны уже в первой версии, а какие могут быть отложены до следующих версий или вообще вынесены за рамки области ответственности системы. Эта информация выявляется при анализе потребностей возможных пользователей и заказчиков.

Потребности определяются на основе наиболее актуальных проблем и задач, которые пользователи и заказчики видят перед собой. При этом требуется аккуратное выявление значимых проблем, определение того, насколько хорошо они решаются при текущем положении дел, и расстановка приоритетов при рассмотрении недостаточно хорошо решаемых, поскольку чаще всего решить сразу все проблемы невозможно.

Формулировка потребностей может быть разбита на следующие этапы.

- Выделить одну-две-три основных проблемы.

- Определить причины возникновения проблем, оценить степень их влияния и выделить наиболее существенные из проблем, влекущие появление остальных.
- Определить ограничения на возможные решения.

Формулировка потребностей не должна накладывать лишних ограничений на возможные решения, удовлетворяющие им. Нужно попытаться сформулировать, что именно является проблемой, а не предлагать сразу возможные решения.

При выявлении потребностей пользователей анализируются модели деятельности пользователей и организаций, в которых они работают, для выявления проблемных мест. Также используются такие приемы, как анкетирование, демонстрация возможных сеансов работы будущей системы, интерактивные опросы, где пользователям предоставляется возможность самим предложить варианты внешнего вида системы и ее работы или поменять предложенные кем-то другим, демонстрация прототипа системы и др.

После выделения основных потребностей нужно решить вопрос о разграничении области ответственности будущей системы, т.е. определить, какие из потребностей надо пытаться удовлетворить в ее рамках, а какие — нет.

При этом все заинтересованные лица делятся на пользователей, которые будут непосредственно использовать создаваемую систему для решения своих задач, и вторичных заинтересованных лиц, которые не решают своих задач с ее помощью, но чьи интересы так или иначе затрагиваются ею. Потребности пользователей нужно удовлетворить в первую очередь и на это нужно выделить больше усилий, а интересы вторичных заинтересованных лиц должны быть только адекватно учтены в итоговой системе.

На основе выделенных потребностей пользователей, отнесенных к области ответственности системы, формулируются возможные функции будущей системы, которые представляют собой услуги, предоставляемые системой и удовлетворяющие потребности одной или нескольких групп пользователей (или других заинтересованных лиц). Идеи для определения таких функций можно брать из имеющегося опыта разработчиков (наиболее часто используемый источник) или из результатов мозговых штурмов и других форм выработки идей.

При этом часто нужно учитывать, что ПО является частью программно-аппаратной системы, требования к которой надо преобразовать в требования к программной и аппаратной ее составляющим. В последнее время, в связи со значительным падением

цен на мощное аппаратное обеспечение общего назначения, фокус внимания переместился, в основном, на программное обеспечение. Во многих проектах аппаратная платформа определяется из общих соображений, а поддержку большинства нужных функций осуществляет ПО.

Каждое требование раскрывает детали поведения системы при выполнении ею некоторой функции в некоторых обстоятельствах. При этом часть требований исходит из потребностей и пожеланий заинтересованных лиц и решений, удовлетворяющих эти потребности и пожелания, а часть — из внешних ограничений, накладываемых на систему, например, основными законами той предметной области, в рамках которой системе придется работать, государственным законодательством, корпоративной политикой и пр.

Еще до перехода от функций к требованиям полезно расставить приоритеты и оценить трудоемкость их реализации и рискованность. Это позволит отказаться от реализации наименее важных и наиболее трудоемких, не соответствующих бюджету проекта функций еще до их детальной проработки, а также выявить возможные проблемные места проекта — наиболее трудоемкие и неясные из вошедших в него функций.

Для представления архитектуры, а точнее — различных входящих в нее структур, удобно использовать графические языки. На настоящий момент наиболее проработанным и наиболее широко используемым из них является унифицированный язык моделирования (Unified Modeling Language, UML), хотя достаточно часто архитектуру системы описывают просто набором именованных прямоугольников, соединенных линиями и стрелками, которые представляют возможные связи.

UML предлагает использовать для описания архитектуры 8 видов диаграмм. 9-й вид UML диаграмм, диаграммы вариантов использования, не относится к архитектурным представлениям. Кроме того, и другие виды диаграмм можно использовать для описания внутренней структуры компонентов или сценариев действий пользователей и прочих элементов, к архитектуре часто не относящихся. В этом курсе мы не будем разбирать диаграммы UML в деталях, а ограничимся обзором их основных элементов, необходимым для общего понимания смысла того, что изображено на таких диаграммах.

Диаграммы UML делятся на две группы — статические и динамические диаграммы.

Статические диаграммы представляют либо постоянно присутствующие в системе сущности и связи между ними, либо

суммарную информацию о сущностях и связях, либо сущности и связи, существующие в какой-то определенный момент времени. Они не показывают способов поведения этих сущностей. К этому типу относятся диаграммы классов, объектов, компонентов и диаграммы развертывания.

Диаграммы классов (class diagrams) показывают классы или типы сущностей системы, характеристики классов (поля и операции) и возможные связи между ними. Пример диаграммы классов изображен на Рис. 31.

Классы представляются прямоугольниками, поделенными на три части. В верхней части показывают имя класса, в средней — набор его полей, с именами, типами, модификаторами доступа (public ‘+’, protected ‘#’, private ‘-’) и начальными значениями, в нижней — набор операций класса. Для каждой операции показывается ее модификатор доступа и сигнатура.

На Рис. 1 изображены классы Account, Person, Organization, Address, CreditAccount и абстрактный класс Client.

Класс CreditAccount имеет private поле maximumCredit типа double, а также public, метод getCredit() и protected метод setCredit().

Интерфейсы, т.е. типы, имеющие только набор операций и не определяющие способов их реализации, часто показываются в виде небольших кружков, хотя могут изображаться и как обычные классы. На Рис. 1 представлен интерфейс AccountInterface.

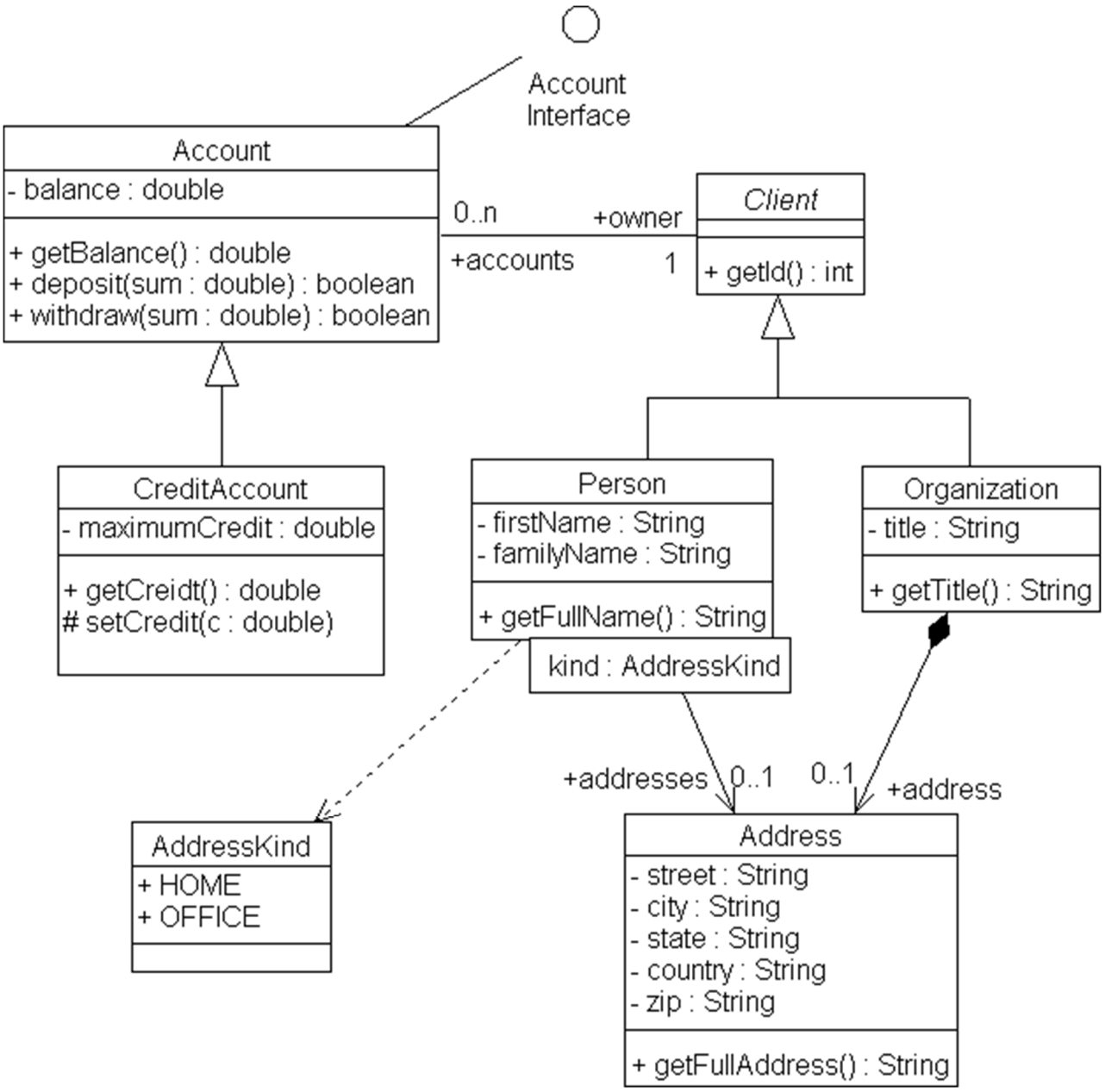


Рисунок 1. Диаграмма классов

Наиболее часто используется три вида связей между классами — связи по композиции, ссылки, связи по наследованию и реализации.

Композиция описывает ситуацию, в которой объекты класса A включают в себя объекты класса B, причем последние не могут разделяться (объект класса B, являющийся частью объекта класса A, не может являться частью другого объекта класса A) и существуют только в рамках объемлющих объектов (уничтожаются при уничтожении объемлющего объекта).

Композицией на Рис. 1 является связь между классами Organization и Address. Ссылочная связь (или слабая агрегация) обозначает, что объект некоторого класса A имеет в качестве поля ссылку на объект

другого (или того же самого) класса B, причем ссылки на один и тот же объект класса B могут иметься в нескольких объектах класса A.

И композиция, и ссылочная связь изображаются стрелками, ведущими от класса A к классу B. Композиция дополнительно имеет закрашенный ромбик у начала этой стрелки. Двусторонние ссылочные связи, обозначающие, что объекты могут иметь ссылки друг на друга, показываются линиями без стрелок. Такая связь показана на Рис. 1 между классами Account и Client.

Эти связи могут иметь описание множественности, показывающее, сколько объектов класса B может быть связано с одним объектом класса A. Оно изображается в виде текстовой метки около конца стрелки, содержащей точное число или нижние и верхние границы, причем бесконечность изображается звездочкой или буквой n. Для двусторонних связей множественности могут показываться с обеих сторон. На Рис. 1 множественности, изображенные для связи между классами Account и Client, обозначают, что один клиент может иметь много счетов, а может и не иметь ни одного, и счет всегда привязан ровно к одному клиенту.

Наследование классов изображается стрелкой с пустым наконечником, ведущей от наследника к предку. На Рис. 1 класс CreditAccount наследует классу Account, а классы Person и Organization — классу Client.

Реализация интерфейсов показывается в виде пунктирной стрелки с пустым наконечником, ведущей от класса к реализуемому им интерфейсу, если тот показан в виде прямоугольника. Если же интерфейс изображен в виде кружка, то связь по реализации показывается обычной сплошной линией (в этом случае неоднозначности в ее толковании не возникает). Такая связь изображена на Рис. 1 между классом Account и интерфейсом AccountInterface.

Один класс использует другой, если этот другой класс является типом параметра или результата операции первого класса. Иногда связи по использованию показываются в виде пунктирных стрелок. Пример такой связи между классом Person и перечислимым типом AddressKind можно видеть на Рис. 1.

Ссылочные связи, реализованные в виде ассоциативных массивов или отображений (map)— такая связь в зависимости от некоторого набора ключей определяет набор ссылок- значений — показываются при помощи стрелок, имеющих прямоугольник с перечислением типов и имен ключей, примыкающий к изображению класса, от которого идет стрелка.

Множественность на конце стрелки при этом обозначает количество ссылок, соответствующее одному набору значений ключей.

На Рис. 1 такая связь ведет от класса Person к классу Address, показывая, что объект класса Person может иметь один адрес для каждого значения ключа kind, т.е. один домашний и один рабочий адреса.

Диаграммы классов используются чаще других видов диаграмм.

Диаграммы объектов (object diagrams) показывают часть объектов системы и связи между ними в некотором конкретном состоянии или суммарно, за некоторый интервал времени. Объекты изображаются прямоугольниками с идентификаторами ролей объектов (в контексте тех состояний, которые изображены на диаграмме) и типами. Однородные коллекции объектов могут изображаться накладывающимися друг на друга прямоугольниками.

Такие диаграммы используются довольно редко.

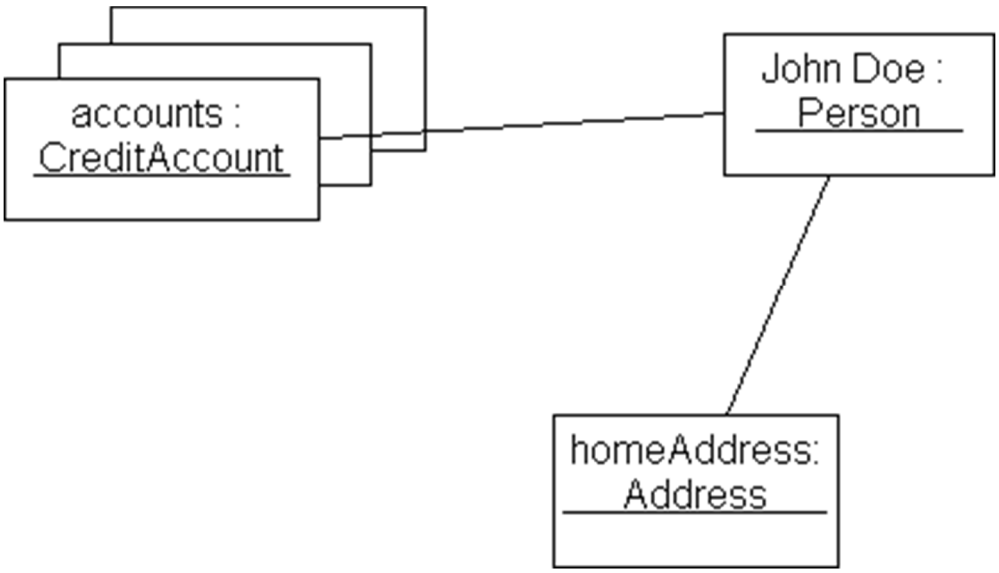


Рисунок 2. Диаграмма объектов.

Диаграммы компонентов (component diagrams) представляют компоненты в нескольких смыслах — атомарные составляющие системы с точки зрения ее сборки, конфигурационного управления и развертывания. Компоненты сборки и конфигурационного управления обычно представляют собой файлы с исходным кодом, динамически подгружаемые библиотеки, HTML-странички и пр., компоненты развертывания — это компоненты JavaBeans, CORBA, COM и т.д.

Компонент изображается в виде прямоугольника с несколькими прямоугольными или другой формы «зубами» на левой стороне.

Связи, показывающие зависимости между компонентами, изображаются пунктирными стрелками. Один компонент зависит от

другого, если он не может быть использован в отсутствии этого другого компонента в конфигурации системы. Компоненты могут также реализовывать интерфейсы.

Диаграммы этого вида используются редко.

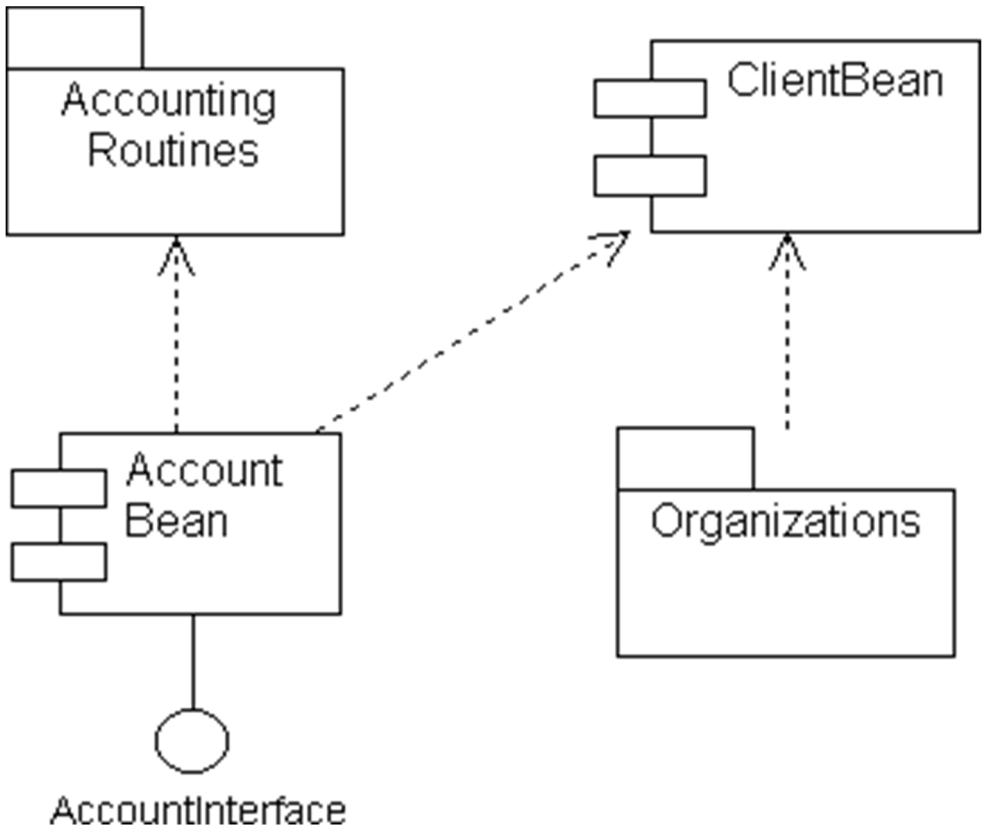


Рисунок 3. Диаграмма компонентов.

На диаграмме компонентов, изображенной на Рис. 3, можно также увидеть пакеты, изображаемые в виде «папок», точнее — прямоугольников с прямоугольными «наростами» над левым верхним углом. Пакеты являются пространствами имен и средством группировки диаграмм и других модельных элементов UML — классов, компонентов и пр. Они могут появляться на диаграммах классов и компонентов для указания зависимостей между ними и отдельными классами и компонентами. Иногда на такой диаграмме могут присутствовать только пакеты с зависимостями между ними.

Диаграммы развертывания (deployment diagrams) показывают декомпозицию системы на физические устройства различных видов — серверы, рабочие станции, терминалы, принтеры, маршрутизаторы и пр. — и связи между ними, представленные различного рода сетевыми и индивидуальными соединениями.

Физические устройства, называемые узлами системы (nodes), изображаются в виде кубов или параллелепипедов, а физические соединения между ними — в виде линий.

На диаграммах развертывания может быть показана привязка (в некоторый момент времени или постоянная) компонентов развертывания системы к физическим устройствам

— например, для указания того, что компонент EJB AccountEJB исполняется на сервере приложений, а аплет AccountInfoEditor — на рабочей станции оператора банка.

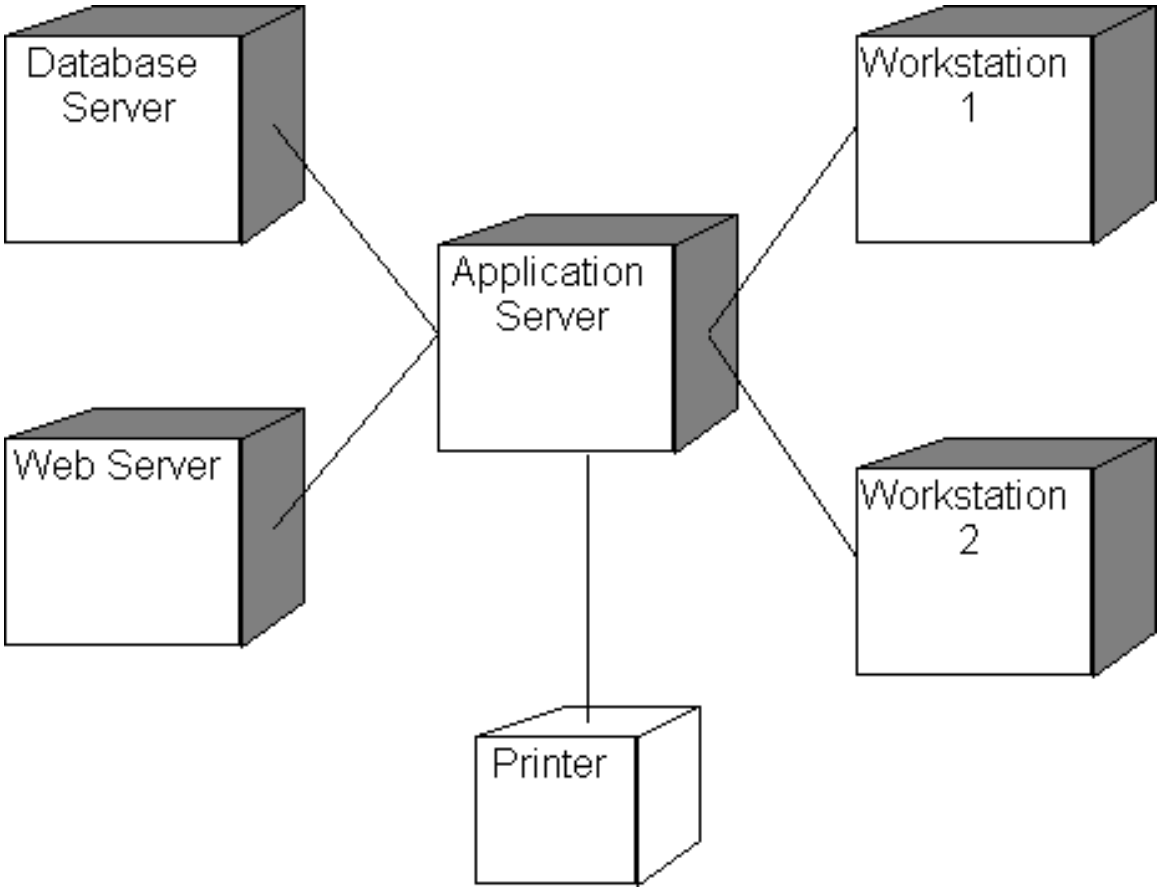


Рисунок 4. Диаграмма развертывания.

Эти диаграммы используются достаточно редко. Пример диаграммы развертывания изображен на Рис. 4.

Динамические диаграммы описывают происходящие в системе процессы. К ним относятся диаграммы деятельности, сценариев, диаграммы взаимодействия и диаграммы состояний.

Диаграммы деятельности (activity diagrams) иллюстрируют набор процессов- деятельностей и потоки данных между ними, а также возможные их синхронизации друг с другом.

Деятельность изображается в виде прямоугольника с закругленными сторонами, слева и справа, помеченного именем деятельности.

Потоки данных показываются в виде стрелок. Синхронизации двух видов — развилки (forks) и слияния (joins) — показываются жирными короткими линиями (кто-то может посчитать их и тонкими закрашенными прямоугольниками), к которым сходятся или от которых расходятся потоки данных. Кроме синхронизаций, на диаграммах деятельности могут быть показаны разветвления потоков данных, связанных с выбором того или иного направления в зависимости от некоторого условия. Такие разветвления показываются в виде небольших ромбов.

Диаграмма может быть поделена на несколько горизонтальных или вертикальных областей, называемых дорожками (swimlanes). Дорожки служат для группировки деятельностей в соответствии с выполняющими их подразделением организации, ролью, приложением, подсистемой и пр.

Диаграммы деятельности могут заменять часто используемые диаграммы потоков данных, поэтому применяются достаточно широко. Пример такой диаграммы показан на Рис. 5.

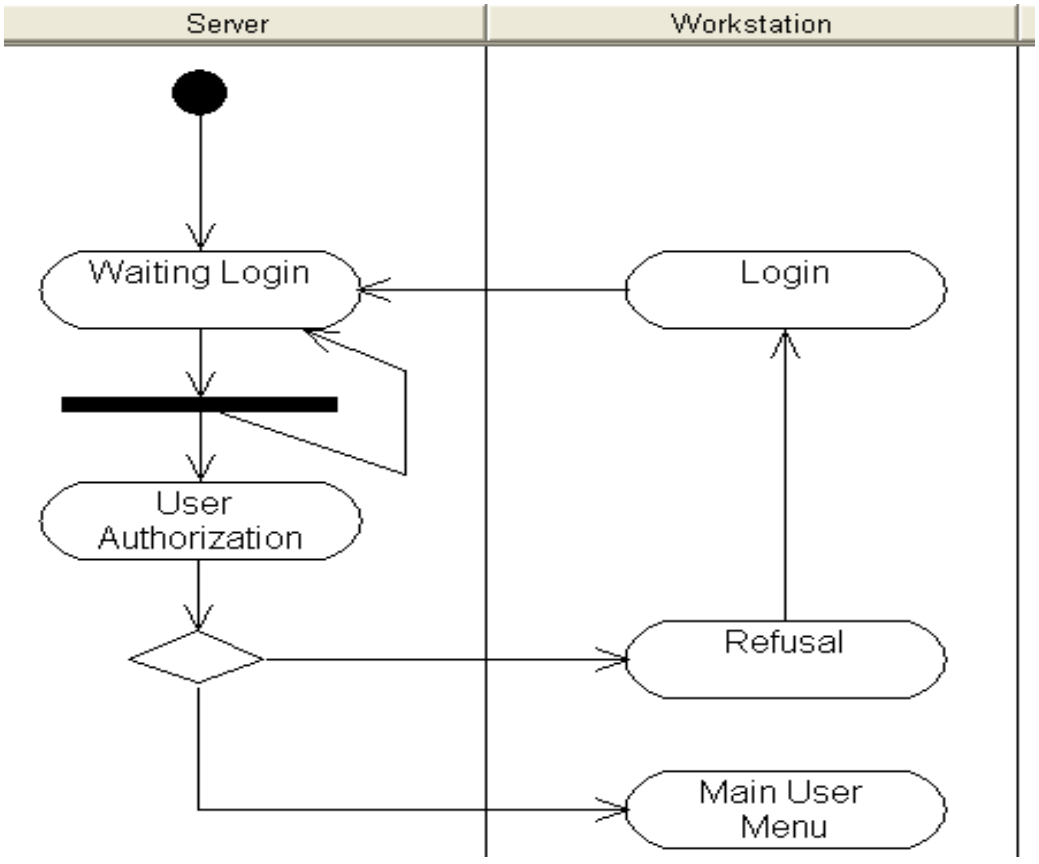


Рисунок 5. Диаграмма деятельности.

Диаграммы сценариев (или диаграммы последовательности, sequence diagrams) показывают возможные сценарии обмена сообщениями или вызовами во времени между различными компонентами системы (здесь имеются в виду архитектурные компоненты, компоненты в широком смысле — это могут быть компоненты развертывания, обычные объекты, подсистемы и пр.). Эти диаграммы являются подмножеством специального графического языка — языка диаграмм последовательностей сообщений (Message Sequence Charts, MSC), который был придуман раньше UML и достаточно долго развивается параллельно ему.

Компоненты, участвующие во взаимодействии, изображаются прямоугольниками вверху диаграммы. От каждого компонента вниз идет вертикальная линия, называемая его линией жизни. Считается, что ось времени направлена вертикально вниз. Интервалы времени, в которые компонент активен, т.е. управление находится в одной из его операций, представлены тонким прямоугольником, для которого линия жизни компонента является осью симметрии.

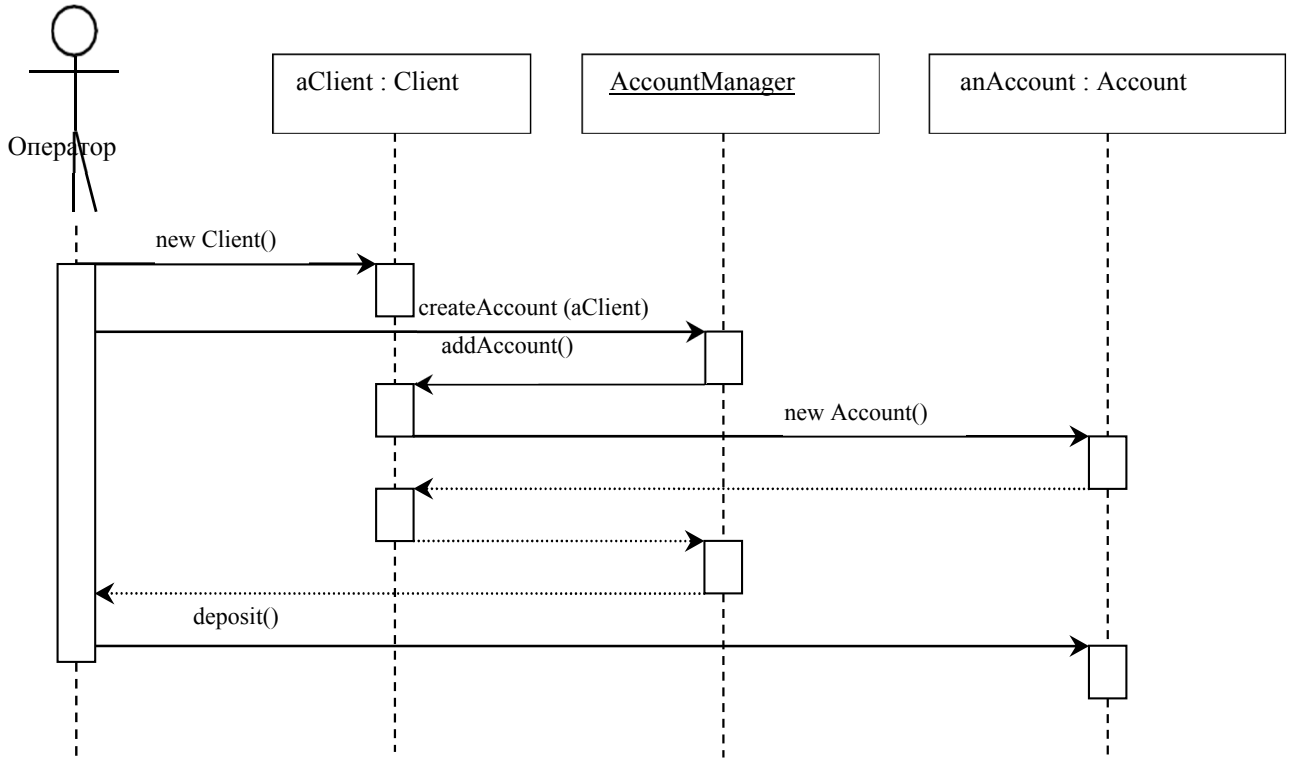


Рисунок 6. Пример диаграммы сценария открытия счета.

Передача сообщения или вызов изображаются стрелкой от компонента-источника к компоненту-приемнику. Возврат управления показан пунктирной стрелкой, обратной к соответствующему вызову.

Эти диаграммы используются достаточно часто, например, при детализации сценариев, входящих в варианты использования. Пример такой диаграммы изображен на Рис. 6.

Диаграммы взаимодействия (collaboration diagrams) показывают ту же информацию, что и диаграммы сценариев, но привязывают обмен сообщениями/вызовами не к времени, а к связями между компонентами.

На диаграмме изображаются компоненты в виде прямоугольников и связи между ними. Вдоль связей могут передаваться сообщения, показываемые в виде небольших стрелок, параллельных связи. Стрелки нумеруются в соответствии с порядком происходящих событий. Нумерация может быть иерархической, чтобы показать вложенность действий друг в друга (т.е. если вызов некоторой операции имеет номер 1, то вызовы, осуществляемые при выполнении этой операции, будут нумероваться как 1.1, 1.2, и т.д.). Диаграммы взаимодействия используются довольно редко.

Диаграммы состояний (statechart diagrams) показывают возможные состояния отдельных компонентов или системы в целом, переходы между ними в ответ на какие-либо события и выполняемые при этом действия.

Состояния показываются в виде прямоугольников с закругленными углами, переходы — в виде стрелок. Начальное состояние представляется как небольшой темный кружок, конечное — как пустой кружок с концентрически вложенным темным кружком. Вы могли обратить внимание на темный кружок на диаграмме деятельности на Рис. 5 — он тоже изображает начальное состояние: дело в том, что диаграммы деятельности являются диаграммами состояний специального рода, а деятельности — частный случай состояний. Пример диаграммы состояний приведен на Рис. 7.

Состояния могут быть устроены иерархически: они могут включать в себя другие состояния, даже целые отдельные диаграммы вложенных состояний и переходов между ними. Пребывая в таком состоянии, система находится ровно в одном из его подсостояний. На Рис. 7 почти все изображенные состояния являются подсостояниями состояния Site.

Кроме того, в нижней части диаграммы три состояния объединены, чтобы показать, что переход по действию cancel возможен в каждом из них и приводит в одно и то же состояние Basket.

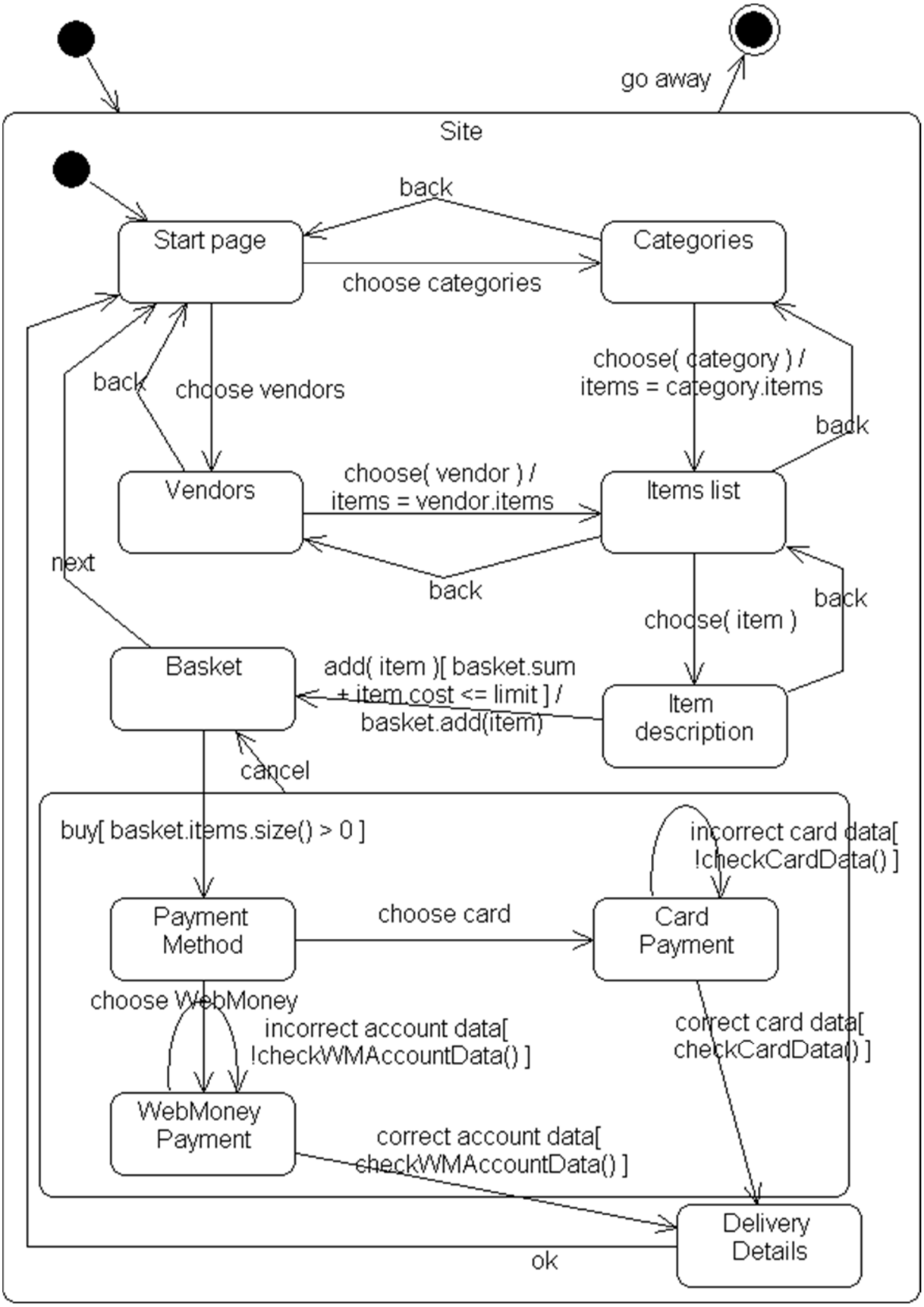


Рисунок 7. Пример диаграммы состояний, моделирующей сайт Интернет-магазина.

Состояние может декомпозироваться и на параллельные подсостояния. Они изображаются как области внутри объемлющего состояния, разделенные пунктирными линиями, их аналогом на диаграммах деятельности являются дорожки. Пребывая в объемлющем состоянии, система должна находиться одновременно в каждом из его параллельных подсостояний.

Помимо показанных на диаграмме состояний изображаемая подсистема может иметь глобальные (в ее рамках) переменные, хранящие какие-то данные. Значения этих переменных являются общими частями всех изображаемых состояний.

На Рис.7 примерами переменных являются список видимых пользователем товаров, items, и набор уже отобранных товаров с количеством для каждого, корзина, basket. Переходы могут происходить между состояниями одного уровня, но могут также вести из некоторого состояния в подсостояние соседнего или, наоборот, из подсостояния в некоторое состояние, находящее на том же уровне, что и объемлющее состояние.

На переходе между состояниями указываются следующие данные:

Событие, приводящее к выполнению этого перехода. Обычно событие — это вызов некоторой операции в одном из объектов или приход некоторого сообщения, хотя могут указываться и абстрактные события.

Например, из состояния Categories на Рис. 7 можно выйти, выполнив команду браузера «Назад». Она соответствует событию back, инициирующему переход в состояние Start page. Другой переход из состояния Categories происходит при выборе категории товаров пользователем. Соответствующее событие имеет параметр — выбранную категорию. Оно изображено как choose(category).

Условие выполнения (охранное условие, guardian). Это условие, зависящее от параметров события и текущих значений глобальных переменных, выполнение которого необходимо для выполнения перехода. При наступлении нужного события переход выполняется, только если его условие тоже выполнено.

Условие перехода показывается в его метке в квадратных скобках.

На Рис. 7 примером условного перехода является переход из состояния Basket в состояние Payment Method. Он выполняется, только если пользователь выполняет команду «Оплатить» (событие buy) и при этом в его корзине есть хотя бы один товар.

Действие, выполняемое в дополнение к переходу между состояниями. Обычно это вызовы каких-то операций и изменения

значения глобальных переменных. Действие показывается в метке перехода после слеша (символа ‘/’). При этом изменения значений переменных перечисляются в начале, затем, после знака ‘^’, указывается вызов операции.

Например, на Рис. 7 при выборе пользователем категории товаров происходит переход из состояния Categories в Items list. При этом список товаров, видимый пользователю, инициализируется списком товаров выбранной категории.

Диаграммы состояний используются часто, хотя требуется довольно много усилий, чтобы разработать их с достаточной степенью детальности.

Рассмотрим проектирование диаграмм в среде PlantUML.

Диаграмма последовательности

Основные примеры

Последовательность -> используется, чтобы отобразить сообщение между двумя участниками (participants). Не обязательно явно объявлять участников.

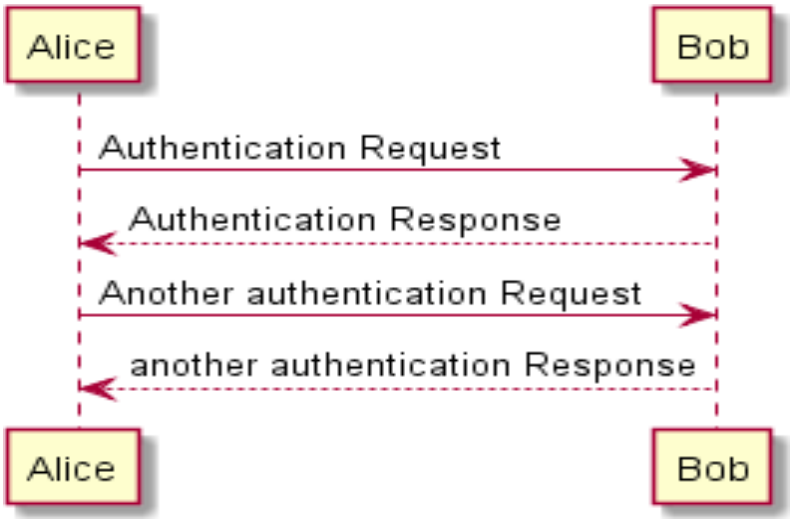
Для получения пунктирной стрелки (dotted arrow), используйте -->.

Также возможно использовать <- и <--. Это не изменит отображение, но может улучшить читабельность. Заметьте, что это верно только для диаграмм последовательности, для других диаграмм правила другие.

@startuml

Alice -> Bob: Authentication Request Bob --> Alice: Authentication Response

Alice -> Bob: Another authentication Request Alice <-- Bob: another authentication Response @enduml



Объявление участников

Возможно изменить порядок участников, используя ключевое слово participant. Так же возможно использование других ключевых слов, для объявления participant'a:

actor

boundary

control

entity

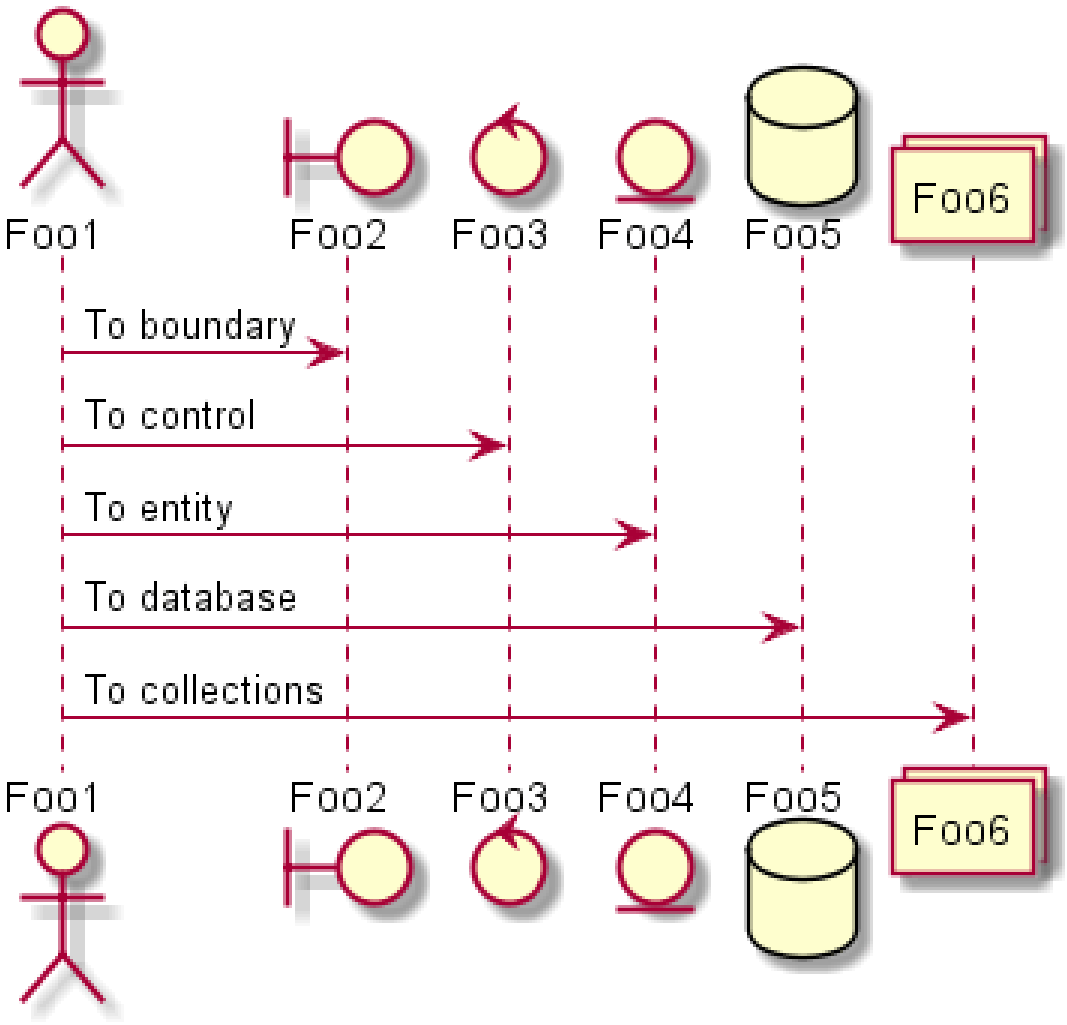
database

@startuml actor Foo1 boundary Foo2 control Foo3 entity Foo4 database Foo5

collections Foo6

Foo1 -> Foo2 : To boundary Foo1 -> Foo3 : To control Foo1 -> Foo4 : To entity Foo1 -> Foo5 : To database Foo1 -> Foo6 : To collections

@enduml



Можно переименовать участника используя ключевое слово as

Также возможно изменить цвет фона actor-a или участника, используя имя цвета или его html-код

@startuml actor Bob #red

' The only difference between actor 'and participant is the drawing participant Alice

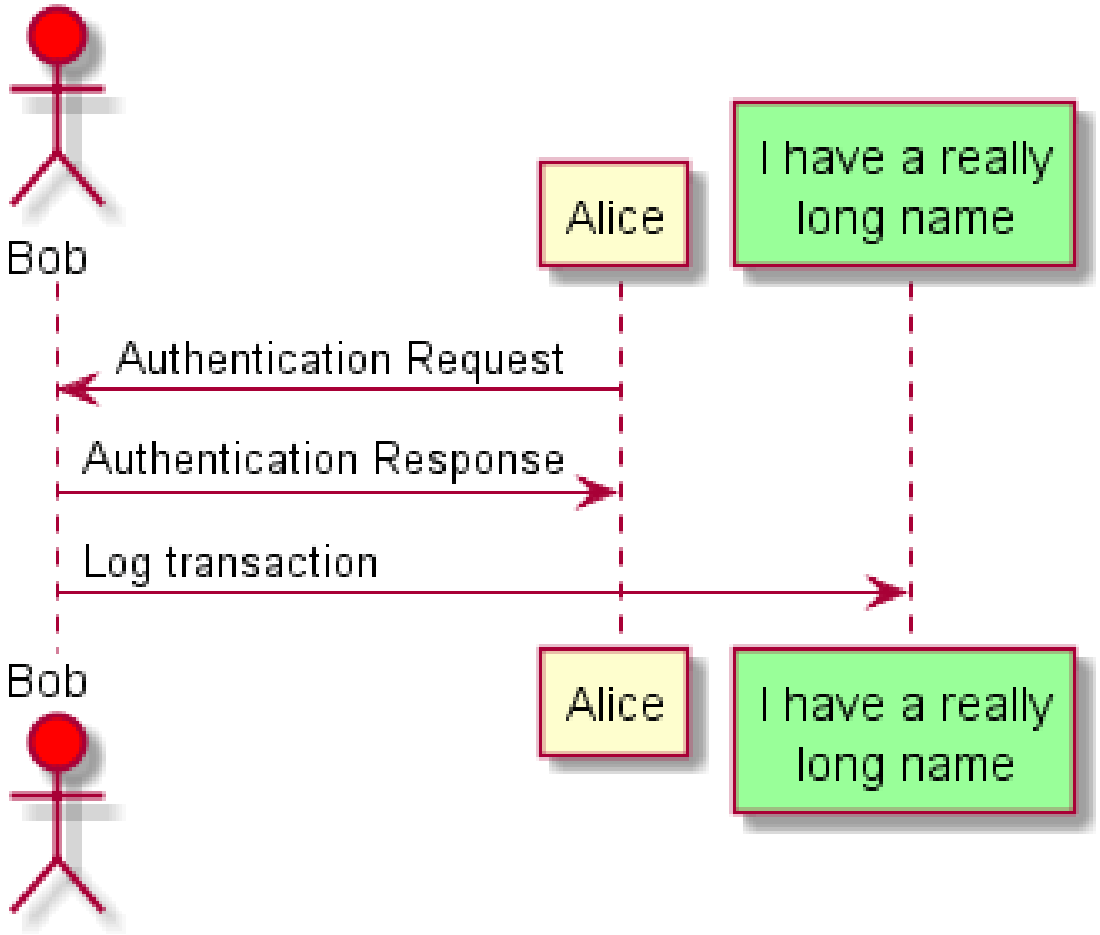
participant "I have a really\nlong name" as L #99FF99

/' You can also declare:

participant L as "I have a really\nlong name" #99FF99 '/

Alice->Bob: Authentication Request Bob->Alice: Authentication Response Bob->L: Log transaction

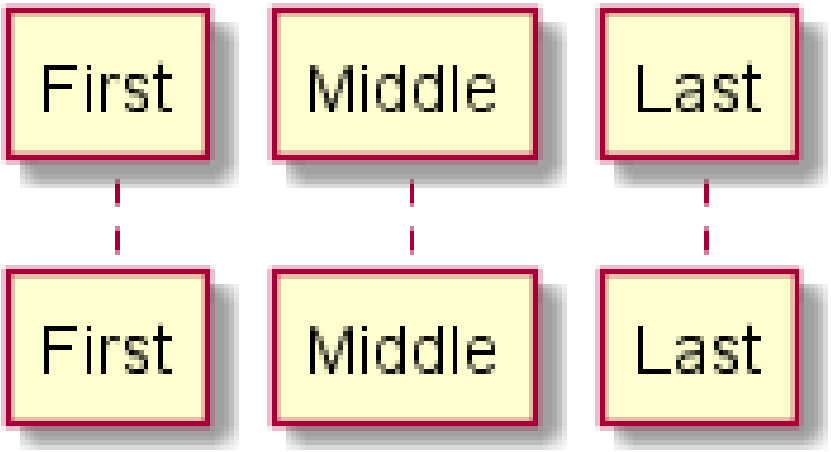
@enduml



Использовав ключевое поле order (порядок) можно настроить порядок печати participant (участника).

@startuml

participant Last order 30 participant Middle order 20 participant First order 10 @enduml

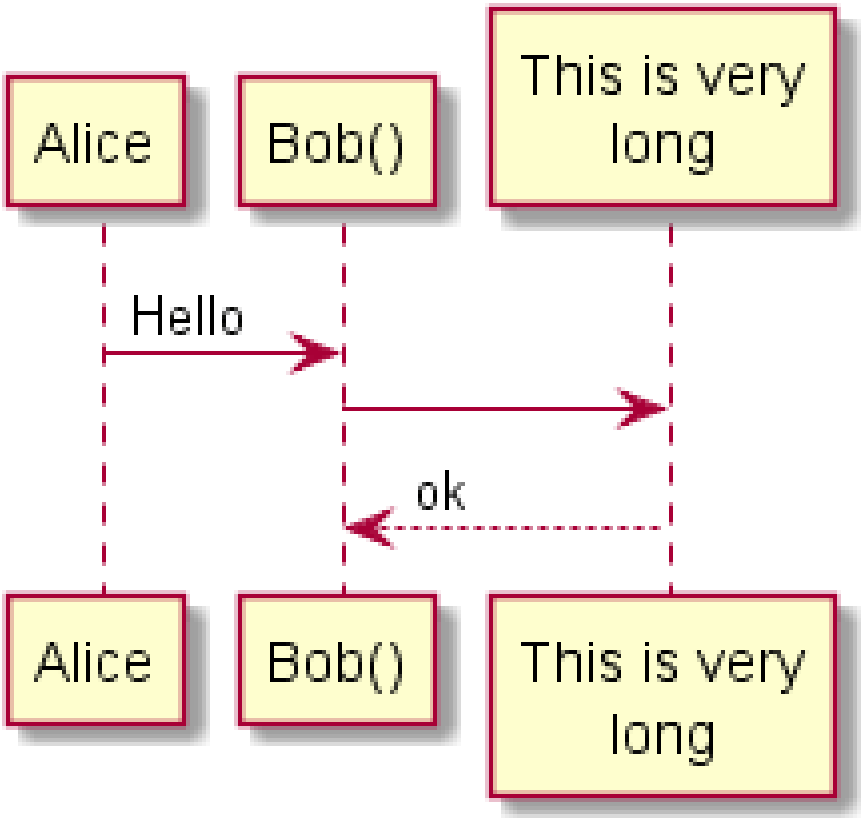


Использование небуквенных символов в названиях участников

Вы можете использовать кавычки для задания участников. Также Вы можете использовать ключевое слово

as для присвоения псевдонимов к этим участникам.

```
@startuml
Alice -> "Bob()" : Hello
"Bob()" -> "This is very\nlong" as Long ' You can also declare:
' "Bob()" -> Long as "This is very\nlong"
Long --> "Bob()" : ok @enduml
```

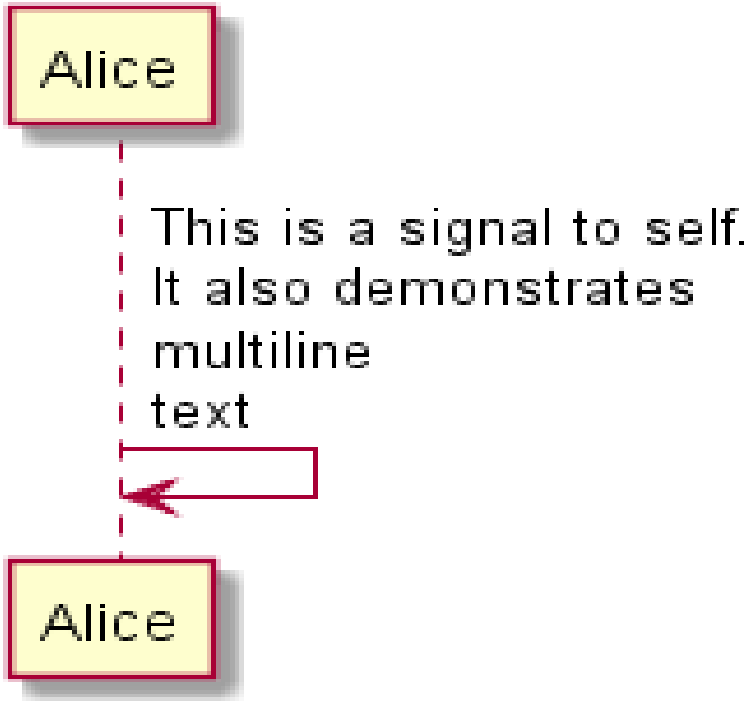


Сообщения к самому себе

Участник может посылать сообщения сам себе.

Также возможно создание многострочных используя \n. @startuml

Alice->Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext @enduml



Изменить стиль стрелок

Вы можете изменить стиль стрелок следующими способами:

закончить стрелку с помощью x для обозначения потерянного соотбщения

используя \ или / вместо < или > для создания только верхней или нижней части стрелки.

повторите окончание стрелки (например, >> or //) для тонкой отрисовки.

используйте -- вместо - для создания пунктирной стрелки

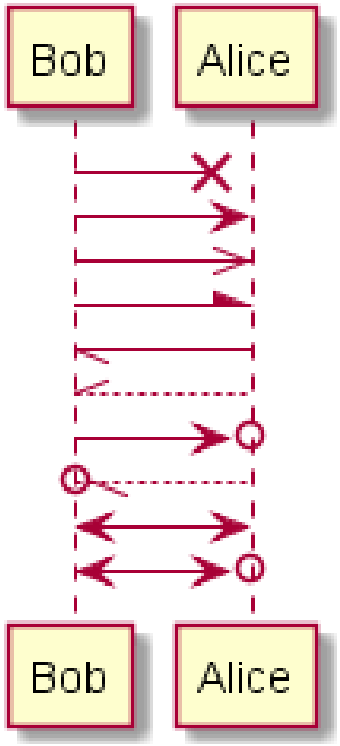
заканчивать символом "o" в острие стрелки

использовать двунаправленные стрелки <-> @startuml

Bob ->x Alice Bob -> Alice Bob ->> Alice Bob -\ Alice Bob \\- Alice Bob //-- Alice

Bob ->o Alice Bob o\\-- Alice

Bob <-> Alice Bob <->o Alice @enduml

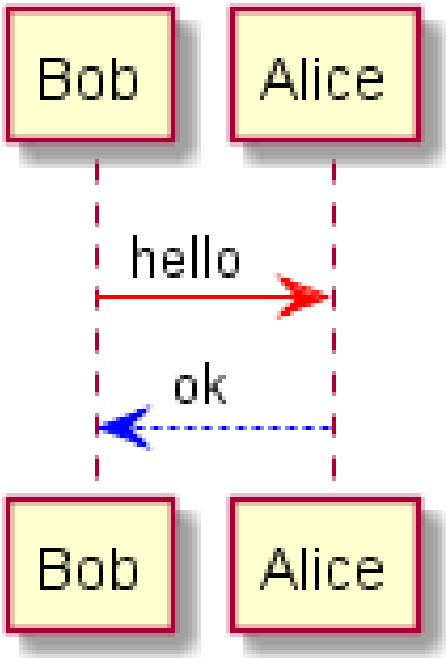


Изменить цвет стрелок

Вы можете изменить цвет отдельных стрелок, используя следующие правила:

@startuml

Bob -[#red]> Alice : hello Alice -[#0000FF]->Bob : ok @enduml

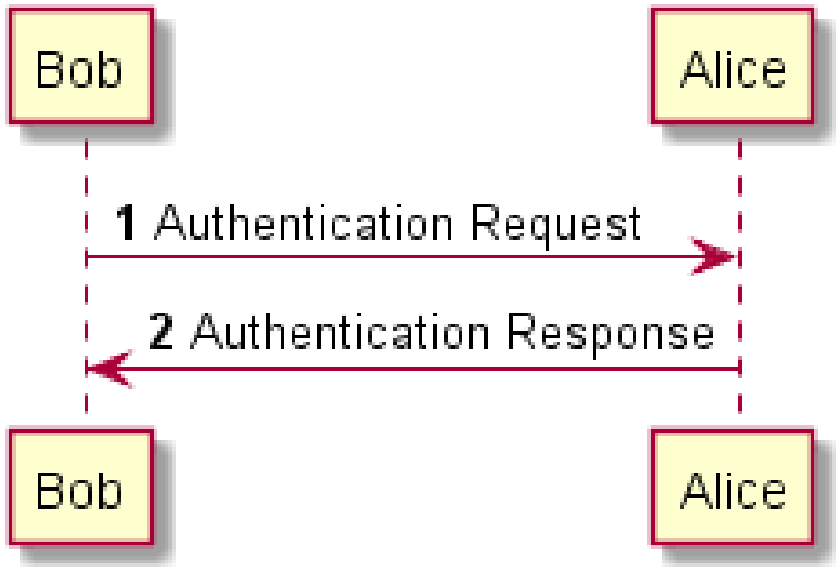


Нумерация сообщений в последовательностях

Ключевое слово autonumber используется для автоматической нумерации сообщений.

@startuml autonumber

Bob -> Alice : Authentication Request Bob <- Alice : Authentication Response @enduml



Вы можете обозначить число с которого начнется отсчет autonumber start , и число которое будет использоваться в качестве инкремента autonumber start increment.

@startuml autonumber

Bob -> Alice : Authentication Request Bob <- Alice : Authentication Response

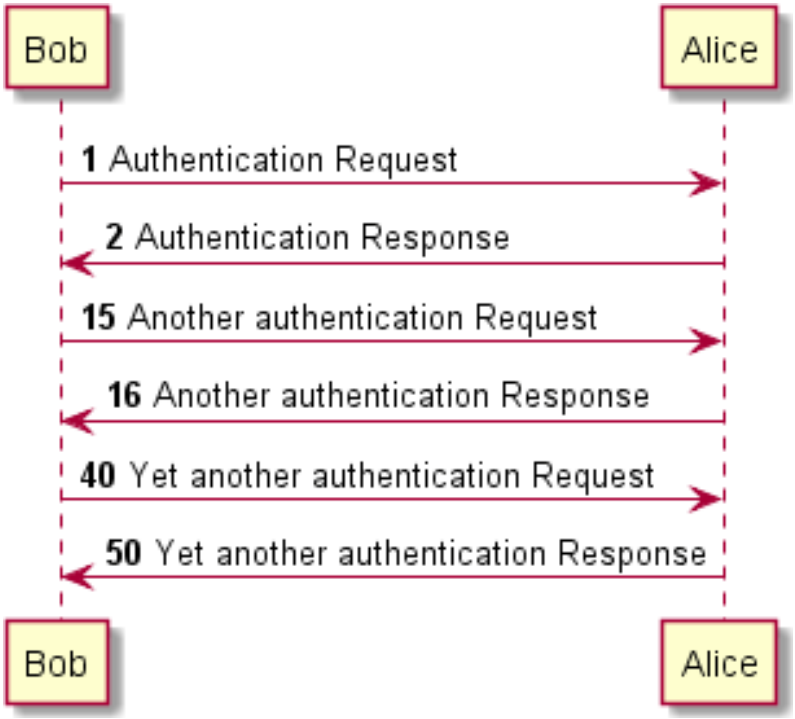
autonumber 15

Bob -> Alice : Another authentication Request Bob <- Alice : Another authentication Response

autonumber 40 10

Bob -> Alice : Yet another authentication Request Bob <- Alice : Yet another authentication Response

@enduml



Можно задавать формат чисел, указав его в двойных кавычках.

Форматирование выполнено с использованием класса Java DecimalFormat (0 означает цифру, # означает цифру или ноль если отсутствует).

При форматировании также можно использовать теги html.

@startuml

autonumber "<b>[000]"

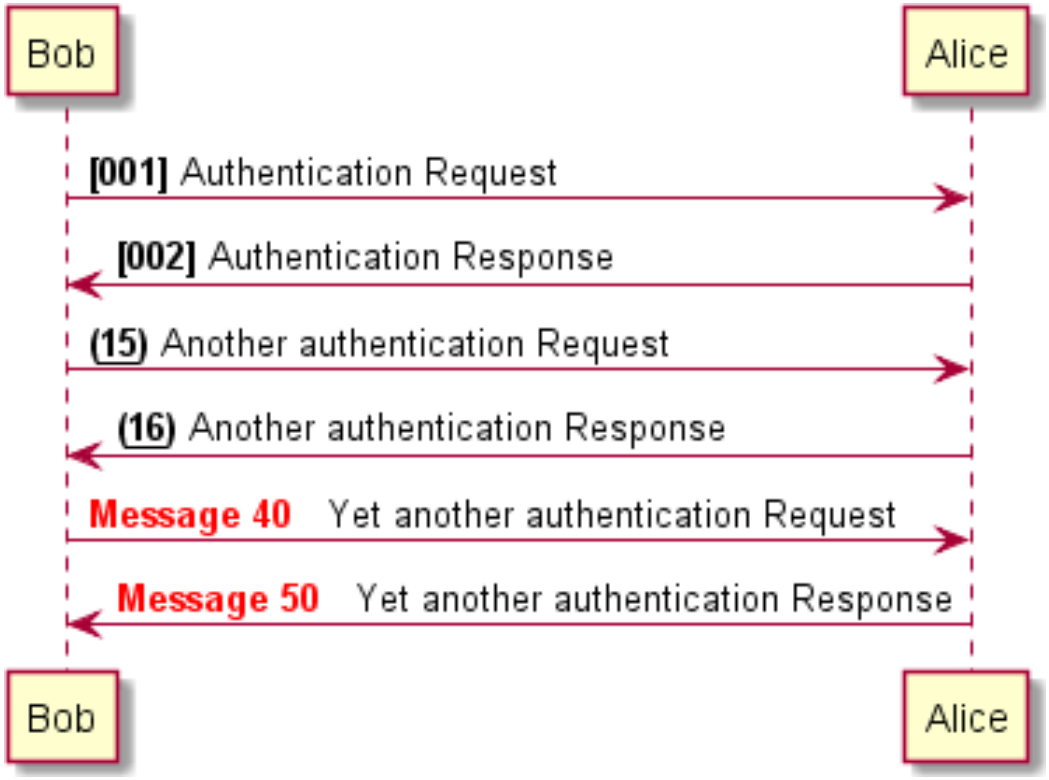
Bob -> Alice : Authentication Request Bob <- Alice : Authentication Response

autonumber 15 "<b>(<u>##</u>)"

Bob -> Alice : Another authentication Request Bob <- Alice : Another authentication Response

autonumber 40 10 "<font color=red><b>Message 0 " Bob -> Alice : Yet another authentication Request Bob <- Alice : Yet another authentication Response

@enduml



Вы так же можете использовать autonumber stop и autonumber resume increment format чтобы соотсетственно остановить и продолжить автоматическое нумерование.

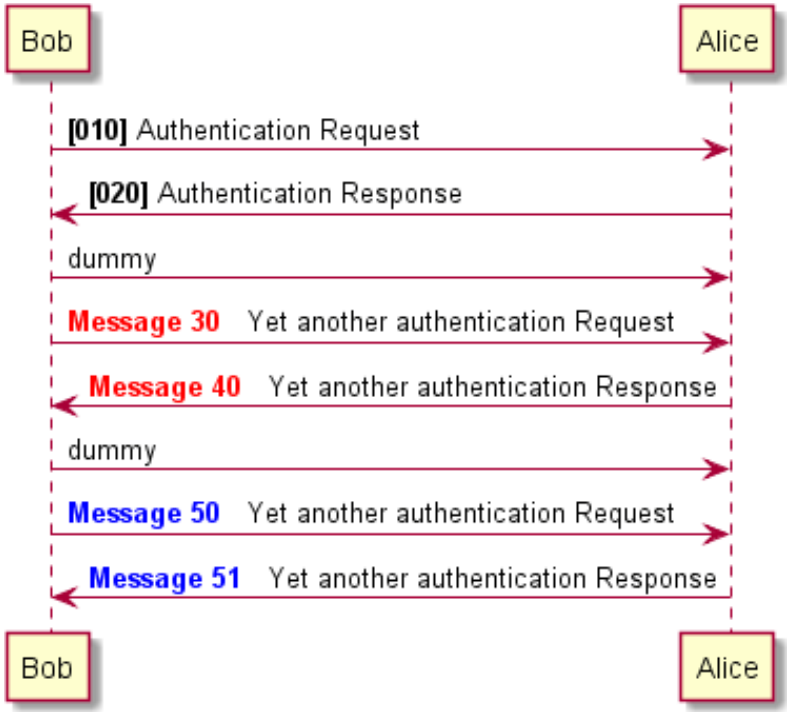
@startuml
autonumber 10 10 "<b>[000]"
Bob -> Alice : Authentication Request Bob <- Alice : Authentication Response

autonumber stop
Bob -> Alice : dummy

autonumber resume "<font color=red><b>Message 0 " Bob -> Alice : Yet another authentication Request Bob <- Alice : Yet another authentication Response

autonumber stop
Bob -> Alice : dummy

autonumber resume 1 "<font color=blue><b>Message 0 " Bob -> Alice : Yet another authentication Request Bob <- Alice : Yet another authentication Response
@enduml



Page Title, Header and Footer

The title keyword is used to add a title to the page.

Pages can display headers and footers using header and footer.

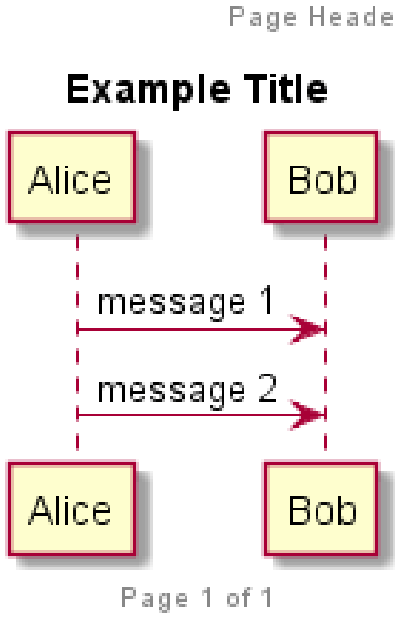
@startuml

header Page Header

footer Page % page % of lastpage title Example Title

Alice -> Bob : message 1 Alice -> Bob : message 2

@enduml



Разбиение диаграм

Ключевое слово newpage используется для разбиения диаграм на несколько изображений. Вы можете указать название страницы сразу после ключевого слова newpage.

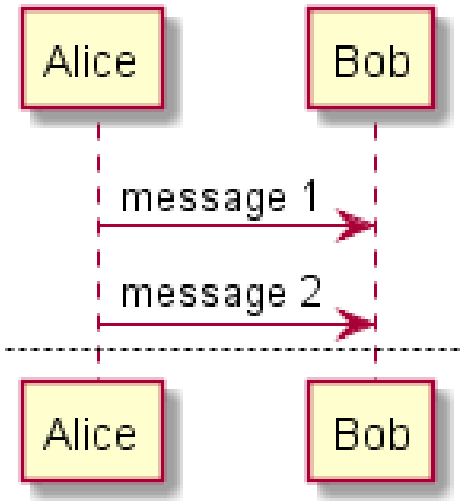
Это очень полезно для печати длинных диаграмм на нескольких страницах.

@startuml

Alice -> Bob : message 1 Alice -> Bob : message 2

newpage

Alice -> Bob : message 3 Alice -> Bob : message 4
newpage A title for the\nlast page Alice -> Bob : message 5
Alice -> Bob : message 6
@enduml

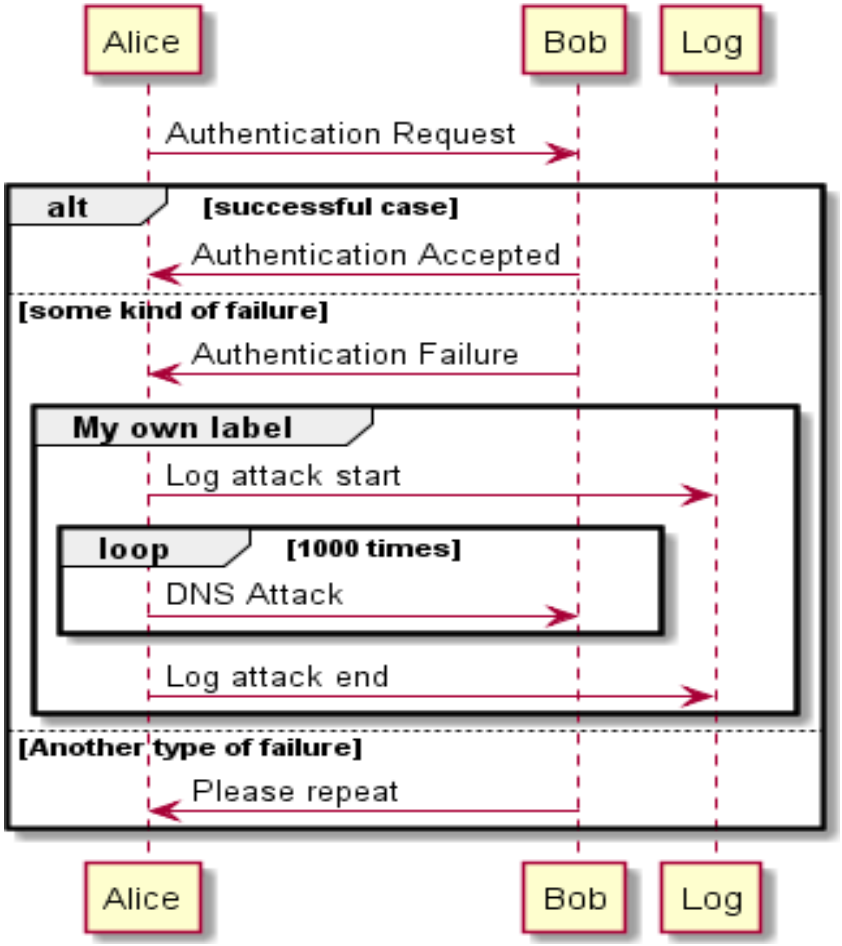


Группировка сообщений

Группировать сообщения возможно используя следующие ключевые слова:

alt/else
opt
loop
par

break
critical
group, соответствует тексту который должен быть отображен
Имеется возможность добавить текст который должен быть отображен в заголовке. Ключевое слово end
используется для завершения группы. Имейте ввиду что допускаются вложенные группы. Ключевое слово end закрывает группу.
Допустимо вложение группы в группу.
@startuml
Alice -> Bob: Authentication Request alt successful case
Bob -> Alice: Authentication Accepted else some kind of failure
Bob -> Alice: Authentication Failure group My own label
Alice -> Log : Log attack start loop 1000 times
Alice -> Bob: DNS Attack
end
Alice -> Log : Log attack end end
else Another type of failure Bob -> Alice: Please repeat
end @enduml

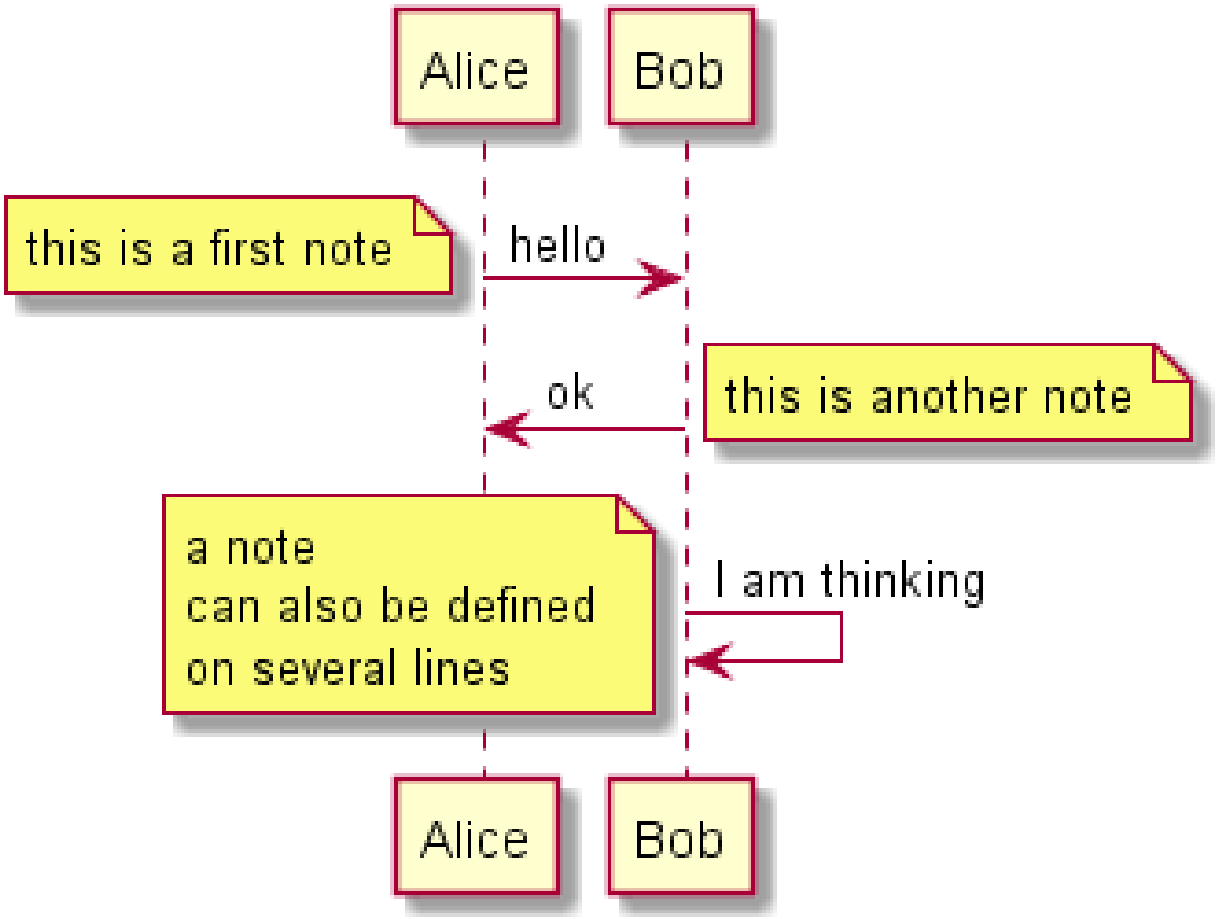


Примечания в сообщениях

Можно помещать заметки к сообщениям, используя ключевые слова note left или note right сразу после сообщения.

Можно делать многострочные заметки используя ключевое слово end note для завершения.

```
@startuml
Alice->Bob : hello
note left: this is a first note

Bob->Alice : ok
note right: this is another note

Bob->Bob : I am thinking note left
a note
can also be defined on several lines end note
@enduml
```



Другие примечания

Так же возможно размещение примечаний относитьельно участников с использованием ключевых слов

<code>note left of</code> , note right of или note over. Возможно выделить примечание изменив цвет фона.

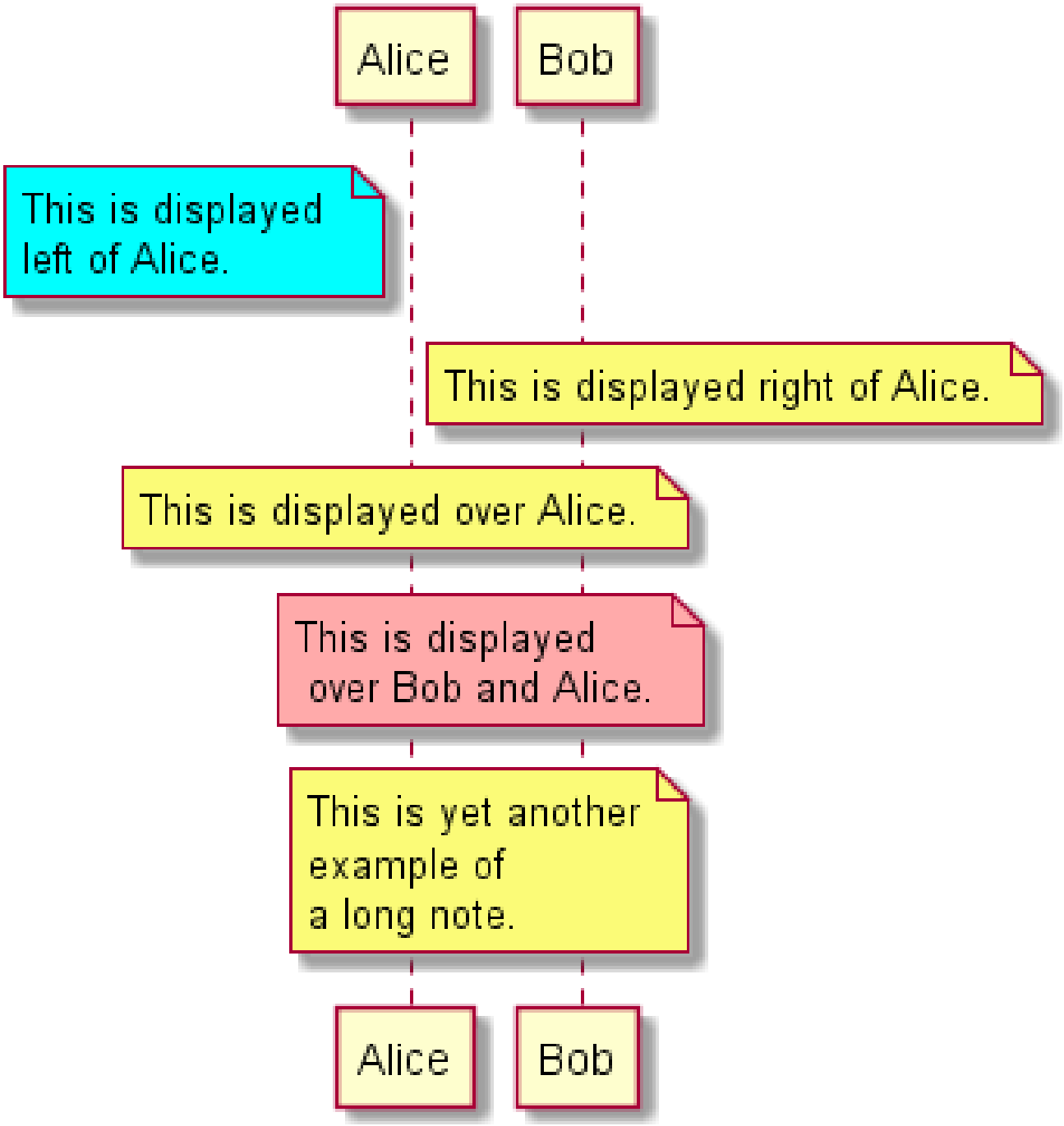
Так же возможно многостройное примечани, для этого существует ключевое слово end note.

@startuml participant Alice participant Bob
note left of Alice #aqua This is displayed
left of Alice. end note

note right of Alice: This is displayed right of Alice. note over Alice: This is displayed over Alice.

note over Alice, Bob #FFAAAA: This is displayed\n over Bob and Alice.

note over Bob, Alice This is yet another example of
a long note. end note
@enduml



Изменение формы примечаний

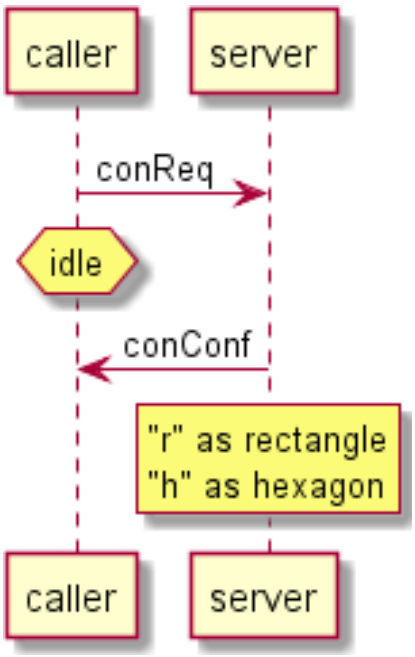
Вы можете использовать hnote и rnote для изменения формы примечаний.

@startuml

caller -> server : conReq hnote over caller : idle caller <- server : conConf rnote over server

"r" as rectangle "h" as hexagon

endrnote @enduml



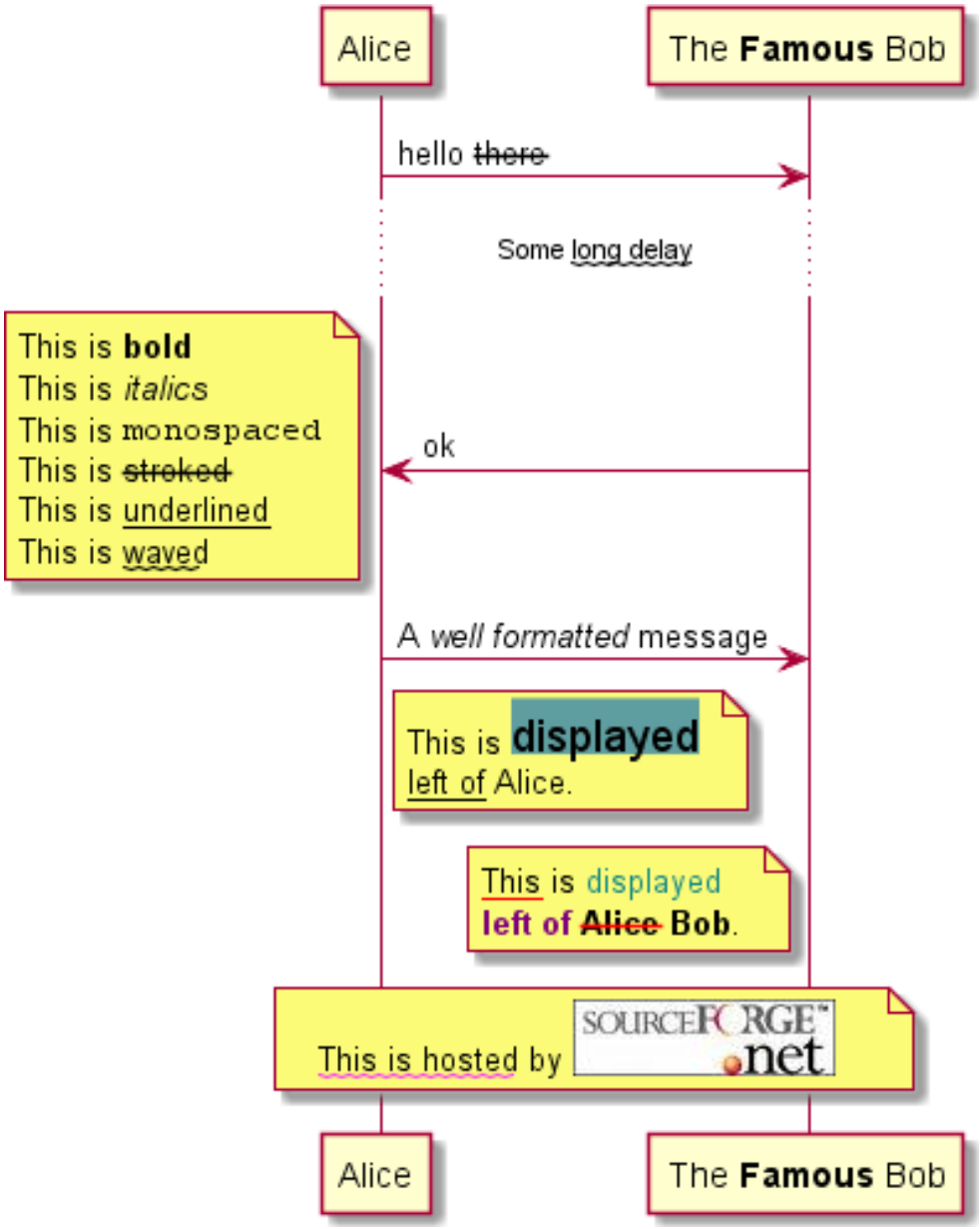
Creole и HTML

Так же можно использовать форматирование на Creole:
@startuml participant Alice
participant "The **Famous** Bob" as Bob

Alice -> Bob : hello --there--
... Some ~~long delay~~ ...
Bob -> Alice : ok note left
This is **bold** This is //italics//
This is ""monospaced"" This is --stroked-- This is underlined This is ~~waved~~
end note

Alice -> Bob : A //well formatted// message note right of Alice
This is <back:cadetblue><size:18>displayed</size></back>
left of Alice. end note
note left of Bob
<u:red>This</u> is <color #118888>displayed</color>
<color purple>left of</color> <s:red>Alice</strike> Bob. end note
note over Alice, Bob
<w:#FF33FF>This is hosted</w> by <img sourceforge.jpg> end note

@enduml



Разделитель

Вы можете использовать разделитель "==", чтобы разбить диаграмму на несколько этапов.

@startuml

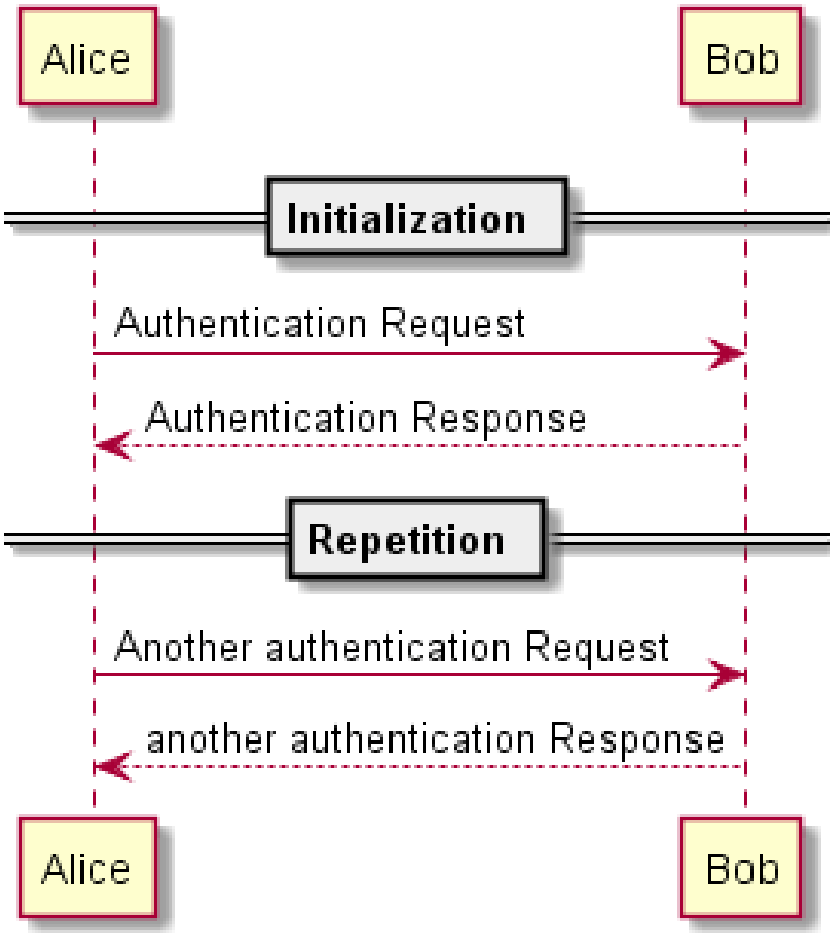
== Initialization ==

Alice -> Bob: Authentication Request Bob --> Alice: Authentication Response

== Repetition ==

Alice -> Bob: Another authentication Request Alice <-- Bob: another authentication Response

@enduml



Ссылки

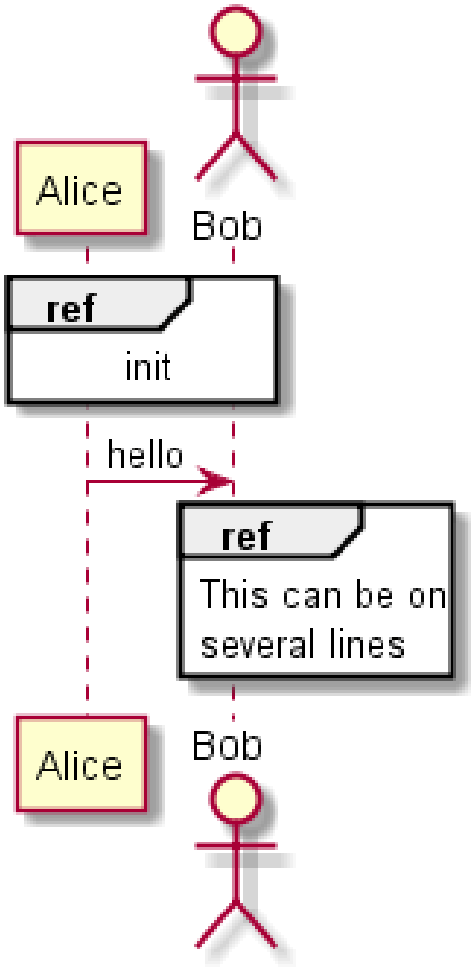
Вы можете использовать ссылки в диаграммах с помощью ключевого слова ref over. @startuml

participant Alice actor Bob

ref over Alice, Bob : init Alice -> Bob : hello

ref over Bob This can be on several lines

end ref @enduml



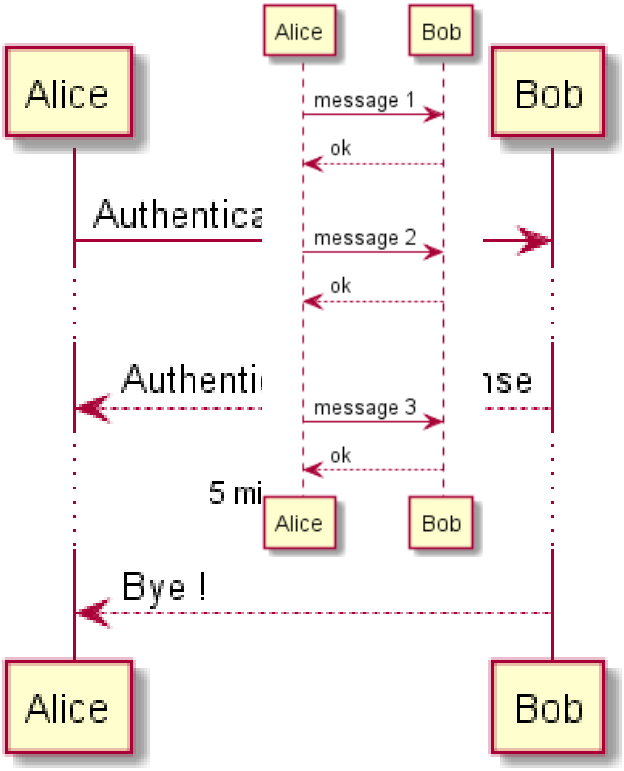
Задержка на диаграммах

Вы можете использовать конструкцию ... для представления временной задержки в процессе на диаграмме. При необходимости можно снабдить задержку комментарием.

@startuml

Alice -> Bob: Authentication Request
...
Bob --> Alice: Authentication Response
...5 minutes later...
Bob --> Alice: Bye !

@enduml



Промежутки

Вы можете использовать ||| чтобы показать промежутки в диаграммах.. Так же возможно указать промежуток в пикселях.

@startuml

Alice -> Bob: message 1 Bob --> Alice: ok
|||
Alice -> Bob: message 2 Bob --> Alice: ok
||45||
Alice -> Bob: message 3 Bob --> Alice: ok

@enduml

Активация и деактивация линии существования

activate и deactivate используются чтобы обозначить активацию участника. Линия существования появляется в момент активации участника.

activate и deactivate применяются к предыдущему сообщению.

destroy обозначает конец линии существования участника.

@startuml participant User

User -> A: DoWork activate A

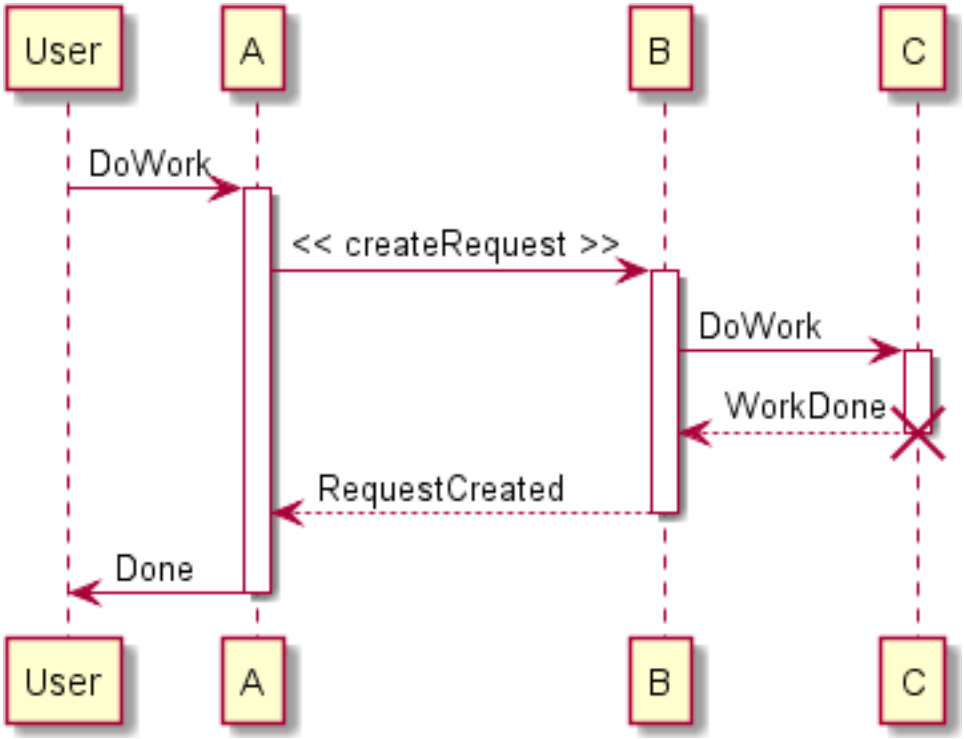
-> B: << createRequest >> activate B

-> C: DoWork activate C
--> B: WorkDone destroy C

B --> A: RequestCreated deactivate B

A -> User: Done deactivate A

@enduml



Можно использовать вложенные линии существования, и возможно добавлять цвет линии существования

@startuml participant User

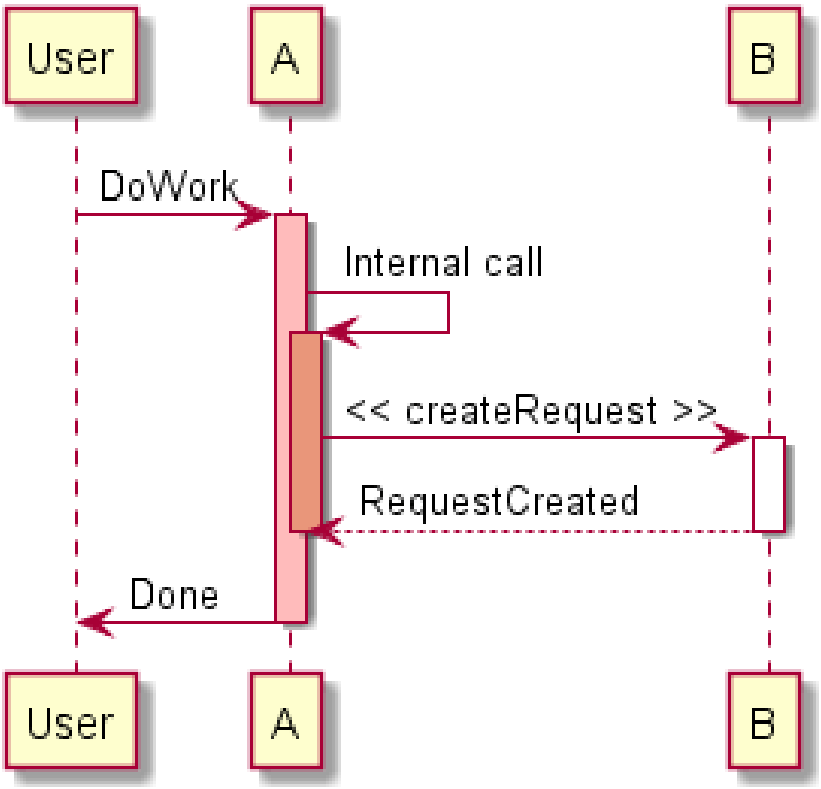
User -> A: DoWork activate A #FFBBBB

A -> A: Internal call activate A #DarkSalmon

-> B: << createRequest >> activate B

--> A: RequestCreated
deactivate B deactivate A
A -> User: Done deactivate A

@enduml

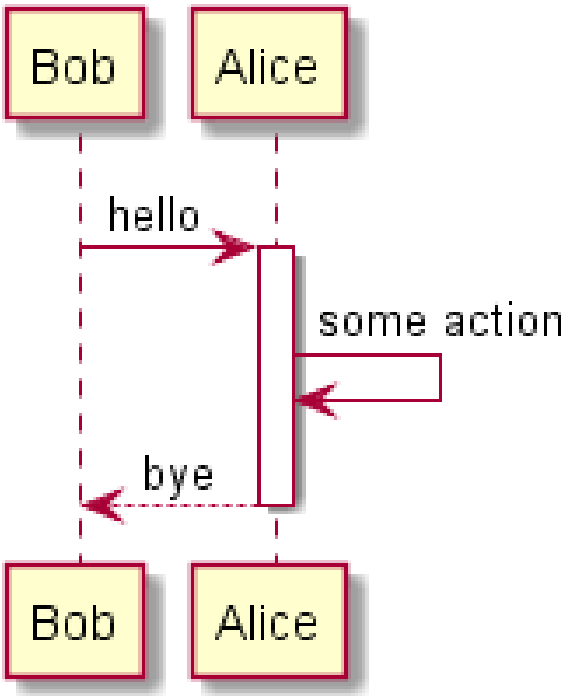


Return

A new command return for generating a return message with optional text label. The point returned to is the point that cause the most recently activated life-line. The syntax is simply return label where label, if provided, can be any string acceptable on conventional messages.

@startuml
Bob -> Alice : hello activate Alice
Alice -> Alice : some action return bye

@enduml



Отображение создания участника процессом

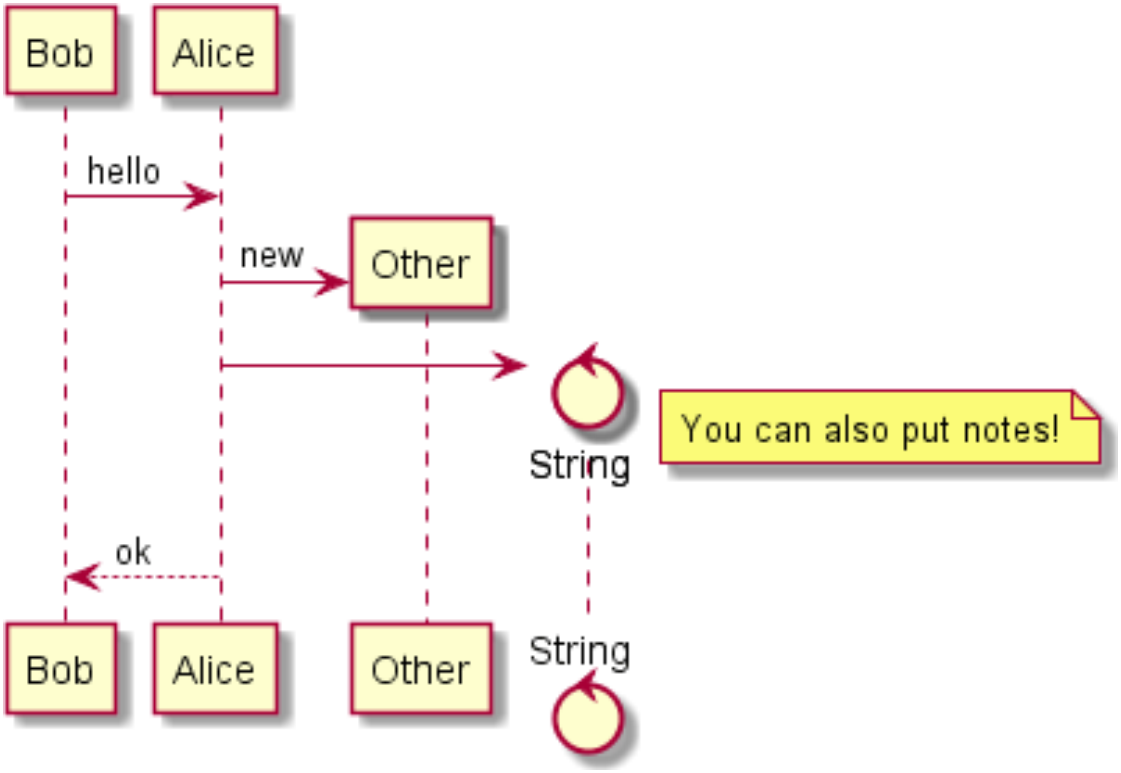
Вы можете использовать ключевое слово create перед декларацией сообщения для акцентирования факта, что принимающий участник создается данным сообщением.

@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

create control String Alice -> String
note right : You can also put notes!

Alice --> Bob : ok @enduml



Входящие и исходящие сообщения

Вы можете использовать входящие или исходящие стрелки если вы хотите сфокусироваться на части диаграммы.

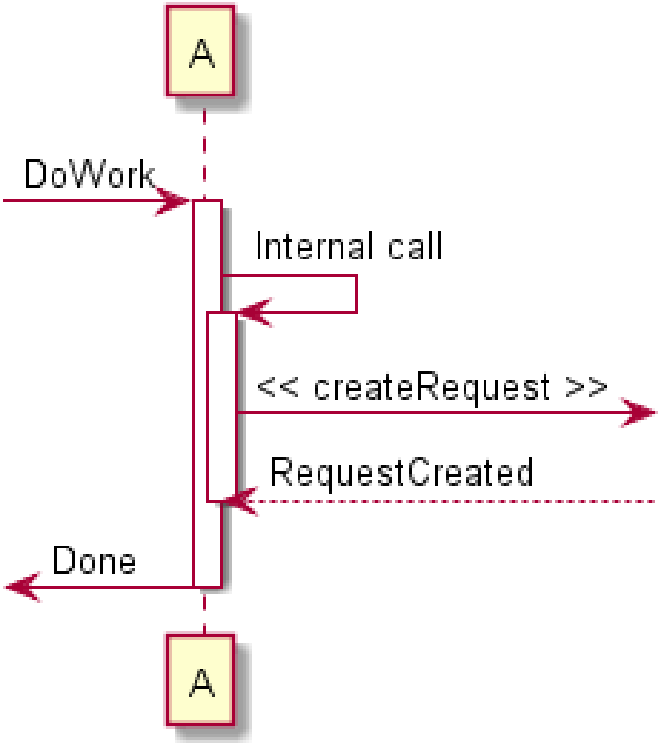
Используйте квадратные скобки для указания левой "[" или правой "]" стороны диаграммы

@startuml
[-> A: DoWork activate A
A -> A: Internal call activate A

A ->] : << createRequest >>

A<--] : RequestCreated deactivate A
[<- A: Done deactivate A @enduml
Вы также можете использовать следующий синтаксис:
@startuml [-> Bob

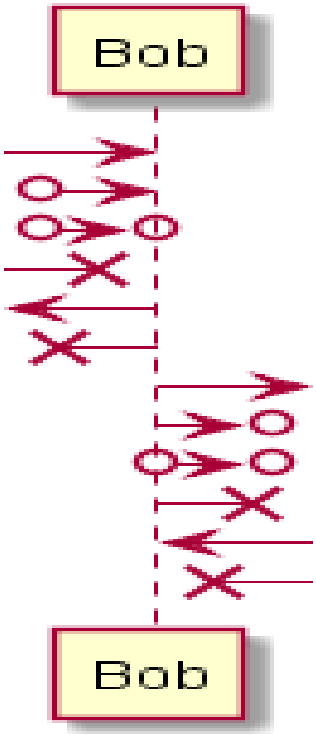
[o-> Bob [o->o Bob [x-> Bob



[<- Bob [x<- Bob

Bob ->] Bob ->o] Bob o->o] Bob ->x]

Bob <-] Bob x<-] @enduml



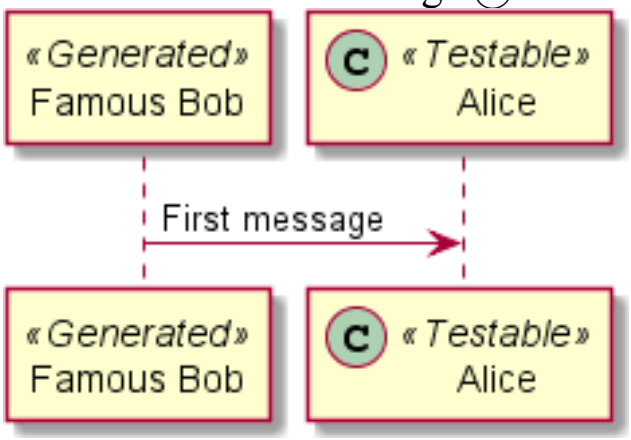
Шаблоны и отметки

Можно добавить шаблоны к участникам используя << и >>.

В шаблоне вы можете добавить отмеченного участника в цветном круге используя синтаксис (X,color). @startuml

participant "Famous Bob" as Bob << Generated >> participant Alice << (C,#ADD1B2) Testable >>

Bob->Alice: First message @enduml



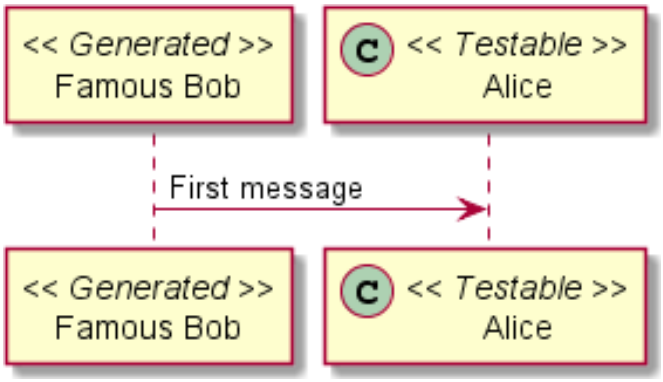
По умолчанию, символ guillemet используется для отображения шаблона. Вы можете изменить это поведение, используя skinparam guillemet:

@startuml

skinparam guillemet false

participant "Famous Bob" as Bob << Generated >> participant Alice << (C,#ADD1B2) Testable >>

Bob->Alice: First message @enduml



@startuml

participant Bob << (C,#ADD1B2) >> participant Alice << (C,#ADD1B2) >>

Bob->Alice: First message @enduml

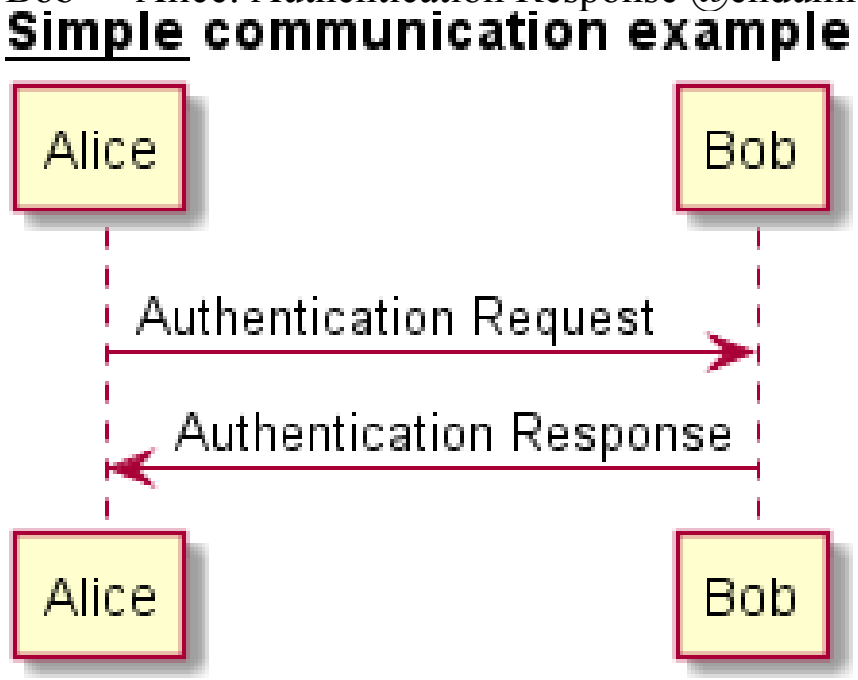
Больше информации в заголовках

Вы можете использовать форматирование на Creole для заголовков.

@startuml

title Simple **communication** example Alice -> Bob: Authentication Request

Bob -> Alice: Authentication Response @enduml



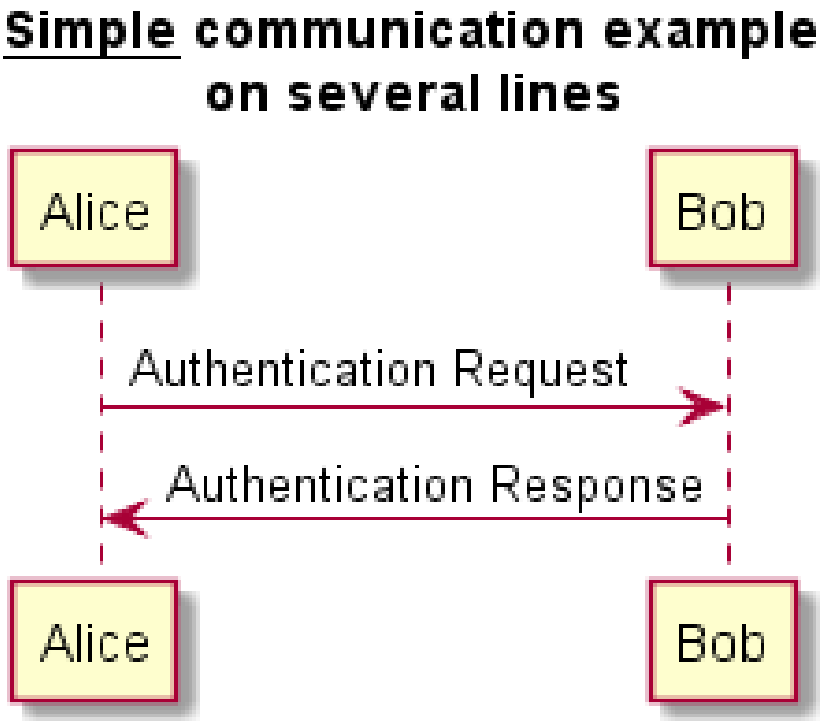
С помощью последовательности символов \n вы можете добавить перевод строки в заголовок.

@startuml

title Simple communication example\non several lines

Alice -> Bob: Authentication Request Bob -> Alice: Authentication Response

@enduml



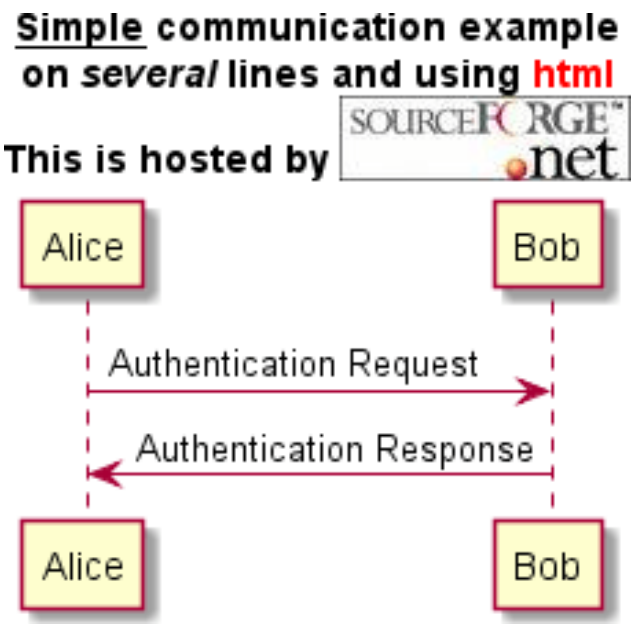
Вы также можете задать заголовок на нескольких строках, используя ключевые слова title и end title .

@startuml

title
<u>Simple</u> communication example
on several lines and using html This is hosted by <img:sourceforge.jpg>
end title

Alice -> Bob: Authentication Request Bob -> Alice: Authentication Response

@enduml



Группировка участников

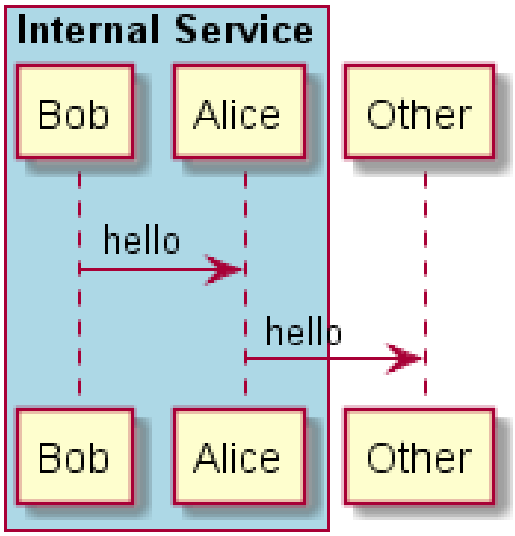
Можно создать прямоугольник вокруг участников, используя комманды box и end box. Вы можете задать опциональный заголовок и цвет фона, после соммады the box. @startuml

box "Internal Service" #LightBlue participant Bob

participant Alice end box participant Other

Bob -> Alice : hello

Alice -> Other : hello @enduml



Удаление футера

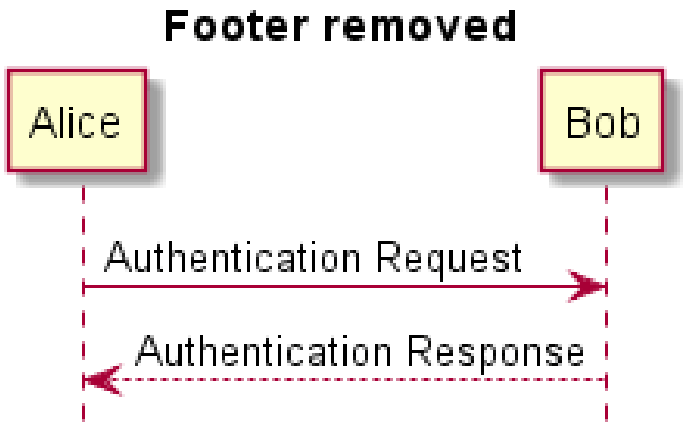
Вы можете использовать ключевое слово hide footbox для удаления футера из диаграммы.

@startuml

hide footbox
title Footer removed

Alice -> Bob: Authentication Request Bob --> Alice: Authentication Response

@enduml



Skinparam

Вы можете использовать команду skinparam для изменения шрифтов и цветов диаграммы Вы можете использовать данную команду :

В определении диаграммы, как любую другую команду,

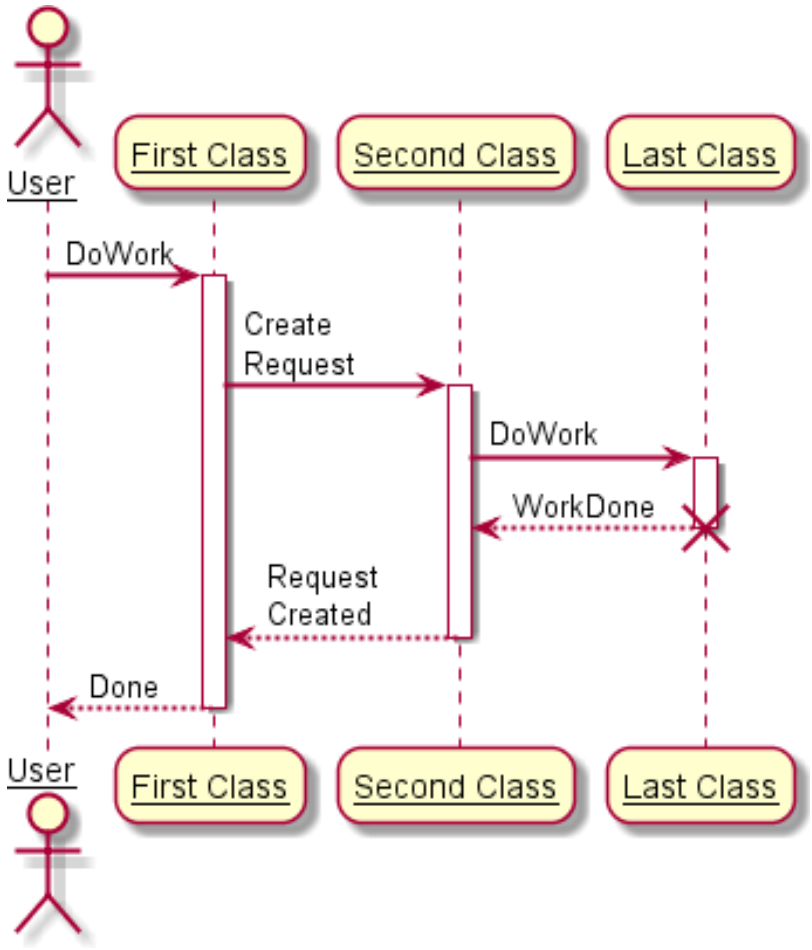
В подключенном файле,

В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете изменить другие параметры отображения, как видно из следующих примеров:

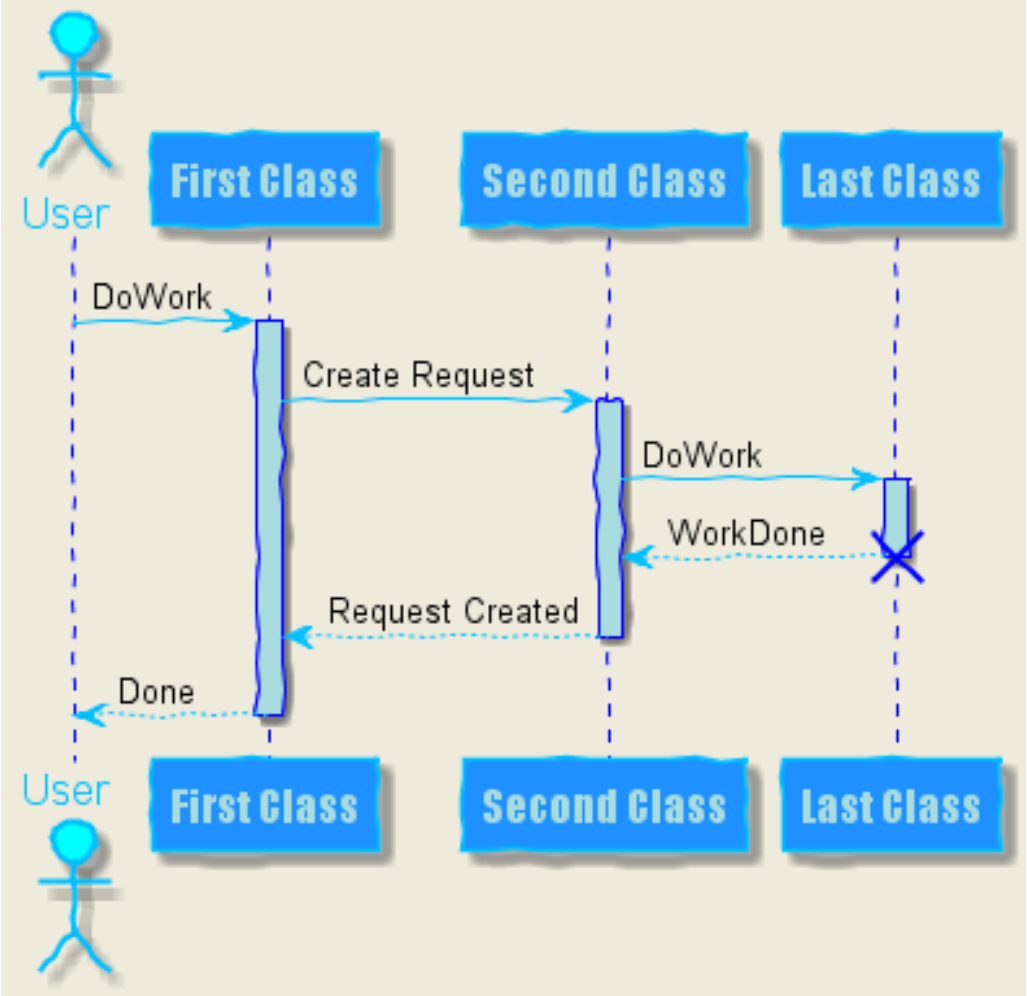
@startuml
skinparam sequenceArrowThickness 2
skinparam roundcorner 20

```
skinparam maxmessagesize 60
skinparam sequenceParticipant underline
actor User
participant "First Class" as A participant "Second Class" as B participant "Last Class" as C

User -> A: DoWork activate A
-> B: Create Request activate B
-> C: DoWork activate C
--> B: WorkDone destroy C

B --> A: Request Created deactivate B

A --> User: Done deactivate A

@enduml
@startuml
skinparam backgroundColor #EEEBDC skinparam handwritten true
```



```
skinparam sequence { ArrowColor DeepSkyBlue ActorBorderColor DeepSkyBlue LifeLineBorderColor blue LifeLineBackgroundColor #A9DCDF
ParticipantBorderColor DeepSkyBlue ParticipantBackgroundColor DodgerBlue ParticipantFontName Impact ParticipantFontSize 17 ParticipantFontColor #A9DCDF
ActorBackgroundColor aqua ActorFontColor DeepSkyBlue
ActorFontSize 17 ActorFontName Aapex
}

actor User
participant "First Class" as A participant "Second Class" as B participant "Last Class" as C

User -> A: DoWork activate A

-> B: Create Request activate B

-> C: DoWork activate C
--> B: WorkDone destroy C

B --> A: Request Created deactivate B

A --> User: Done deactivate A
```

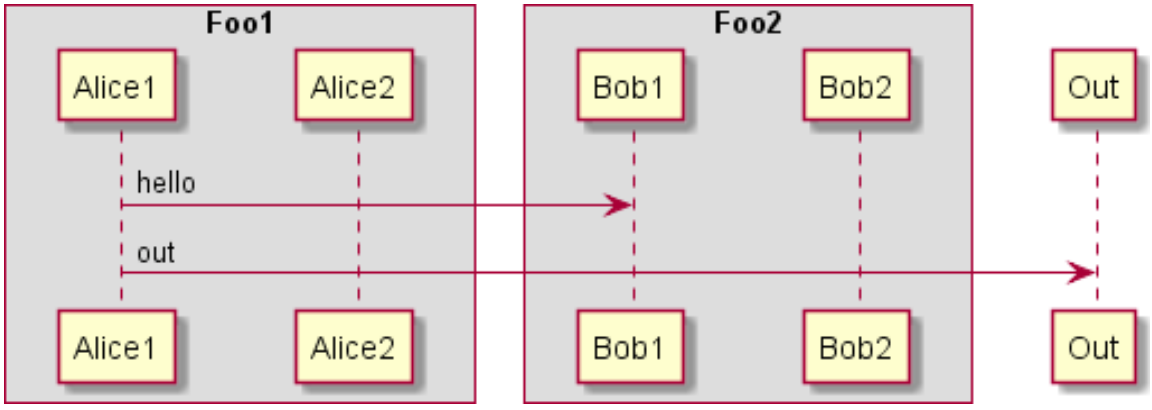
@enduml



Изменение отступов

Вы можете изменить некоторые настройки отступов
@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

box "Foo1" participant Alice1
participant Alice2 end box
box "Foo2" participant Bob1 participant Bob2 end box
Alice1 -> Bob1 : hello Alice1 -> Out : out @enduml
Диаграмма прецедентов



Рассмотрим несколько примеров:

Заметьте, что Вы можете отключить тени, используя команду skinparam shadowing false.

Прецеденты

Прецеденты заключаются в две скобки (потому что две скобки выглядят как овал).

Вы можете использовать usecase для создания прецедента. также вы можете создать псевдоним, используя

as keyword. Этот пседоним будет использоваться позже во время определения связей

@startuml

(First usecase)
(Another usecase) as (UC2) usecase UC3
usecase (Last\nusecase) as UC4 @enduml

Актёры

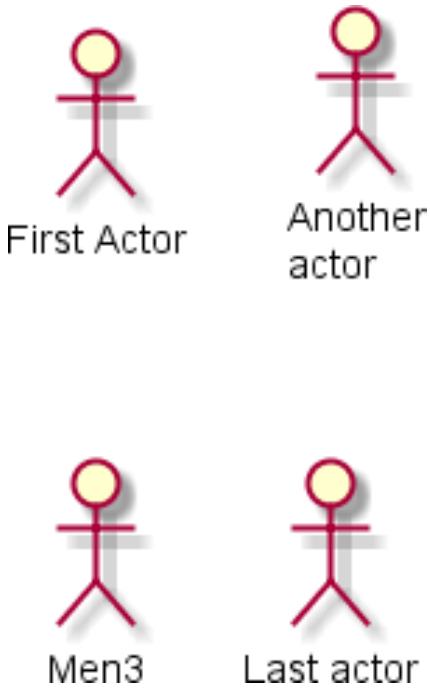
Актеры обозначаются заключёнными между двумя точками.

Также Вы можете использовать ключевое слово actor для определения актёра. И вы можете создать псевдоним, используя ключевое слово as. Этот псевдоним будет использован позднее, при определении отношений.

Мы увидим, что определения актеров не обязательны.

@startuml

:First Actor:
:Another\nactor: as Men2 actor Men3
actor :Last actor: as Men4 @enduml



Описание прецедентов

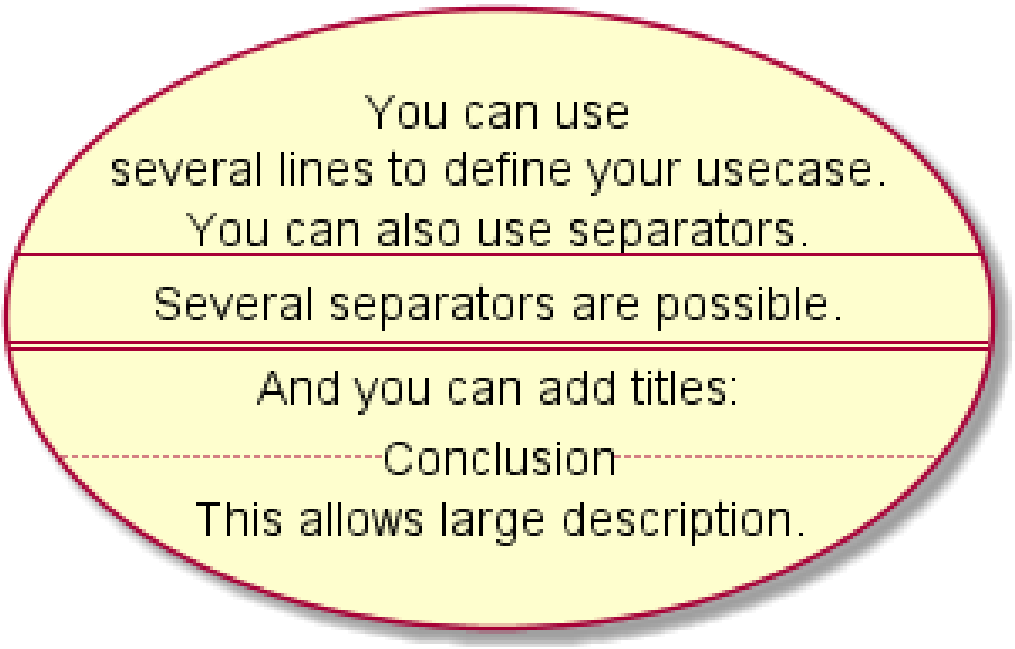
Если вы хотите описание на несколько строк, можете использовать кавычки.

Вы также можете использовать следующие разделители: -- .. == . И вы можете вставлять заголовки внутри разделителей.

@startuml

usecase UC1 as "You can use

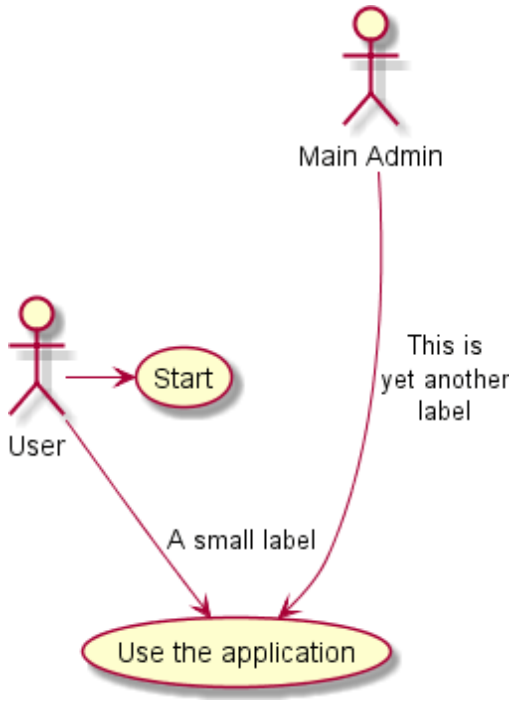
several lines to define your usecase. You can also use separators.
--
Several separators are possible.
==
And you can add titles:
..Conclusion..
This allows large description." @enduml



Простой пример

Для соединения актеров и прецедентов, используется стрелка -->.
Чем больше тире - в стрелке, тем она длиннее. Вы можете добавить метку на стрелку, добавив символ :
при определении стрелки.
В этом примере, вы можете видеть, что User не определён ранее и используется как актёр.
@startuml

User -> (Start)
User --> (Use the application) : A small label
:Main Admin: ---> (Use the application) : This is\nyet another\nlabel
@enduml



Расширение

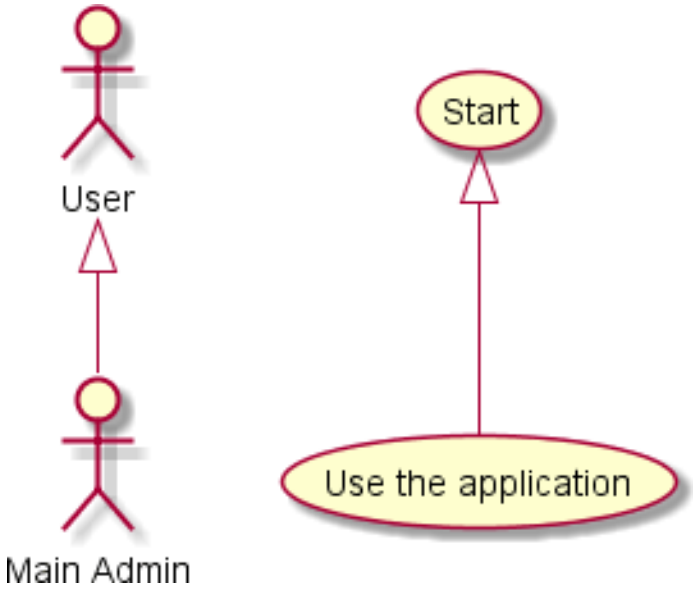
Если один актёр/прецедент расширяют другой, вы можете использовать символ <|--. @startuml

:Main Admin: as Admin

(Use the application) as (Use)

User <|-- Admin (Start) <|-- (Use)

@enduml

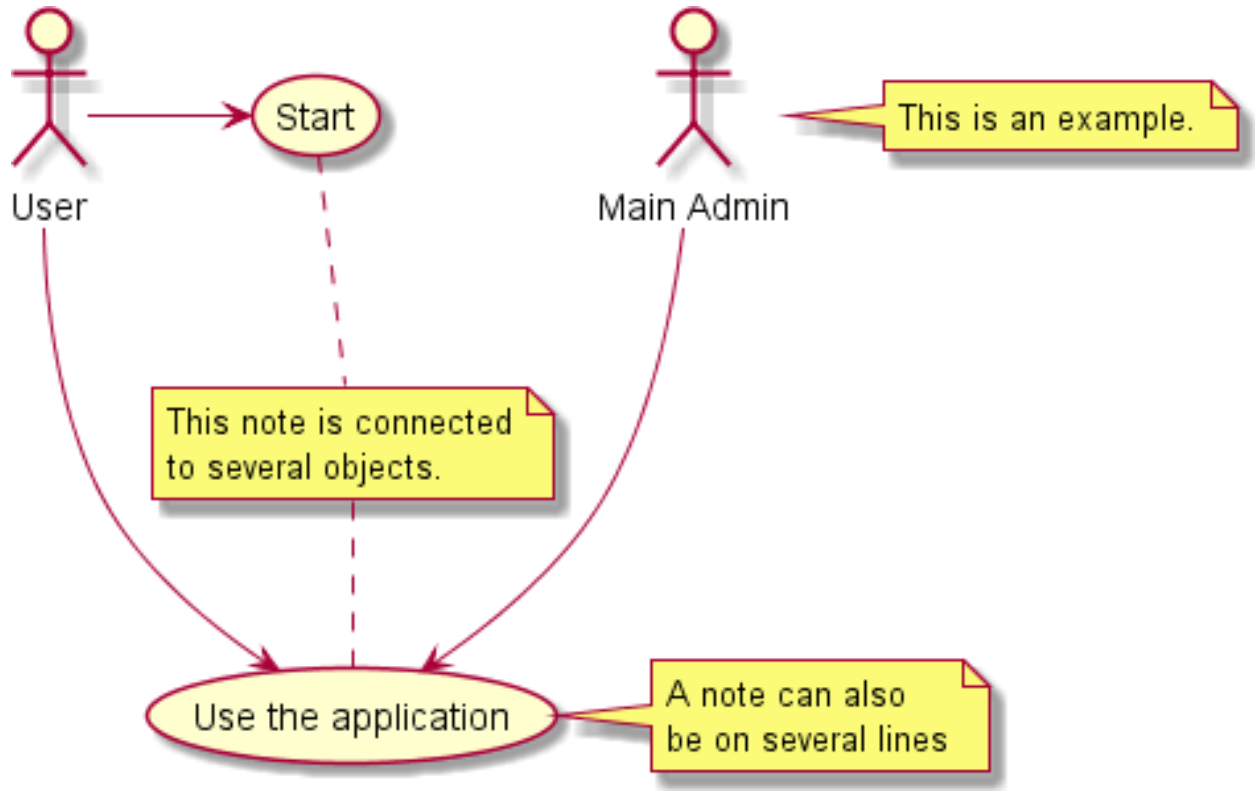


Использование заметок

Вы можете использовать ключевые слова note left of , note right of , note top of , note bottom of чтобы создать заметку относящуюся к одному объекту.

Заметка так же может быть создана с помощью ключевого слова note , а затем прикреплена к другому объекту используя символ ...

```
@startuml
:Main Admin: as Admin
(Use the application) as (Use) User -> (Start)
User --> (Use)

Admin ---> (Use)
note right of Admin : This is an example. note right of (Use)
A note can also
be on several lines end note

note "This note is connected\nto several objects." as N2 (Start) .. N2
N2 .. (Use)
@enduml
```



Шаблоны

Вы можете добавить шаблоны когда определяете актёров и прецеденты, используя << и >>. @startuml

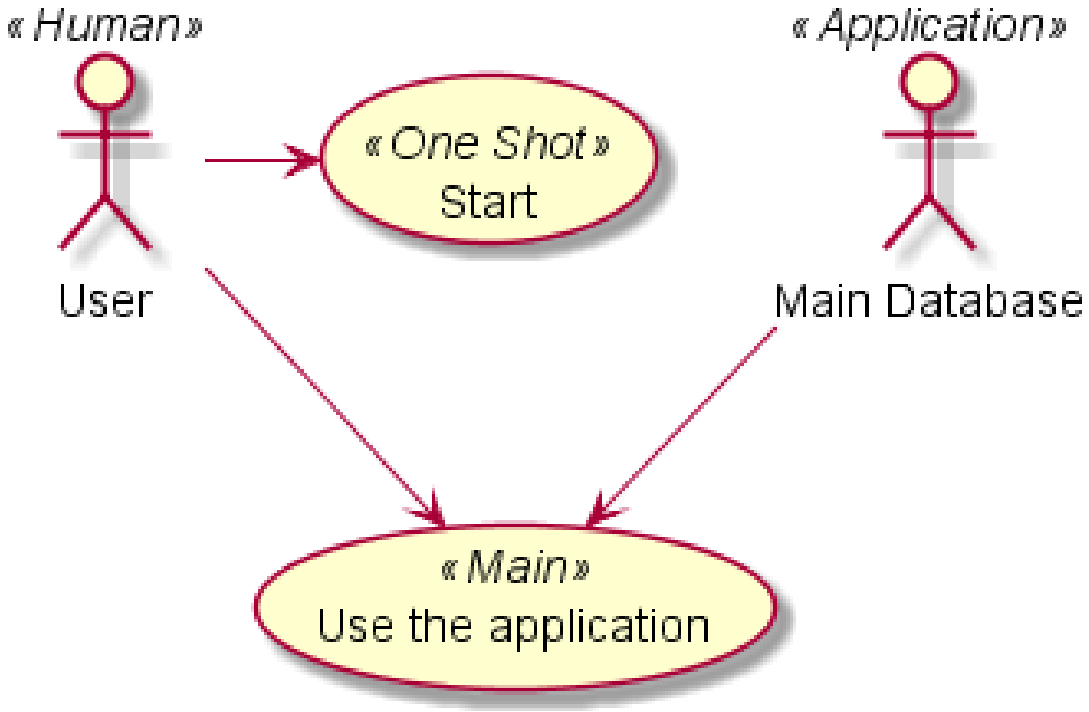
User << Human >>

:Main Database: as MySql << Application >> (Start) << One Shot >>

(Use the application) as (Use) << Main >>

User -> (Start) User --> (Use)

MySql --> (Use) @enduml



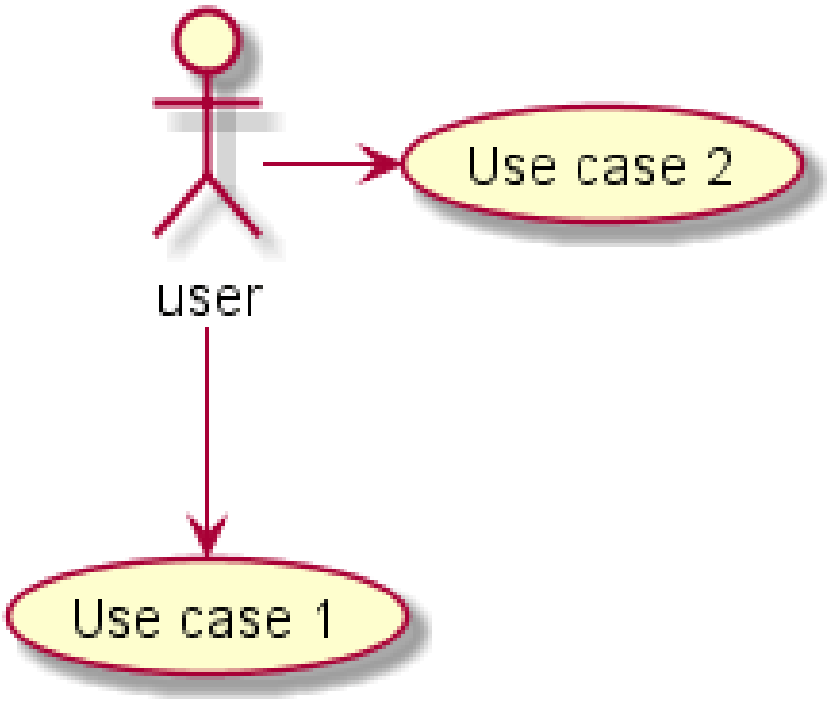
Смена направления стрелок

По умолчанию, связи между классами имеют два тире -- и вертикально ориентированны. можно использовать горизонтальные связи, с помощью написание одного тире (или точки), вот так:

@startuml

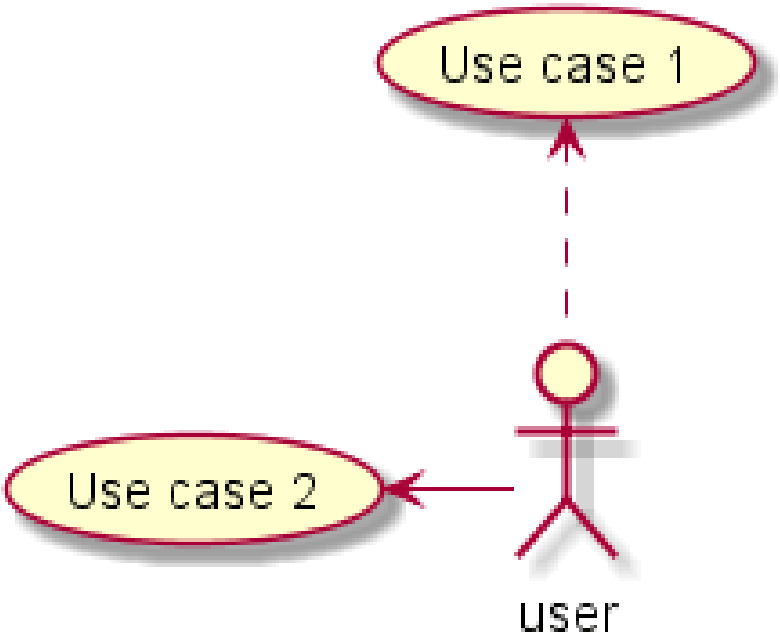
:user: --> (Use case 1)

:user: -> (Use case 2) @enduml



Вы так же можете изменить направление с помощью переворачивания связи:

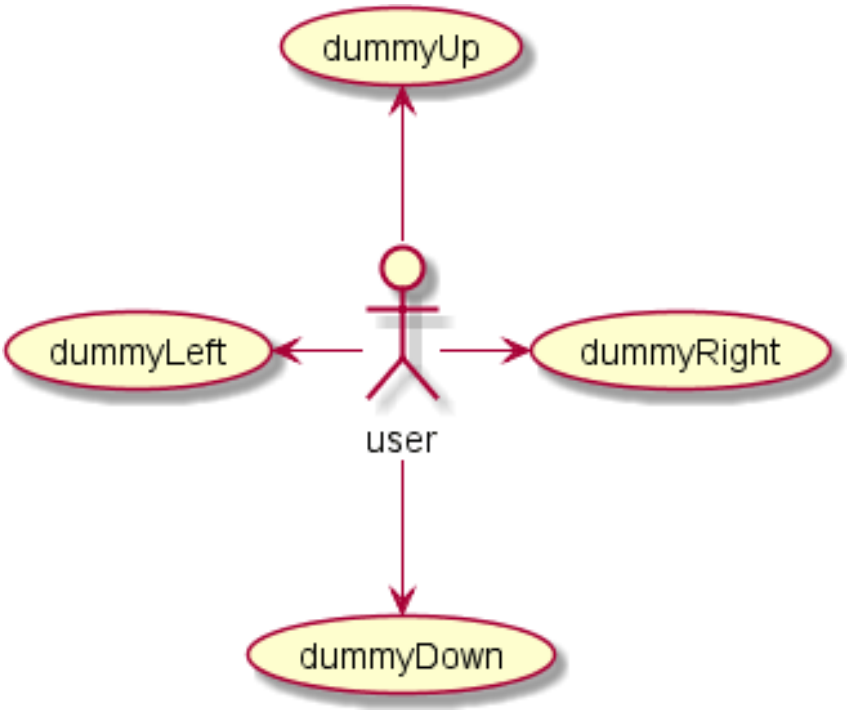
@startuml
(Use case 1) <.. :user: (Use case 2) <- :user: @enduml



Так же возможно сменить направление добавляя ключевые слова left, right, up или down внутри стрелки:

@startuml
:user: -left-> (dummyLeft)
:user: -right-> (dummyRight)

:user: -up-> (dummyUp)
:user: -down-> (dummyDown) @enduml



Вы можете записать короче, используя только первый символ названия направления (например, -d- вместо
-down-) или первые два символа (-do-).

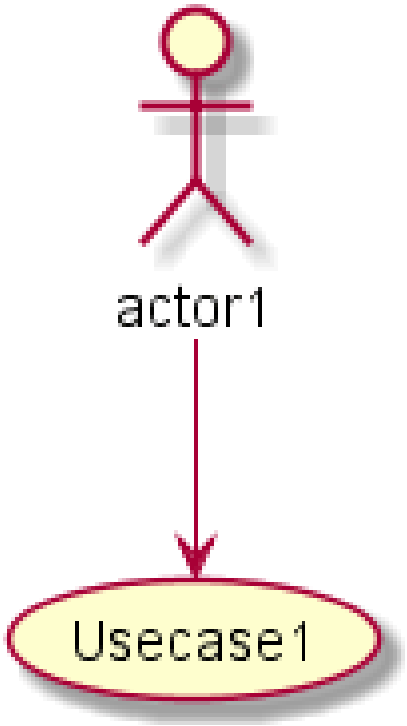
Пожалуйста, помните, что Вы не должны использовать эту функциональность без реальной необходимости:

GraphViz обычно даёт хороший результат без дополнительных настроек.

Разделение диаграмм

Ключевое слово newpage используется для разделения диаграмм на несколько страниц или изображений.

@startuml
:actor1: --> (Usecase1) newpage
:actor2: --> (Usecase2) @enduml

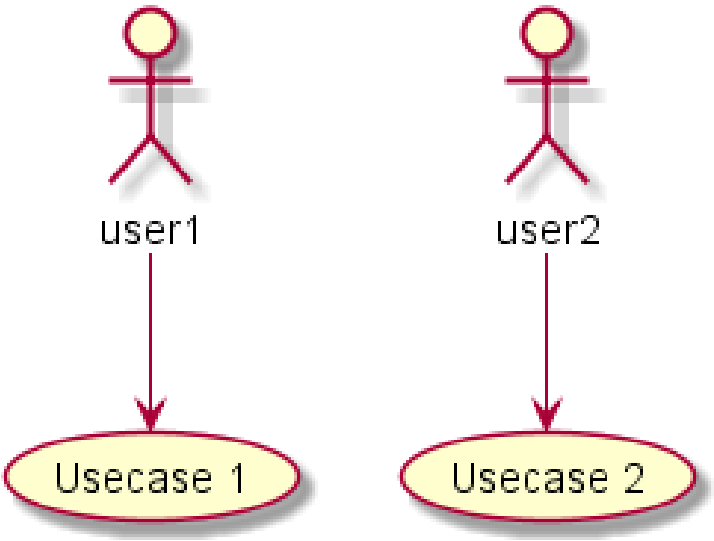


Направление слева направо

Общее поведение по умолчанию - построение диаграмм сверху вниз.

@startuml 'default
top to bottom direction user1 --> (Usecase 1) user2 --> (Usecase 2)

@enduml

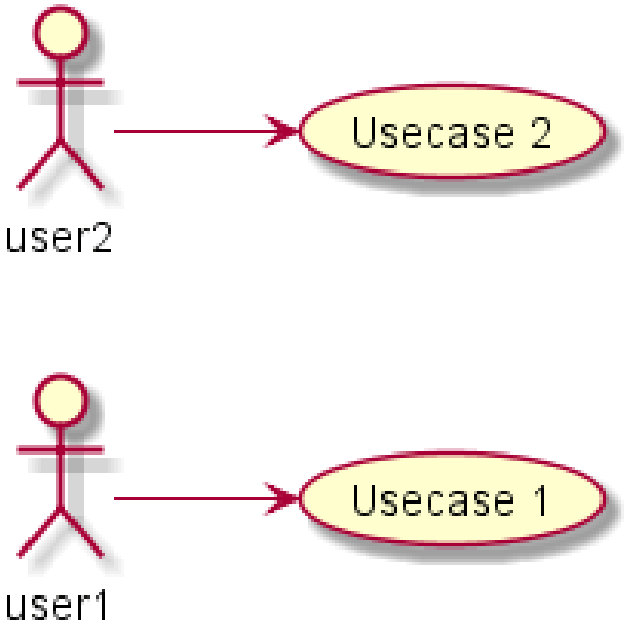


Вы можете изменить направление на слева напрво используя команду left to right direction. Часто результат с таким направлением выглядит лучше.

@startuml

left to right direction user1 --> (Usecase 1) user2 --> (Usecase 2)

@enduml



Skinparam

Вы можете использовать команду skinparam для изменения шрифтов и цветов диаграммы Вы можете использовать данную команду :

В определении диаграммы, как любую другую команду,

В подключенном файле,

В конфигурационном файле, указанном в командной строке в задании ANT. Вы можете задать цвет или шрифт для актёров или прецедентов с шаблонами. @startuml

skinparam handwritten true

skinparam usecase { BackgroundColor DarkSeaGreen BorderColor DarkSlateGray

BackgroundColor<< Main >> YellowGreen BorderColor<< Main >> YellowGreen

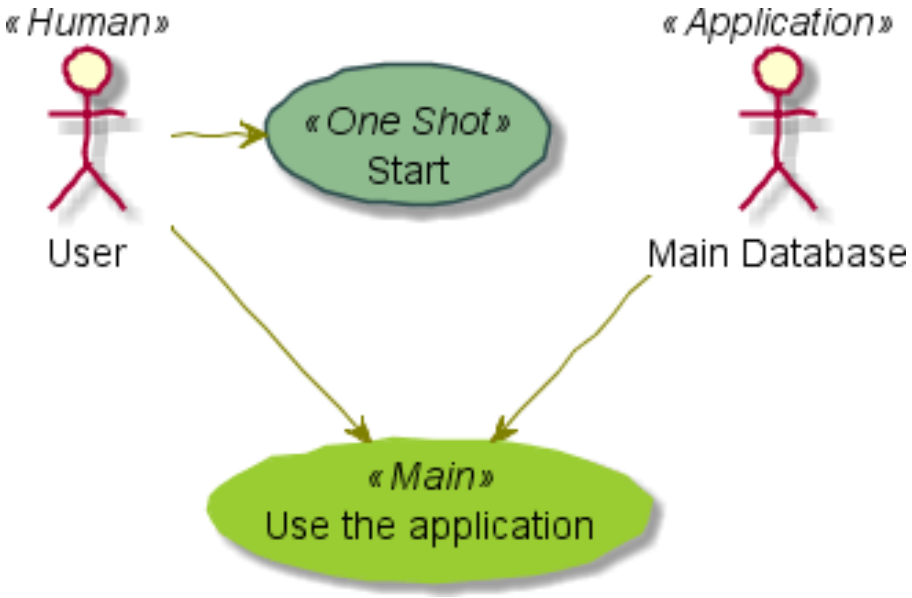
ArrowColor Olive ActorBorderColor black ActorFontName Courier

ActorBackgroundColor<< Human >> Gold
}

User << Human >>
:Main Database: as MySql << Application >> (Start) << One Shot >>
(Use the application) as (Use) << Main >>

User -> (Start) User --> (Use)

MySql --> (Use) @enduml



Полноценный пример

@startuml
left to right direction skinparam packageStyle rectangle actor customer
actor clerk rectangle checkout {
customer -- (checkout) (checkout) .> (payment) : include (help) .> (checkout) : extends (checkout) -- clerk
}

@enduml

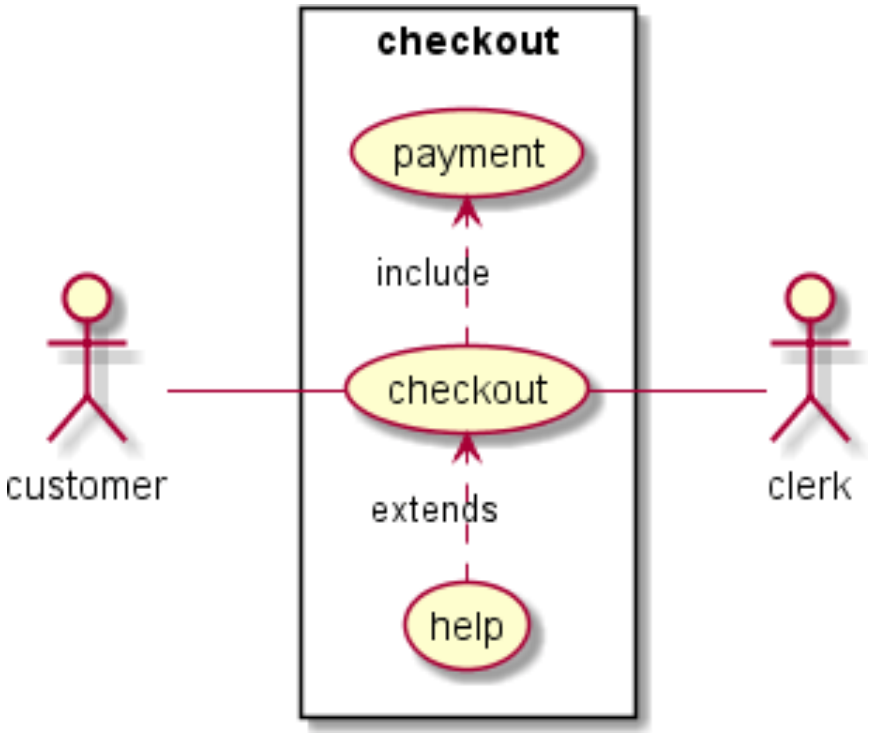


Диаграмма классов

Отношения между классами

Отношения между классами определяются с помощью следующих символов:

Type	Symbol	Drawing
Extension	<|--	
Composition	*--	
Aggregation	o--	

Можно заменить - на .., чтобы создать пунктирную линию. Зная эти правила можно нарисовать следующие изображения: @startuml

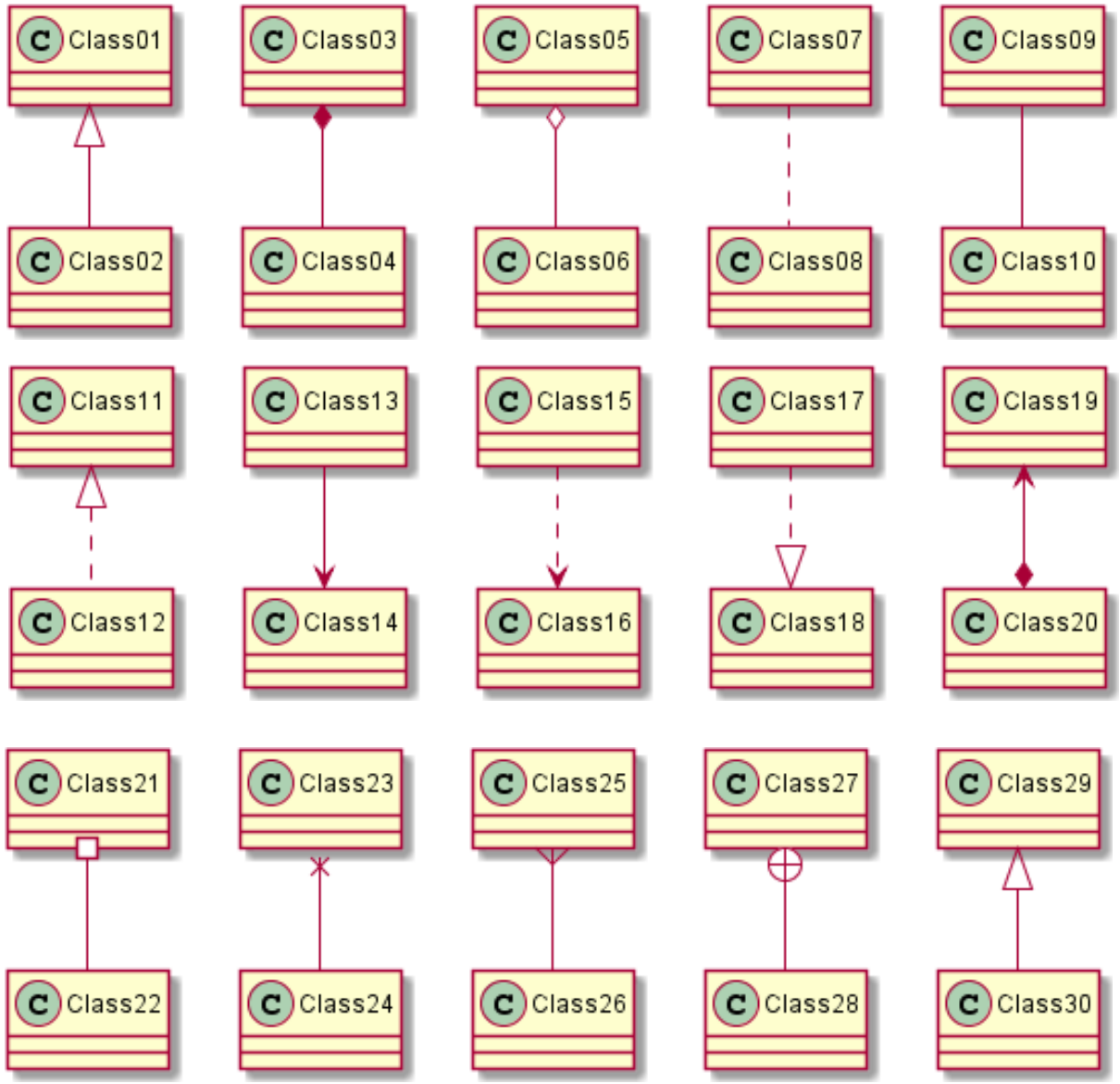
Class01 <|-- Class02 Class03 *-- Class04 Class05 o-- Class06 Class07 .. Class08 Class09 -- Class10 @enduml

@startuml

Class11 <|.. Class12 Class13 --> Class14 Class15 ..> Class16 Class17 ..|> Class18 Class19 <--* Class20 @enduml

@startuml

Class21 #-- Class22 Class23 x-- Class24 Class25 }-- Class26 Class27 +-- Class28 Class29 ^-- Class30 @enduml



Метки на отношениях

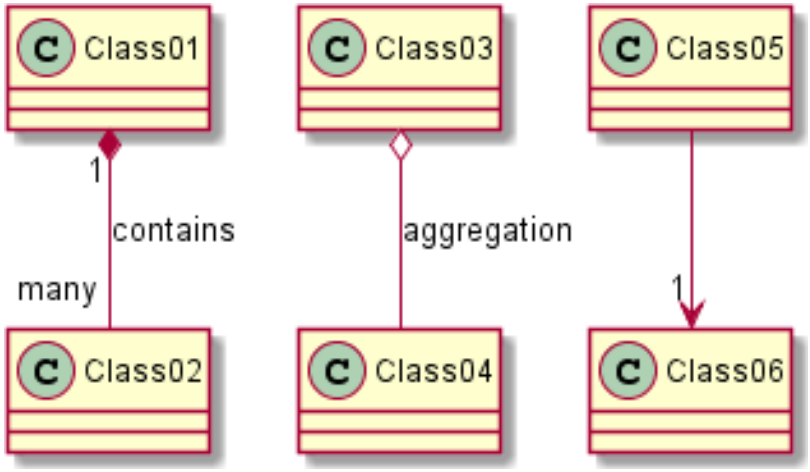
Для отношения можно добавить метку. Делается это с помощью указания символа :, после которого указывается текст метки.

Для указания количества элементов на каждой стороне отношения можно использовать двойные кавычки

"".

@startuml

Class01 "1" *-- "many" Class02 : contains Class03 o-- Class04 : aggregation Class05 --> "1" Class06



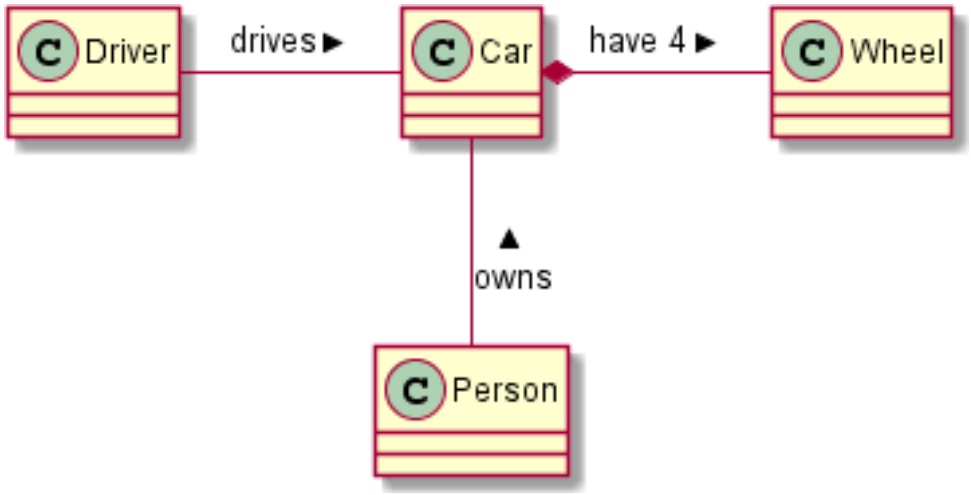
@enduml

Вы можете добавить дополнительные стрелки < или > в начале или в конце метки, указывающие на использование одного из объектов другим объектом.

@startuml class Car

Driver - Car : drives > Car *- Wheel : have 4 > Car -- Person : < owns

@enduml

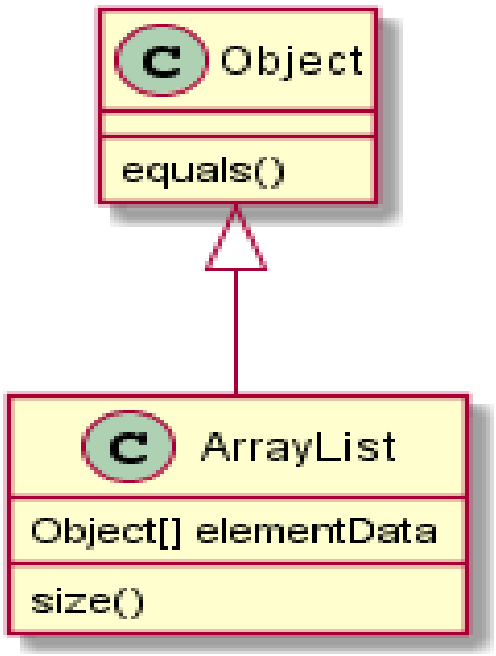


Добавление методов

Для объявления полей и методов вы можете использовать символ :, после которого указывается имя поля или метода.

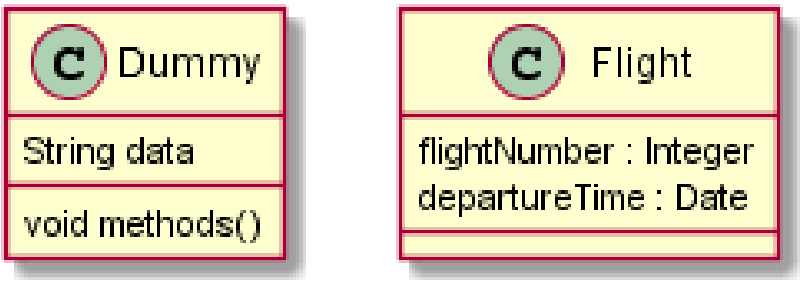
Для определения того, что вы указали метод или поле, система ищет скобки.

```
@startuml
Object <|-- ArrayList

Object : equals()
ArrayList : Object[] elementData ArrayList : size()

@enduml
```



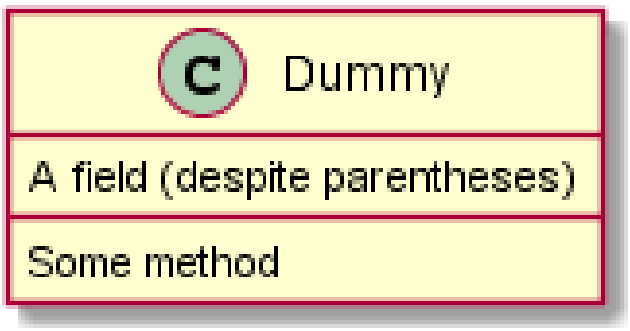
Также можно группировать все поля и методы между фигурными скобками {}. Синтаксис порядка описания типа/имени довольно гибок.

```
@startuml class Dummy {
String data void methods()
}

class Flight { flightNumber : Integer departureTime : Date
}
@enduml
```



You can use {field} and {method} modifiers to override default behaviour of the parser about fields and methods.

```
@startuml class Dummy {
{field} A field (despite parentheses)
{method} Some method
}

@enduml
```

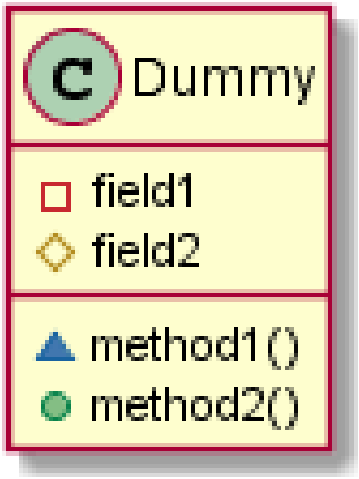


Указание видимости

Определяя методы и поля, вы можете использовать символы указания видимости, приведённые в таблице ниже:

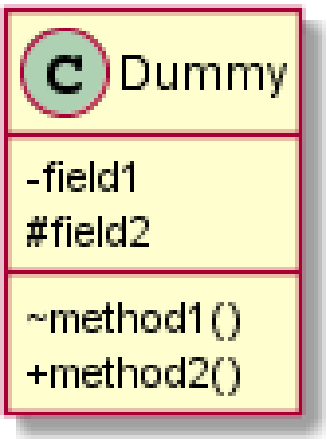
Character	Icon for field	Icon for method	Visibility
-			private
#			protected
~			package private
+			public

```
@startuml class Dummy {
-field1 #field2
~method1()
+method2()
}

@enduml
```



Убрать значки можно командой skinparam classAttributeIconSize 0:

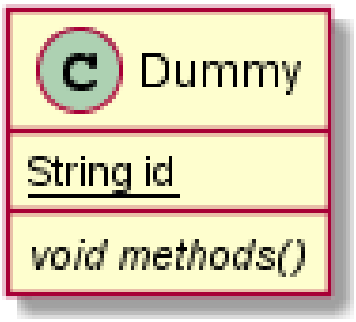
```
@startuml
skinparam classAttributeIconSize 0 class Dummy {
-field1 #field2
~method1()
+method2()
}

@enduml
```



Абстрактные и статические

Вы можете определить статические или абстрактные методы и поля используя модификаторы {static} и
{abstract} соответственно.
Эти модификаторы могут располагаться как в начале так и в конце строки. Вы так же можете использовать
{classifier} как замену для {static}.

```
@startuml class Dummy {
{static} String id
{abstract} void methods()
}
@enduml
```



Расширенное тело класса

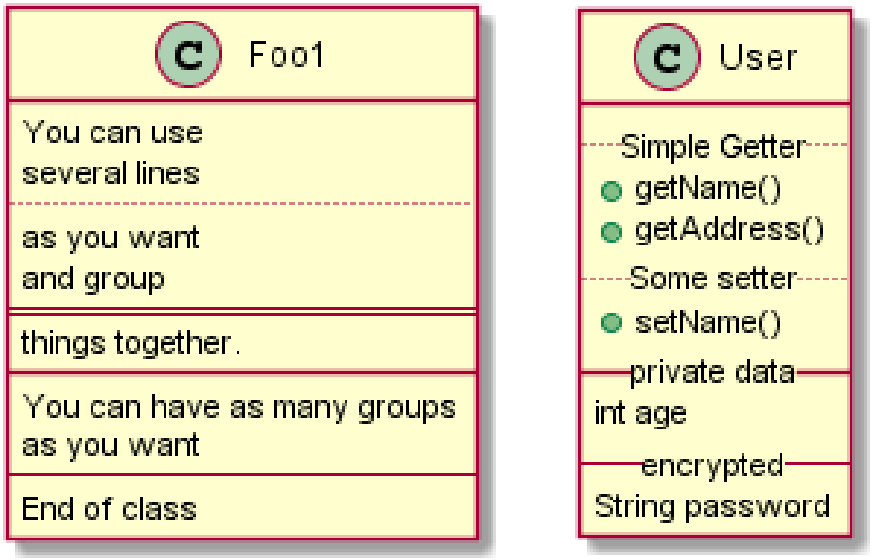
По умолчанию, методы и поля автоматически группируются PlantUML. Вы можете использовать разделители, чтобы определить собственный порядок полей и методов. Можно использовать следующие разделители:

-- .. == .

Вы также можете использовать заголовки внутри разделителей:

```
@startuml class Foo1 {
You can use several lines
..
as you want and group
==
things together.
—
```

```
You can have as many groups as you want
--
End of class
}

class User {
.. Simple Getter ..
+ getName()
+ getAddress()
.. Some setter ..
+ setName()
        private data
int age
-- encrypted -- String password
}

@enduml
```



Заметки и шаблоны

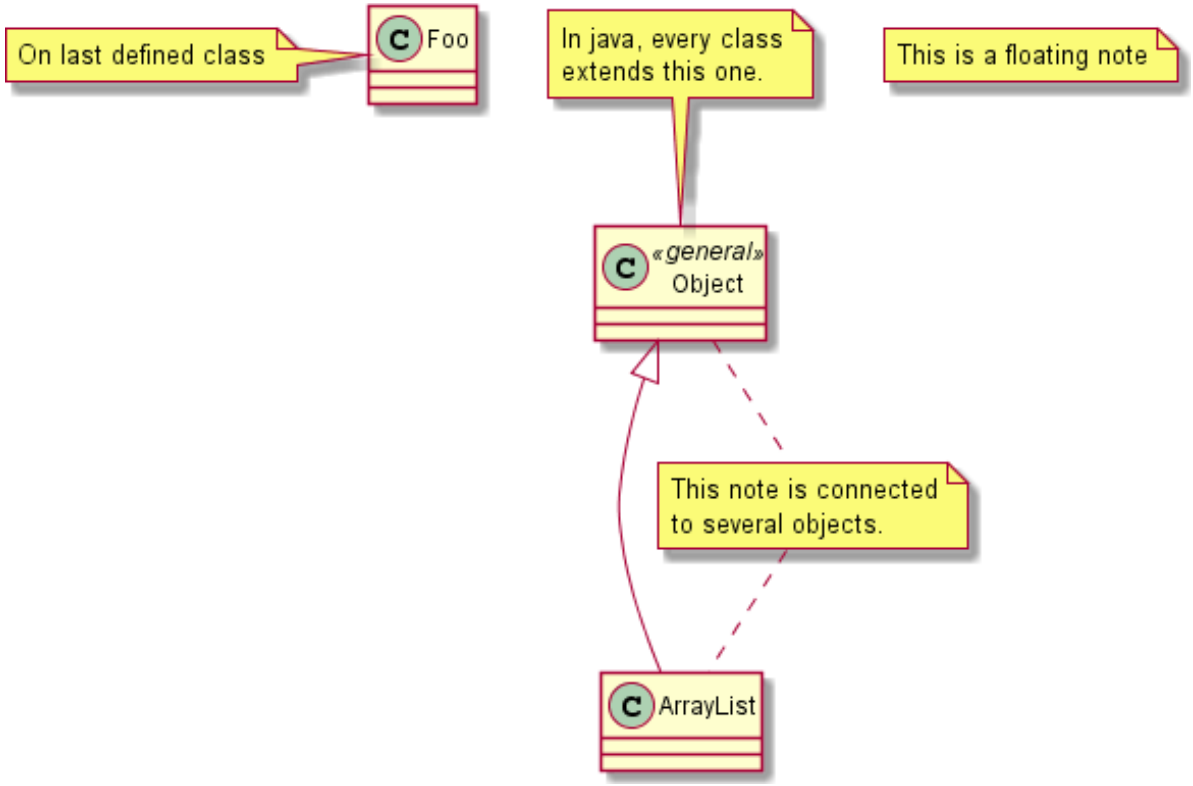
Шаблоны задаются ключевым словом class, << и >>.

Также вы можете создать заметку, используя ключевые слова note left of , "note right of , note top of ,
note bottom of".

Вы также можете добавить заметку к последнему определённому классу, используя note left, note right, note top, note bottom.

Ключевым словом note легко создать заметку без привязи, а после, используя символ .., привязать её к другим объектам.

```
@startuml
class Object << general >>
Object <|--- ArrayList
note top of Object : In java, every class\nextends this one. note "This is a floating note" as N1
note "This note is connected\nto several objects." as N2 Object .. N2
N2 .. ArrayList

class Foo
note left: On last defined class @enduml
```



Больше о заметках

Также допускается использование некоторых HTML-тегов, таких как:

<b>

<u>

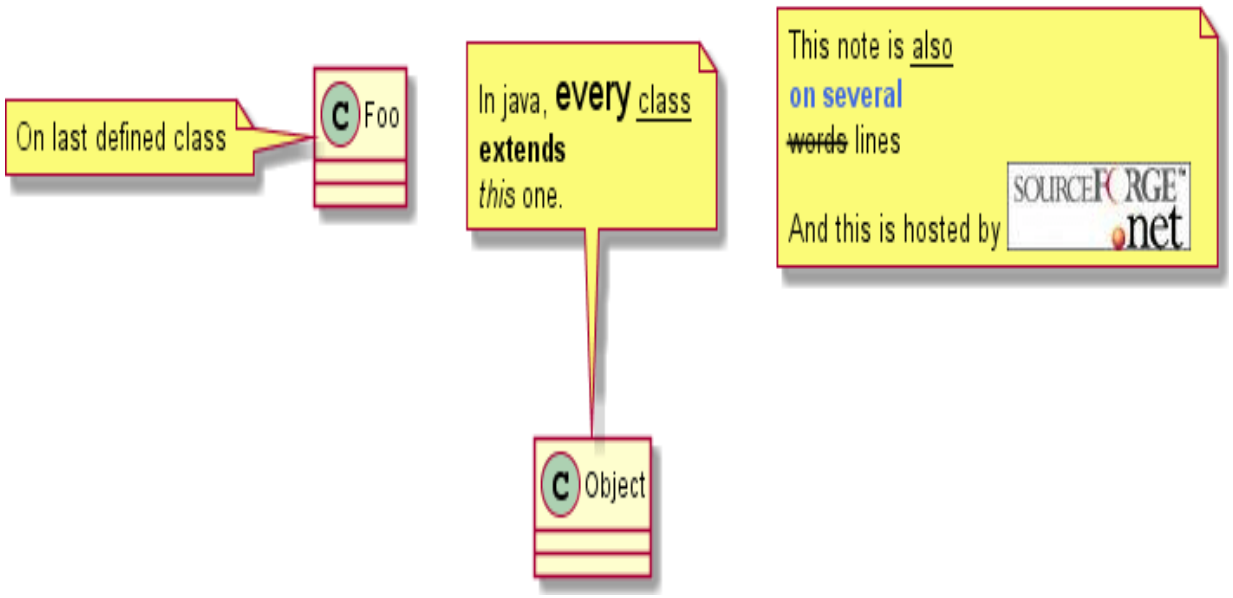
<i>

<s>, <del>, <strike>

<font color="#AAAAAA"> or <font color="colorName">
<color:#AAAAAA> or <color:colorName>
<size:nn> to change font size
<img src="file"> or <img:file>: the file must be accessible by the filesystem Заметка может быть из нескольких строк.

Можно определить заметку для класса, заданного последним, с помощью note left, note right, note top, note bottom.

```
@startuml

class Foo
note left: On last defined class

note top of Object
In java, <size:18>every</size> <u>class</u>
<b>extends</b>
<i>this</i> one. end note

note as N1
This note is <u>also</u>
<b><color:royalBlue>on several</color>
<s>words</s> lines
And this is hosted by <img:sourceforge.jpg> end note

@enduml
```



Заметки на связях

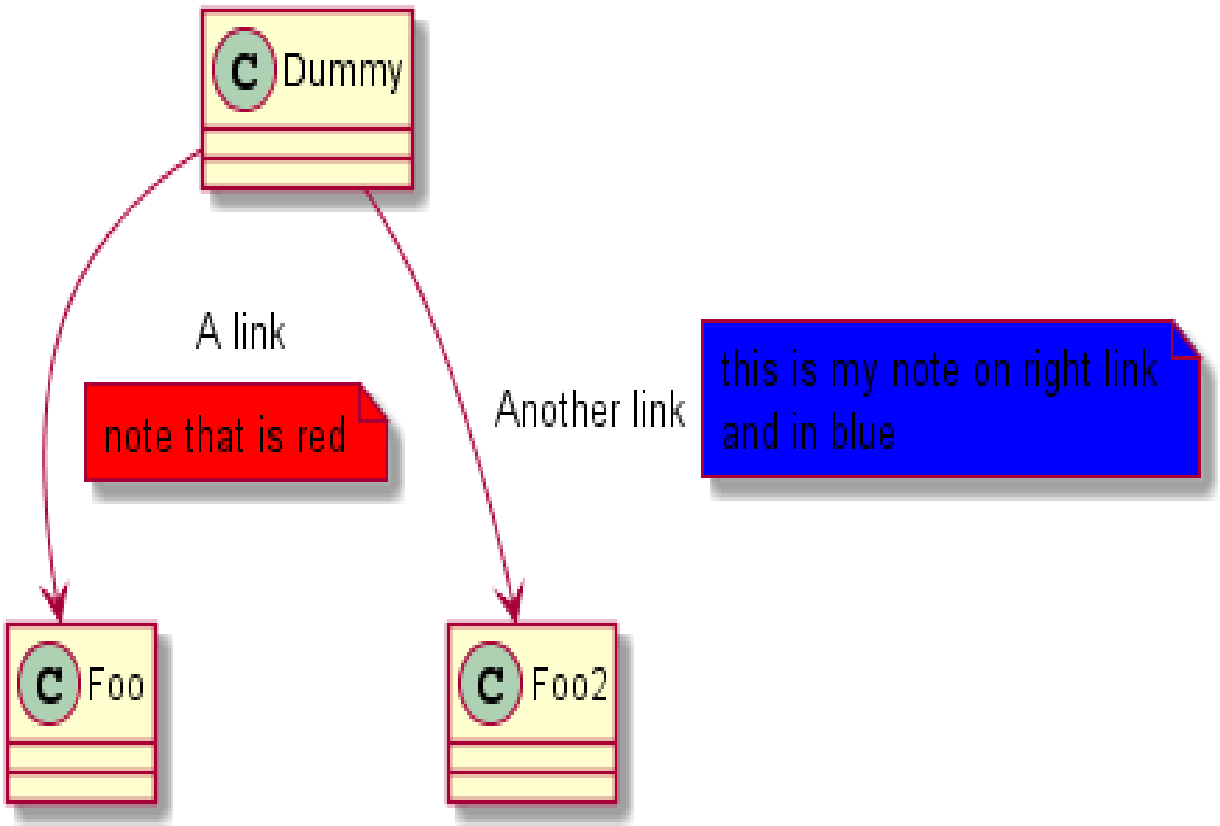
Возможно добавить заметку на связь, сразу после определения связи, используя note on link.

Вы также можете использовать note left on link, note right on link, note top on link, note bottom on link если вы хотите изменить относительную позицию заметки с надписью.

@startuml

class Dummy
Dummy --> Foo : A link
note on link #red: note that is red

Dummy --> Foo2 : Another link note right on link #blue this is my note on right link and in blue
end note
@enduml



Абстрактные классы и интерфейсы

Вы можете определить класс как абстрактный, используя ключевые слова abstract или abstract class. Классы будут нарисованы курсивом.

Вы также можете использовать ключевые слова interface, annotation и enum. @startuml

abstract class AbstractList abstract AbstractCollection interface List interface Collection

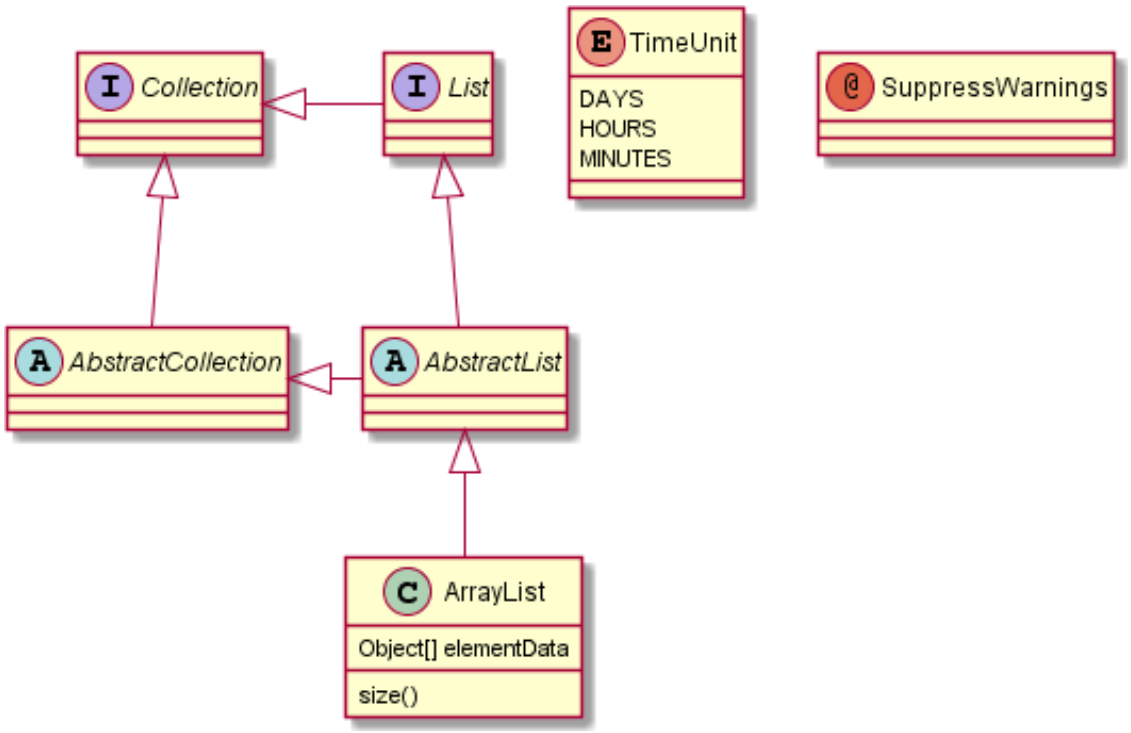
List <|-- AbstractList
Collection <|-- AbstractCollection

Collection <|- List AbstractCollection <|- AbstractList AbstractList <|-- ArrayList

class ArrayList { Object[] elementData size()
}

enum TimeUnit { DAYS
HOURS MINUTES
}

annotation SuppressWarnings @enduml



Использование не буквенных символов

Если вы хотите использовать не буквенные символы в названии класса (или другого объекта), вы можете использовать 2 способа :

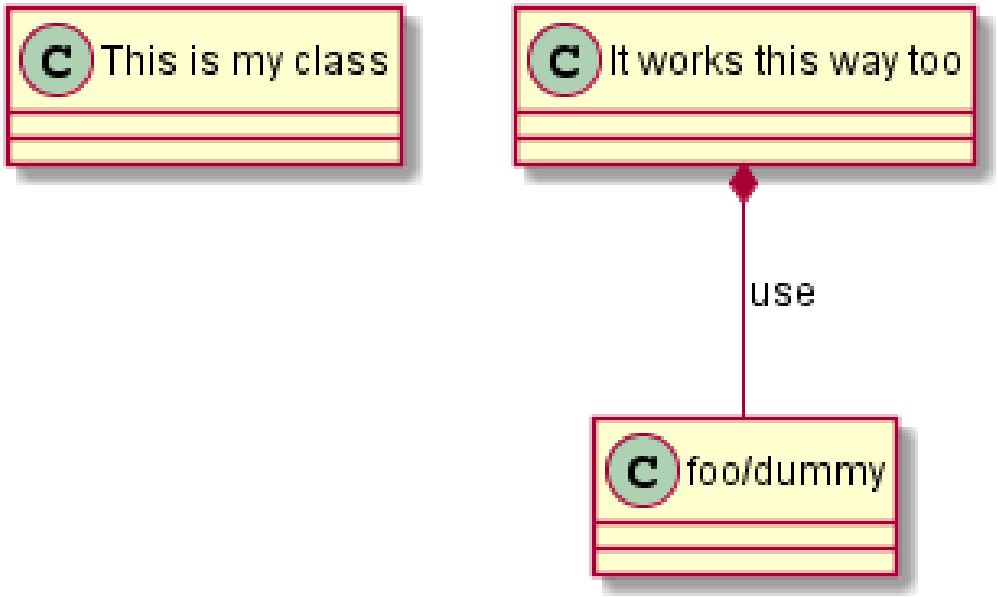
Использовать ключевое слово as в определении класса

Поставить кавычки "" вокруг имени класса

@startuml

class "This is my class" as class1 class class2 as "It works this way too"

class2 *-- "foo/dummy" : use @enduml



Скрытие атрибутов, методов...

Вы можете управлять видимостью классов с помощью команды hide/show.

Базовая команда это - hide empty members. Команда скроет атрибуты или методы, если они пусты. Вместо empty members, вы можете использовать:

empty fields или empty attributes для пустых полей,

empty methods для пустых методов,

fields или attributes, которые скроют поля, даже если они были описаны,

methods, которые скроют методы, даже если они были описаны,

members, которые скроют поля и методы, даже если они были описаны,

circle для круглых символов перед именем класса,

stereotype для шаблона.

Вы также можете указать ключевое слово, сразу за hide или show:

class для всех классов,

interface для всех интерфейсов,

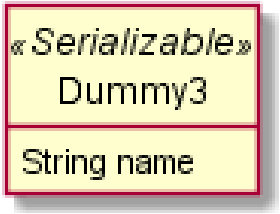
enum для всех перечислений,

<<foo1>> для классов, к которым применен шаблон с помощью foo1,

имя существующего названия класса.

Для определения большого набора, состоящего из правил и исключений, можно использовать несколько команд show/hide.

```
@startuml

class Dummy1 {
+myMethods()
}

class Dummy2 {
+hiddenMethod()
}

class Dummy3 <<Serializable>> { String name
}

hide members
hide <<Serializable>> circle show Dummy1 methods
show <<Serializable>> fields @enduml
```



Скрытие классов

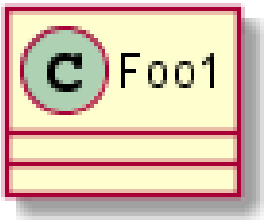
Вы также можете использовать команду show/hide, чтобы скрывать классы.

Это может быть полезно, если вы определяете большой !подключенный файл, и если вы хотите скрыть некоторые классы после включения.

@startuml

class Foo1 class Foo2

Foo2 *-- Foo1
hide Foo2 @enduml

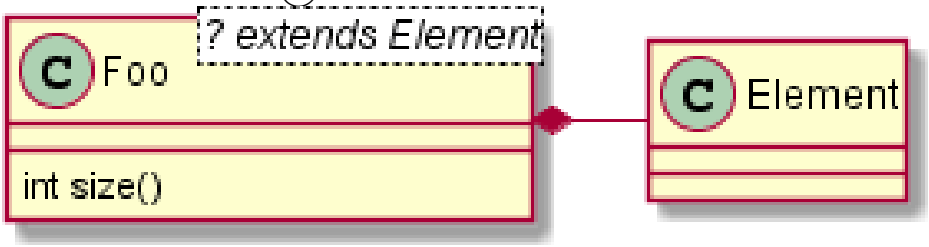


Использование дженериков

Вы также можете использовать скобки < и > чтобы указать на использование дженериков в классе.

@startuml

class Foo<? extends Element> { int size()
}
Foo *- Element @enduml



Вы можете отключить отрисовку этих элементов, используя команду skinparam genericDisplay old.

Определение метки

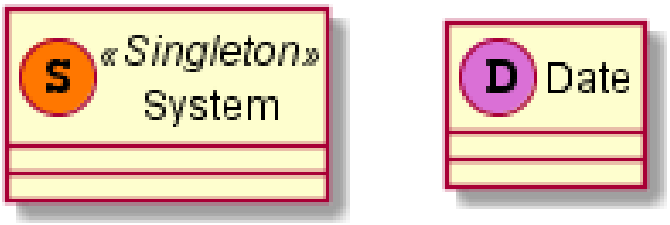
Обычно, метка с буквой (C, I, E or A) испольуется для классов, интерфейсов, перечисления и абстрактных классов.

Но также вы можете использовать свою собственную метку для класса, когда создаёте шаблон, добавляя одну букву и цвет, как в этом примере:

@startuml

class System << (S,#FF7700) Singleton >> class Date << (D,orchid) >>

@enduml



Пакеты

Вы можете определить пакет, используя ключевое слово package, с возможностью объявить ещё и цвет его фона, (используя html-код цвета или его имя).

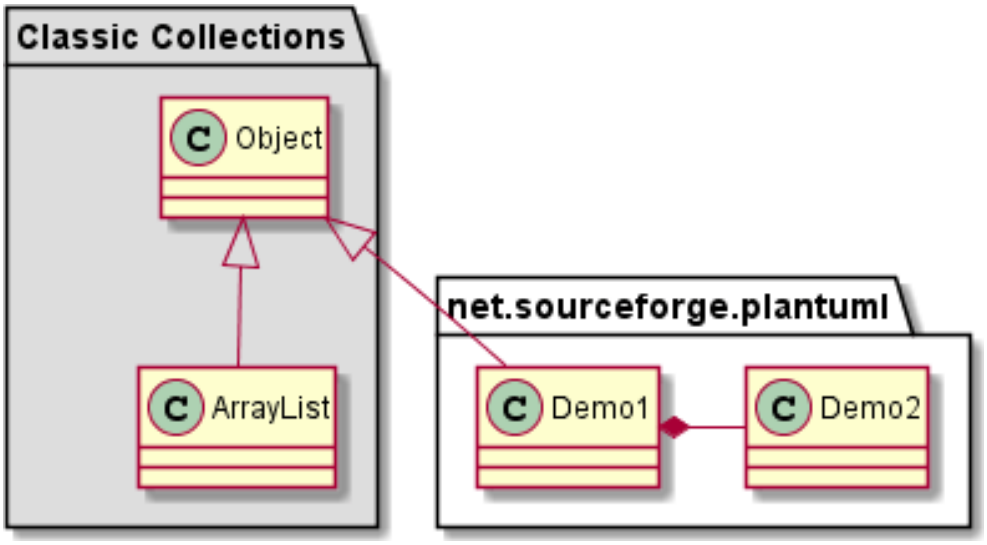
Обратите внимание, что определения пакета могут быть вложенными.

@startuml

package "Classic Collections" #DDDDDD { Object <|-- ArrayList
}

package net.sourceforge.plantuml { Object <|-- Demo1
Demo1 *- Demo2
}

@enduml



Стили пакетов

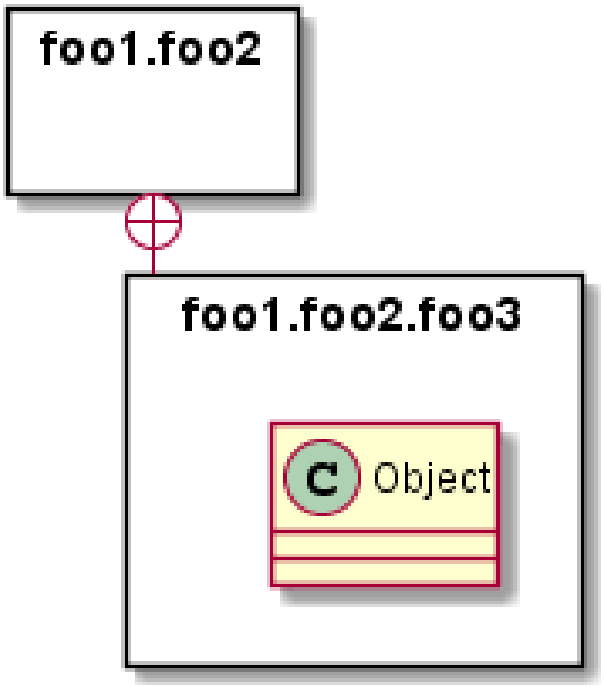
Доступны различные стили для пакетов.

Можно задать стили по умолчанию с помощью команды: skinparam packageStyle, или применить шаблоны на пакет:

```
@startuml scale 750 width
package foo1 <<Node>> { class Class1
}

package foo2 <<Rectangle>> { class Class2
}

package foo3 <<Folder>> { class Class3
}

package foo4 <<Frame>> { class Class4
}

package foo5 <<Cloud>> { class Class5
}

package foo6 <<Database>> { class Class6
}
@enduml
```

Вы также можете определить связи между пакетами, как в данном примере:

```
@startuml
skinparam packageStyle rectangle package foo1.foo2 {
}

package foo1.foo2.foo3 { class Object
}

foo1.foo2 +-- foo1.foo2.foo3 @enduml
```



Пространства имён

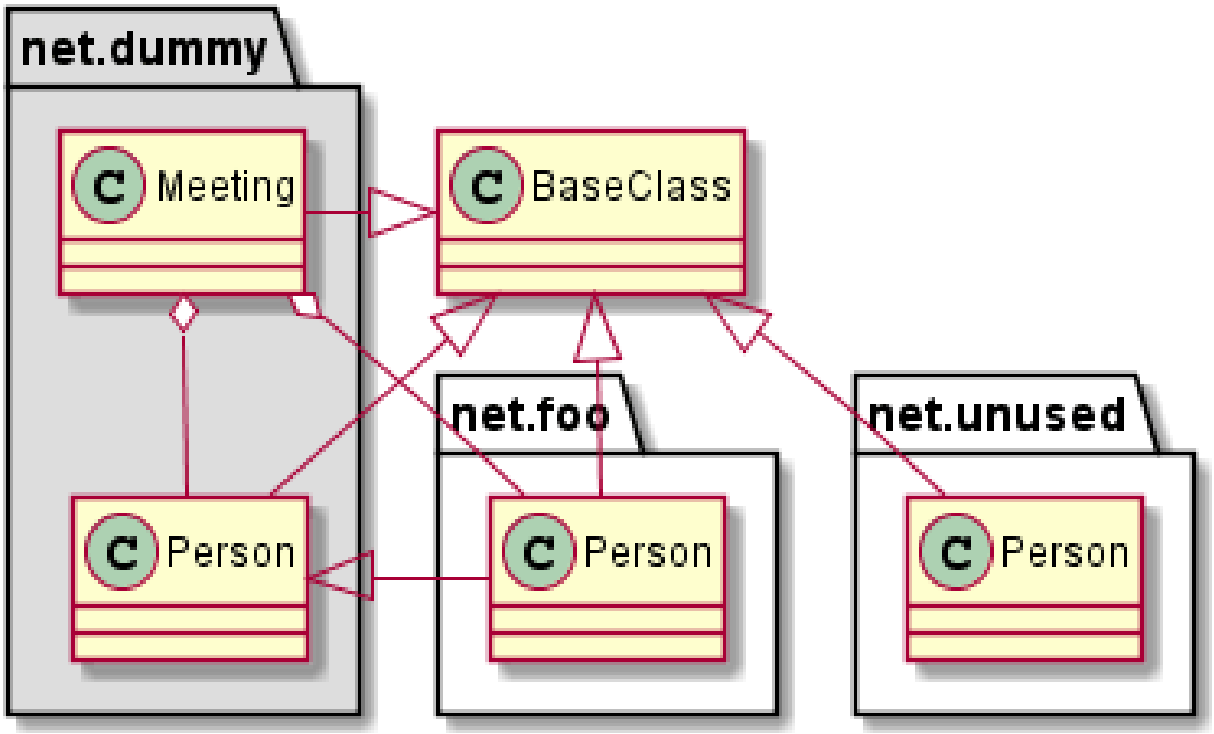
В пакетах, имя класса является уникальным идентификатором этого класса. Это значит, что у вас не может быть двух одноименных классов в разных блоках.

В этом случае, вам следует использовать пространства имен вместо пакетов.

Вы можете ссылаться на классы из других пространств имён по их полному определению. Классы из пространства имён по умолчанию определяются ведущей точкой.

Обратите внимание, что вы не обязаны явно создавать пространство имен: полностью определенный класс автоматически попадает в правильное пространство имен.

```
@startuml class BaseClass
namespace net.dummy #DDDDDD {
.BaseClass <|-- Person Meeting o-- Person

.BaseClass <|- Meeting
}

namespace net.foo { net.dummy.Person       <|- Person
.BaseClass <|-- Person

net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person @enduml
```



Автоматическое создание пространств имён

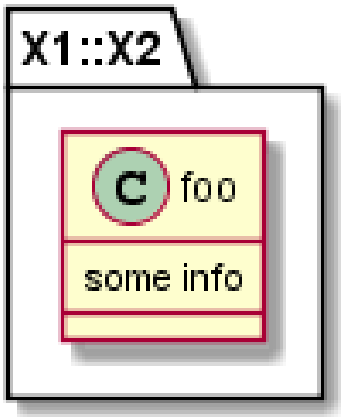
Вы также можете задать другой разделитель (не точку) используя команду : set namespaceSeparator

???.

@startuml

set namespaceSeparator :: class X1::X2::foo {
some info
}

@enduml



Вы можете отключить автоматическое создание пакетов используя команду set namespaceSeparator none.

@startuml

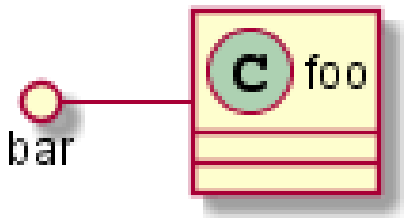
set namespaceSeparator none class X1.X2.foo {
some info
}

@enduml

Lollipop интерфейс

Вы также можете задать lollipops интерфейсы на классах, используя следующий синтаксис:

```
bar ()- foo
bar ()-- foo
foo -() bar @startuml
class foo bar ()- foo @enduml
```

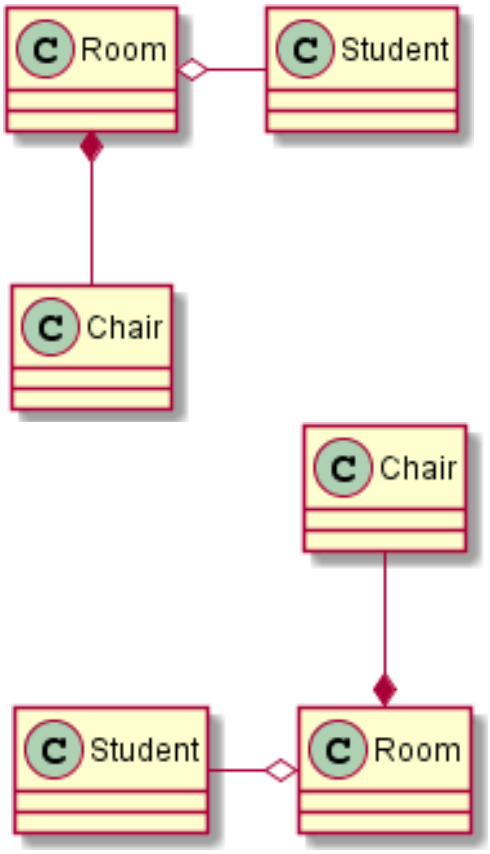


Изменение направления стрелок

По умолчанию, связи между классами имеют два тире --и вертикально ориентированны. Возможно создать горизонтальную связь, используя одно тире (or dot) вот так:

```
@startuml
Room o- Student Room *-- Chair @enduml
```

Вы можете изменить направление перевернув связь:



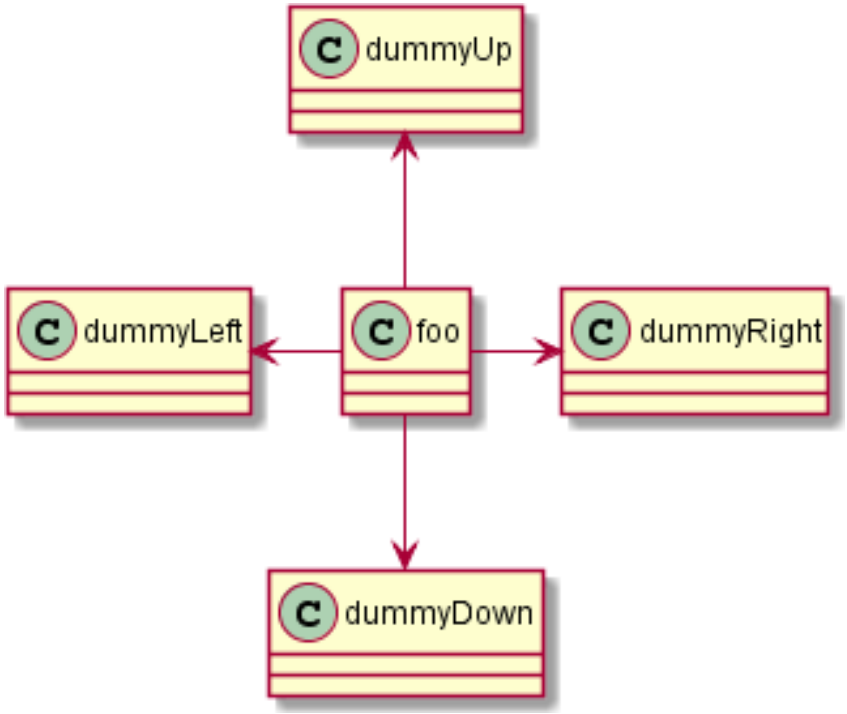
@startuml Student -o Room Chair --* Room @enduml

Также возможно изменять направление стрелок, добавляя ключевые слова left, right, up или down внутри стрелки:

@startuml

foo -left-> dummyLeft foo -right-> dummyRight foo -up-> dummyUp

foo -down-> dummyDown @enduml



Вы можете укоротить запись, используя только первую букву направления (например, -d- вместо -down-) или две первые буквы (-do-).

Заметьте, что вам не стоит пользоваться этой функциональностью без особой надобности: Graphviz обычно предоставляет хорошие результаты без дополнительной настройки.

Ассоциация классов

Вы можете задать ассоциацию класса после того, как была задана связь между двумя классами, как в примере:

@startuml class Student {

Name

}

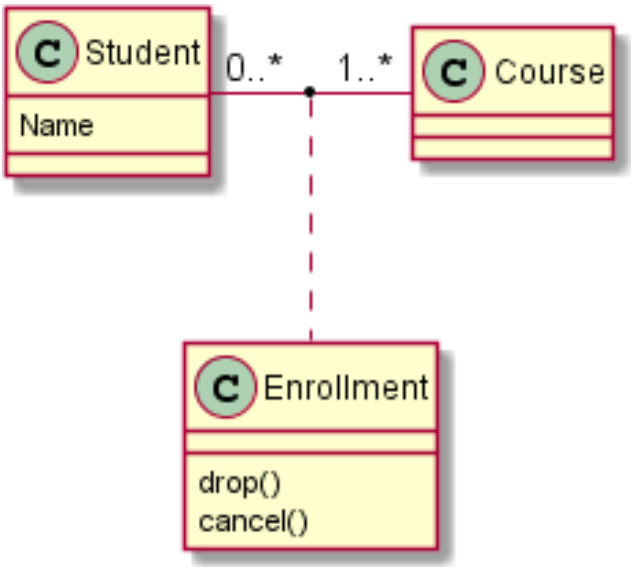
Student "0..*" - "1..*" Course (Student, Course) .. Enrollment

class Enrollment { drop()

cancel()

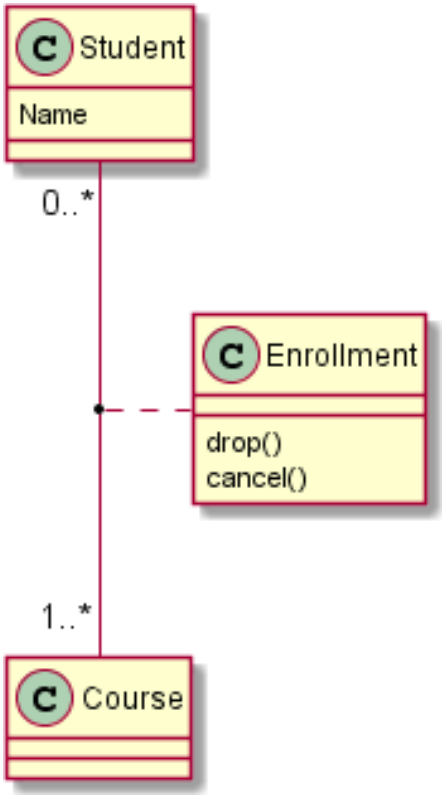
}

@enduml



Вы можете задать это в другом направлении:
@startuml class Student {
Name
}
Student "0..*" -- "1..*" Course (Student, Course) . Enrollment

class Enrollment { drop()
cancel()
}
@enduml



Skinparam

Вы можете использовать команду skinparam для изменения шрифтов и цветов диаграммы Вы можете использовать данную команду :

В определении диаграммы, как любую другую команду,

В подключенном файле,

В конфигурационном файле, указанном в командной строке в задании ANT.

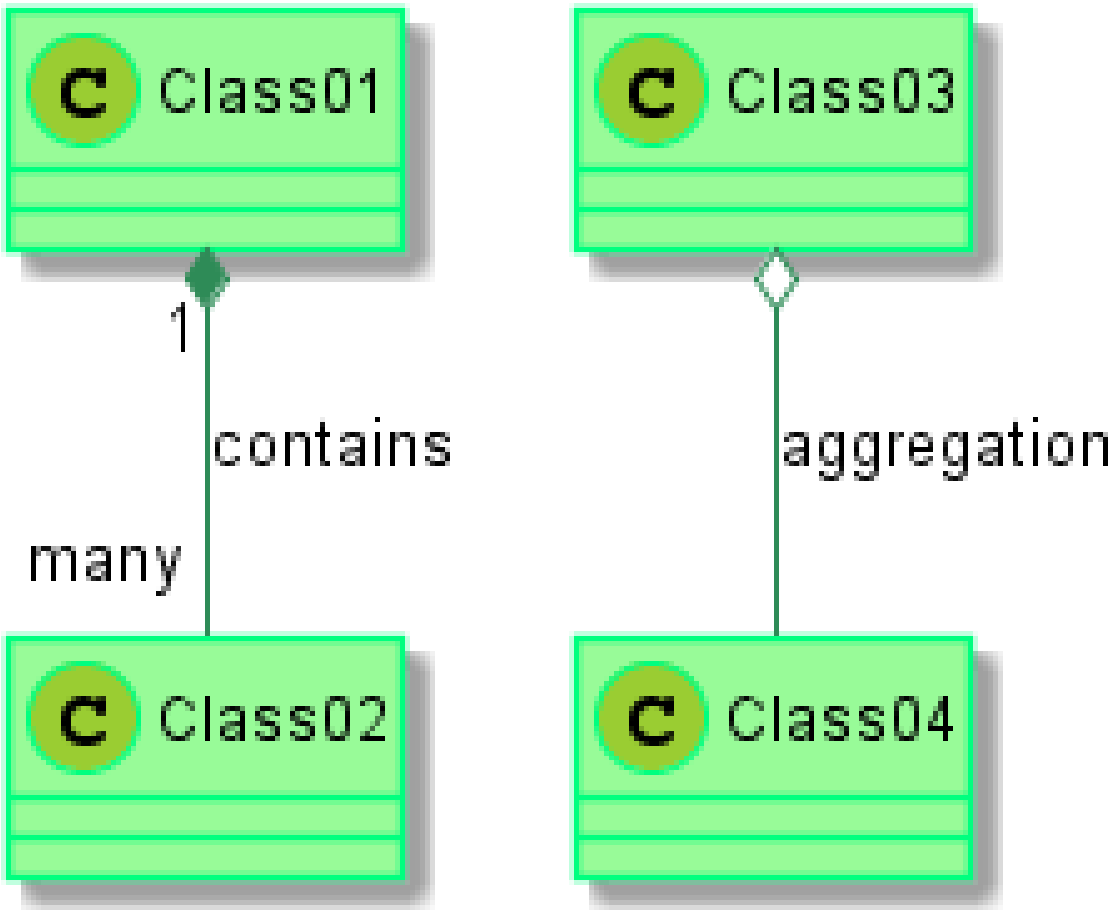
@startuml

skinparam class { BackgroundColor PaleGreen ArrowColor SeaGreen BorderColor SpringGreen

}

skinparam stereotypeCBackgroundColor YellowGreen Class01 "1" *-- "many" Class02 : contains Class03 o-- Class04 : aggregation

@enduml



Шаблоны со Skinparam

Вы можете задать цвет или шрифт для шаблонов классов.

@startuml

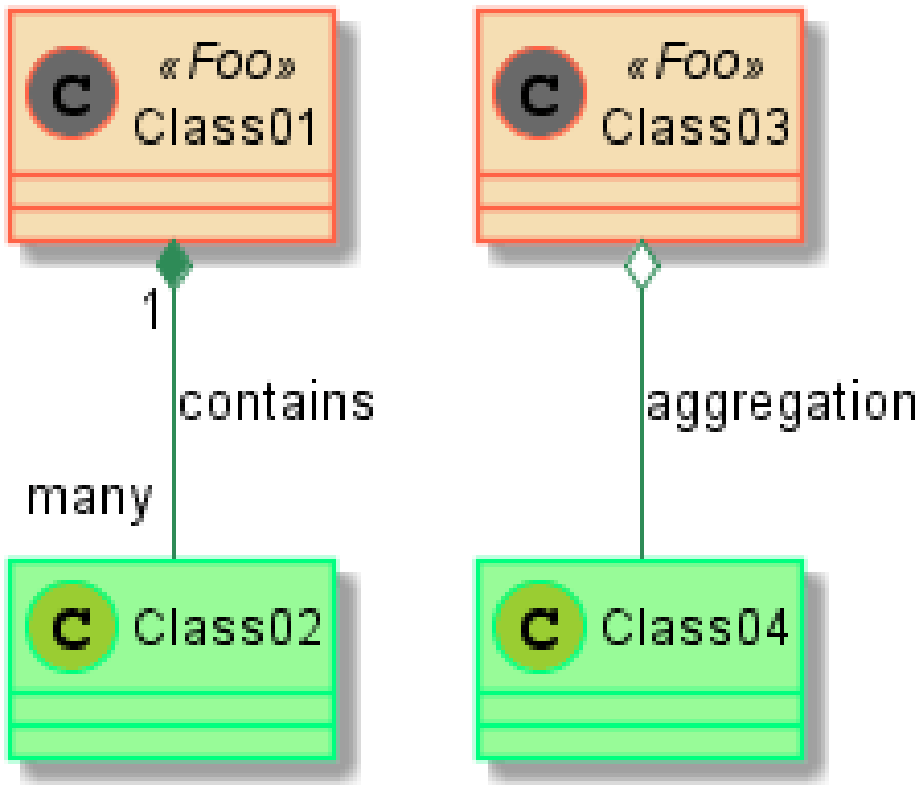
skinparam class { BackgroundColor PaleGreen ArrowColor SeaGreen BorderColor SpringGreen BackgroundColor<<Foo>> Wheat BorderColor<<Foo>> Tomato

}

skinparam stereotypeCBackgroundColor YellowGreen skinparam stereotypeCBackgroundColor<< Foo >> DimGray

Class01 <<Foo>> Class03 <<Foo>>

Class01 "1" *-- "many" Class02 : contains Class03 o-- Class04 : aggregation @enduml



Цветовой градиент

Можно объявить индивидуальный цвет для классов или примечаний, используя # обозначения. Можно использовать как стандартные названия цветов, так и RGB-код.

Так же возможно использование градиента для фона, используя следующие символы для разделения пары цветов:

|,
/,
\,
or -

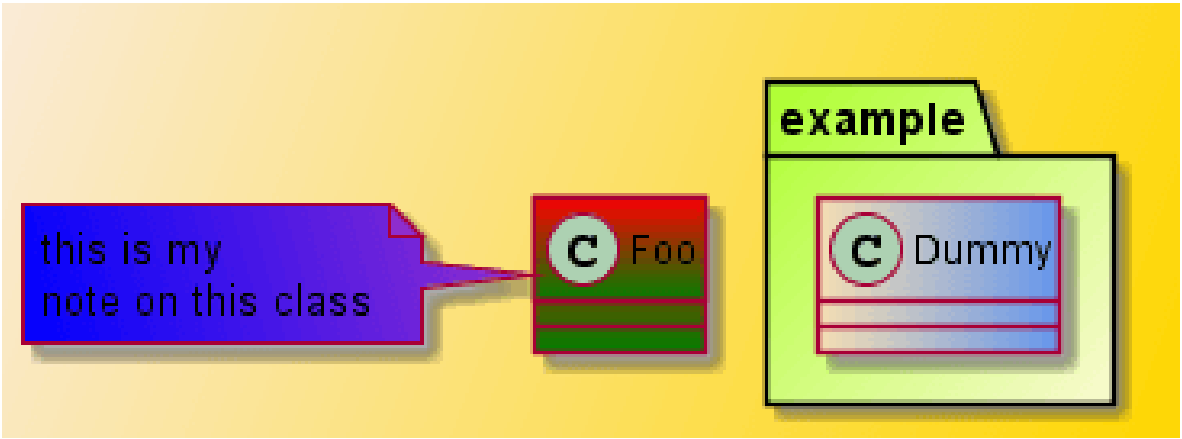
в зависимости от направления градиента Например так :
@startuml

skinparam backgroundcolor AntiqueWhite/Gold skinparam classBackgroundColor Wheat|CornflowerBlue

class Foo #red-green
note left of Foo #blue\9932CC this is my
note on this class end note

package example #GreenYellow/LightGoldenRodYellow { class Dummy
}

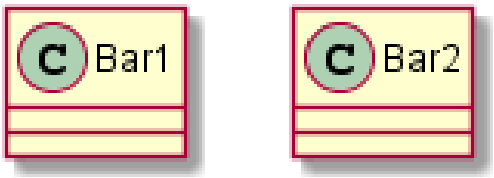
@enduml



Помощь в расположении классов

Sometimes, the default layout is not perfect...
You can use together keyword to group some classes together : the layout engine will try to group them (as if they were in the same package).
You can also use hidden links to force the layout.
@startuml

```
class Bar1 class Bar2 together {
class Together1 class Together2 class Together3
}
Together1 - Together2 Together2 - Together3 Together2 -[hidden]--> Bar1 Bar1 -[hidden]> Bar2

@enduml
```



Разделение больших файлов

Иногда могут получиться очень большие файлы изображений.

Вы можете использовать команду page (hpages)x(vpages) чтобы разделить создаваемое изображение на несколько файлов (страниц) :

hpages - это задание числа горизонтальных страниц, и vpages - это задание числа вертикальных страниц..

Здесь также можно использовать специфику skinparam настроек как цвета разделённых страниц, так и их границы (смотри пример).

```
@startuml
' Split into 4 pages page 2x2
```

skinparam pageMargin 10 skinparam pageExternalColor gray skinparam pageBorderColor black

class BaseClass

namespace net.dummy #DDDDDD {
.BaseClass <|-- Person Meeting o-- Person

.BaseClass <|- Meeting

}

namespace net.foo { net.dummy.Person <|- Person
.BaseClass <|-- Person

net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person @enduml

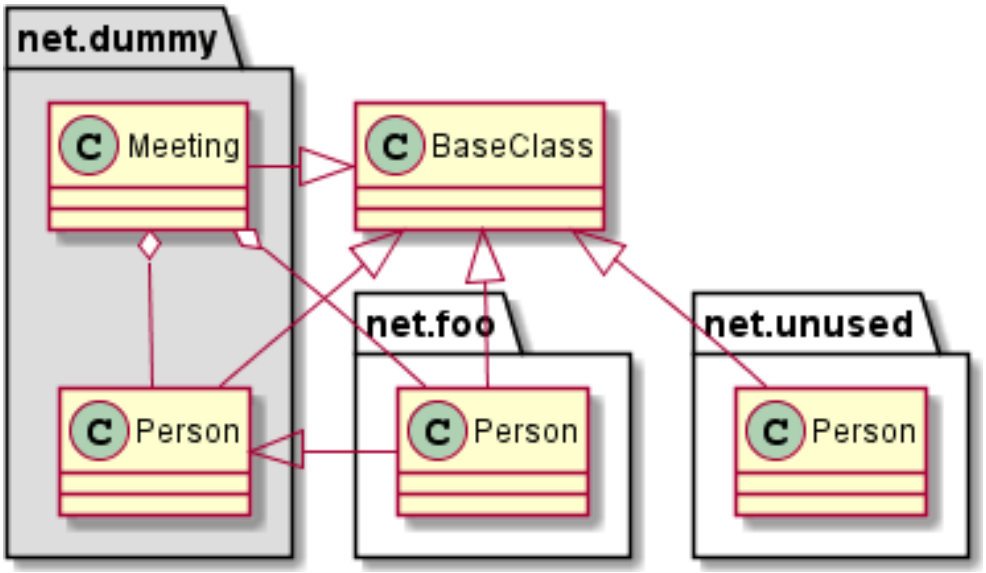


Диаграмма деятельности

Простая деятельность

Вы можете использовать (*) для начальных и конечных точек диаграммы деятельности.

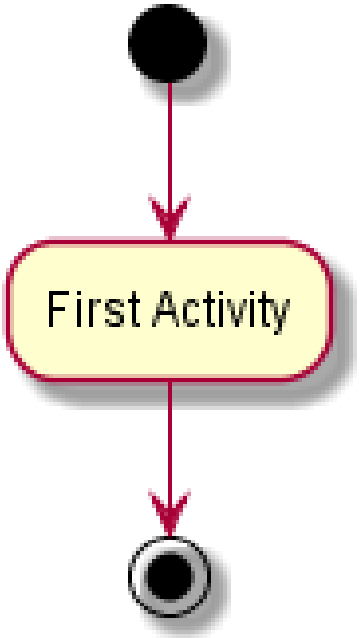
В некоторых случаях,вы можете использовать (*top) чтобы указать что начальная точка должна быть в верху диаграммы.

Используйте --> для стрелок.

@startuml

(*) --> "First Activity" "First Activity" --> (*)

@enduml



Метка на стрелках

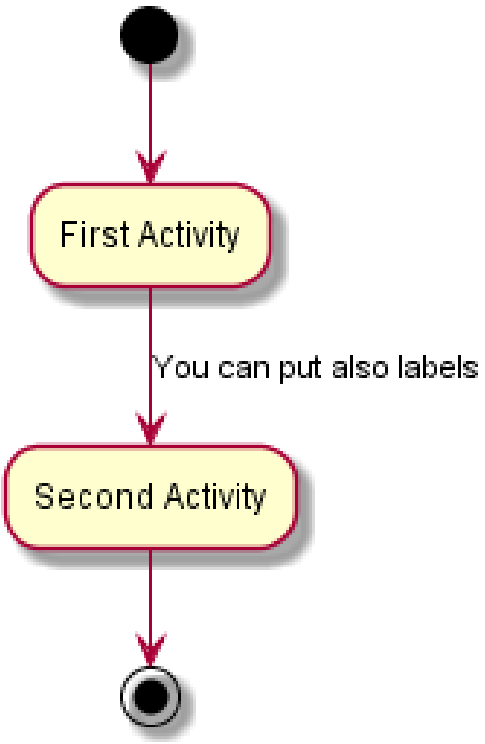
По умолчанию, стрелка начинается с последней использованной активности.

Вы можете пометить стрелку при помощи скобок [и] сразу после определения стрелки.

@startuml

(*) --> "First Activity"
-->[You can put also labels] "Second Activity"
--> (*)

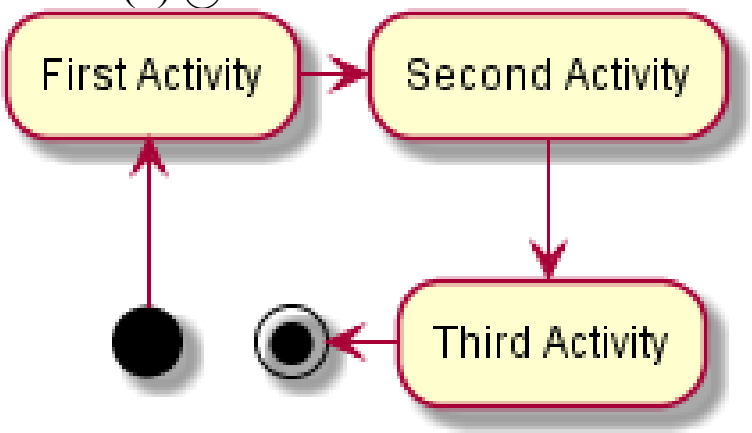
@enduml



Изменение направления стрелки

Вы можете использовать -> для горизонтальных стрелок. Возможно задать направление стрелки используя следующий синтаксис:

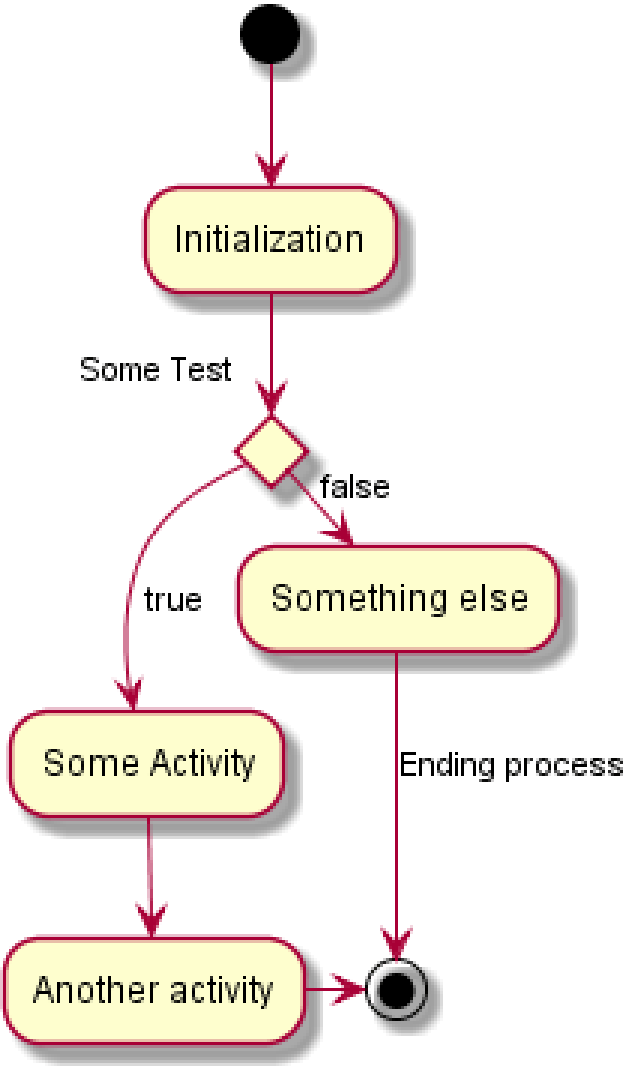
-down-> (default arrow)
-right-> or ->
-left->
-up-> @startuml
(*) -up-> "First Activity"
-right-> "Second Activity"
--> "Third Activity"
-left-> (*) @enduml



Ветвления

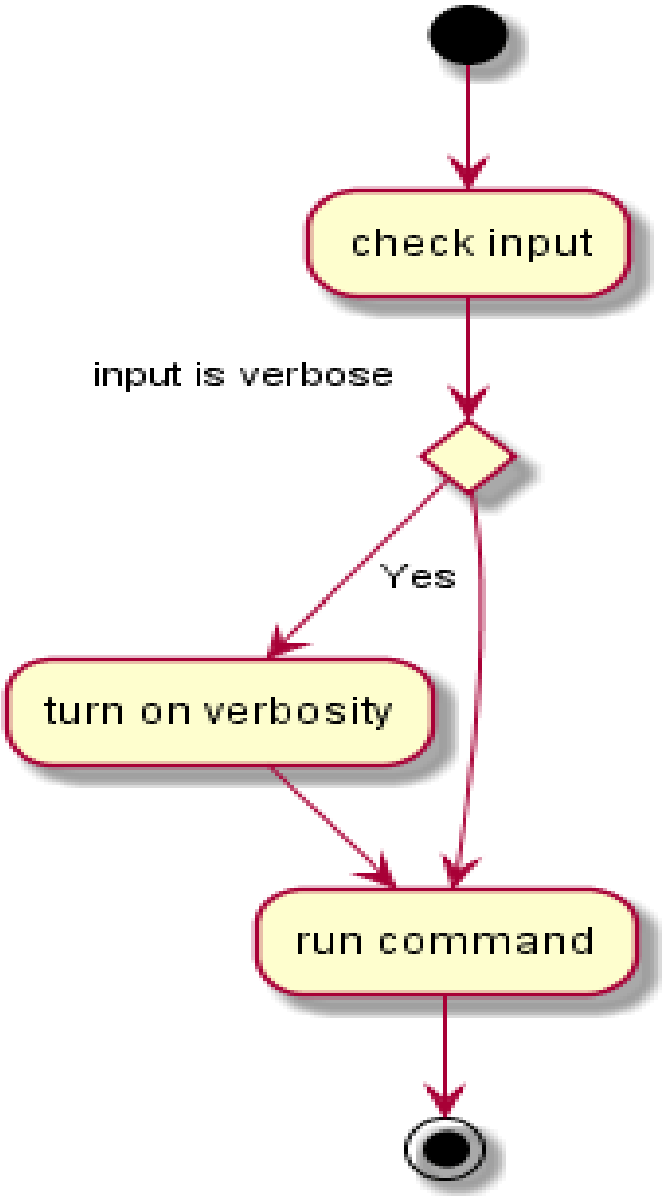
Вы можете использовать ключевые слова if/then/else чтобы определять ветки.

```
@startuml
(*) --> "Initialization"

if "Some Test" then
-->[true] "Some Activity"
--> "Another activity"
-right-> (*) else
->[false] "Something else"
-->[Ending process] (*) endif

@enduml
```



К сожалению, вам иногда придётся повторять ту же активность в тексте диаграммы:

```
@startuml
(*)    --> "check input"
If "input is verbose" then
--> [Yes] "turn on verbosity"
--> "run command" else
--> "run command" Endif
-->(*)
@enduml
```



Больше о ветках

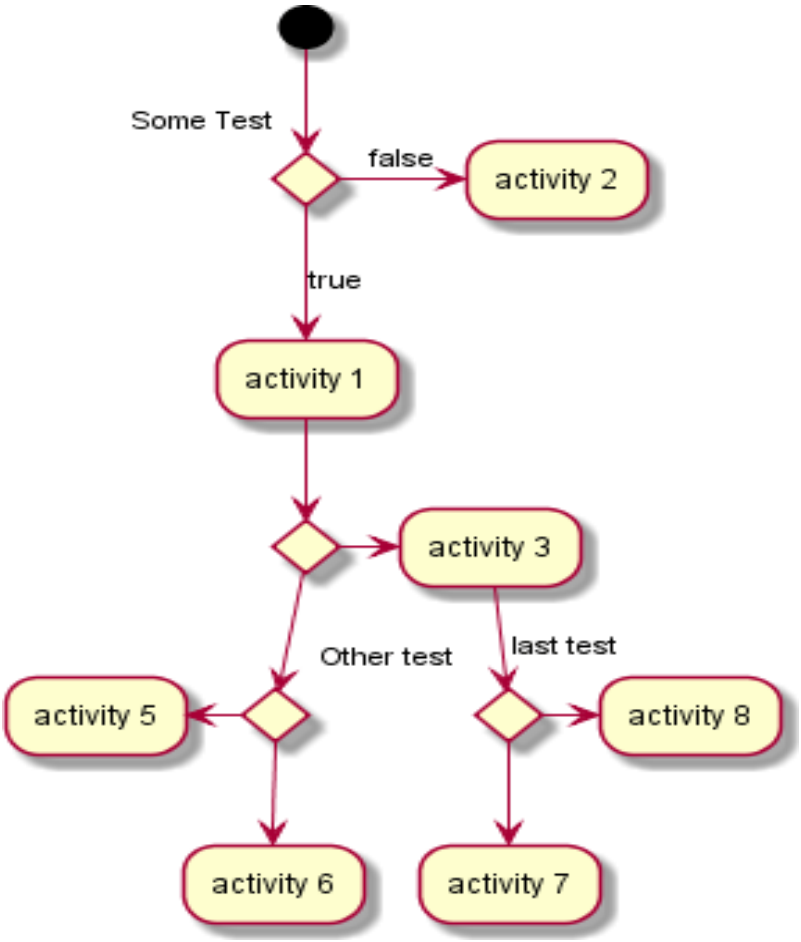
По умолчанию, ветка соединена к последней заданной активности, но возможно переопределить это и задать связь с помощью ключевого слова if.

Также возможно создавать вложенные ветки.

@startuml

(*) --> if "Some Test" then
-->[true] "activity 1" if "" then
-> "activity 3" as a3 else
if "Other test" then
-left-> "activity 5" else
--> "activity 6" endif
endif else
->[false] "activity 2" endif
a3 --> if "last test" then
--> "activity 7" else
-> "activity 8" endif

@enduml



Синхронизация

Вы можете использовать === code ===, чтобы отобразить барьеры синхронизации.

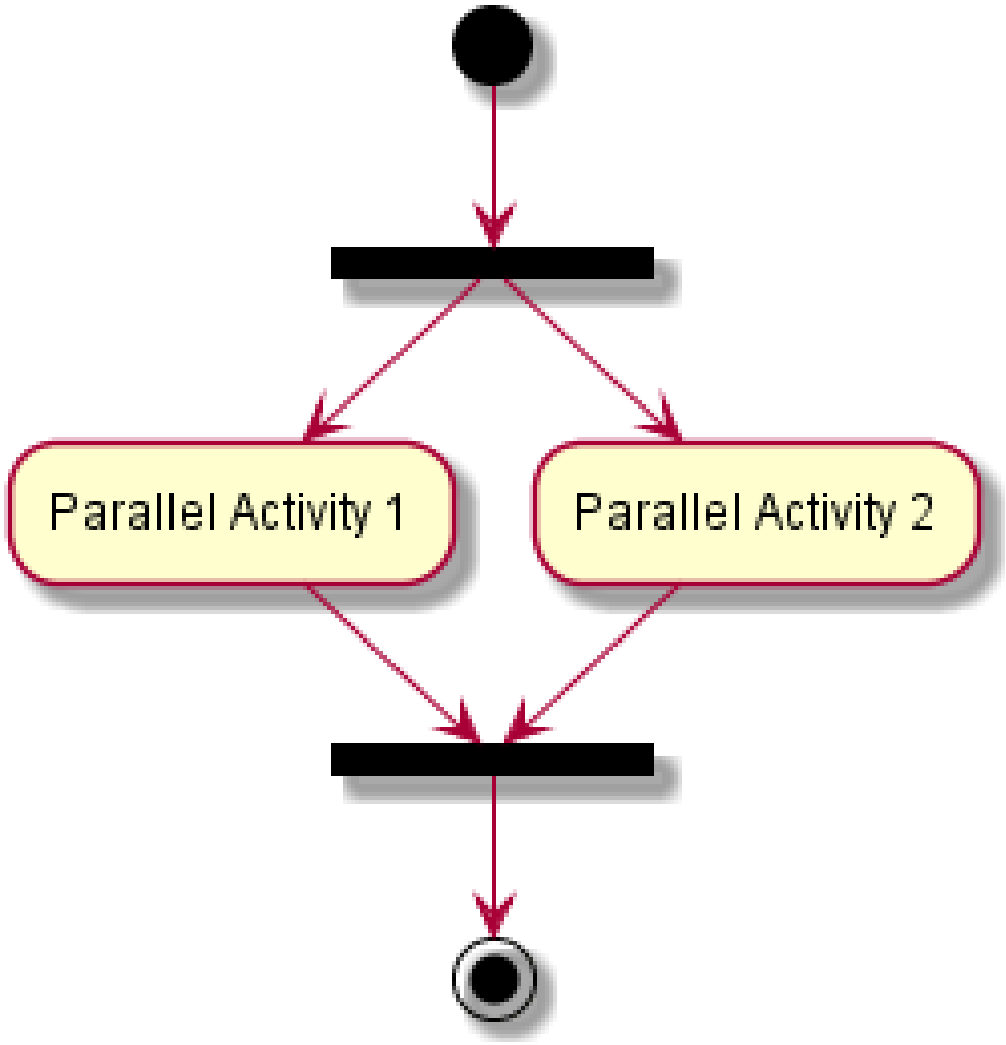
@startuml

(*) --> ===B1===
--> "Parallel Activity 1"
--> ===B2===

===B1=== --> "Parallel Activity 2"
--> ===B2===

--> (*)

@enduml



Длинное описание активности

Когда вы задаёте активность, вы можете разделить её описание на несколько линий. Вы также можете добавть \n в описание.

Вы также можете задать короткий код активности в помощью ключевого слова as. Этот код может быть использован позже в описании диаграммы.

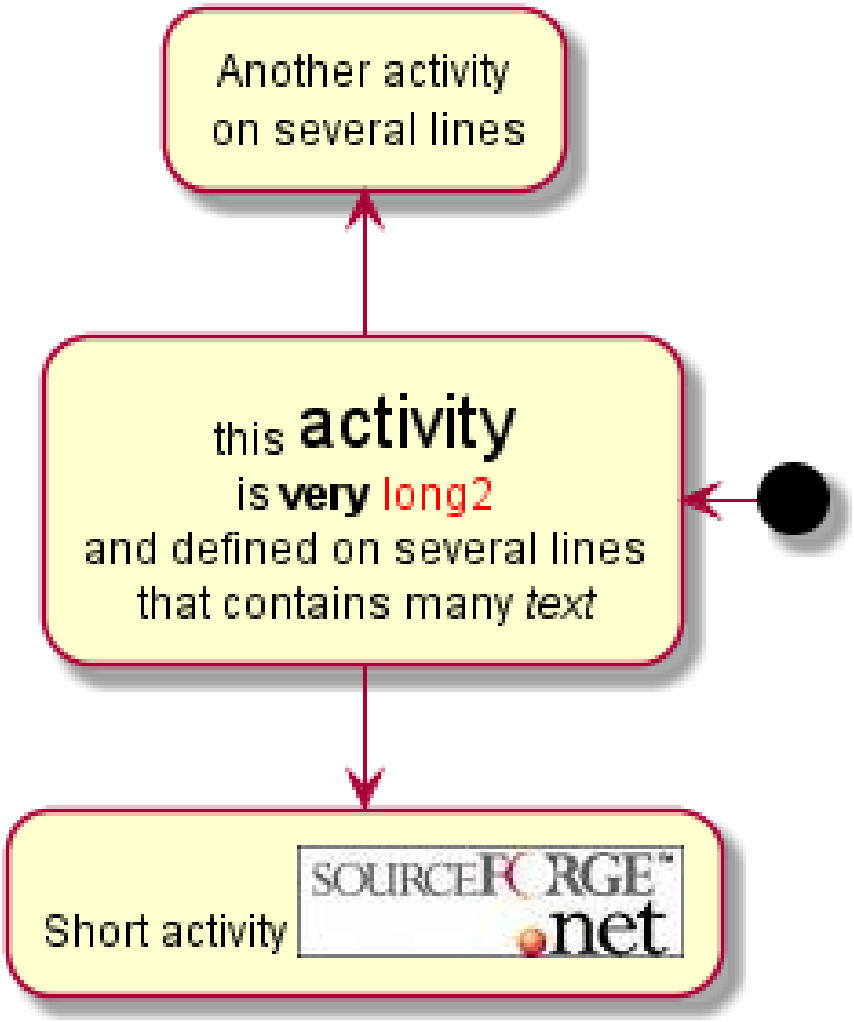
@startuml

(*) -left-> "this <size:20>activity</size> is very <color:red>long2</color> and defined on several lines

that contains many text" as A1

-up-> "Another activity\n on several lines"

A1 --> "Short activity <img:sourceforge.jpg>" @enduml



Заметки

Вы можете добавить заметки к активности используя команды note left, note right, note top or note bottom, Сразу после описания активности. к которой вы хотите прикрепить заметку.

Если вы хотите прикрепить заметку к точку начала, задайте метку в самом начале описания диаграммы. Вы также можете создать заметку на нескольких линиях, используя ключевое слово endnote. @startuml

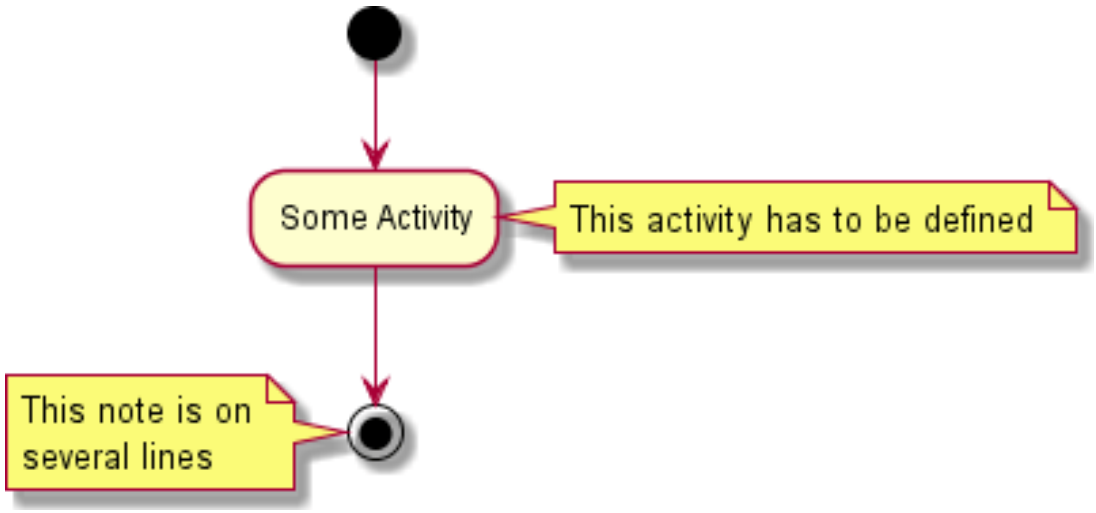
(*) --> "Some Activity"

note right: This activity has to be defined "Some Activity" --> (*)

note left

This note is on several lines end note

@enduml



Разделы

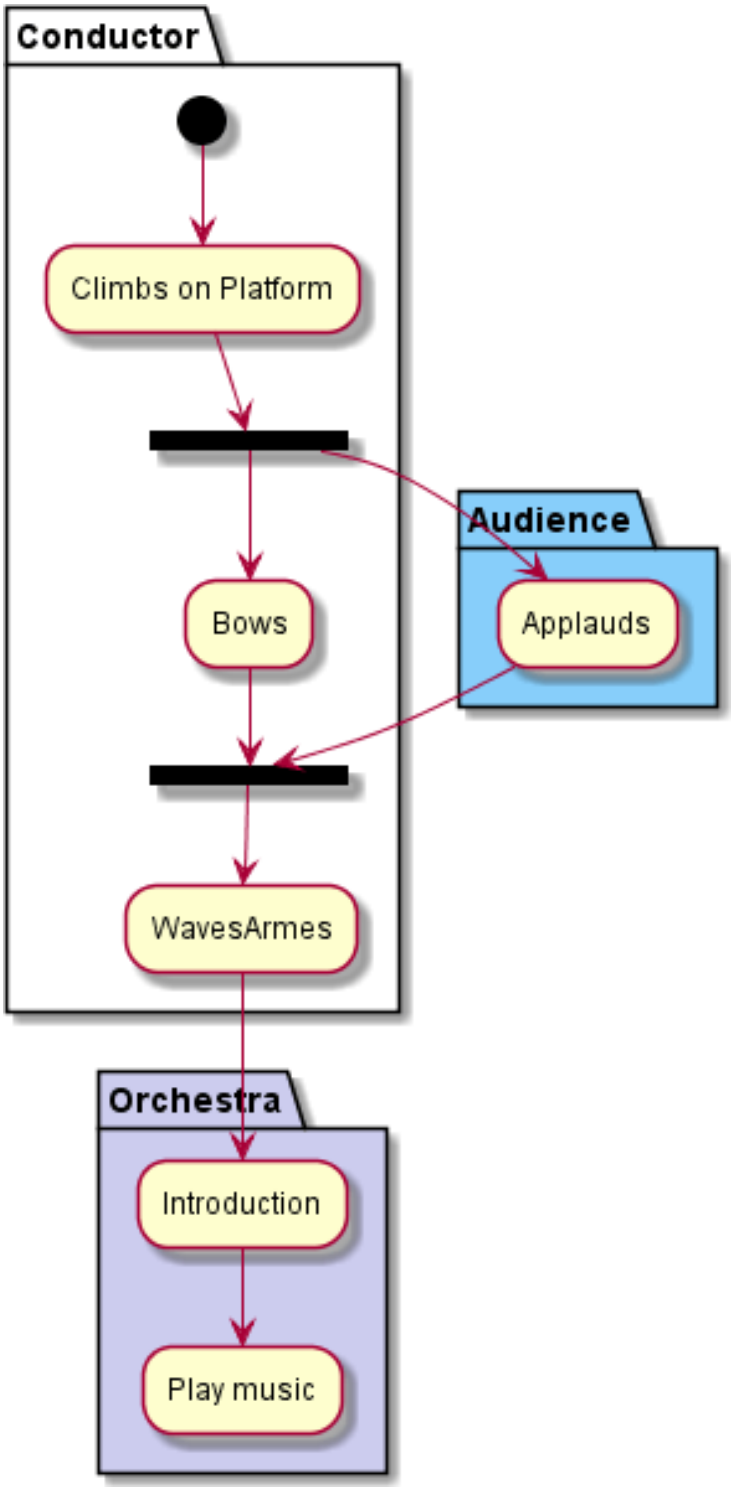
Вы можете задать раздел используя ключевое слово partition, и опционально задать цвет фона для своего раздела (Используя код цвета html или название цвета)

Когда вы задаёте активность, они автоматически попадают в последнюю заданную активность. Вы можете закрыть раздел используя закрывающую скобку }.

```
@startuml

partition Conductor {
(*) --> "Climbs on Platform"
--> === S1 ===
--> Bows
}

partition Audience #LightSkyBlue {
=== S1 === --> Applauds
}

partition Conductor { Bows --> === S2 ===
--> WavesArmes Applauds --> === S2 ===
}

partition Orchestra #CCCCEE { WavesArmes --> Introduction
```

--> "Play music"
}

@enduml



Skinparam

Вы можете использовать команду skinparam чтобы изменить цвет и шрифт рисования. Вы можете использоват команду :

В определении диаграммы, как любую другую команду,

В подключаемом файле,

В конфигурационном файле, подставленный в командной строке ANT задания. Вы можете задать определённый цвет и шрифт для активностей с шаблоном. @startuml

skinparam backgroundColor #AAFFFF skinparam activity {

StartColor red BarColor SaddleBrown EndColor Silver BackgroundColor Peru

BackgroundColor<< Begin >> Olive BorderColor Peru

FontName Impact

}

(*) --> "Climbs on Platform" << Begin >>

--> === S1 ===

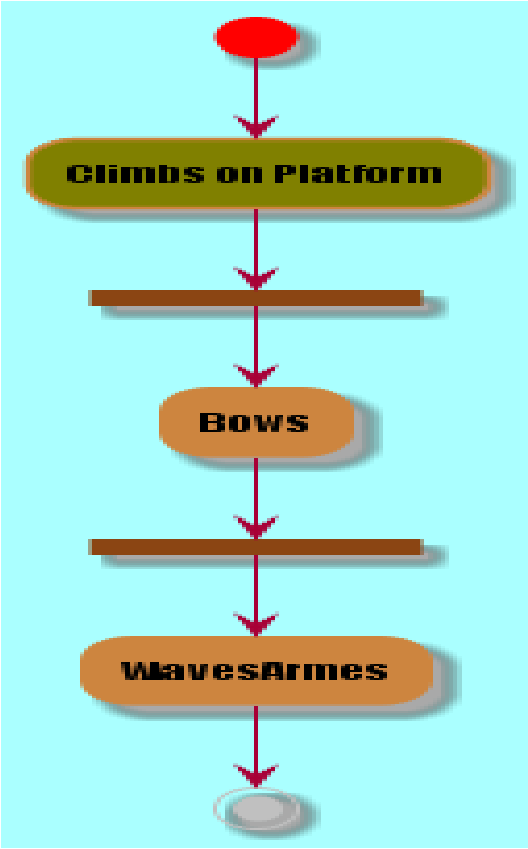
--> Bows

--> === S2 ===

--> WavesArmes

--> (*)

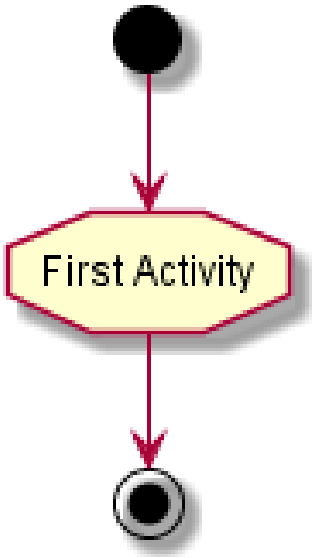
@enduml



Восьмиугольник

Вы можете изменить форму активностей на восьмиугольник, используя команду skinparam activityShape octagon.

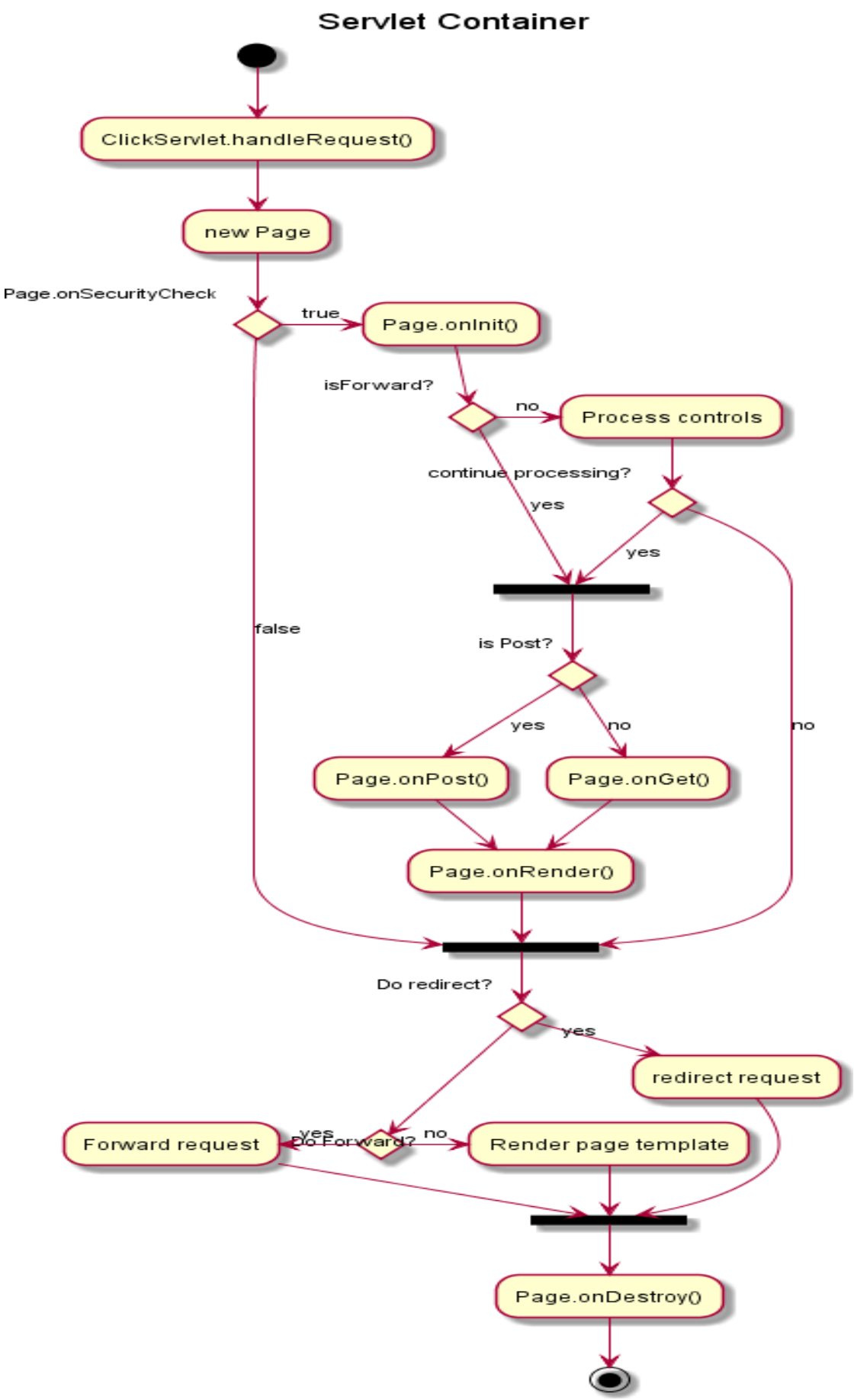
```
@startuml
'Default is skinparam activityShape roundBox skinparam activityShape octagon

(*) --> "First Activity" "First Activity" --> (*)

@enduml
```



Полноценный пример

```
@startuml
title Servlet Container
(*) --> "ClickServlet.handleRequest()"
--> "new Page"

if "Page.onSecurityCheck" then
->[true] "Page.onInit()"

if "isForward?" then
->[no] "Process controls"
```

```
if "continue processing?" then
-->[yes] ===RENDERING===
else
-->[no] ===REDIRECT_CHECK===
endif

else
-->[yes] ===RENDERING===
endif

if "is Post?" then
-->[yes] "Page.onPost()"
--> "Page.onRender()" as render
--> ===REDIRECT_CHECK===
else
-->[no] "Page.onGet()"
--> render endif

else
-->[false] ===REDIRECT_CHECK===
endif

if "Do redirect?" then
->[yes] "redirect request"
--> ==BEFORE_DESTROY===
else
if "Do Forward?" then
-left->[yes] "Forward request"
--> ==BEFORE_DESTROY===
else
-right->[no] "Render page template"
--> ==BEFORE_DESTROY===
endif endif

--> "Page.onDestroy()"
-->(*)

@enduml
```


Servlet Container
ClickServlet.handleRequest()
new Page
Page.onSecurityCheck
true
Page.onInit()
isForward?
no
Process controls
continue processing?
yes
yes
false
is Post?
yes
no
no
Page.onPost()
Page.onGet()
Page.onRender()
Do redirect?
yes
redirect request
Forward request
yes
Do Forward?
no
Render page template
Page.onDestroy()

Платформы для проектирования и реализации баз данных

Современные системы анализа и проектирования могут быть разделены на две большие категории. Первую составляют CASE-системы (как независимые (upper CASE), так и интегрированные с СУБД), обе системы анализа и проектированияечивающие проектирование БД и приложений в комплексе с интегрированными средствами разработки приложений "клиент-сервер" (например, Westmount I-CASE+Uniface, Designer/2000+Developer/2000). Их основное достоинство заключается в том, что они позволяют разрабатывать всю ИС целиком (функциональные системы анализа и проектирования спецификации, логику процессов, интерфейс с пользователем и базу данных), оставаясь в одной технологической среде. Инструменты этой категории, как правило, обладают существенной сложностью, широкой сферой применения и высокой гибкостью.

Вторую категорию составляют собственно средства проектирования БД, реализующие ту или иную методологию, как правило, "сущность-связь" ("entity-relationship") и рассматриваемые в комплексе со средствами разработки приложений. К средствам этой категории можно отнести такие, как SILVERRUN+JAM, ERwin/ERX+PowerBuilder и др.

Помимо указанных категорий, системы анализа и проектирования можно классифицировать по следующим признакам:

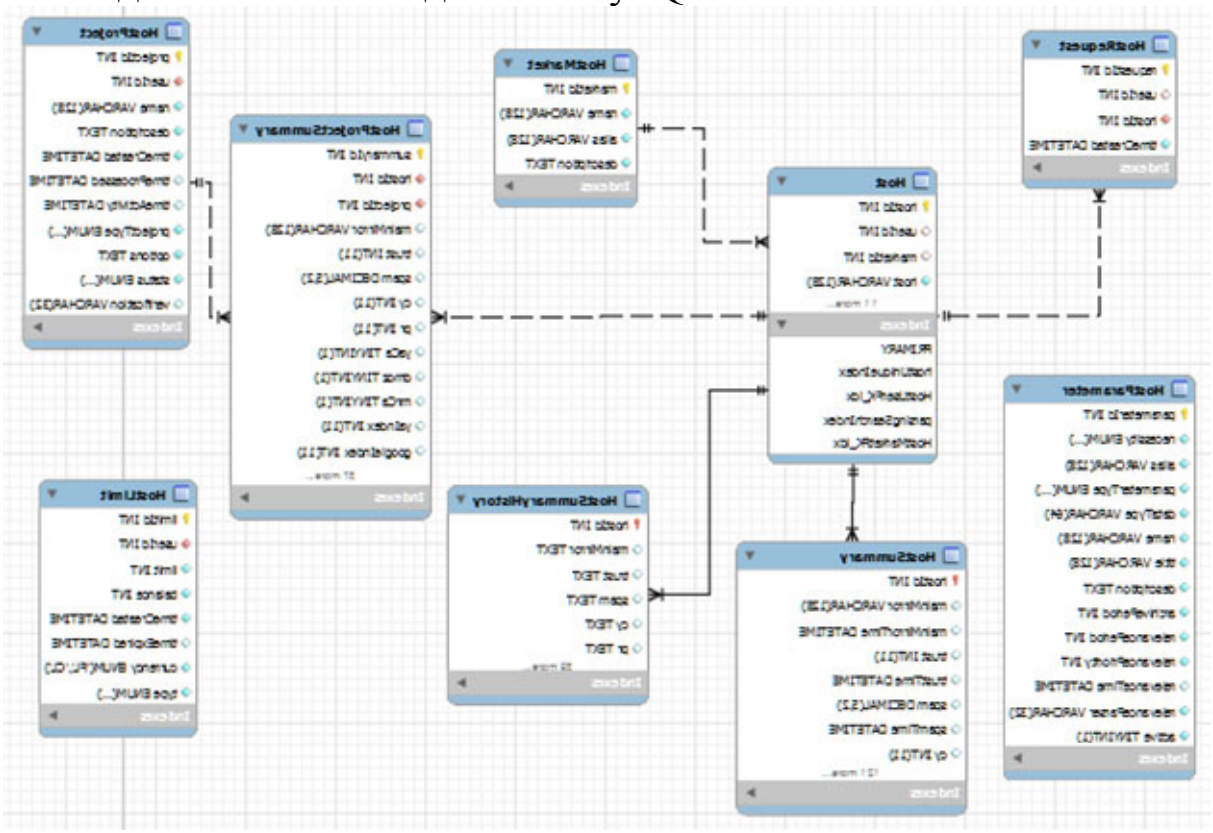
- степени интегрированности: (отдельные локальные средства, набор частично интегрированных средств, охватывающих большинство этапов жизненного цикла ИС и полностью интегрированные средства, связанные общей базой проектных данных - репозиторием);
- применяемым методологиям и моделям систем и БД;
- степени интегрированности с СУБД;
- степени открытости;
- доступным платформам.

В разряд системы анализа и проектирования попадают как относительно дешевые системы для персональных компьютеров (ПК) с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред. Так, современный рынок программных средств насчитывает около 300 различных CASE-систем, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами. Применение системы анализа и проектирования требует от потенциальных пользователей системы анализа и проектирования специальной подготовки и обучения. Опыт показывает, что внедрение системы анализа и проектирования осуществляется медленно, однако по

мере приобретения практических навыков и общей культуры проектирования эффективность применения этих средств резко возрастает, причем наибольшая потребность в использовании систем проектирования испытывается на начальных этапах разработки, а именно на этапах анализа и спецификации требований. Это объясняется тем, что цена ошибок, допущенных на начальных этапах, на несколько порядков превышает цену ошибок, выявленных на более поздних этапах разработки.

Практическое задание

MySQL Workbench — инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL.

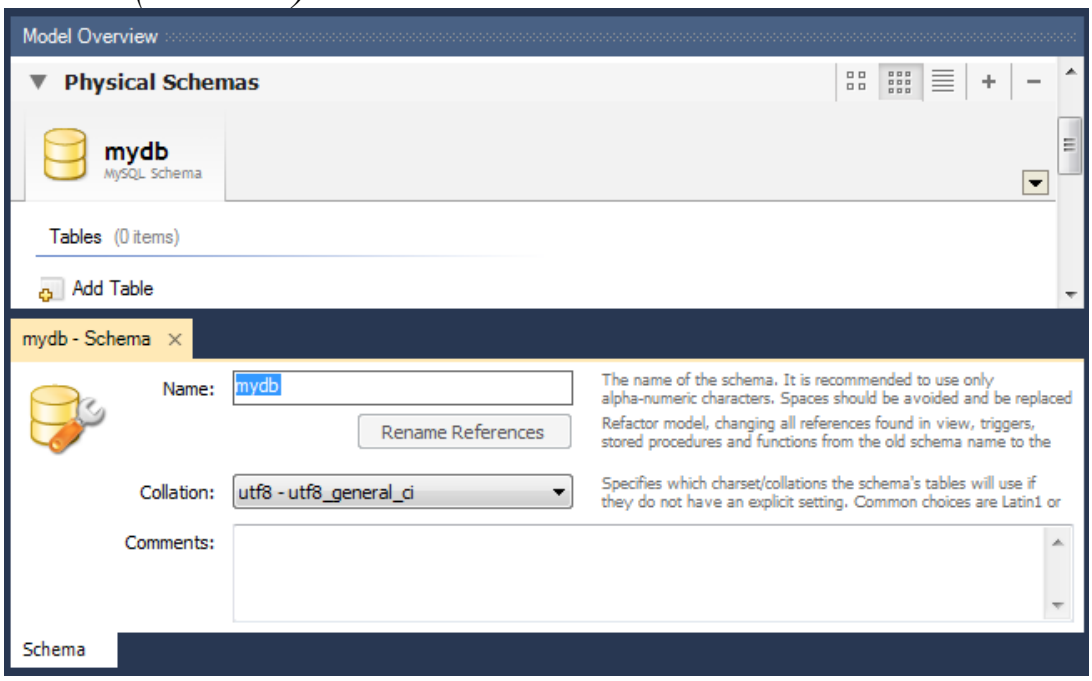


Программа позволяет быстро и с удовольствием накидывать **схемы данных проекта**, проектировать **сущности и связи** между ними, безболезненно **внедрять изменения** в схему и так же быстро и безболезненно **синхронизировать** её с удалённым

сервером. А графический редактор **EER-диаграмм** позволяет увидеть общую картину модели данных.

Создание и редактирование модели данных

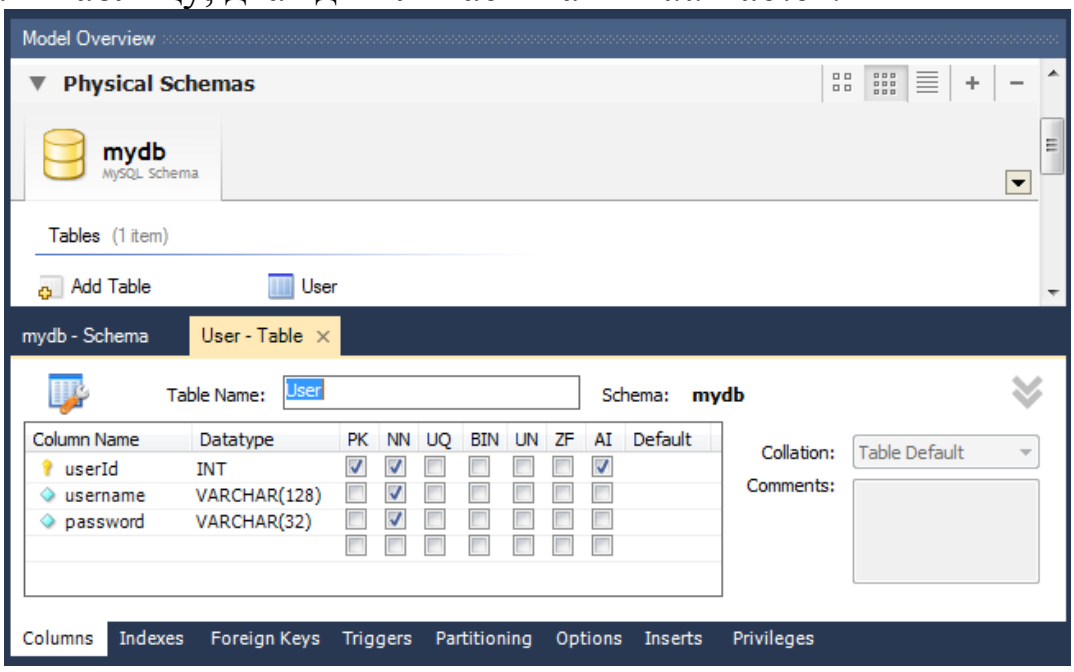
Для добавления модели нажимаем "Models" или выбираем *"File → New Model" (Ctrl + N):*



На этом экране вводим имя базы данных, выбираем кодировку по умолчанию и, если нужно, заполняем поле комментария. Можно приступать к созданию таблиц.

Добавление и редактирование таблицы

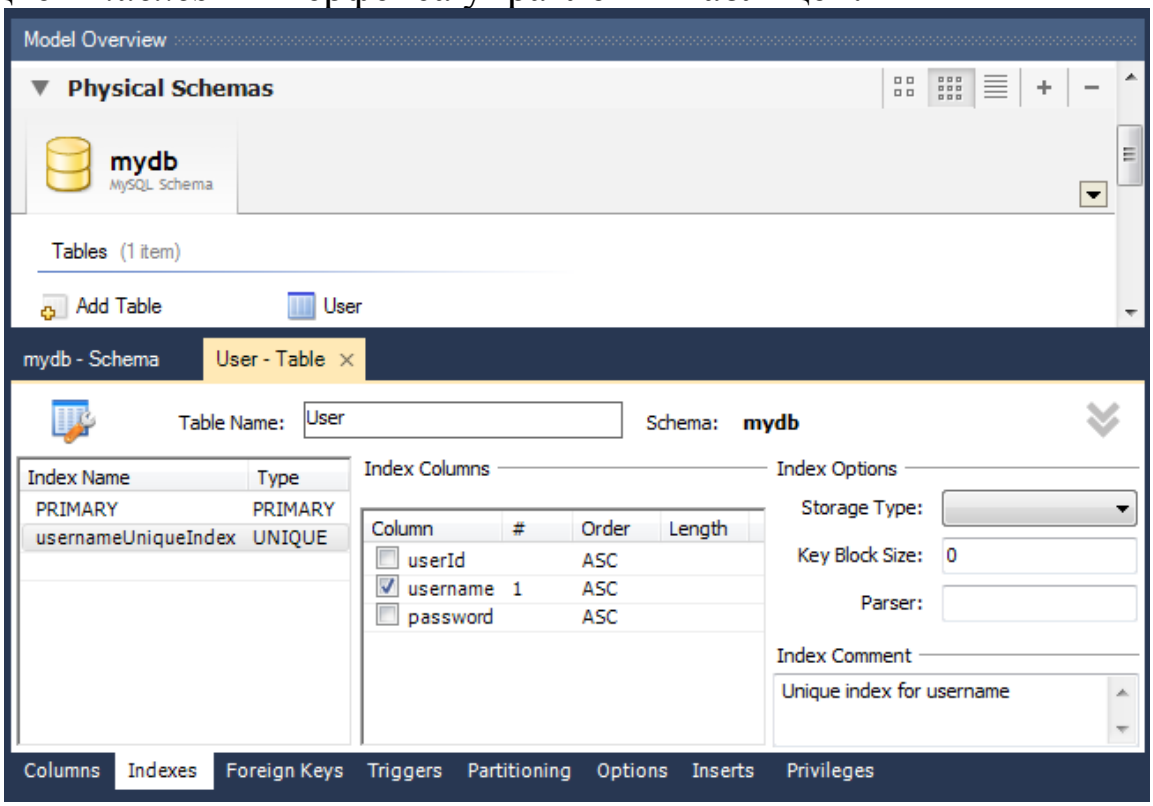
Список баз данных проекта и список таблиц в пределах базы данных будет располагаться во вкладке *"Physical Schemas"*.Чтобы создать таблицу, дважды кликаем на *"+Add Table":*



Откроется удобный интерфейс для редактирования списка полей и их свойств. Здесь мы можем задать название поля, тип данных, а так же установить для полей различные атрибуты: назначить поле *первичным ключом (PK)*, пометить его *Not Null (NN)*, *бинарным (BIN)*, *уникальным (UQ)* и другие, установить для поля *авто-инкремирование (AI)* и *значение по умолчанию (Default)*.

Управление индексами

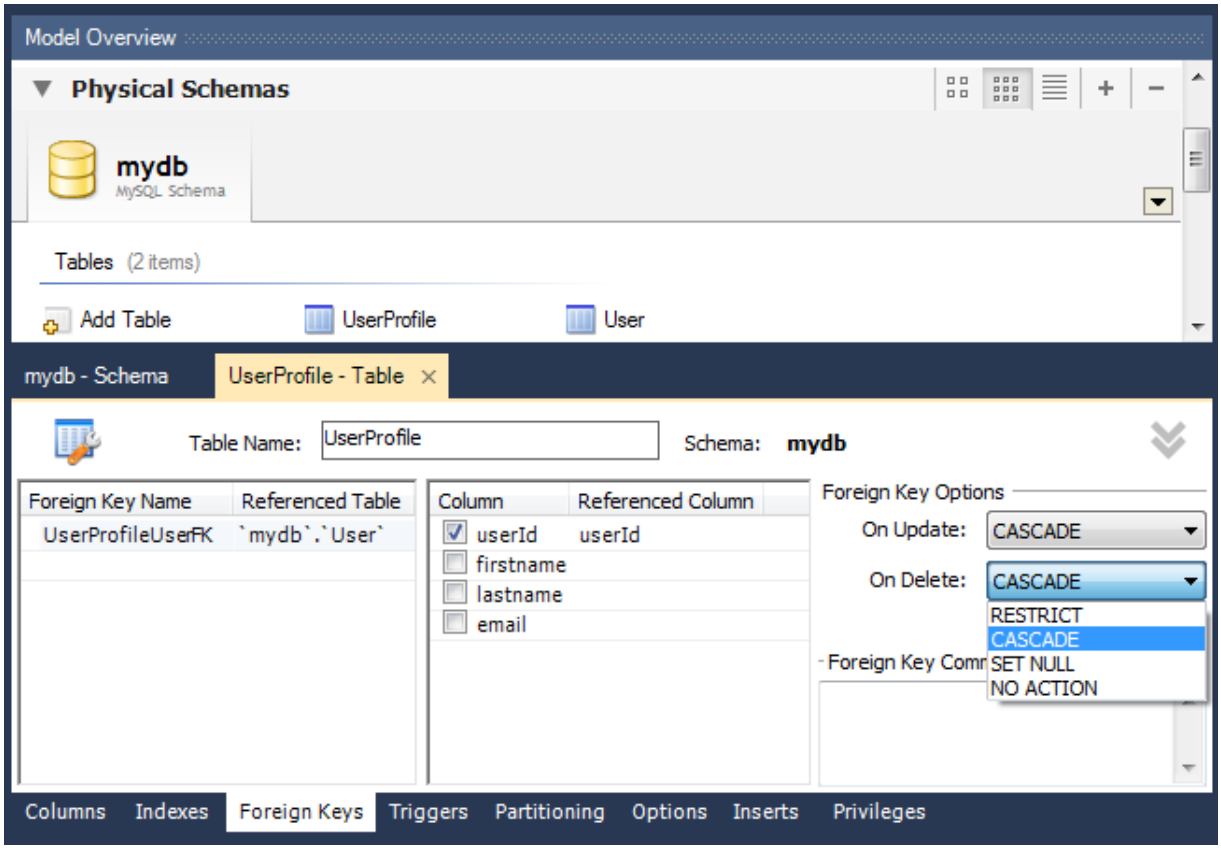
Добавлять, удалять и редактировать индексы таблиц можно во вкладке *"Indexes"* интерфейса управления таблицей:



Вводим название индекса, выбираем его тип, затем галочками помечаем в нужном порядке список полей, участвующих в данном индексе. Порядок полей будет соответствовать порядку, в котором были проставлены галочки. В данном примере добавлен уникальный индекс к полю *username*.

Связи между таблицами

Установка внешних ключей и связывание таблиц возможно только для таблиц *InnoDB* (эта система хранения данных выбирается по умолчанию). Для управления связями в каждой таблице находится вкладка *"Foreign Keys"*:



Для добавления связи открываем вкладку *"Foreign Keys"* **дочерней таблицы**, вводим имя внешнего ключа и выбираем **таблицу-родителя**. Далее в средней части вкладки в графе *Column* выбираем поле-ключ из дочерней таблицы, а в графе *Referenced Column* - соответствующее поле из родительской таблицы (тип полей должен совпадать). При создании внешних ключей **в дочерней таблице автоматически создаются соответствующие индексы**.

В разделе *"Foreign Key Options"* настраиваем поведение внешнего ключа при изменении соответствующего поля *(ON UPDATE)* и удалении *(ON DELETE)* родительской записи:

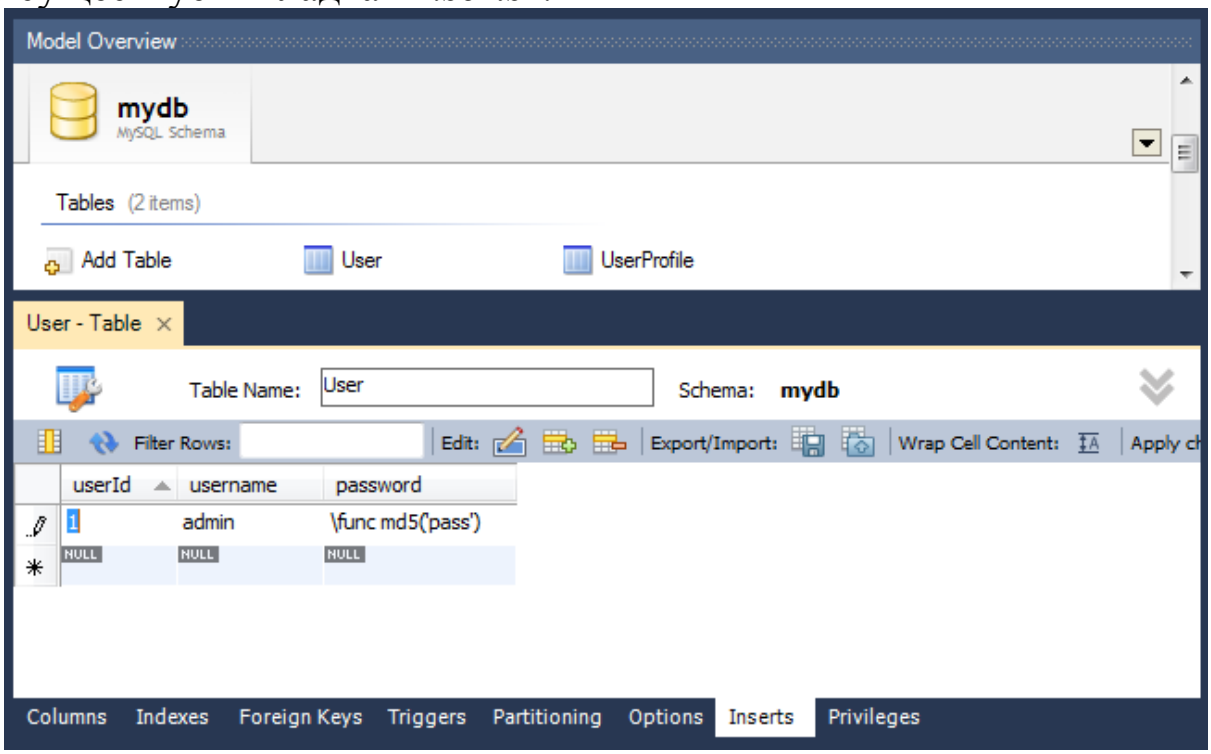
- *RESTRICT* - выдавать ошибку при изменении / удалении родительской записи
- *CASCADE* - обновлять внешний ключ при изменении родительской записи, удалять дочернюю запись при удалении родителя
- *SET NULL* - устанавливать значение внешнего ключа *NULL* при изменении / удалении родителя **(неприемлемо для полей, у которых установлен флаг** *NOT NULL!***)**
- *NO ACTION* - не делать ничего, однако по факту эффект аналогичен RESTRICT

В приведённом примере добавили к дочерней таблице *UserProfile* внешний ключ для связи с родительской

таблицей *User*. При редактировании поля *userId* и удалении позиций из таблицы *User* аналогичные изменения будут **автоматически** происходить и со связанными записями из таблицы *UserProfile*.

Наполнение таблицы базовыми данными

При создании проекта в базу данных часто нужно добавлять стартовые данные. Это могут быть корневые категории, пользователи-администраторы и т.д. В управлении таблицами MySQL Workbench для этого существует вкладка *"Inserts":*

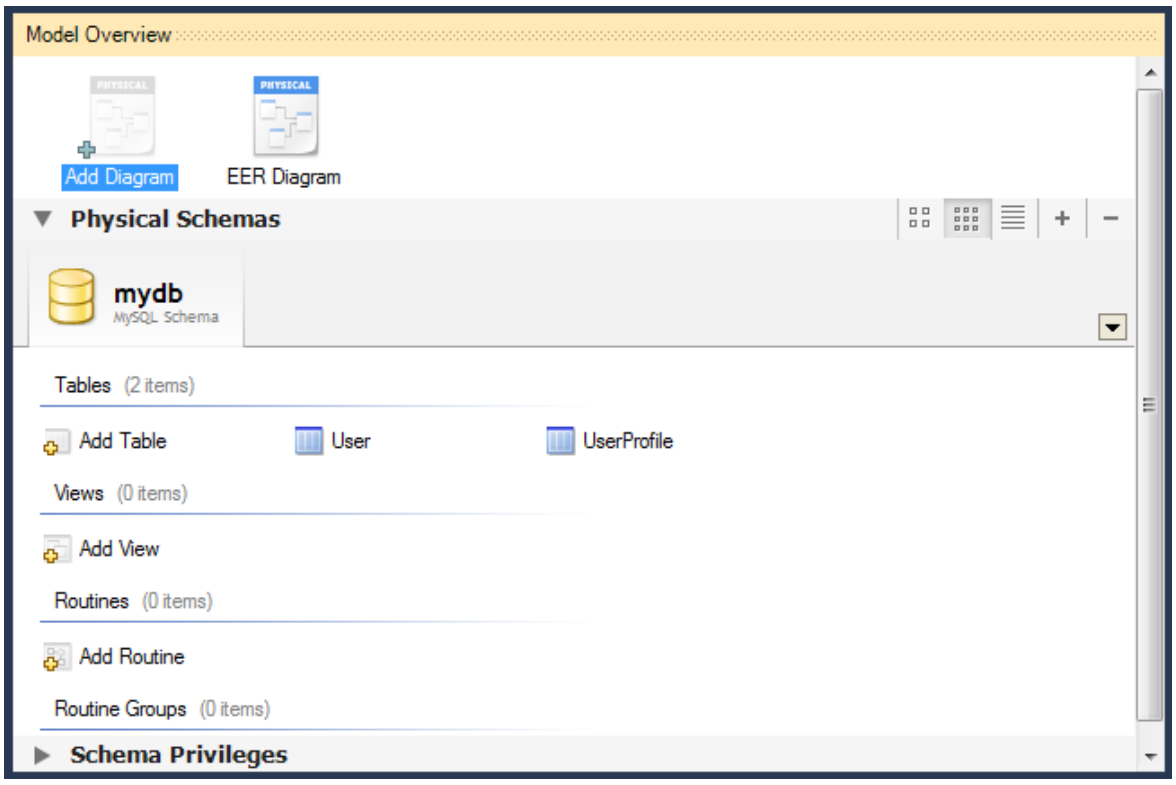


Как видно из примера, в случае, если перед записью в базу данных к данным нужно применить какую-то функцию MySQL, это делается с помощью синтаксиса *\func functionName('data')*, например, *\func md5('password')*.

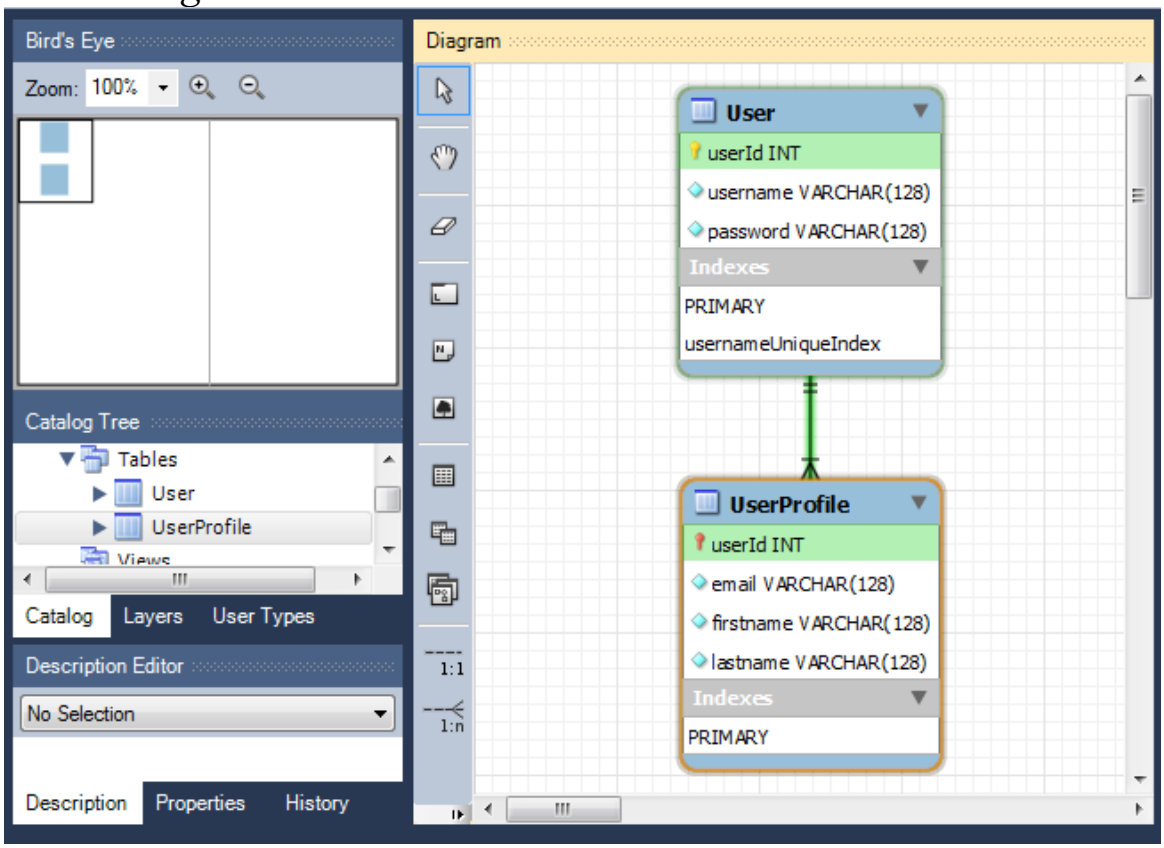
После ввода данных необходимо сохранить их в локальную базу данных нажатием на кнопку *"Apply Changes"*.

Создание ER диаграммы (диаграммы "сущность-связь")

Для представления схемы данных, сущностей и их связей в графическом виде в MySQL Workbench существует редактор EER-диаграмм. Для создания диаграммы в верхней части экрана управления базой данных дважды кликаем на иконку *"+Add Diagram":*



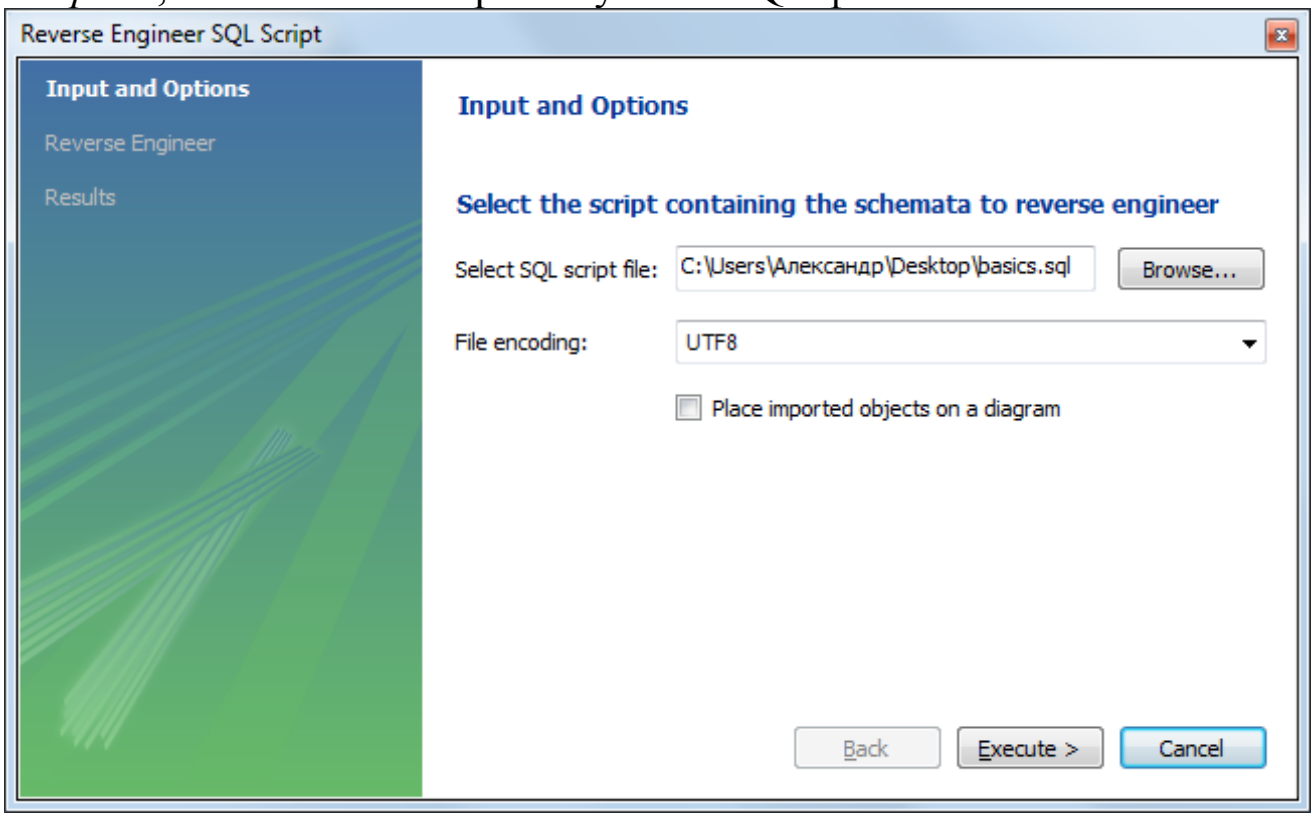
В его интерфейсе можно создавать и редактировать таблицы, добавлять между ними связи различных типов. Чтобы добавить уже существующую в схеме таблицу на диаграмму, просто перетащите её из панели *"Catalog Tree"*.



Для экспорта схемы данных в графический файл выберите *"File → Export"*, а затем один из вариантов *(PNG, SVG, PDF, PostScript File)*.

Импорт существующей схемы данных (из SQL дампа)

Если уже есть схема данных, её можно без труда импортировать в MySQL Workbench для дальнейшей работы. Для импорта модели из SQL файла выбираем *"File → Import → Reverse Engineer MySQL Create Script..."*, после чего выбираем нужный SQL файл и жмём *"Execute >"*



В MySQL Workbench так же предусмотрен импорт и синхронизация модели данных нарямую с удалённым сервером.

Работа с MS SQL Server

Создание базы данных

Базу данных часто отождествляют с набором таблиц, которые хранят данные. Но это не совсем так. Лучше сказать, что база данных представляет хранилище объектов.

Основные из них:

- Таблицы: хранят собственно данные
- Представления (Views): выражения языка SQL, которые возвращают набор данных в виде таблицы
- Хранимые процедуры: выполняют код на языке SQL по отношению к данным к БД (например, получает данные или изменяет их)

- Функции: также код SQL, который выполняет определенную задачу

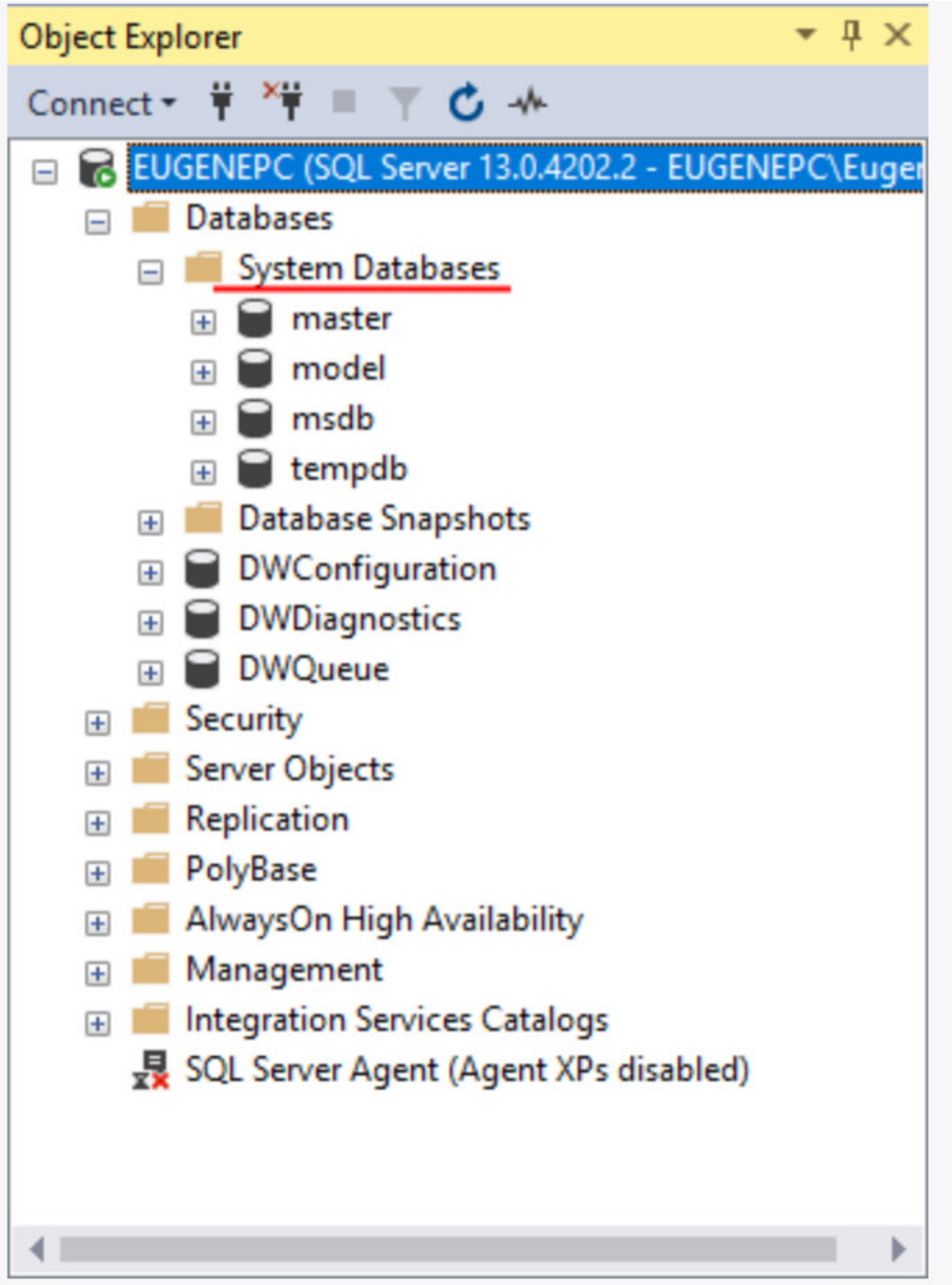
В SQL Server используется два типа баз данных: системные и пользовательские. Системные базы данных необходимы серверу SQL для корректной работы. А пользовательские базы данных создаются пользователями сервера и могут хранить любую произвольную информацию. Их можно изменять и удалять, создавать заново.

Системные базы данных

В MS SQL Server по умолчанию создается четыре системных баз данных:

- master: эта главная база данных сервера, в случае ее отсутствия или повреждения сервер не сможет работать. Она хранит все используемые логины пользователей сервера, их роли, различные конфигурационные настройки, имена и информацию о базах данных, которые хранятся на сервере, а также ряд другой информации.
- model: эта база данных представляет шаблон, на основе которого создаются другие базы данных. То есть когда мы создаем через SSMS свою БД, она создается как копия базы model.
- msdb: хранит информацию о работе, выполняемой таким компонентом как планировщик SQL. Также она хранит информацию о бэкапах баз данных.
- tempdb: эта база данных используется как хранилище для временных объектов. Она заново пересоздается при каждом запуске сервера.

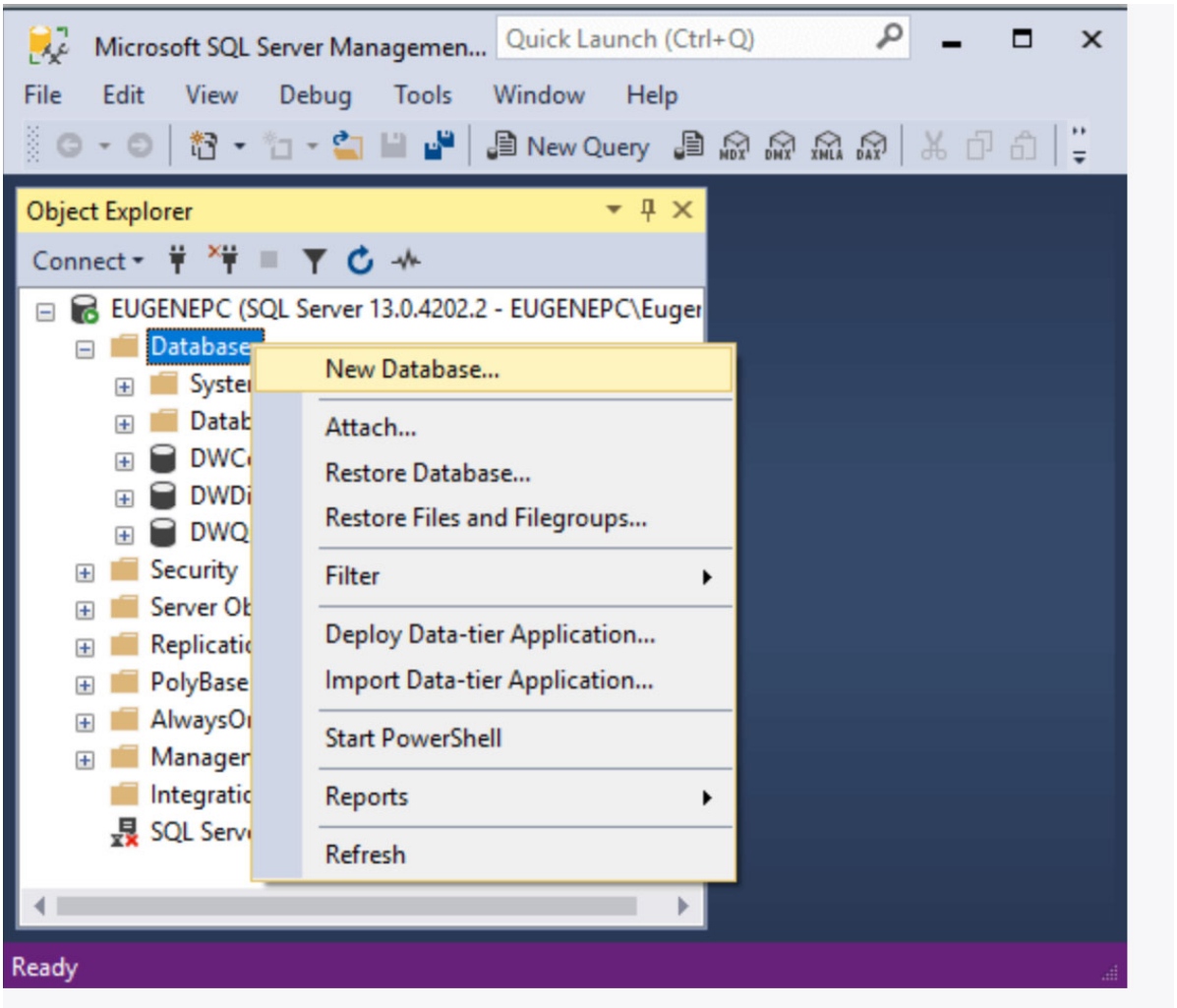
Все эти базы можно увидеть через SQL Server Management Studio в узле Databases -> System Databases:



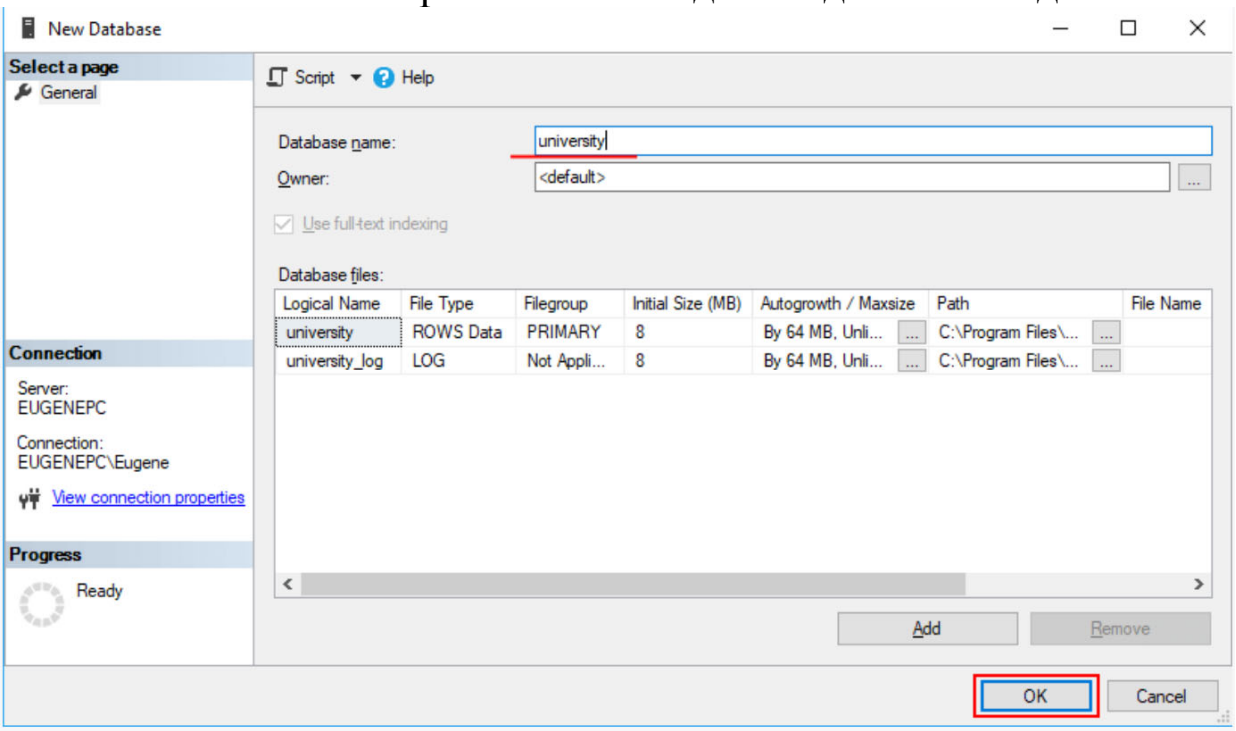
Эти базы данных не следует изменять.

Создание базы данных в SQL Management Studio

Теперь создадим свою базу данных. Для этого мы можем использовать скрипт на языке SQL, либо все сделать с помощью графических средств в SQL Management Studio. В данном случае мы выберем второй способ. Для этого откроем SQL Server Management Studio и нажмем правой кнопкой мыши на узел Databases. Затем в появившемся контекстном меню выберем пункт New Database:



После этого нам открывается окно для создания базы данных:



В поле Database необходимо ввести название новой БД. Пусть у нас база данных называется university.

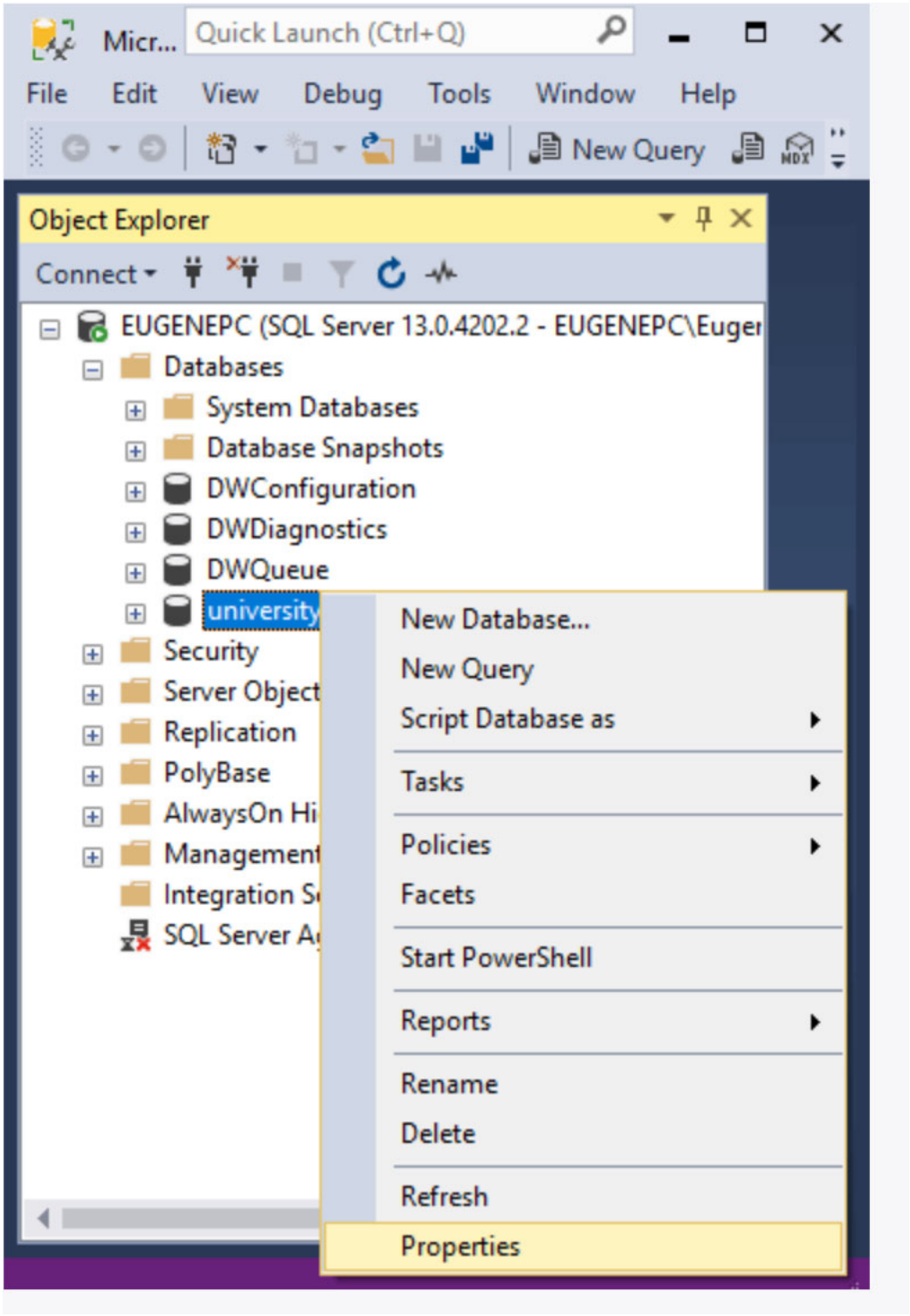
Следующее поле Owner задает владельца базы данных. По умолчанию оно имеет значение <defult>, то есть владельцем будет тот, кто создает эту базу данных. Оставим это поле без изменений.

Далее идет таблица для установки общих настроек базы данных. Она содержит две строки - первая для установки настроек для главного файла, где будут храниться данные, и вторая строка для конфигурации файла логгирования. В частности, мы можем установить следующие настройки:

- Logical Name: логическое имя, которое присваивается файлу базы данных.
- File Type: есть несколько типов файлов, но, как правило, основная работа ведется с файлами данных (ROWS Data) и файлом лога (LOG)
- Filegroup: обозначет группу файлов. Группа файлов может хранить множество файлов и может использоваться для разбиения базы данных на части для размещения в разных местах.
- Initial Size (MB): устанавливает начальный размер файлов при создании (фактический размер может отличаться от этого значения).
- Autogrowth/Maxsize: при достижении базой данных начального размера SQL Server использует это значение для увеличения файла.
- Path: каталог, где будут храниться базы данных.
- File Name: непосредственное имя физического файла. Если оно не указано, то применяется логическое имя.

После ввода названия базы данных нажмем на кнопку ОК, и БД будет создана.

После этого она появится среди баз данных сервера. Если эта БД впоследствии не потребуется, то ее можно удалить, нажав на нее правой кнопкой мыши и выбрав в контекстном меню пункт Delete:

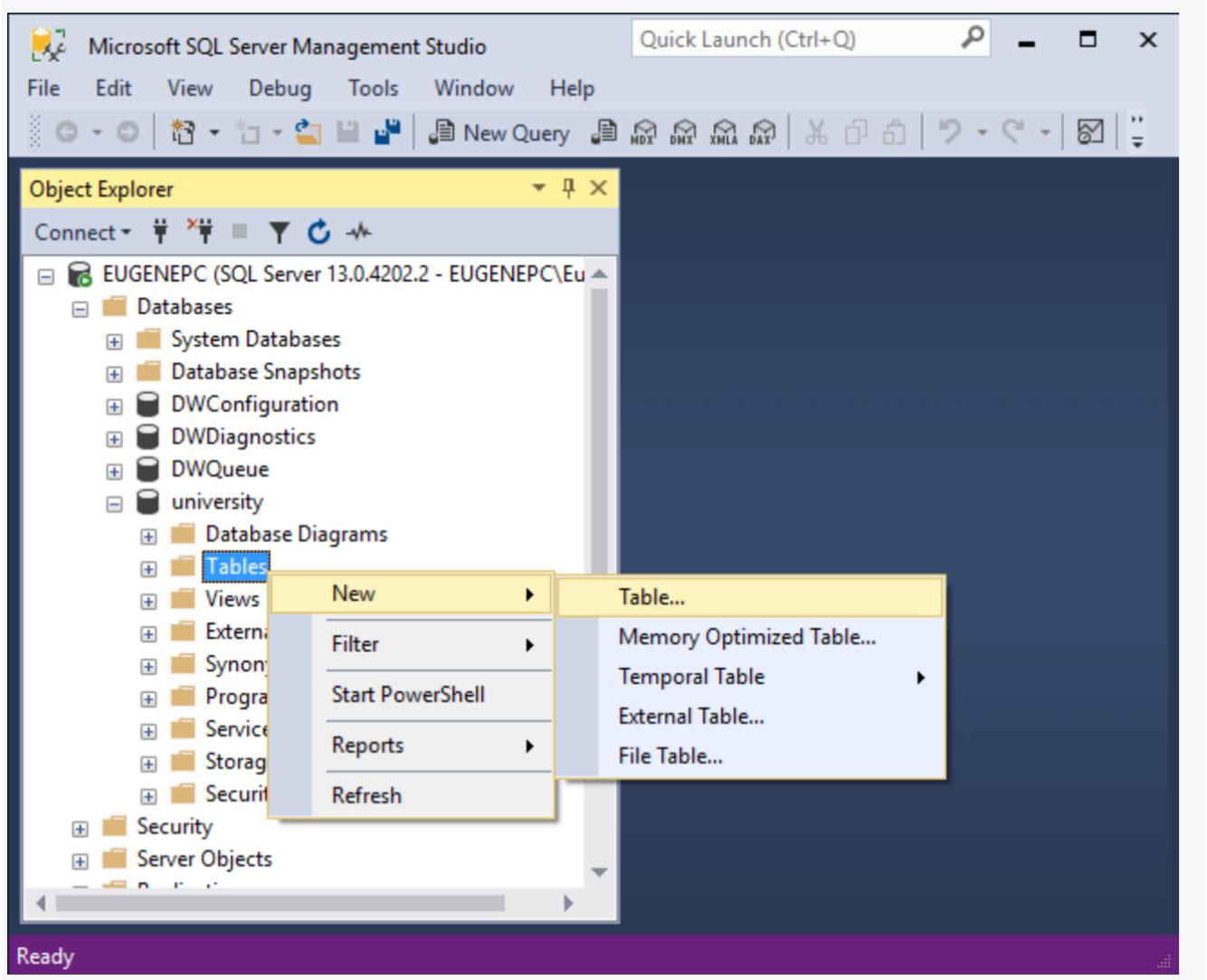


Ключевым объектом в базе данных являются таблицы. Таблицы состоят из строк и столбцов. Столбцы определяют тип информации, которая хранится, а строки содержат значения для этих столбцов.

Нами была создана база данных university. Теперь определим в ней первую таблицу. Опять же для создания таблицы в SQL Server Management Studio можно применить скрипт на языке SQL, либо

воспользоваться графическим дизайнером. В данном случае выберем второе.

Для этого раскроем узел базы данных university в SQL Server Management Studio, нажмем на его подузел Tables правой кнопкой мыши и далее в контекстном меню выберем New -> Table

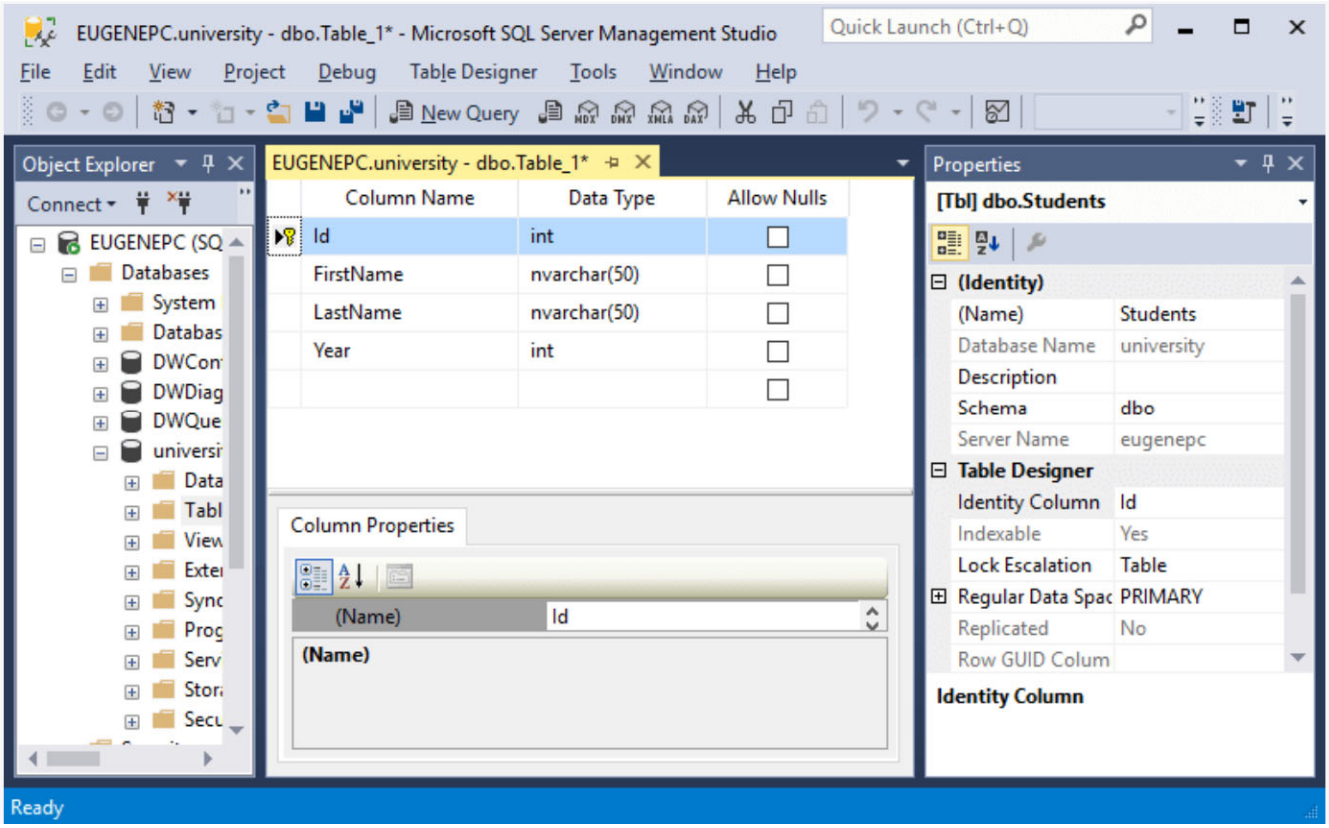


После этого нам откроется дизайнер таблицы. В центральной части в таблице необходимо ввести данные о столбцах таблицы. Дизайнер содержит три поля:

Column Name: имя столбца

Data Type: тип данных столбца. Тип данных определяет, какие данные могут храниться в этом столбце. Например, если столбец представляет числовой тип, то он может хранить только числа.

Allow Nulls: может ли отсутствовать значение у столбца, то есть может ли он быть пустым



Допустим, нам надо создать таблицу с данными учащихся в учебном заведении. Для этого в дизайнере таблицы четыре столбца: Id, FirstName, LastName и Year, которые будут представлять соответственно уникальный идентификатор пользователя, его имя, фамилию и год рождения. У первого и четвертого столбца надо указать тип int (то есть целочисленный), а у столбцов FirstName и LastName - тип nvarchar(50) (строковый).

Соответствие типов данных Microsoft Access и Microsoft SQL

№	Тип данных Microsoft Access	Тип данных Microsoft SQL	Описание типа данных Microsoft SQL
1	Текстовый	nvarchar	Тип данных для хранения текста до 4000 символов
2	Поле MEMO	ntext	Тип данных для хранения символов в кодировке Unicode до 1 073 741 823 символов
3	Числовой	int	Численные значения (целые) в диапазоне от -2 147 483 648 до +2 147 483 647
4	Дата/время	smalldatetime	Дата и время от 1 января 1900 г. до 6 июня 2079 года с точностью до одной минуты
5	Денежный	money	Денежный тип данных, значения которого лежат в диапазоне от -922 337 203 685 477.5808 до +922 337 203 685 477.5807, с точностью до одной десятитысячной
6	Счетчик	int	См. пункт 3
7	Логический	bit	Переменная, способная принимать только два значения - 0 или 1
8	Поле объекта OLE	image	Переменная для хранения массива байтов от 0 до 2 147 483 647 байт
9	Гиперссылка	ntext	См. пункт 2
10	Мастер подстановок	nvarchar	См. пункт 1

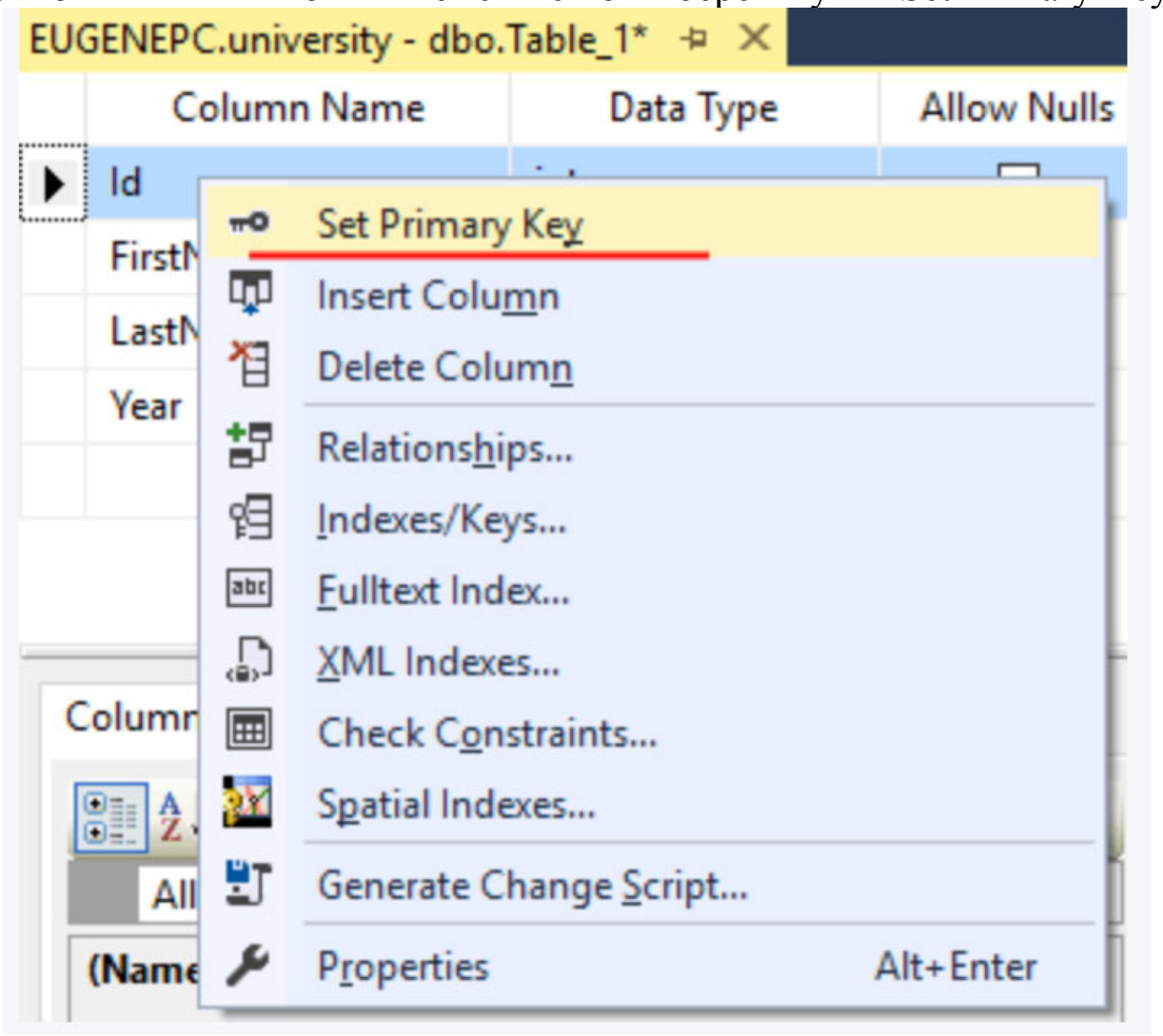
Затем в окне Properties, которая содержит свойства таблицы, в поле Name надо ввести имя таблицы - Students, а в поле Identity ввести Id, то есть тем самым указывая, что столбец Id будет идентификатором.

Имя таблицы должно быть уникальным в рамках базы данных. Как правило, название таблицы отражает название сущности, которая в ней хранится. Например, мы хотим сохранить студентов, поэтому таблица называется Students (слово студент во множественном числе на английском языке). Существуют разные мнения по поводу того, стоит

использовать название сущности в единственном или множественном числе (Student или Students). В данном случае вопрос наименования таблицы всецело ложится на разработчика базы данных.

И в конце нам надо отметить, что столбец Id будет выполнять роль первичного ключа (primary key). Первичный ключ уникально идентифицирует каждую строку. В роли первичного ключа может выступать один столбец, а может и несколько.

Для установки первичного ключа нажмем на столбец Id правой кнопкой мыши и в появившемся меню выберем пункт Set Primary Key.



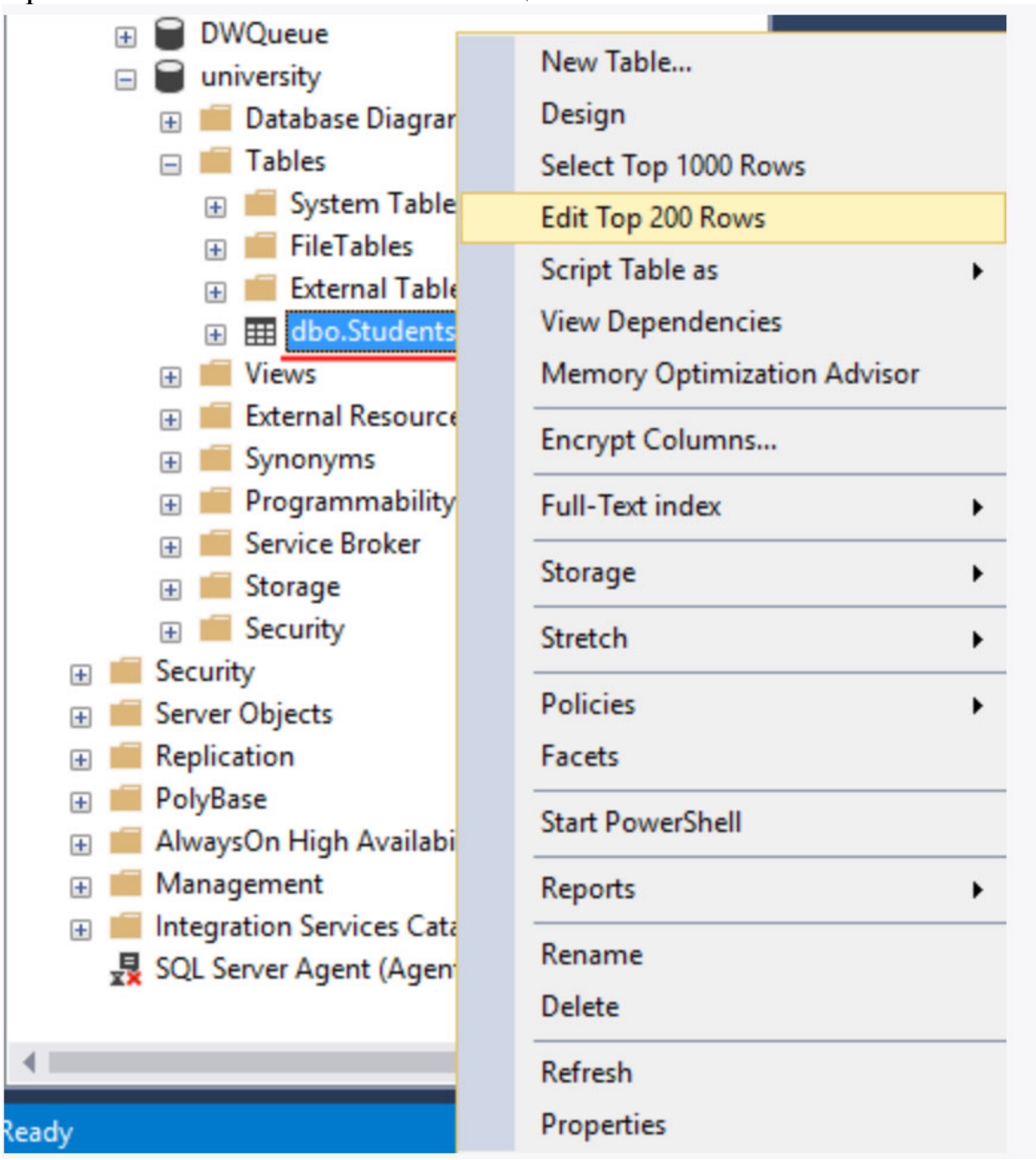
После этого напротив поля Id должен появиться золотой ключик. Этот ключик будет указывать, что столбец Id будет выполнять роль первичного ключа.

И после сохранения в базе данных university появится таблица Students:

Мы можем заметить, что название таблицы на самом деле начинается с префикса dbo. Этот префикс представляет схему. Схема определяет контейнер, который хранит объекты. То есть схема

логически разграничивает базы данных. Если схема явным образом не указывается при создании объекта, то объект принадлежит схеме по умолчанию - схеме dbo.

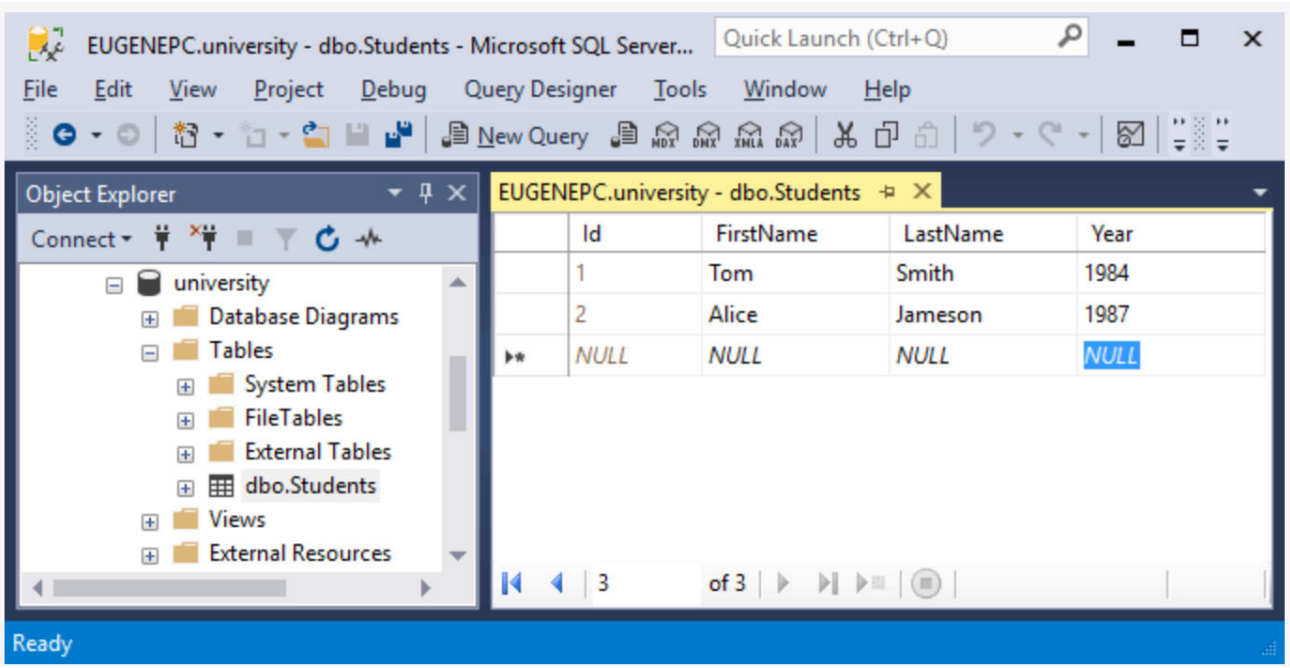
Нажмем правой кнопкой мыши на название таблицы, и нам отобразится контекстное меню с опциями:



С помощью этих опций можно управлять таблицей. Так, опция Delete позволяет удалить таблицу. Опция Design откроет окно дизайнера таблицы, где мы можем при необходимости внести изменения в ее структуру.

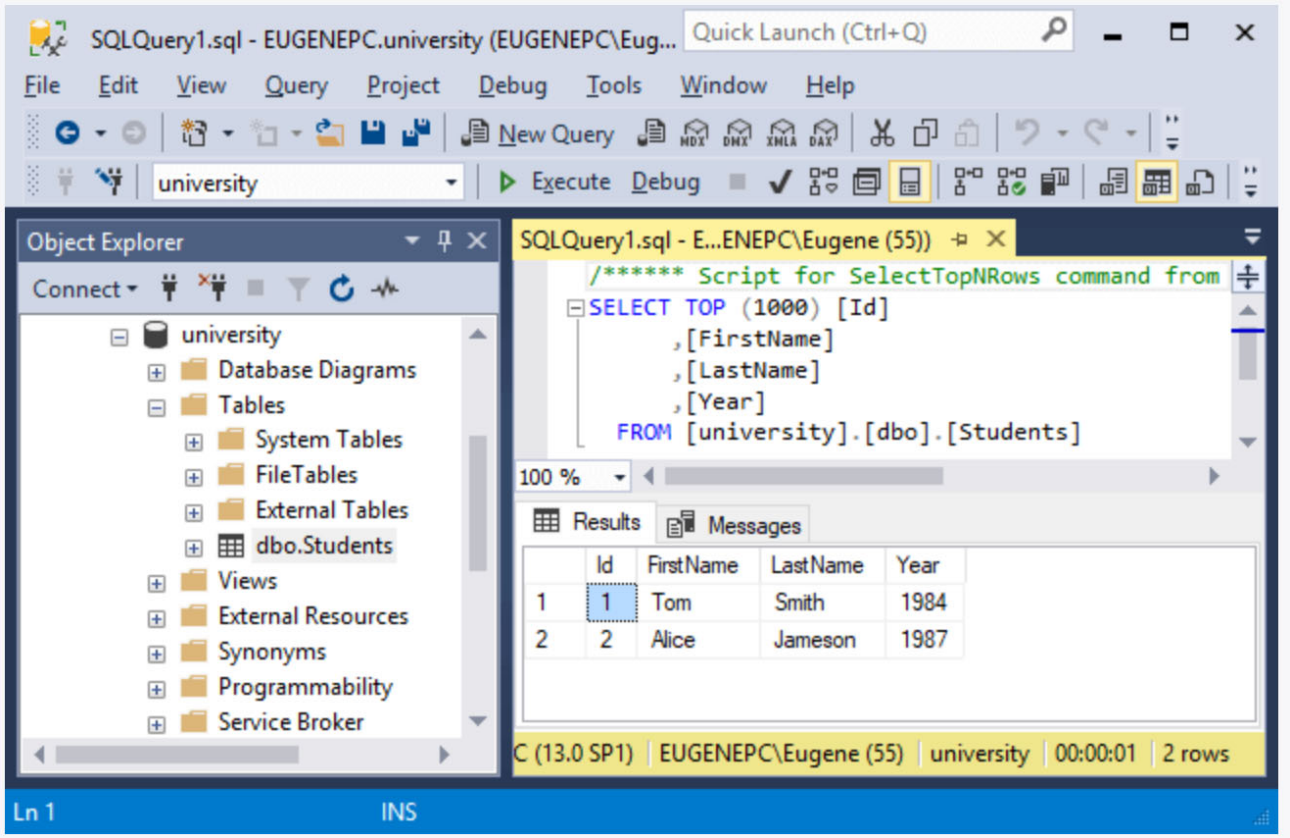
Для добавления начальных данных можно выбрать опцию Edit Top 200 Rows. Она открывает в виде таблицы 200 первых строк и позволяет

их изменить. Но так как у нас таблица только создана, то естественно в ней будет никаких данных. Введем пару строк - пару студентов, указав необходимые данные для столбцов:

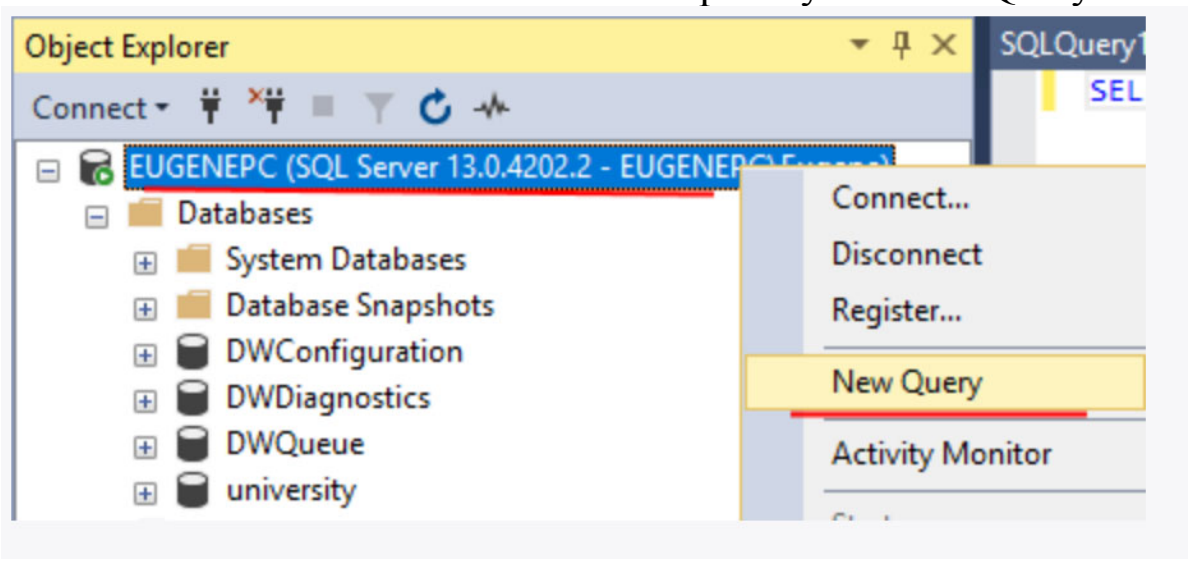


В данном случае мы добавили две строки.

Затем опять же по клику на таблицу правой кнопкой мыши мы можем выбрать в контекстном меню пункт Select To 1000 Rows, и будет запущен скрипт, который отобразит первые 1000 строк из таблицы:



Теперь определим и выполним первый SQL-запрос. Для этого откроем SQL Management Studio, нажмем правой кнопкой мыши на элемент самого верхнего уровня в Object Explorer (название сервера) и в появившемся контекстном меню выберем пункт New Query:

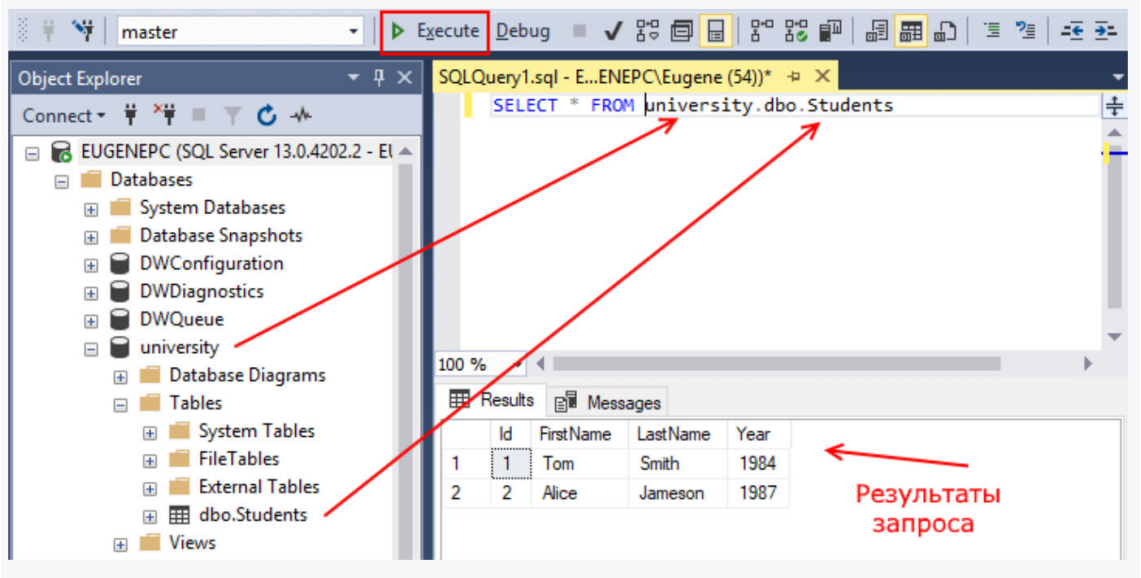


После этого в центральной части программы откроется окно для ввода команд языка SQL.

Выполним запрос к таблице, которая была создана в прошлой теме, в частности, получим все данные из нее. База данных у нас называется university, а таблица - dbo.Students, поэтому для получения данных из таблицы введем следующий запрос:

```
1       SELECT * FROM university.dbo.Students
```

Оператор SELECT позволяет выбирать данные. FROM указывает источник, откуда брать данные. Фактически этим запросом мы говорим "ВЫБРАТЬ все ИЗ таблицы university.dbo.Students". Стоит отметить, что для названия таблицы используется полный ее путь с указанием базы данных и схемы.



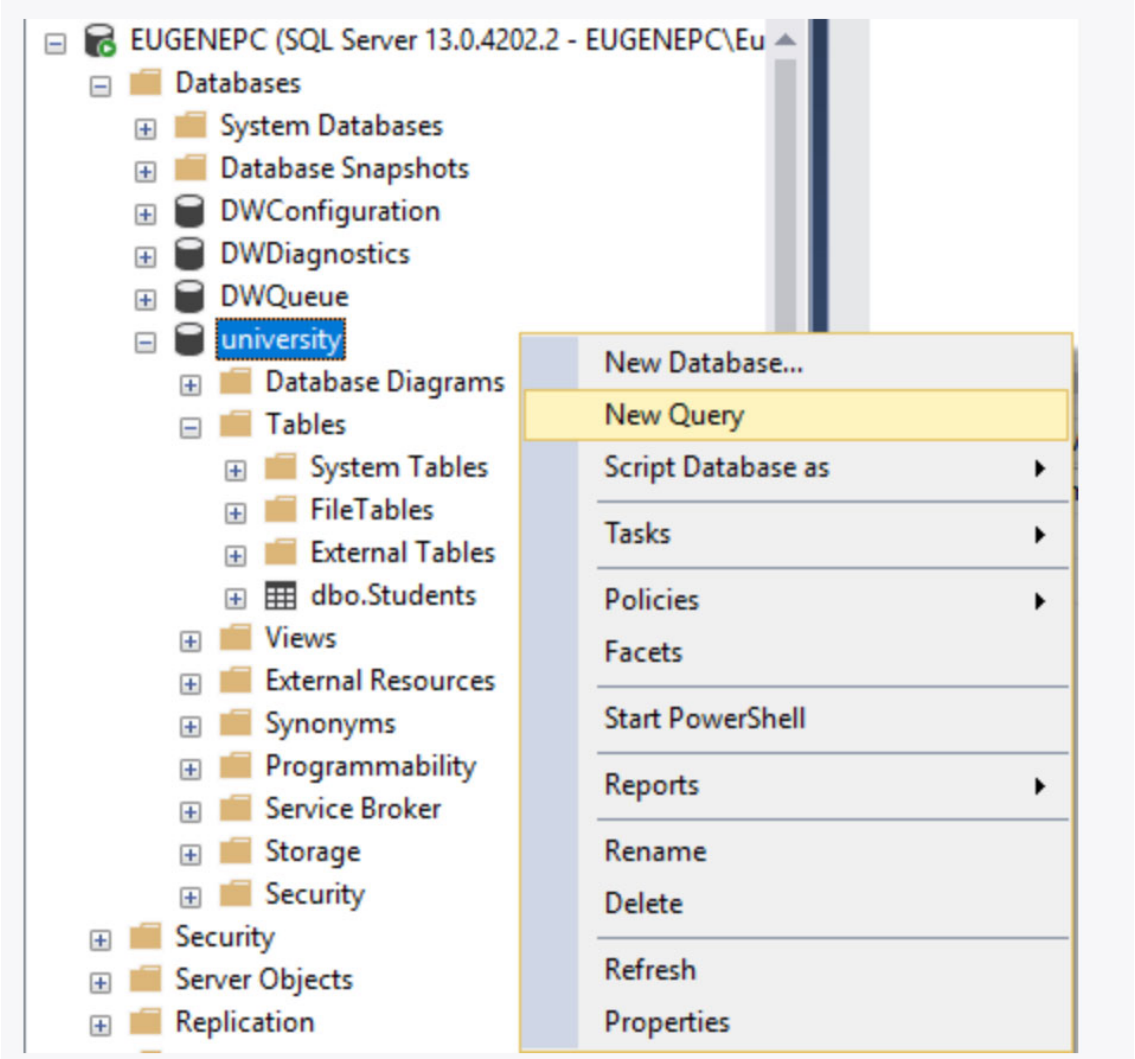
После ввода запроса нажмем на панели инструментов на кнопку Execute, либо можно нажать на клавишу F5.

В результате выполнения запроса в нижней части программы появится небольшая таблица, которая отобразит результаты запроса - то есть все данные из таблицы Students.

Если необходимо совершить несколько запросов к одной и той же базе данных, то мы можем использовать команду USE, чтобы зафиксировать базу данных. В этом случае при запросах к таблицам достаточно указать их имя без имени бд и схемы:

```
1    USE university
2    SELECT * FROM Students
```

В данном случае мы выполняем запрос в целом для сервера, мы можем обратиться к любой базе данных на сервере. Но также мы можем выполнять запросы только в рамках конкретной базы данных. Для этого необходимо нажать правой кнопкой мыши на нужную БД и в контекстном меню выбрать пункт New Query:



Если в этом случае мы захотим выполнить запрос к выше использованной таблице Students, то нам не пришлось бы указывать в запросе название базы данных и схему, так как эти значения итак уже были бы понятны:

```
1    SELECT * FROM Students
```

Для выполнения практического задания можно ознакомиться с обучающими видео

Задание:

1) разработать ER-диаграмму к информационной системе на основании описания бизнес-процесса (https://drive.google.com/drive/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-lkn);
2) разработать элемент рабочей тетради по теме “UML-диаграммы’’(действующие лица, варианты использования, инструментальные средства и т.д.) .

МОДУЛЬ КОМПЕТЕНЦИИ 2: «РАЗРАБОТКА ОКОННЫХ ПРИЛОЖЕНИЙ»

Платформы для разработки программных решений

Компьютерная платфо́рма — в общем смысле, это любая существующая среда выполнения, в которой должен выполняться вновь разрабатываемый фрагмент программного обеспечения или объектный модуль с учётом накладываемых этой средой ограничений и предоставляемых возможностей. Термин платформа может применяться к разным уровням абстракции, включая определенную аппаратную архитектуру, операционную систему или библиотеку времени выполнения.

Нижний слой многоуровневой организации вычислительной системы (аппаратура, операционная система, прикладное программное обеспечение), на который опираются ОС и прикладное ПО. Аппаратные платформы отличаются друг от друга архитектурой центрального процессора и используемыми шинами связи функциональных блоков.

Каждой аппаратной платформе соответствуют совместимые с ней операционные системы и прикладные программы, которые могут на ней запускаться.

Программная платформа представляет собой общую организацию исполнения прикладных программ, задавая, например, порядок запуска программы, схему использования ею адресного пространства, зафиксированные в архитектуре операционной системы плюс API на уровне операционной системы.

При рассмотрении совместимости, или сходства, на уровне операционных систем, например, системных вызовов, файловых систем и пользовательской среды, при сравнении родственных операционных систем (например, UNIX) или семейства (например, Microsoft Windows), речь идет о совместимости на уровне API операционной системы, например, в рамках семейства ОС, а не абстрактного понятия «платформы»

Примеры платформ ОС

- Win32 — Win32 API,
- API POSIX для ОС UNIX/Linux.

Кроссплатформенность программного обеспечения — возможность исполнять его, без перекомпилирования программы, как на различных аппаратных платформах, так и под управлением разных

операционных систем (иначе говоря, возможность запуска исполняемого файла на платформах различных ОС).

Примерами программного обеспечения, выполняющегося на разных аппаратных платформах и под управлением разных операционных систем, являются разнообразные программы, написанные на языках программирования для виртуальных машин, таких, как, например, PHP, Perl, Python, Java, и многие другие, а также — кроссплатформенные среды разработки приложений.

Примеры

- Qt
- GTK
- Boost
- Java Virtual Machine
- .NET Framework
- Adobe AIR

В рамках компетенции «Программные решения для бизнеса» для разработки используются платформы Java или .NET. В рамках .NET чаще всего используют интерфейсы программирования Windows Forms или WPF.

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 APIв управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, C++, так и на VB.Net, J# и др.

С одной стороны, Windows Forms рассматривается как замена более старой и сложной библиотеке MFC, изначально написанной на языке C++. С другой стороны, WF не предлагает парадигму, сравнимую с MVC. Для исправления этой ситуации и реализации данной функциональности в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является User Interface Process Application Block, выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Внутри .NET Framework, Windows Forms реализуется в рамках пространства имён System.Windows.Forms.

Приложение Windows Forms представляет собой событийно-ориентированное приложение, поддерживаемое Microsoft .NET Framework. В отличие от пакетных программ, большая часть времени тратится на ожидание от пользователя каких-либо действий, как, например, ввод текста в текстовое поле или клика мышкой по кнопке.

Windows Presentation Foundation (WPF) — это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. С помощью WPF можно создавать широкий спектр как автономных, так и размещенных в браузере приложений.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений, в том числе Extensible Application Markup Language (XAML), элементами управления, привязкой данных, макетом, 2-D- и 3-D-графикой, анимацией, стилями, шаблонами, документами, мультимедиа, текстом и оформлением. WPF входит в состав Microsoft .NET Framework и позволяет создавать приложения, включающие другие элементы библиотеки классов .NET Framework.

Windows Presentation Foundation (WPF) — это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. С помощью WPF можно создавать широкий спектр как автономных, так и размещенных в браузере приложений. На следующем рисунке показан пример одного из таких приложений Contoso Healthcare Sample Application.

Программирование с использованием WPF

WPF существует в качестве подмножества типов .NET Framework, которые занимают большую часть в пространстве именSystem.Windows. Пользователи, которые ранее создавали приложения с помощью .NET Framework, используя такие управляемые технологии, как ASP.NET и Windows Forms, должны быть знакомы с основами программирования WPF; создание экземпляров классов, задание свойств, вызов методов и обработка событий осуществляется с помощью одного из хорошо знакомых языков программирования .NET Framework, таких как C# или Visual Basic.

Для поддержки некоторых более мощных возможностей WPF и для упрощения процесса программирования WPF включает дополнительные программные конструкции, которые расширяют свойства и события: свойства зависимостей и перенаправленные события. Дополнительные сведения о свойствах зависимостей см. в разделе Общие сведения о свойствах зависимости. Дополнительные сведения о перенаправленных событиях см. в разделе Общие сведения о перенаправленных событиях.

Разметка и код программной части

В WPF дополнительно совершенствуется процесс программирования для разработки клиентских приложений Windows. Одним очевидным усовершенствованием является возможность разрабатывать приложения с помощью разметки и кода программной части, с которыми разработчики ASP.NET должны быть уже знакомы. Разметка Extensible Application Markup Language (XAML) обычно используется для реализации внешнего вида приложения при реализации его поведения с помощью управляемых языков программирования (кода программной части). Это разделение внешнего вида и поведения имеет следующие преимущества:

Затраты на разработку и обслуживание снижаются, так как разметка определенного внешнего вида тесно не связана с кодом определенного поведения.

Разработка более эффективна, так как разработчики, реализующие внешний вид приложения, могут это делать одновременно с разработчиками, реализующими поведение приложения.

Для реализации и совместного использования разметки XAML применяется множество средств конструирования, чтобы удовлетворить требованиям участников разработки приложений. Microsoft Expression Blend предназначается для конструкторов, в то время как Visual Studio 2005 ориентируется на разработчиков.

Ниже приводится краткое описание разметки и кода программной части WPF. Дополнительные сведения об этой модели программирования см. в Общие сведения о языке XAML (WPF) и в Код программной части и XAML в WPF.

Разметка

XAML — это основанный на XML язык разметки, который используется для декларативной реализации внешнего вида приложения. Обычно он используется для создания окон, диалоговых окон, страниц и

пользовательских элементов управления, а также для их заполнения элементами управления, фигурами и графикой.

Поскольку XAML основан на XML, UI, который формируется с его помощью, организуется в виде иерархии вложенных элементов, называемой деревом элементов. Дерево элементов предоставляет логический и интуитивно понятный способ для создания и управления UIs.

Код программной части

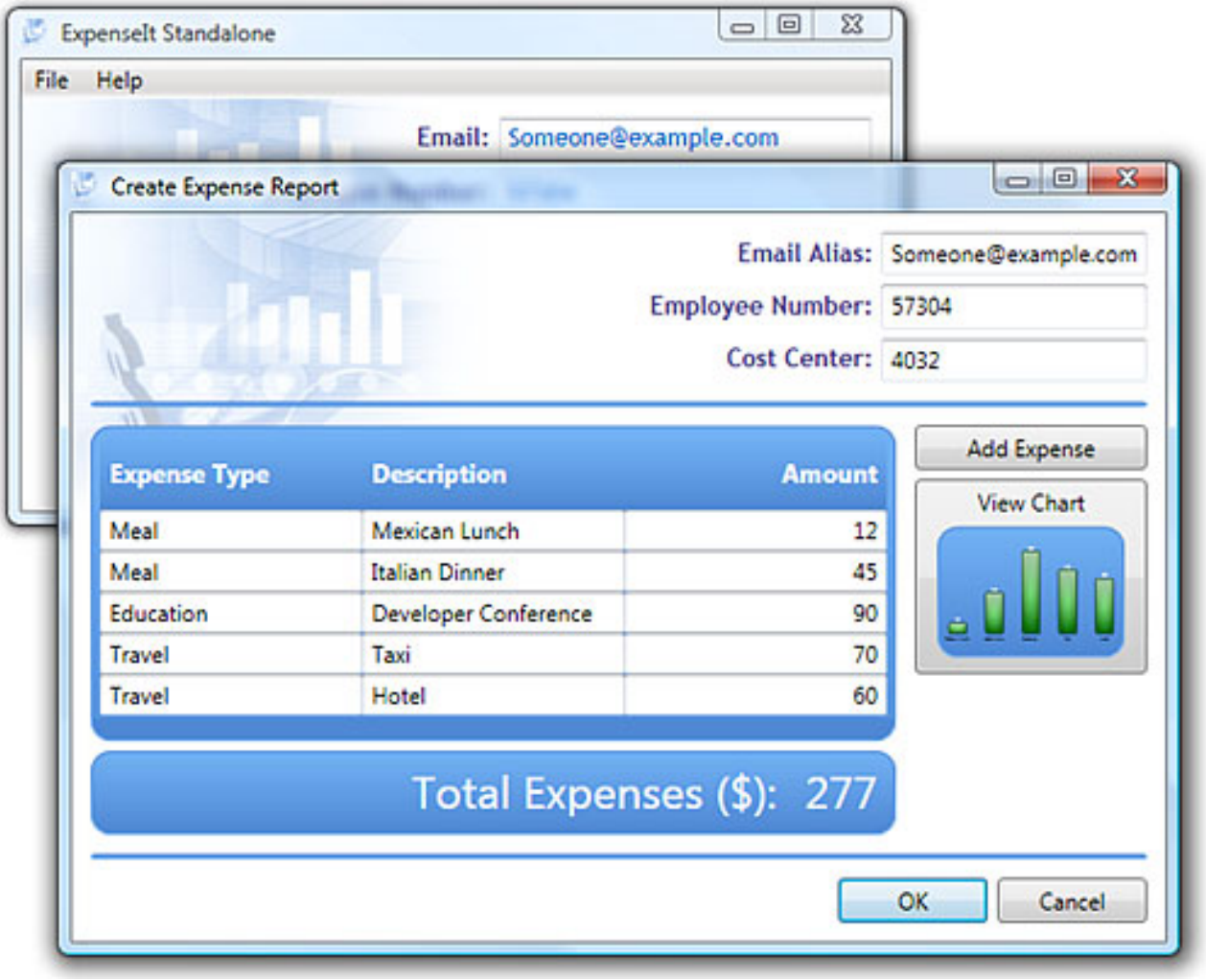
Приложение в основном предназначено для реализации функциональных возможностей, которые отвечают на взаимодействия с пользователем, включая обработку событий (например, нажатие меню, панели инструментов или кнопки) и вызов бизнес-логики и логики доступа к данным в ответ на события. В WPF такое поведение обычно реализуется в коде, который связан с разметкой. Этот тип кода называется кодом программной части.

Приложения

.NET Framework, System.Windows, разметка и выделенный код составляют основу разработки приложений WPF. Кроме того, WPF предоставляет полный набор средств для создания удобных и многофункциональных элементов пользовательского интерфейса. Чтобы упаковать разработанные элементы и предоставить их пользователю в виде приложений, WPF предоставляет типы и службы, вместе называемые моделью приложения. Модель приложения поддерживает разработку как автономных, так и размещенных в браузере приложений.

Автономные приложения

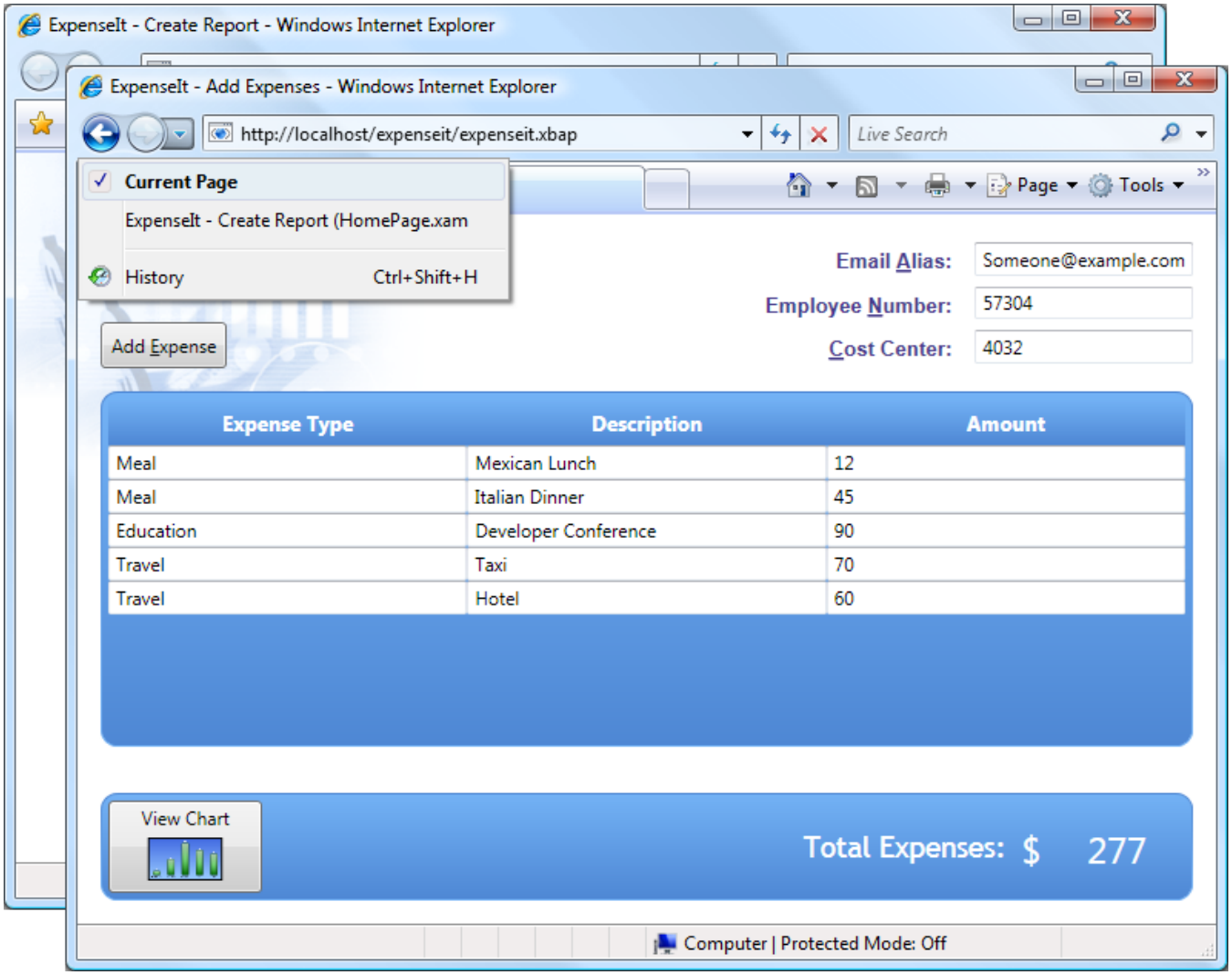
Для автономных приложений можно создавать доступные из меню и панелей инструментов окна и диалоговые окна с помощью класса Window. На следующем рисунке показано автономное приложение с главным и диалоговым окном.



Кроме того, можно использовать следующие диалоговые окна WPF: MessageBox, OpenFileDialog, SaveFileDialog и PrintDialog.

Приложения, размещенные в браузере

Для приложений, размещаемых в браузере, также называемых XAML browser applications (XBAPs), можно создавать страницы (Page) и страничные функции (PageFunction<T>), по которым можно переходить с помощью гиперссылок (классы Hyperlink). На следующем рисунке показана страница в XBAP, размещенная в Internet Explorer 7.



Приложения WPF могут размещаться как в Microsoft Internet Explorer 6, так и в Internet Explorer 7. WPF предлагает два следующих параметра для альтернативных узлов переходов:

- Frame , чтобы размещать блоки содержимого для навигации в окнах или на страницах.
- NavigationWindow , чтобы размещать содержимое для навигации во всем окне.

Элементы управления

Взаимодействия с пользователем, предоставляемые моделью приложения, являются сконструированными элементами управления. В WPF "элемент управления" — это основное понятие, относящееся к категории классов WPF, которые расположены в окне или на странице, имеют user interface (UI) и реализовывают некоторое поведение.

Элементы управления WPF по функциям

Далее перечислены встроенные элементы управления WPF.

- Кнопки: Button и RepeatButton.
- Отображение данных: DataGrid, ListView и TreeView.

- Выбор и отображение дат: Calendar и DatePicker.
- Диалоговые
  окна: OpenFileDialog, PrintDialog и SaveFileDialog.
- Рукописный фрагмент: InkCanvas и InkPresenter.
- Документы: DocumentViewer, FlowDocumentPageViewer, Flo
  wDocumentReader, FlowDocumentScrollViewer и StickyNoteCo
  ntrol.
- Ввод: TextBox, RichTextBox и PasswordBox.
- Структура: Border, BulletDecorator, Canvas, DockPanel, Expan
  der, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip,
  Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox
  , VirtualizingStackPanel, Window и WrapPanel.
- Мультимедиа: Image, MediaElement и SoundPlayerAction.
- Меню: ContextMenu, Menu и ToolBar.
- Переходы: Frame, Hyperlink, Page, NavigationWindow и TabCo
  ntrol.
- Выбор: CheckBox, ComboBox, ListBox, RadioButton и Slider.
- Информация
  пользователя: AccessText, Label, Popup, ProgressBar, StatusBar
  , TextBlock и ToolTip.

Ввод и команды

Элементы управления наиболее часто обнаруживают входные данные пользователя и отвечают на них. Система ввода WPF использует прямые и перенаправленные события для поддержки ввода текста, управления фокусом и позиционирования мыши. Дополнительные сведения см. в разделе Общие сведения о входных данных.

Приложения часто предъявляют сложные требования к вводу данных. WPF предоставляет систему команд, в которой действия пользователя по вводу данных отделяются от кода, который отвечает на эти действия. Дополнительные сведения см. в разделе Общие сведения о системе команд.

Макет

При создании UI происходит упорядочивание элементов управления по расположению и размеру для формирования структуры. Основным требованием любой структуры является адаптируемость к изменениям размера окна и параметрам отображения. Чтобы не пришлось создавать код для адаптации

структуры в таких обстоятельствах, WPF предоставляет первоклассную расширяемую систему структуры.

Основой системы структуры является относительное позиционирование, что увеличивает способность адаптации к изменяемому окну и условиям отображения. Кроме того, система структуры управляет согласованием между элементами управления для определения структуры.Такое согласование состоит из двух этапов: сначала элемент управления сообщает родительскому элементу, какое расположение и размер требуется; затем родительский элемент сообщает элементу управления, какое пространство он может занять.

Дочерние элементы управления получают доступ к системе макета через базовые классы WPF. Для распространенных макетов, таких как сетки, вложение и закрепление, WPF включает несколько элементов управления макетом.

- Canvas : дочерние элементы управления предоставляют свои собственные макеты.
- DockPanel : дочерние элементы управления выравниваются по краям панели.
- Grid : дочерние элементы управления располагаются по строкам и столбцам.
- StackPanel : дочерние элементы управления располагаются либо горизонтально, либо вертикально.
- VirtualizingStackPanel : дочерние элементы управления являются виртуальными и располагаются в одной горизонтальной или вертикальной строке.
- WrapPanel : дочерние элементы управления располагаются в порядке слева-направо и переносятся на следующую строку, когда в текущей строке не хватает места.

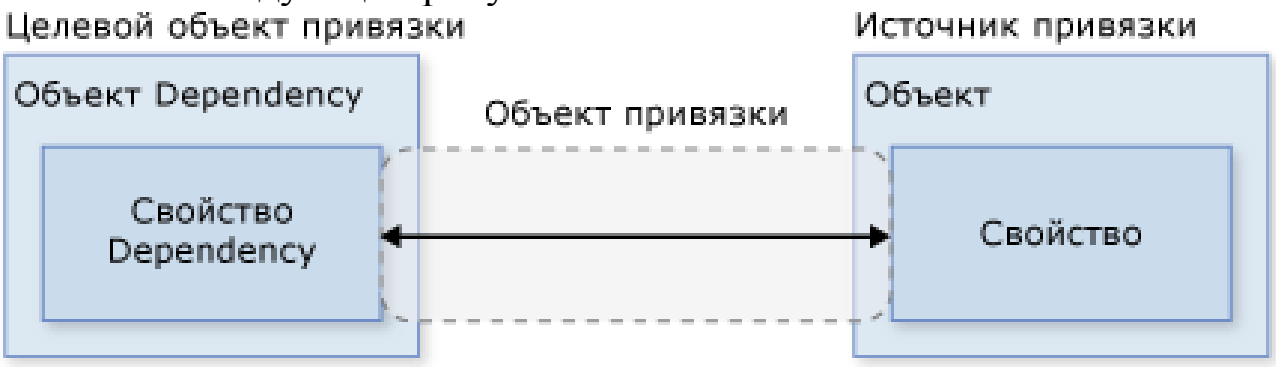
Привязка данных

Большинство приложений создаются для предоставления пользователям средств просмотра и редактирования данных. В приложениях WPF работа по хранению и доступу к данным уже обеспечена такими технологиями, как Microsoft SQL Server и ADO.NET. После обращения к данным и загрузки данных в управляемые объекты приложения начинается основная тяжелая работа для приложений WPF. По существу, она включает в себя две вещи:

- копирование данных из управляемых объектов в элементы управления, где данные могут отображаться и редактироваться;

- проверка того, что изменения, внесенные в данные с помощью элементов управления, скопированы обратно в управляемые объекты.

Чтобы упростить разработку приложений, WPF предоставляет механизм привязки данных для автоматического выполнения этих этапов. Основной единицей механизма привязки данных является класс Binding, назначение которого привязать элемент управления (цель привязки) к объекту данных (источник привязки). Это отношение показано на следующем рисунке.



Механизм привязки данных WPF предоставляет дополнительную поддержку, включающую проверку, сортировку, фильтрацию и группировку. Кроме того, привязка данных поддерживает использование шаблонов данных для создания настраиваемого UI, чтобы связывать данные, когда UI отображается несоответствующими стандартными элементами управления WPF.

Графика

WPF представляет обширный, масштабируемый и гибкий набор графических возможностей, которые имеют следующие преимущества:

Графика, не зависящая от разрешения и устройства. Основной единицей измерения в графической системе WPF является аппаратно-независимая точка, которая составляет 1/96 часть дюйма независимо от фактического разрешения экрана и предоставляет основу для создания изображения, независимого от разрешения и устройства. Каждый аппаратно-независимый пиксель автоматически масштабируется в соответствии с числом точек на дюйм в системе, в которой он отображается.

Повышенная точность. В системе координат WPF используются числа с плавающей запятой двойной точности, вместо одиночной точности. Значения преобразований и прозрачности также выражаются с помощью чисел двойной точности. Кроме того, WPF поддерживает широкую цветовую палитру (scRGB) и предоставляет встроенную

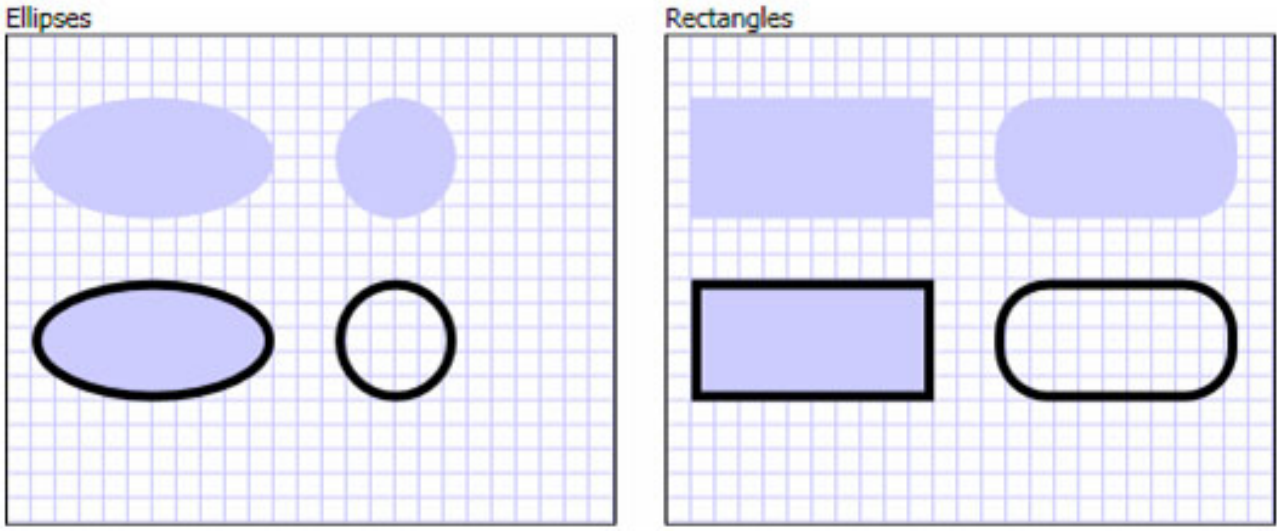
поддержку для управления входными данными из различных цветовых пространств.

Дополнительная поддержка графики и анимации. WPF упрощает программирование графики за счет автоматического управления анимацией. Разработчик не должен заниматься обработкой сцен анимации, циклами визуализации и билинейной интерполяцией. Кроме того, WPF предоставляет поддержку проверки нажатия и полную поддержку альфа-компоновки.

Аппаратное ускорение. Графическая система WPF использует преимущества графического оборудования, чтобы уменьшить использование ЦП.

Двухмерные формы

WPF предоставляет библиотеку общих 2-D фигур, нарисованных с помощью векторов, таких, как прямоугольники и эллипсы, показанные на следующем рисунке.



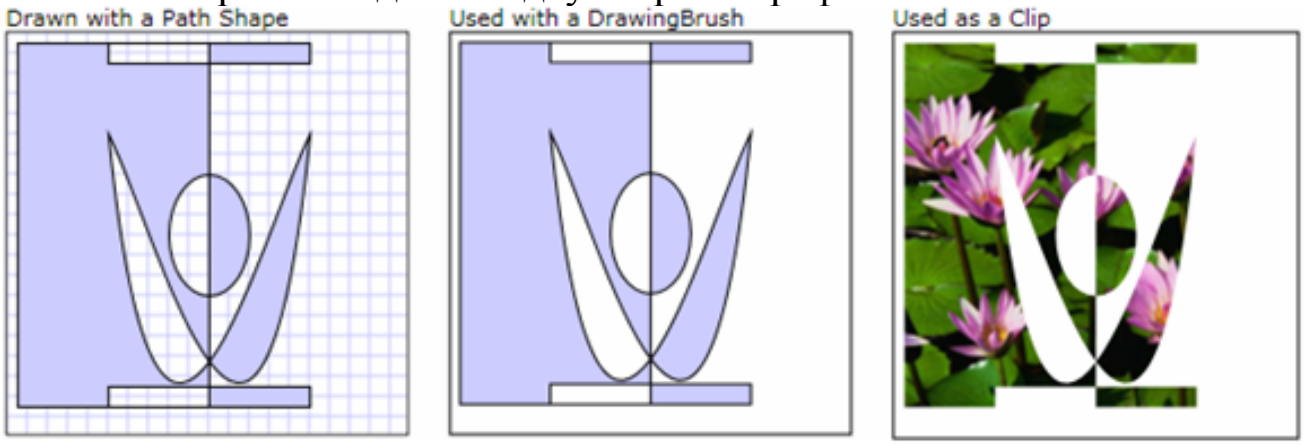
Интересная особенность фигур в том, что они могут не только отображаться; фигуры реализовывают многие возможности, ожидаемые от элементов управления, включая ввод с клавиатуры и ввод с помощью мыши.

Двухмерная геометрия

WPF предоставляет стандартный набор двухмерных (2-D) фигур. Однако, возможно, потребуется создать пользовательские фигуры для облегчения разработки настраиваемого UI. В этих целях WPF предоставляет геометрические объекты. На следующем рисунке показано использование геометрий для создания пользовательской фигуры, которая может быть нарисована непосредственно, использоваться в качестве кисти, или использоваться для отсечения других фигур и элементов управления.

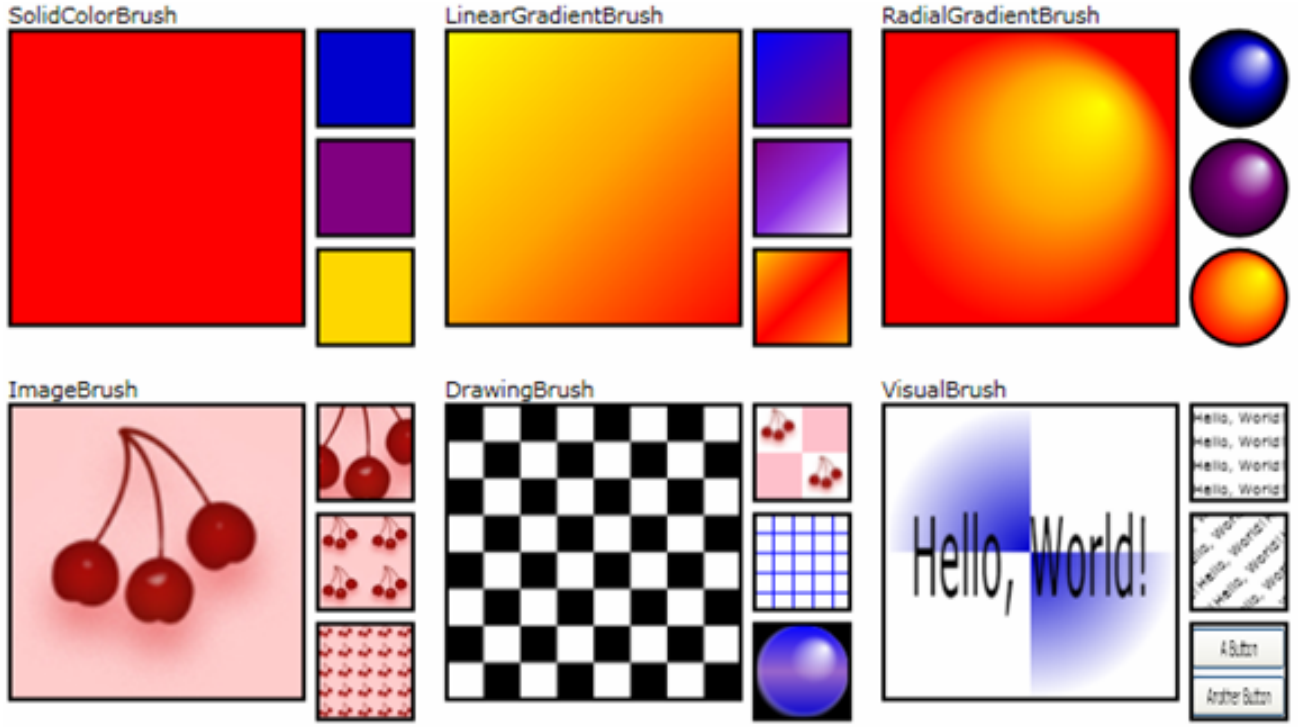
Объекты Path могут быть использованы для рисования замкнутых, открытых, составных фигур и даже кривых поверхностей.

Объекты Geometry могут использоваться для отсечения, проверки нажатия и отрисовки данных двухмерной графики.



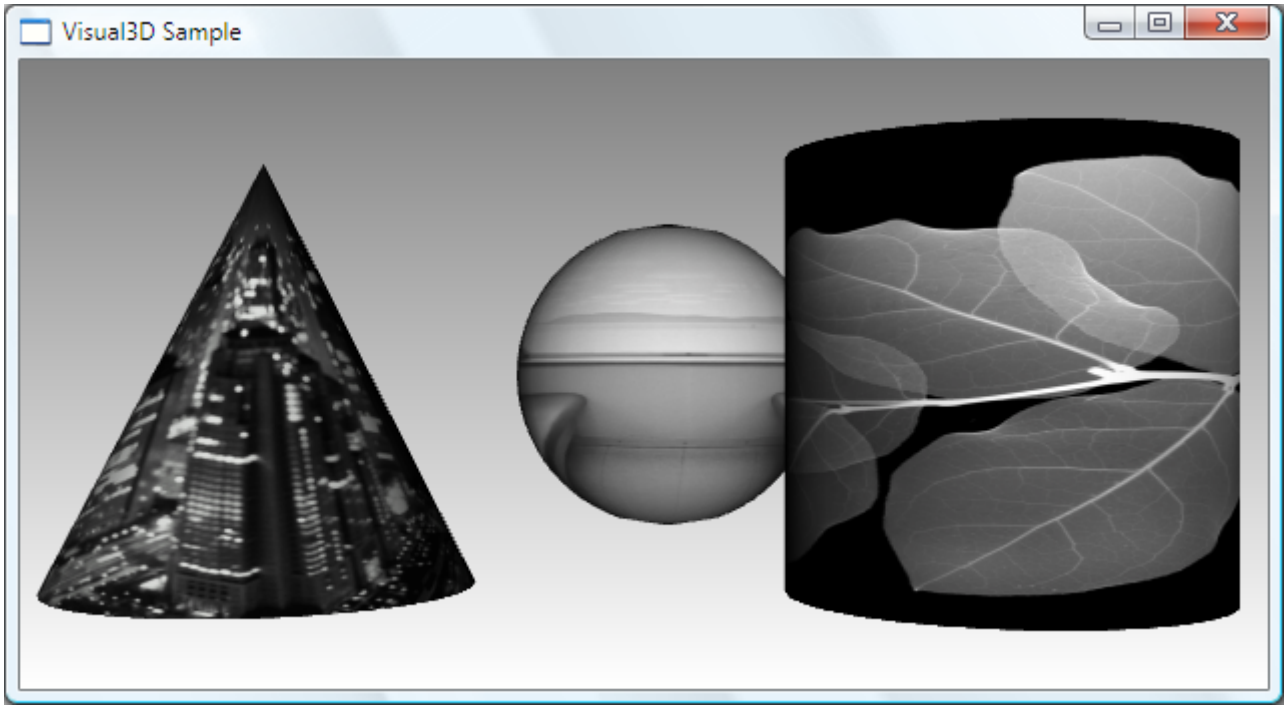
Двухмерные эффекты

Подмножество средств 2-D WPF включает визуальные эффекты, такие как градиенты, точечные рисунки, чертежи, рисунки с видео, поворот, масштабирование и наклон. Все это достигается с помощью кистей; на следующем рисунке показано несколько примеров.



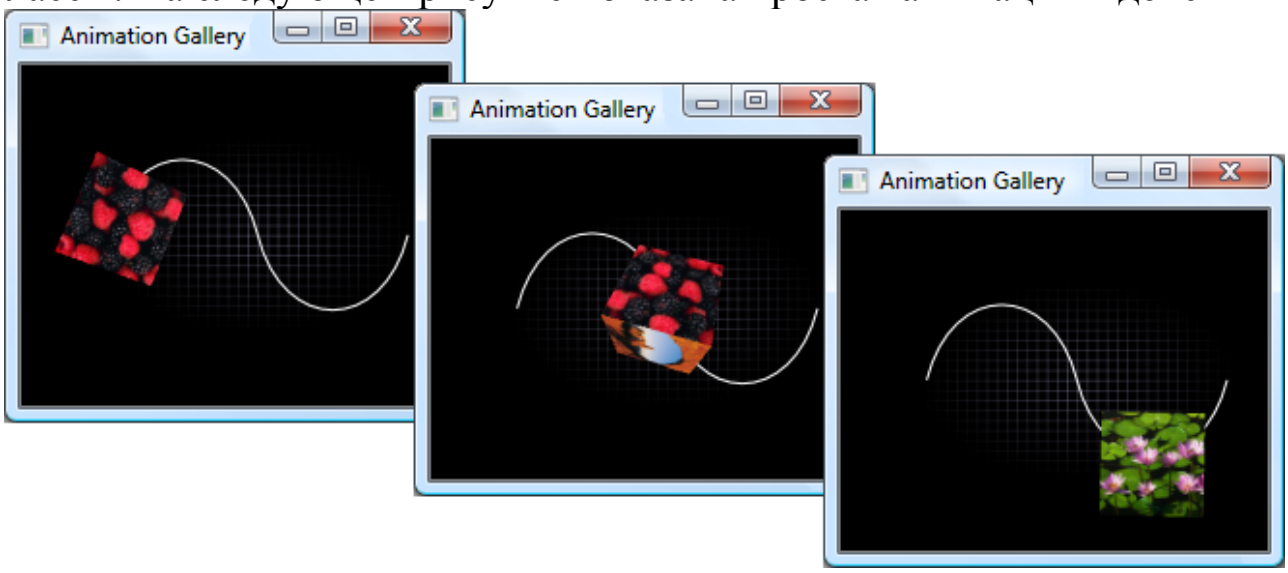
Трехмерная визуализация

WPF также включает возможности трехмерной (3-D) визуализации, интегрированные с двухмерной (2-D) графикой, что позволяет создавать более яркий и интересный UIs. Например, следующий рисунок показывает изображения 2-D, отображаемые в фигурах 3-D.



Анимация

Поддержка анимации WPF позволяет осуществлять рост, вибрацию, вращение и исчезновение элементов управления для создания интересных страничных переходов и других эффектов. Можно анимировать большинство классов WPF, даже настраиваемые классы. На следующем рисунке показана простая анимация в действии.



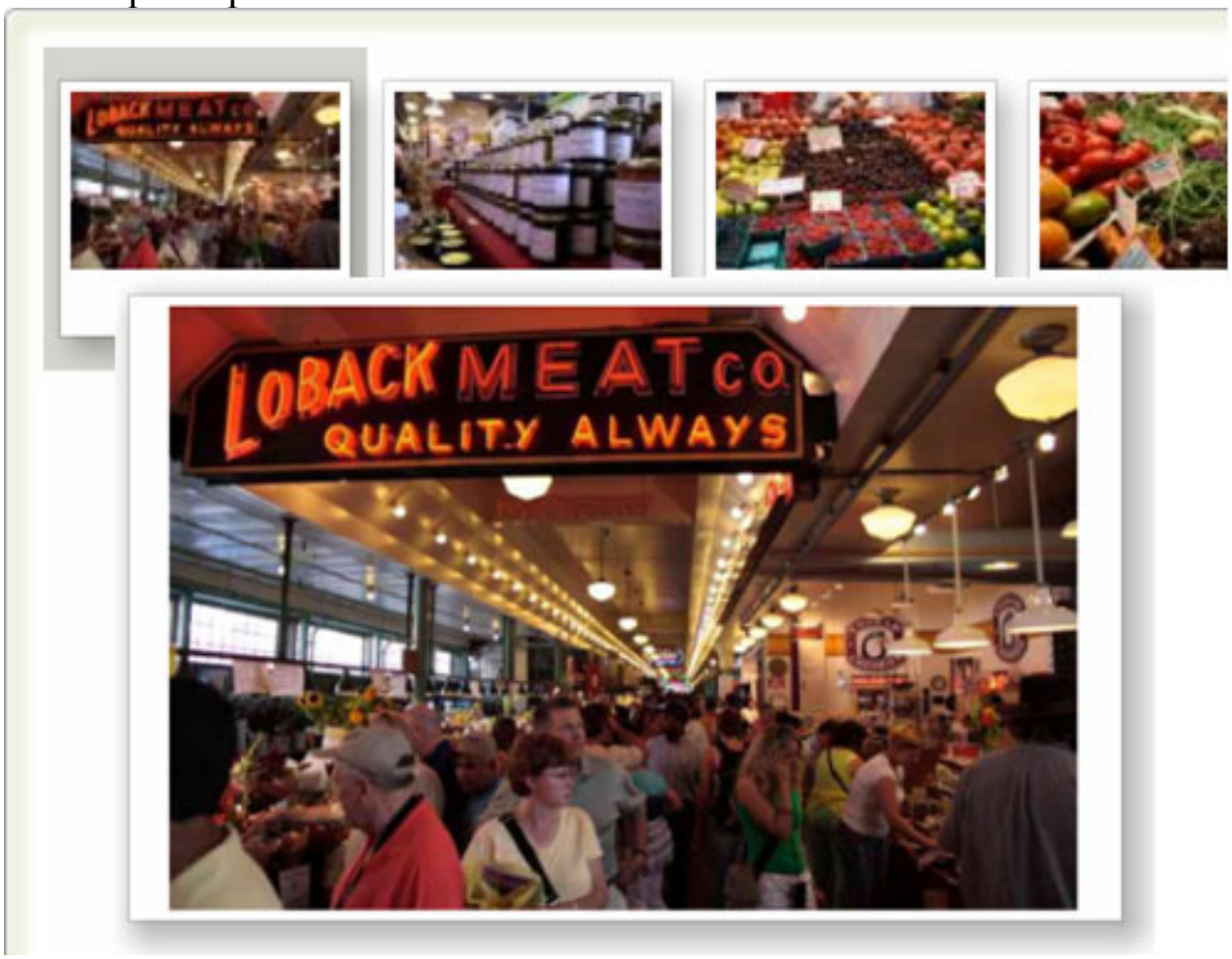
Мультимедиа

Одним из способов передачи богатого содержимого является использование аудиовизуальной среды. WPF предоставляет специальную поддержку для изображений, видео и аудио.

Изображения

Изображения присутствуют в большинстве приложений, и WPF предоставляет несколько способов их использования. На следующем

рисунке показан UI со списком, в котором содержатся эскизные изображения. При выделении эскиза изображение показывается в полном размере.



Видео и аудио

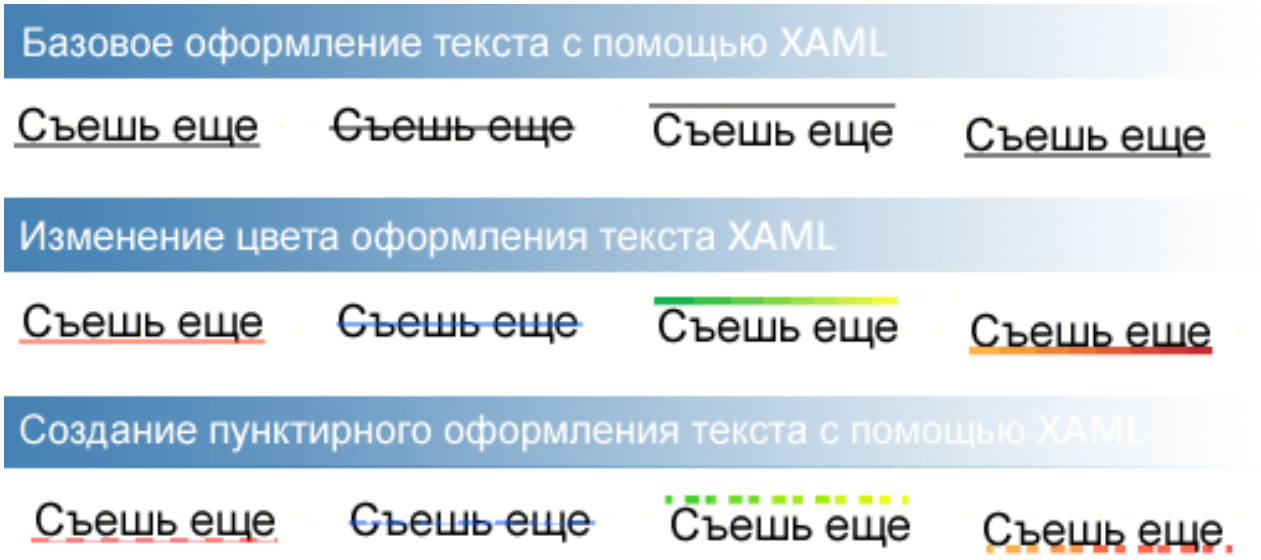
Элемент управления MediaElement способен воспроизводить видео и аудио, и является достаточно гибким, чтобы служить основой для пользовательского проигрывателя.

Текст и типография

Для облегчения отрисовки текста высокого качества WPF предоставляет следующие возможности:

- Поддержка шрифта OpenType.
- Улучшения ClearType.
- Высокая производительность, которая использует преимущества аппаратного ускорения.
- Интеграция текста с мультимедиа, графикой и анимацией.
- Механизмы резервирования и поддержки международного шрифта.

Для демонстрации интеграции текста с графикой на следующем рисунке показано применение художественного оформления текста.



Документы

WPF предоставляет встроенную поддержку работы с тремя типами документов: документами нефиксированного формата, документами фиксированного формата и документами XML Paper Specification (XPS). WPF также предоставляет службы для создания и просмотра документов, управления документами, добавления заметок, упаковки и печати документов.

Документы нефиксированного формата

Документы нефиксированного формата разработаны для оптимизации просмотра и читаемости посредством динамической настройки и обновления содержимого при изменении размера окна и параметров дисплея.

Документы фиксированного формата

Документы фиксированного формата предназначены для приложений, в которых требуется точное представление вида "что видишь, то и получишь" (режим полного соответствия изображения на экране и распечатки WYSIWYG), особенно по отношению к печати. Документы фиксированного формата обычно используются при подготовке публикаций с помощью настольных издательских средств, обработке текста и разметке формы, где строгое соблюдение исходного дизайна страницы является обязательным.

В документах фиксированного формата поддерживается точное размещение содержимого независимо от устройства. Например, документ фиксированного формата отображается на мониторе с разрешением 96 точек на дюйм точно так же, как при печати на лазерном принтере с разрешением 600 точек на дюйм или на фотонаборной машине с разрешением 4800 точек на дюйм. Макет документа остается

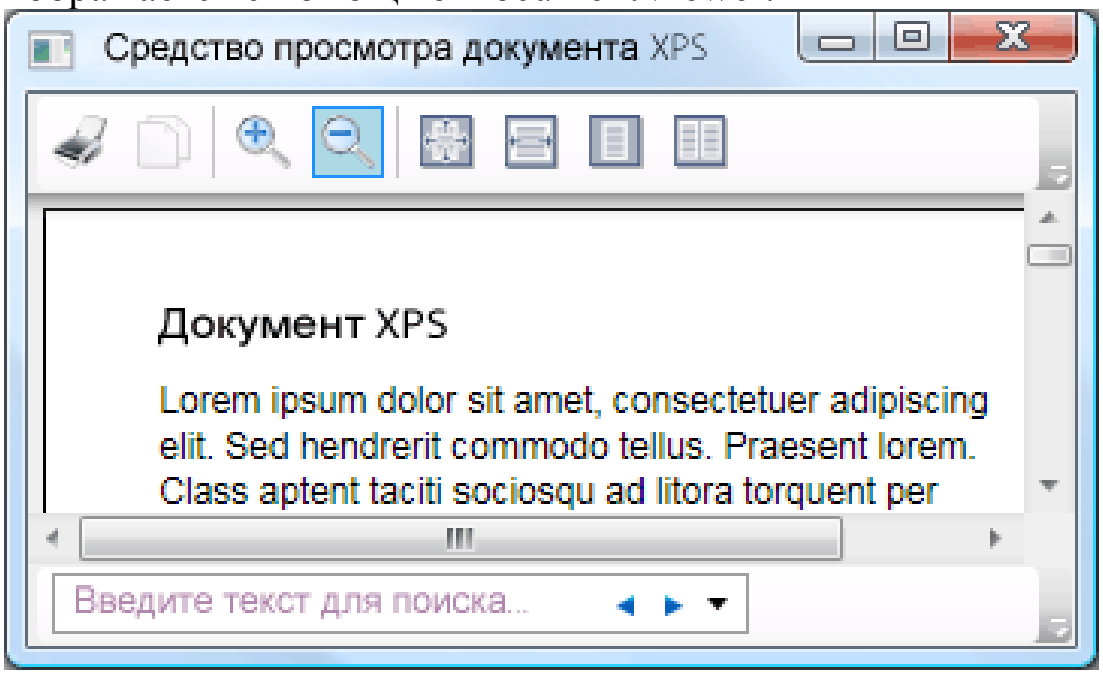
одинаковым во всех случаях, хотя качество документа варьируется в зависимости от возможностей каждого устройства.

Документы XML Paper Specification (XPS) построены на основе документов фиксированного формата WPF. Документы XPS описываются схемой на основе XML, которая фактически представляет разбитый на страницы электронный документ. Открытый кросс-платформенный формат документов XPS предназначен для упрощения создания, печати и архивирования разбитых на страницы документов, а также организации совместного доступа. Технология XPS включает следующие важные возможности:

Упаковка документов XPS в файлы ZipPackage, соответствующие стандарту Open Packaging Conventions (OPC).

- Размещение в автономных и в размещенных в браузере приложениях.
- Создание документов XPS и управление ими из приложений WPF вручную.
- Высокоточная отрисовка путем выбора устройства вывода максимального качества.
- Очередь печати принтера Windows Vista.
- Прямая отправка документов на XPS-совместимые принтеры.
- Интеграция UI с DocumentViewer.

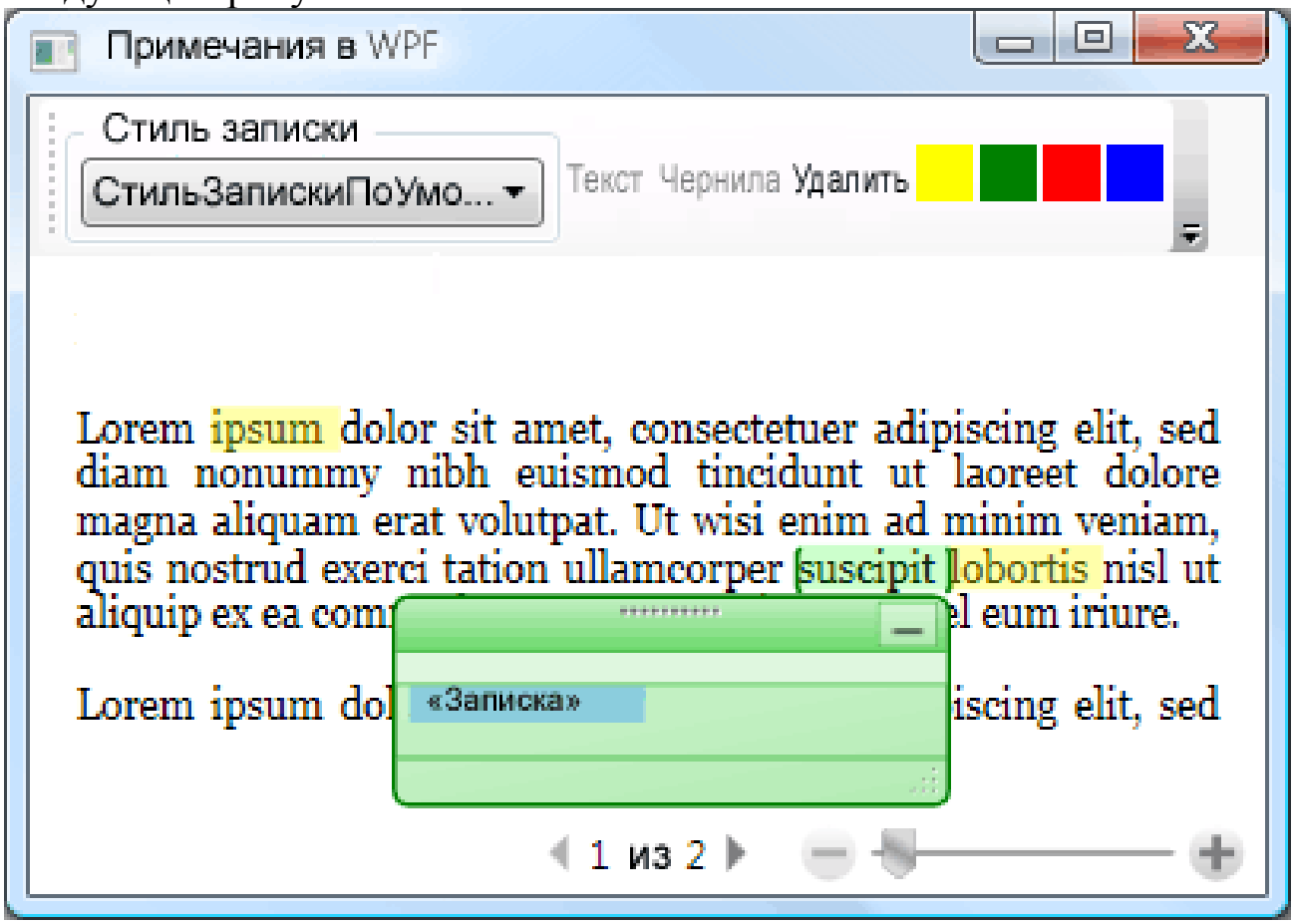
На следующем рисунке показан документ XPS, который отображается с помощью DocumentViewer.



DocumentViewer также дает возможность пользователям изменять просмотр, поиск и печать документов XPS.

Заметки

Заметки — это примечания или комментарии, которые добавляются к документу, чтобы отметить информацию или выделить интересующие элементы для дальнейшего использования. В напечатанных документах делать заметки просто, но в электронных документах возможность создания заметок часто ограничена или отсутствует. Однако в WPF для поддержки возможности создания комментариев-наклеек и выделений предоставляется система заметок. Кроме того, эти заметки можно применять к документам, размещенным в элементе управления DocumentViewer, как показано на следующем рисунке.



Упаковка

WPF System.IO.Packaging APIs позволяет приложениям организовывать данные, содержимое и ресурсы в единые, переносимые, удобные для распространения и для доступа упакованные документы. Для проверки подлинности элементов, содержащихся в пакете, можно включать цифровые подписи, которые гарантируют, что подписанный элемент не был подделан или изменен. Кроме того, можно ограничить доступ к защищенной информации, зашифровав пакеты с помощью системы управления правами.

Печать

.NET Framework включает подсистему печати, которую WPF дополняет поддержкой для расширенного управления системой печати.Улучшения печати включают следующее:

- Установка удаленных серверов и очередей печати в режиме реального времени.
- Динамическое обнаружение возможностей принтера.
- Динамическая установка параметров принтера.
- Перенаправление и изменение приоритета заданий на печать.

В документах XPS также имеется ключевое улучшение производительности. Существующий путь печати Microsoft Windows Graphics Device Interface (GDI) обычно подразумевает два преобразования:

первое — преобразование документа в формат процессора печати, например в Enhanced Metafile (EMF);

второе — преобразование в язык описания страниц принтера, например в Printer Control Language (PCL) или PostScript.

Однако документы XPS обходятся без этих преобразований, поскольку один компонент формата файла XPS является как языком обработчика заданий печати, так и языком описания страницы. Эта поддержка позволяет уменьшить как размер файла очереди, так и загрузки сетевых принтеров.

Главной задачей большей части элементов управления WPF является отображение содержимого. В WPF тип и количество элементов, которые могут составлять содержимое элемента управления, называется моделью содержимого элемента управления. Некоторые элементы управления могут содержать один элемент и один тип содержимого. Например, содержимое TextBox — строковое значение, которое присваивается свойству Text.

Триггеры

Хотя главной задачей разметки XAML и является реализация внешнего вида приложения, XAML также можно использовать для реализации некоторых действий, выполняемых приложением. Например, с помощью триггеров изменять внешний вид приложения в зависимости от действий пользователя.

Шаблоны элементов управления

UIs по умолчанию для элементов управления WPF обычно создается из других элементов управления и

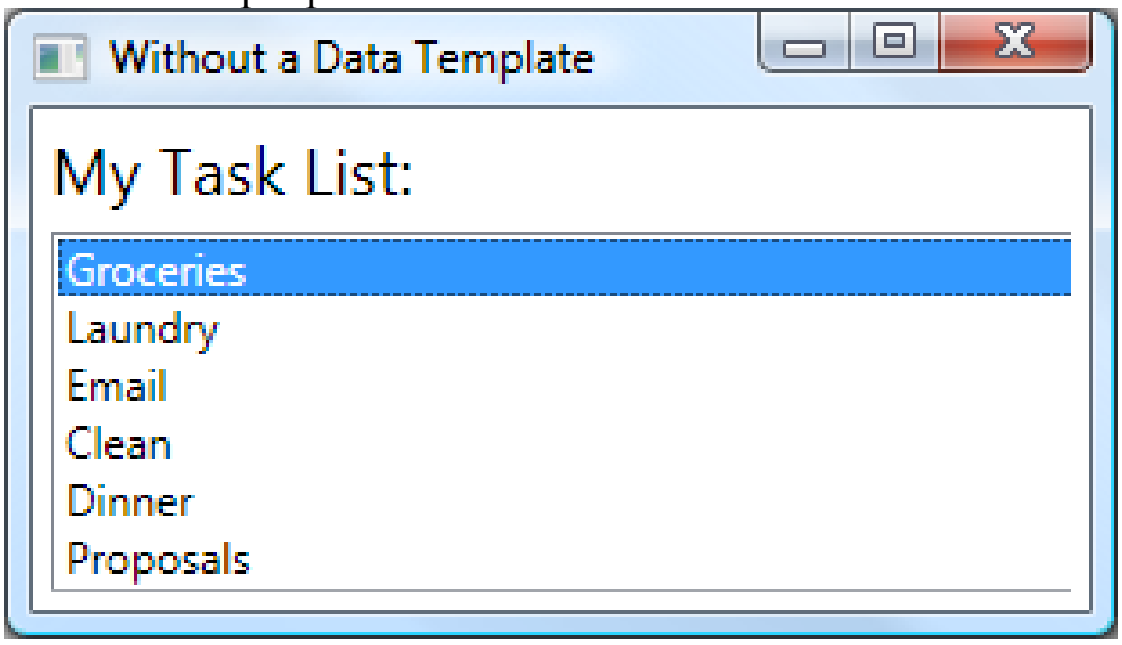
фигур. Например, Button состоит из элементов управления ButtonChrome и ContentPresenter.

ButtonChrome обеспечивает стандартный внешний вид кнопки, в то время какContentPresenter отображает содержимое кнопки, заданное свойством Content.

Иногда внешний вид элемента управления по умолчанию может не сочетаться с общим внешним видом приложения. В этом случае можно с помощью ControlTemplate изменить внешний вид UI элемента управления без изменения его содержимого и поведения.

Шаблоны данных

В то время как шаблон элемента управления позволяет задавать внешний вид элемента управления, шаблон данных позволяет задавать внешний вид содержимого элемента управления. Шаблоны данных часто используются для улучшения способа отображения данных, связанных с элементом управления. На следующем рисунке показан внешний вид по умолчанию элемента управления ListBox, который связан с коллекцией объектов Task, где каждый объект имеет имя, описание и приоритет.



Внешний вид по умолчанию — это вид, который ожидается от ListBox. Однако внешний вид по умолчанию каждого объекта коллекции содержит только имя. Чтобы отобразить имя, описание и приоритет объекта коллекции, внешний вид по умолчанию элементов связанного списка элемента управления ListBoxдолжен быть изменен с помощью DataTemplate.

Ресурсы

Элементы управления в приложении должны использовать один и тот же внешний вид, который может включать любые шрифты и цвета фона для шаблонов элементов управления, шаблонов данных и стилей. С помощью поддержки WPF для ресурсов user interface (UI) можно инкапсулировать эти ресурсы в одном расположении для повторного использования.

Темы и обложки

С точки зрения визуального восприятия тема определяет глобальный внешний вид системы Windows и приложений, которые в ней запускаются. Windows поставляется с несколькими темами. Например, Microsoft Windows XP поставляется с темами Windows XP и Windows Classic, а Windows Vista поставляется с темами Windows Vista и Windows Classic. Внешний вид, определяемый темой, задает внешний вид по умолчанию для приложения WPF. Однако WPF не поддерживает прямую интеграцию с темами Windows. Поскольку внешний вид WPF определяется шаблонами, WPF включает по одному шаблону для каждой известной темы Windows, в том числе Aero (Windows Vista), Classic (Microsoft Windows 2000), Luna (Microsoft Windows XP) и Royale (Microsoft Windows XP Media Center Edition 2005). Эти темы упакованы в словари ресурсов, которые применяются, если ресурсы не найдены в приложении. Внешний вид многих приложений задается с помощью этих тем; сохраняющаяся согласованность с внешним видом Windows помогает пользователям быстрее освоиться с большинством приложений.

С другой стороны, опыт работы пользователя с некоторыми приложениями не обязательно связан с стандартными темами. Например, Microsoft Windows Media Player работает с аудио- и видеоданными, и здесь преимущество имеют пользователи с опытом работы в другом стиле. Такие UIs чаще предоставляют настраиваемые, специфичные для приложения темы. Такие темы называются "обложки", и приложения, которые их используют, часто предоставляют средства настройки различных аспектов обложек. Microsoft Windows Media Player имеет множество собственных обложек и обложек от сторонних производителей.

Пользовательские элементы управления

Хотя WPF и обеспечивает поддержку настройки, могут возникнуть ситуации, в которых существующие элементы управления WPF не удовлетворяют требованиям приложения или его пользователей. Это возможно в следующих ситуациях:

Нужный UI не может быть создан путем настройки внешнего вида и поведения существующих реализаций WPF.

Нужное поведение не поддерживается (или поддерживается частично) существующими реализациями WPF.

Тем не менее, в этом случае можно воспользоваться одной из трех моделей WPF для создания нового элемента управления. Каждая модель предназначена для определенного скрипта и требует, чтобы пользовательский элемент управления был производным от конкретного базового класса WPF. Далее приводится описание этих трех моделей.

Модель пользовательского элемента управления. Пользовательский элемент управления производится из UserControl и состоит из одного или нескольких других элементов управления.

Модель элемента управления. Пользовательский элемент управления производится из Control и используется для построения реализаций, в которых внешний вид и поведение разделены с помощью шаблонов, подобно большей части элементов управления WPF. Создание элемента управления, производного от Control, предоставляет по сравнению с пользовательскими элементами управления большую свободу для создания нестандартного UI, но может потребовать дополнительных усилий.

Модель элемента .NET Framework. Пользовательский элемент управления производится от FrameworkElement, когда его внешний вид определяется пользовательской логикой визуализации (не шаблонами).

Советы и рекомендации по WPF

Как любая платформа разработки, WPF может использоваться множеством способов для достижения нужного результата. Для гарантий, что приложения WPF предоставляют требуемое взаимодействие с пользователем и удовлетворяют требованиям аудитории в целом, в данном разделе предлагаются советы и рекомендации по специальным возможностям, глобализации и локализации, а также производительности.

Рассмотрим примеры работы с платформой .NET на примере выполнения задания для демонстрационного экзамена 2016 года(WPF).

Ознакомиться с полной версией задания и использованными ресурсами можно на ресурсе Академии Ворлдскиллс Россия по ссылке: https://drive.google.com/drive/u/0/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-lkn

Сессия 1 данного Конкурсного задания состоит из следующей документации / файлов:

1. WSR2018_TP09_14+_C1.pdf (Инструкция к первой сессии)

2. marathon-skills-2018-database-mysql.sql (Сценарий SQL для создания таблиц с данными для MySQL)

3. marathon-skills-2018-database-mssql.sql (Сценарий SQL для создания таблиц с данными для Microsoft SQL Server)

4. marathon-skills-2018-staff-import.xlsx (Данные для импорта)

5. marathon-skills-2018-testing-data-s1.pdf (Тестирование системы)

ВВЕДЕНИЕ

В этой сессии вы начнете разработку приложения и базы данных для MarathonSkills 2016. Дизайнер предоставил вам набор системной документации, так что вы можете построить систему в соответствии с потребностями клиента. Найдите время для знакомства с предоставленными материалами.

Создайте базу данных, а затем импортируйте туда необходимые данные. Затем создайте приложение: часть окон, которые будут доступны пользователям системы.

Файл: marathon-skills-2018-testing-data-s1.pdf предоставлен вам для того, чтобы вы смогли протестировать систему

ИНСТРУКЦИЯ УЧАСТНИКУ

К концу этой сессии, у вас должны быть достигнуты следующие результаты, необходимые для того, чтобы заказчик был спокоен, что система будут завершена вовремя. Убедитесь, что вы следуете предоставленному руководству по стилю во всех частях системы. Убедитесь, что вы предоставляете соответствующие проверки и сообщения об ошибках во всех частях системы. Убедитесь, что все соответствующие кнопки / ссылки работают в конце сессии. Убедитесь, что вы используете соответствующие соглашения об именах для всех частей системы по мере необходимости.

ПРАКТИЧЕСКИЕ РЕЗУЛЬТАТЫ

1.1 СОЗДАНИЕ БАЗЫ ДАННЫХ Создайте базу данных, используя знакомую вам платформу (MySQL / MSSQLServer) на сервере баз данных, который вам предоставлен.

1.2 ЗАГРУЗКА ДАННЫХ Сценарий SQLпредоставлен для вас, чтобы создать большинство таблиц и вставки данных в них. Все, что вам нужно сделать, это импортировать сценарий SQL в вашу базу данных. Выберите сценарий SQL, который подходит для вашей платформы: MySQL: marathon-skills-2016-database-mysql.sql СерверSQL: marathon-skills-2016-database-mssql.sql Таблица Сотрудники (персонал, должности) не включены в этот сценарий SQL. См результаты 1.3 и 1.4.

1.3 СОЗДАТЬ ТАБЛИЦЫ ДЛЯ ПЕРСОНАЛА СОГЛАСНО СПЕЦИФИКАЦИИ Обратитесь к диаграмме базы данных (ERD) и словарю данных. Создайте таблицы сотрудников (Position, Staff, Timesheet) согласно спецификации.

1.4 ИМПОРТ ДАННЫХ ПЕРСОНАЛА Все данные сотрудников были представлены в marathon-skills-2018-staff-import.xlsx. Эти данные не отформатированы для импортирования непосредственно в базу данных, вам необходимо отформатировать данные и загрузить их в таблицы, которые вы только что создали. Поле Summary Information не требуется. В поле "FullName" в формате "Имя Фамилия" используются разные символы разделителя. Убедитесь, что адреса электронной почты в правильном формате.

1.5 СОЗДАТЬ ПРИЛОЖЕНИЕ Создайте приложение, используя выбранную вами платформу .NET (или Java).

1.6 СОЗДАНИЕ "1. ГЛАВНЫЙ ЭКРАН СИСТЕМЫ" Создайте главное меню системы, как указано в "1. Главный экран системы"в презентации. Каждое окно / страница приложения, который имеет "? дней? часов и? минут до начала гонки в "в нижней части экрана должно автоматически обновляться в режиме реального времени. Рассчитать количество времени, оставшегося до начала первого MarathonSkills 2018 начинается (2018-11-24 6:00).

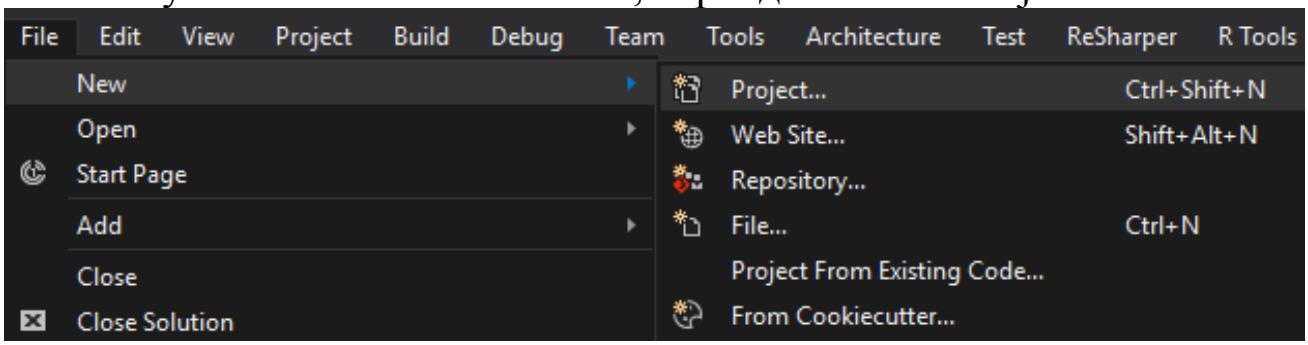
1.7 СОЗДАНИЕ ОКНА "7. ПОДРОБНАЯ ИНФОРМАЦИЯ” Создание окна подробная информация как указано в “7. Подробная информация” в презентации.

1.8 СОЗДАНИЕ ОКНА "10. СПИСОК БЛАГОТВОРИТЕЛЬНЫХ ОРГАНИЗАЦИЙ" Создать страницу, как описано в "10. Список благотворительных организаций»в презентации. Эта страница

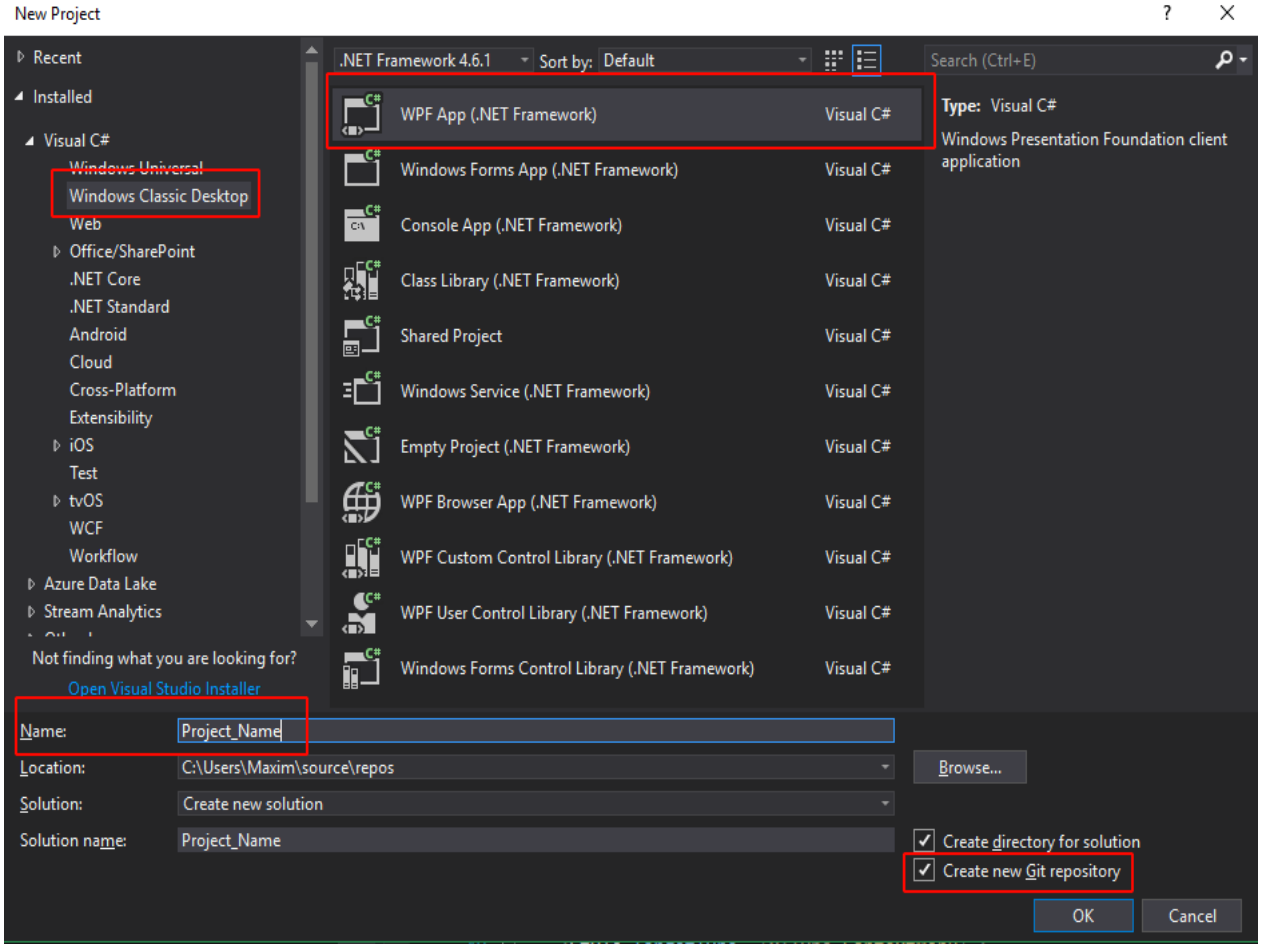
отображает все благотворительные организации, перечисленные в базе данных вместе с их логотипами (при условии, что они есть в общих ресурсах), чтобы показать бегунам благотворительные организации, которые их могут поддержать.

СОЗДАНИЕ ПРОЕКТА

Запустите Vi sual Studio 2017, перейдите File – Project.



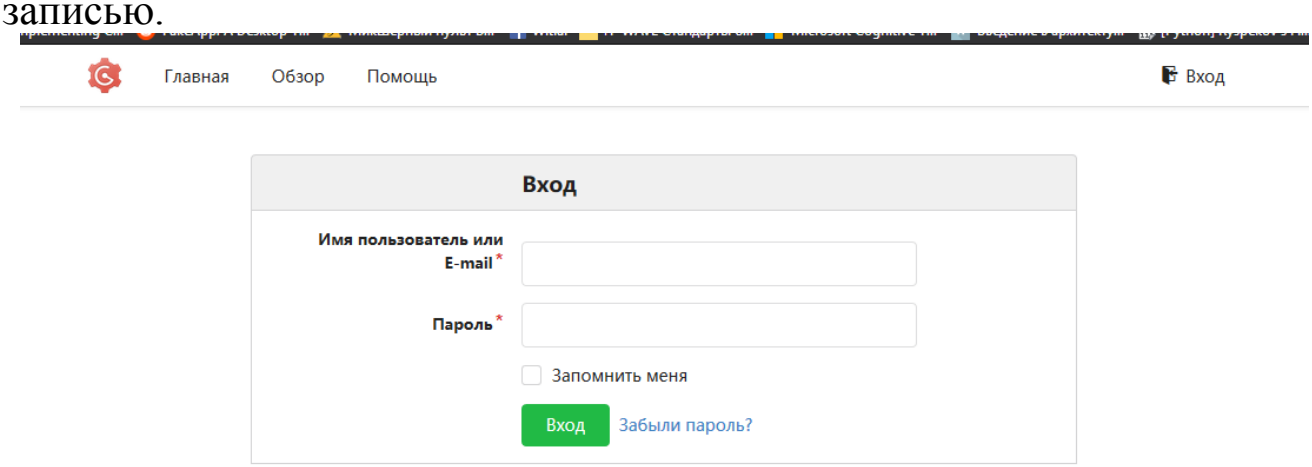
Выберите Visual C# - Windows Classic Desktop – WPF App, укажите имя проекта, а также поставьте галочку «Create new git repositoty» чтобы включить проект в систему контроля версий.



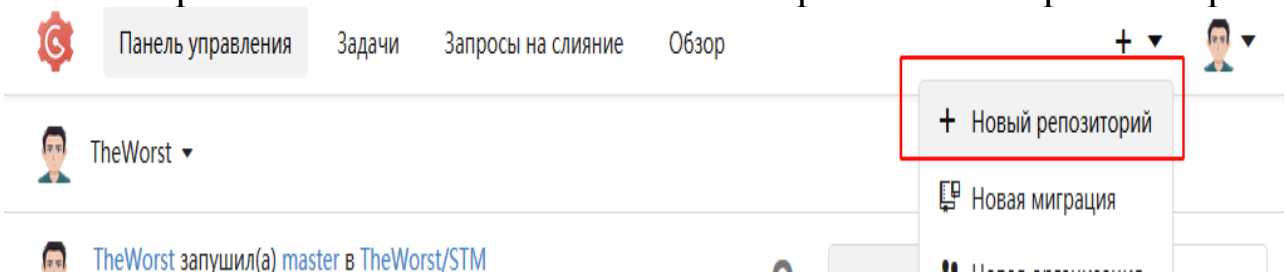
После нажатия кнопки «OK» создастся проект с пустой формой.

СОЗДАНИЕ РЕПОЗИТОРИЯ

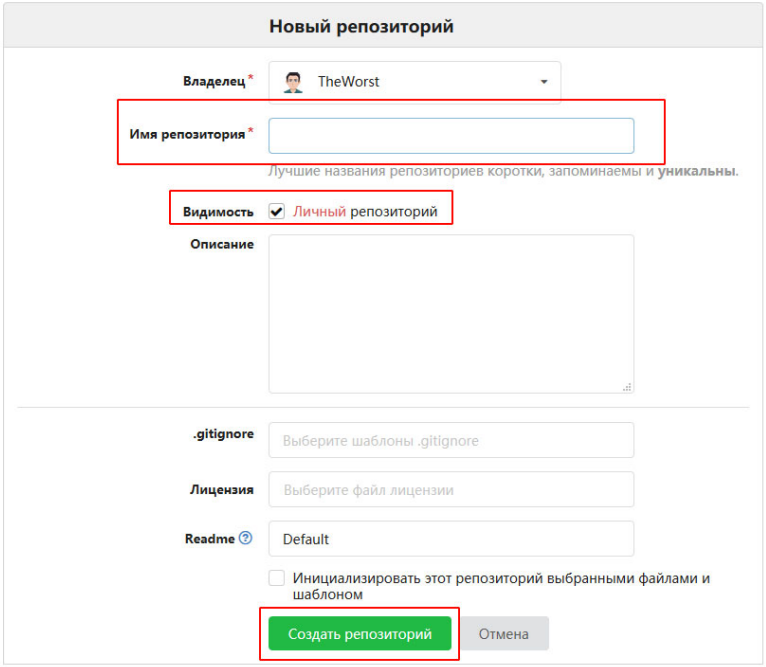
Откройте в браузере GOGS. Авторизуйтесь под своей учетной записью.



На верхней панели нажмите «+» и выберите «Новый репозиторий».



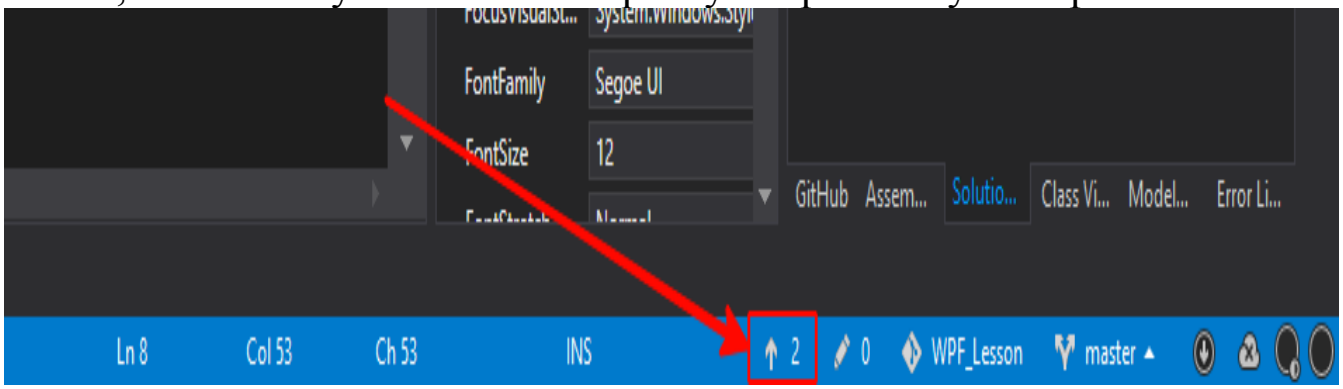
После чего укажите имя репозитория, поставьте галочку что репозиторий будет являться личным. И создайте его путем нажатия на«Создать репозиторий».



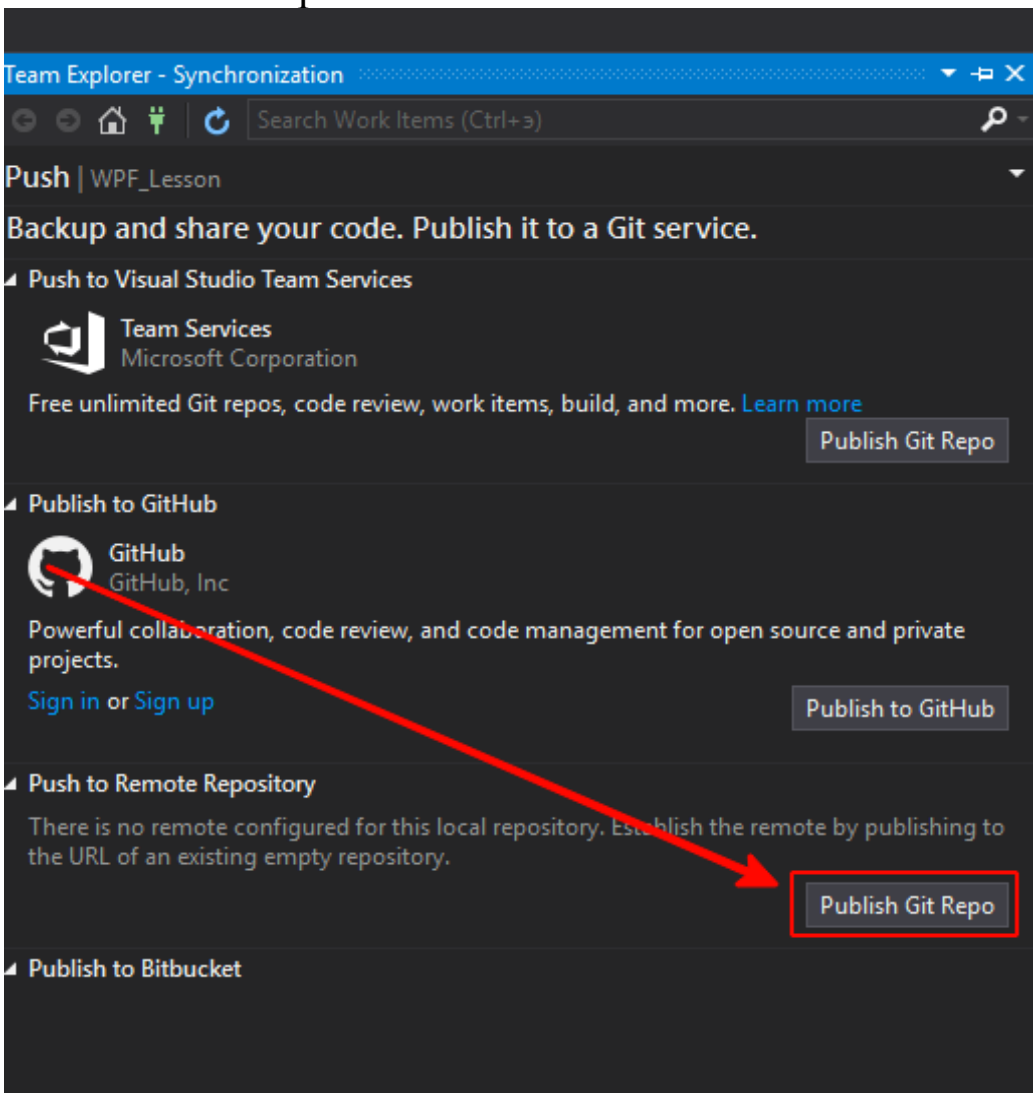
После создания репозитория появится следующее окно.

Создадим в проекте «Resource Dictionary».

Для того чтобы подключить проект, откройте проект в Visual Studio, затем внизу нажмите на стрелку направленную вверх.

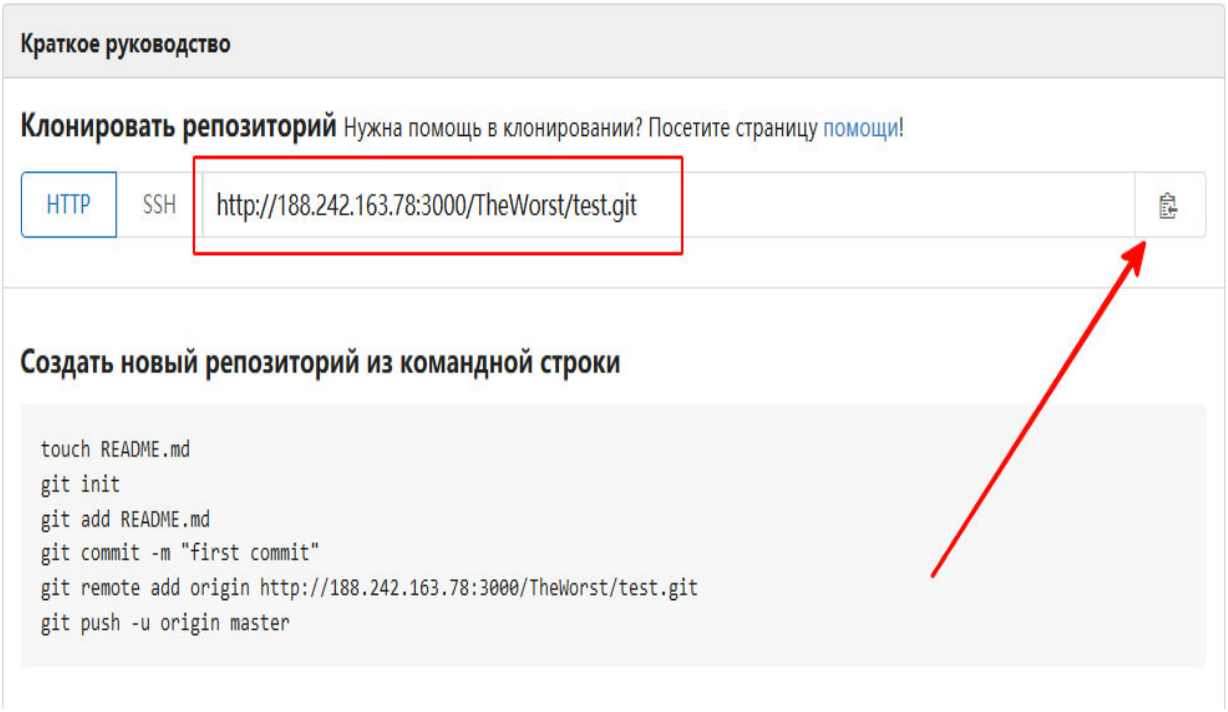


После нажатия у вас откроется окно «Team Explorer», в котором нажмите «Publish Git Repo»

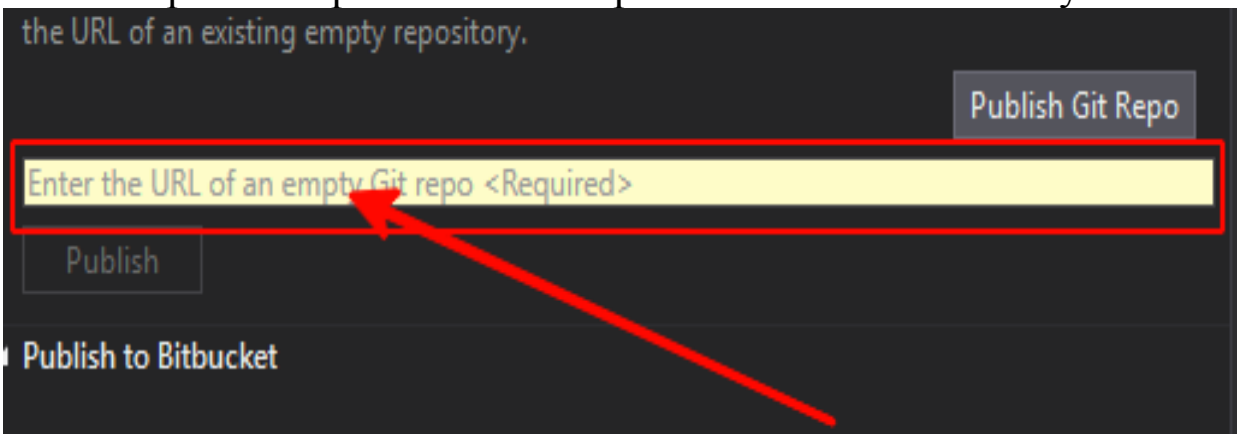


После чего у вас откроется окно в которое необходимо вставить ссылку на репозиторий. Для получения ссылки, откройте GOGS,

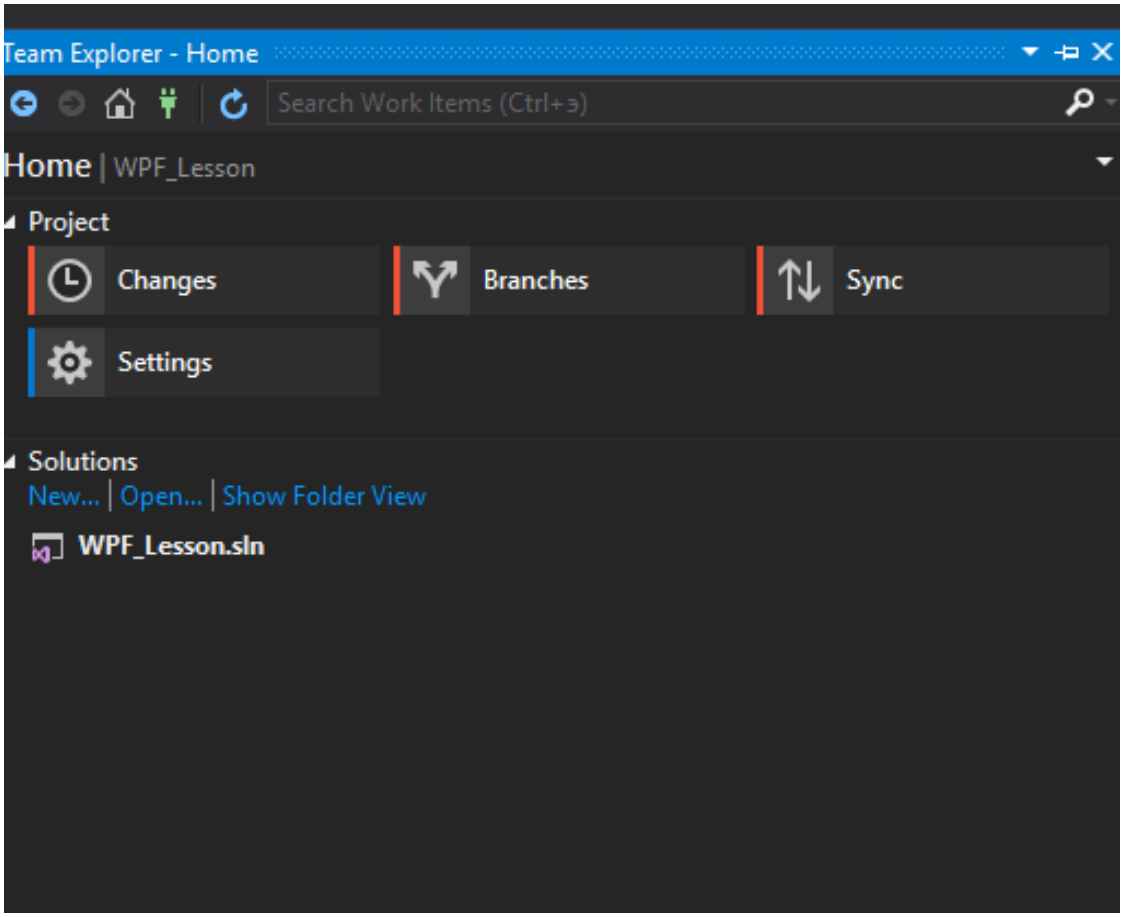
авторизуйтесь, зайдите в созданный репозиторий (урок Создание репозитория) и там скопируйте на него ссылку.



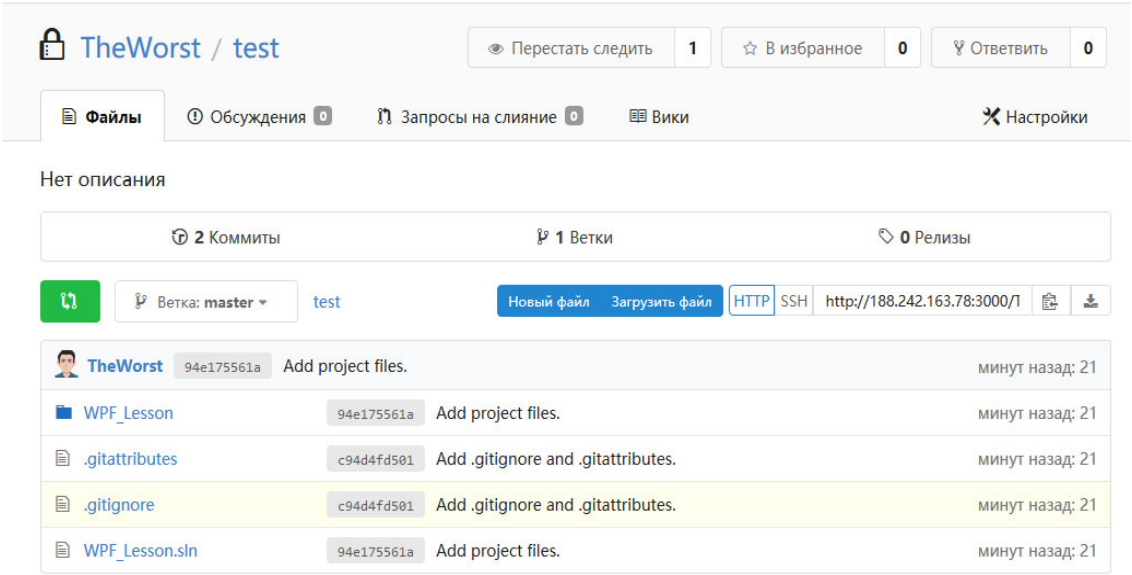
После того как скопируете ссылку вставьте ее в поле подключения удаленного репозитория в «Team Explorer» и нажмите кнопку «Publish»



После нажатия кнопки, проект будет сохранен на удаленном репозитории, а также создастся связь между локальным и удаленным репозиторием.



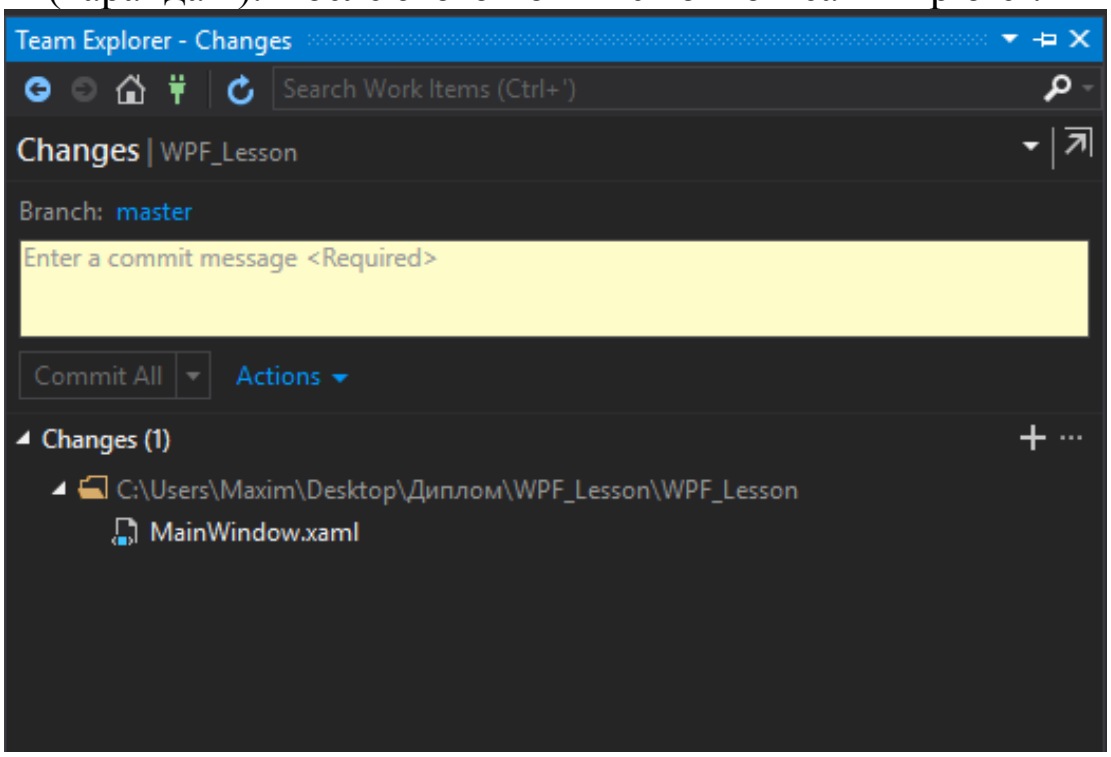
Проверить результат сохранения проекта можно на GOGS в репозитории.



СОХРАНЕНИЕ ИЗМЕНЕНИЙ В РЕПОЗИТОРИИ

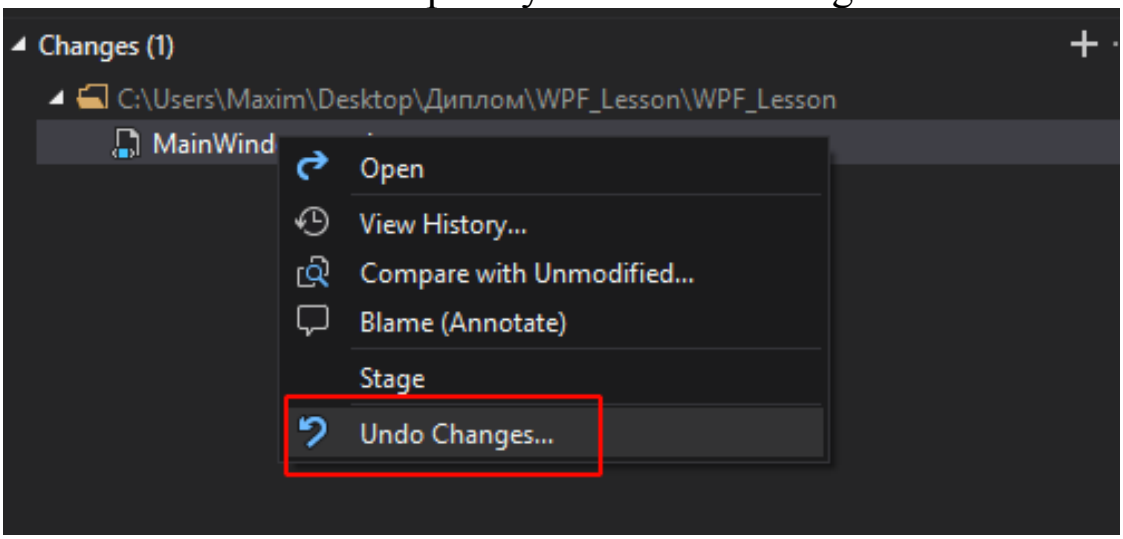
По мере того как будут вноситься правки в проект, счетчик изменений будет расти (указывается внизу Visual Studio)

Чтобы сохранить проект в репозитории, необходимо нажать на счетчик (карандаш). После этого появится окно Team Explorer.

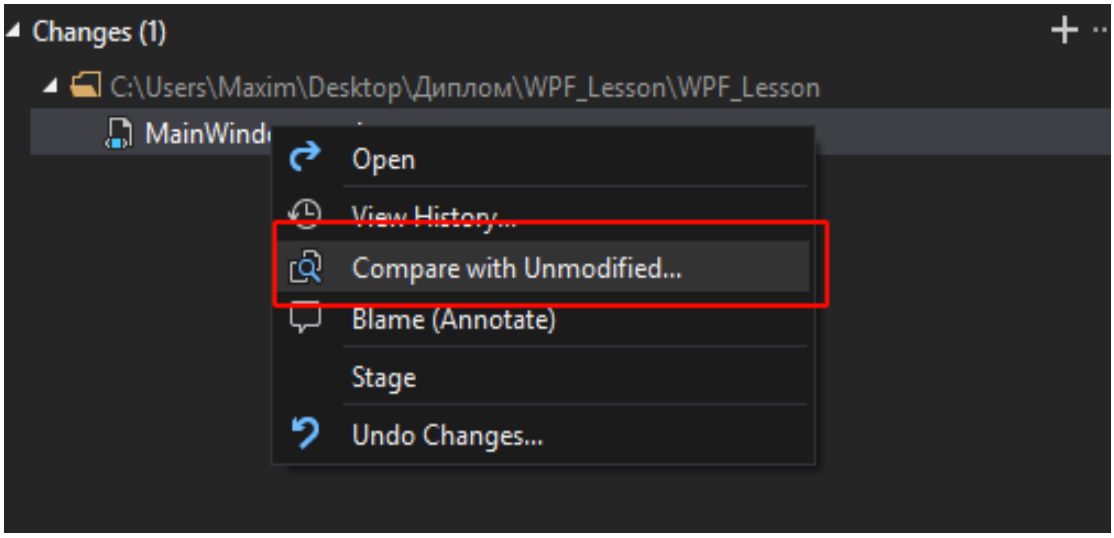


В данном окне показывается ветка (branch) в которую будет сохранятся изменения и файлы (Changes) в которых произошли изменения.

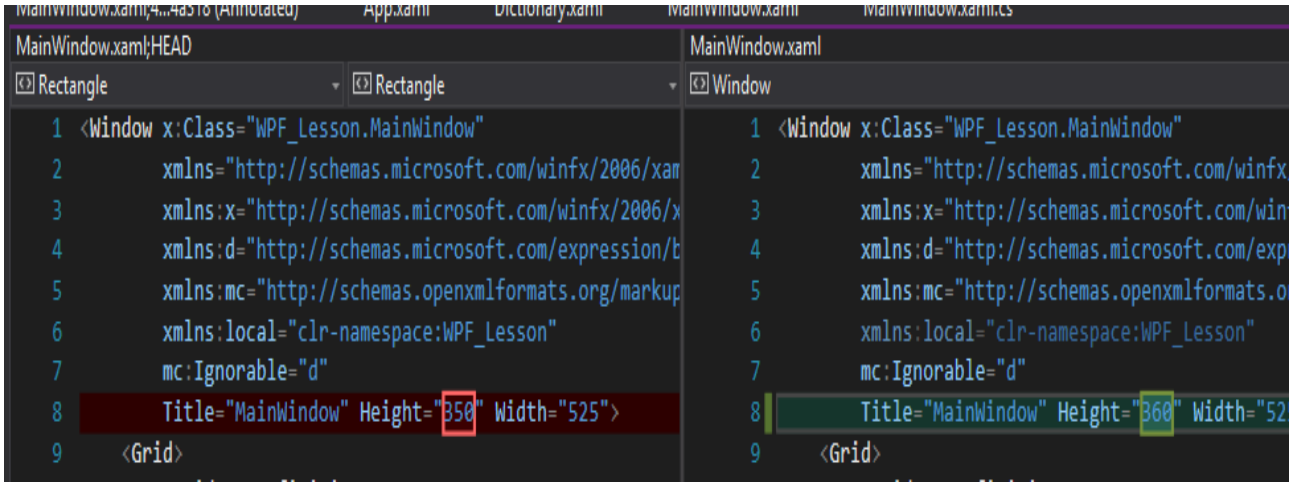
Если изменения были произведены ошибочно, то их можно откатить на предыдущею версию, путем нажатия на необходимом файле правой кнопкой мыши и выбрав пункт «Undo Changes»



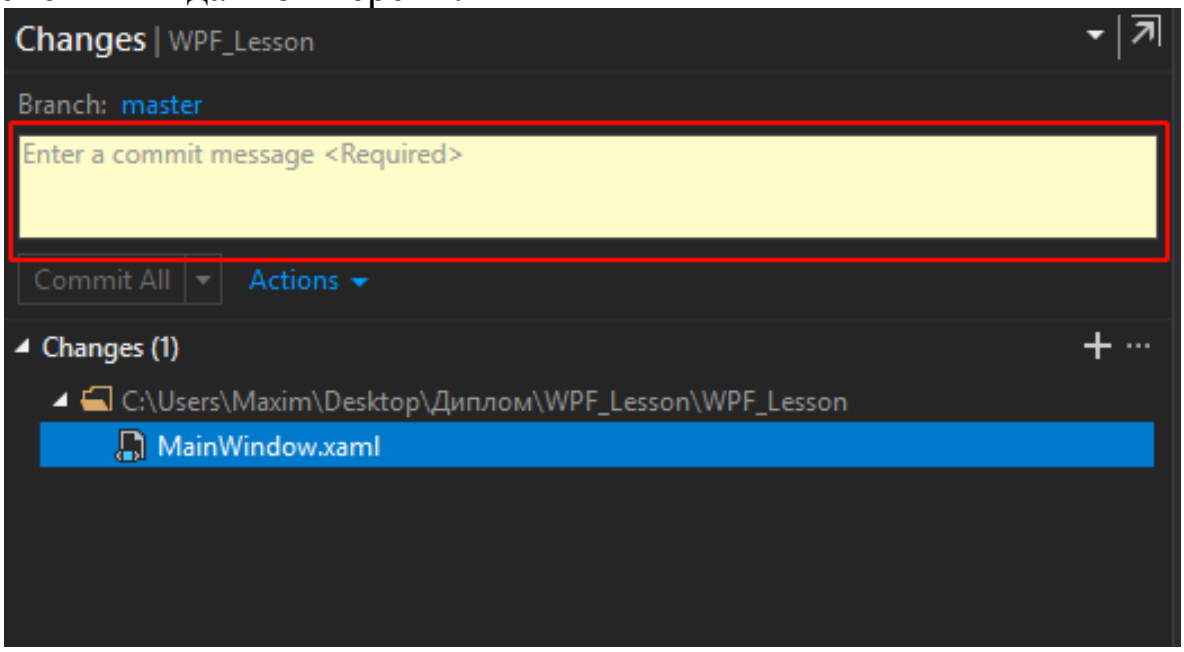
Если необходимо посмотреть изменения в файле (сравнить с предыдущей версией), то необходимо нажать правую кнопку мыши на нужном файле и выбрать пункт меню «Compare with Unmodified»



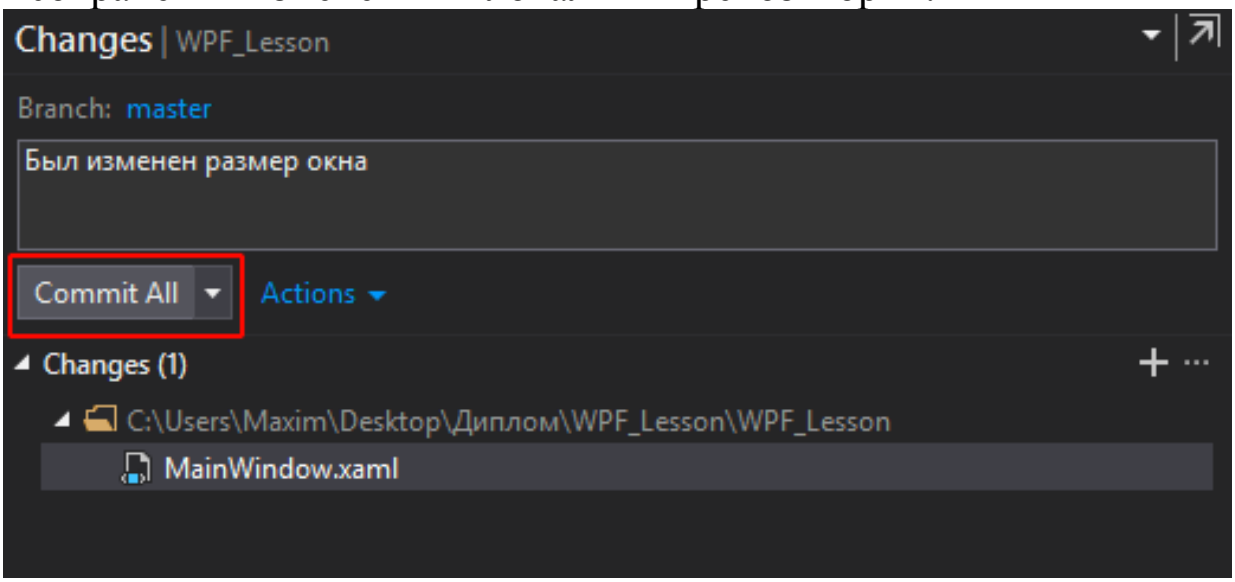
После чего откроется окно с изменениями, в одном окне предыдущая версия, в другом актуальная.



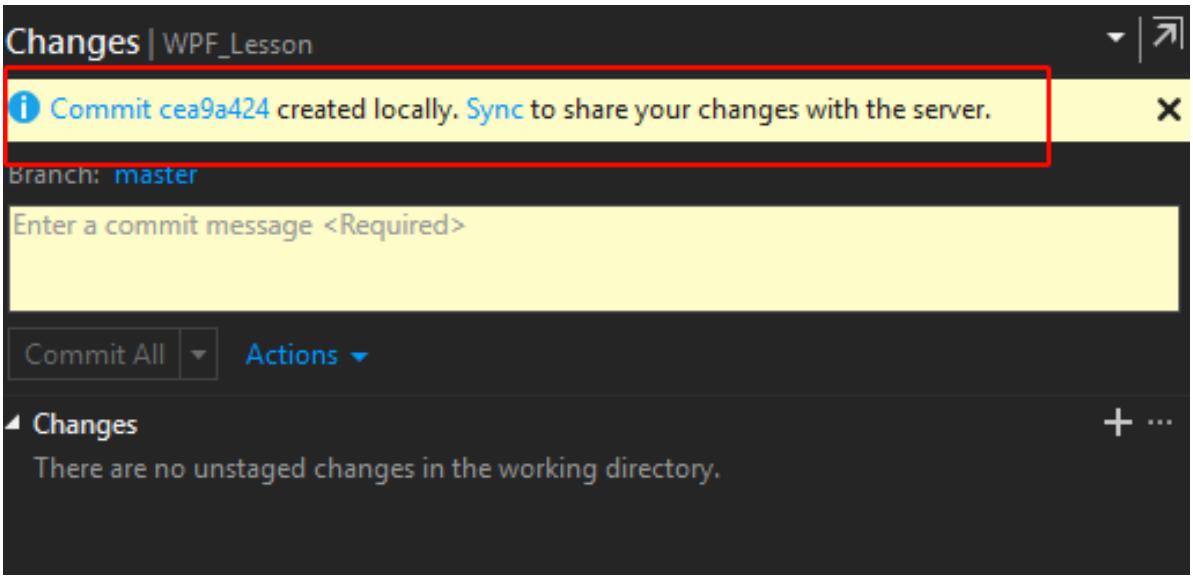
Для того чтобы сохранить изменения необходимо написать текст об изменениях в данной версии.



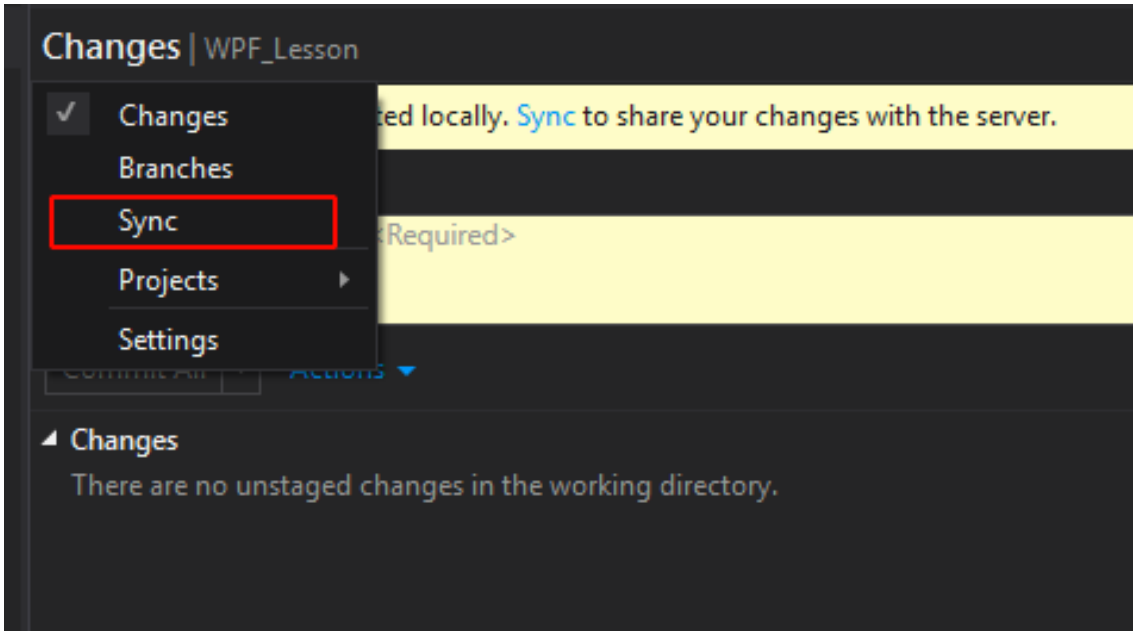
После ввода изменений, необходимо нажать кнопку «Commit All» для сохранения изменений в локальный репозиторий.



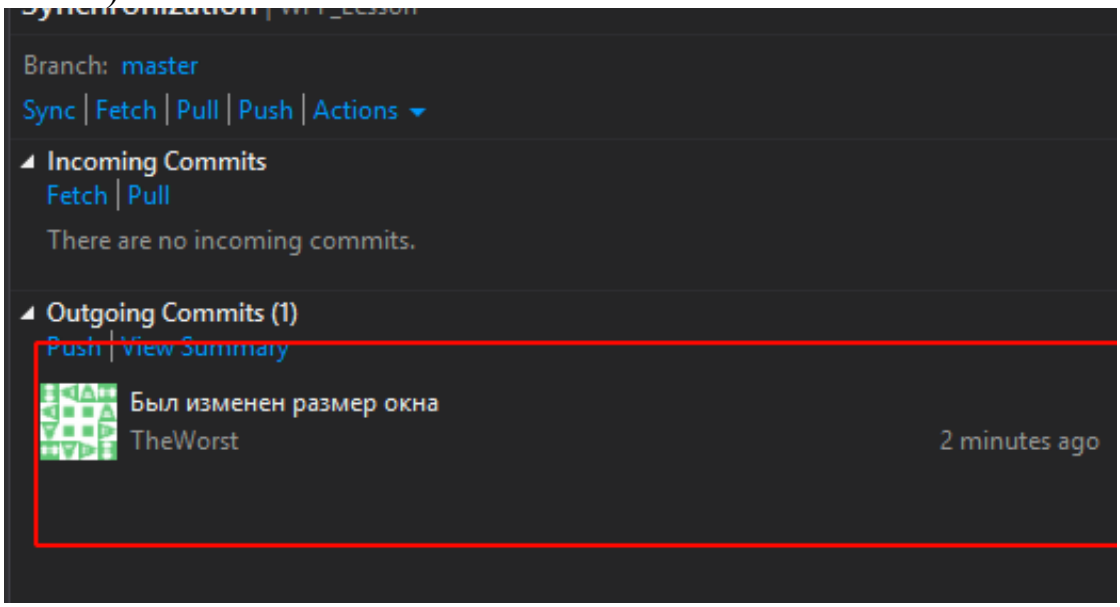
После чего будет выведено сообщение о успешном сохранении проекта. Так же будет предложено синхронизировать локальный репозиторий с удаленным.



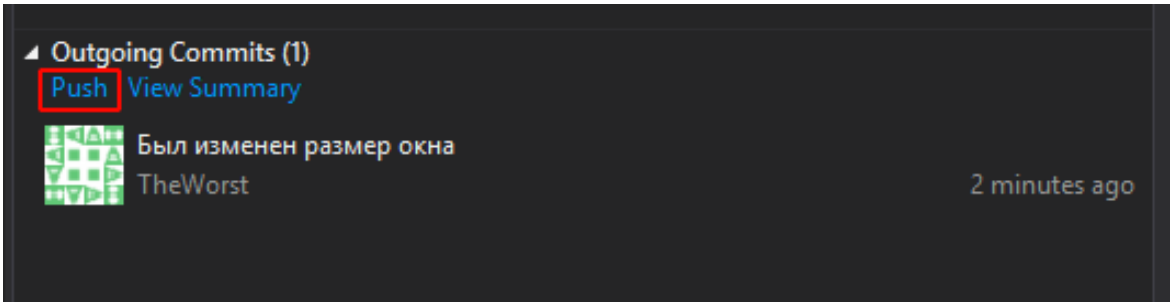
Теперь необходимо отправить изменения локального репозитория на удаленный репозиторий. Для этого необходимо выбрать пункт Sync из выпадающего меню.



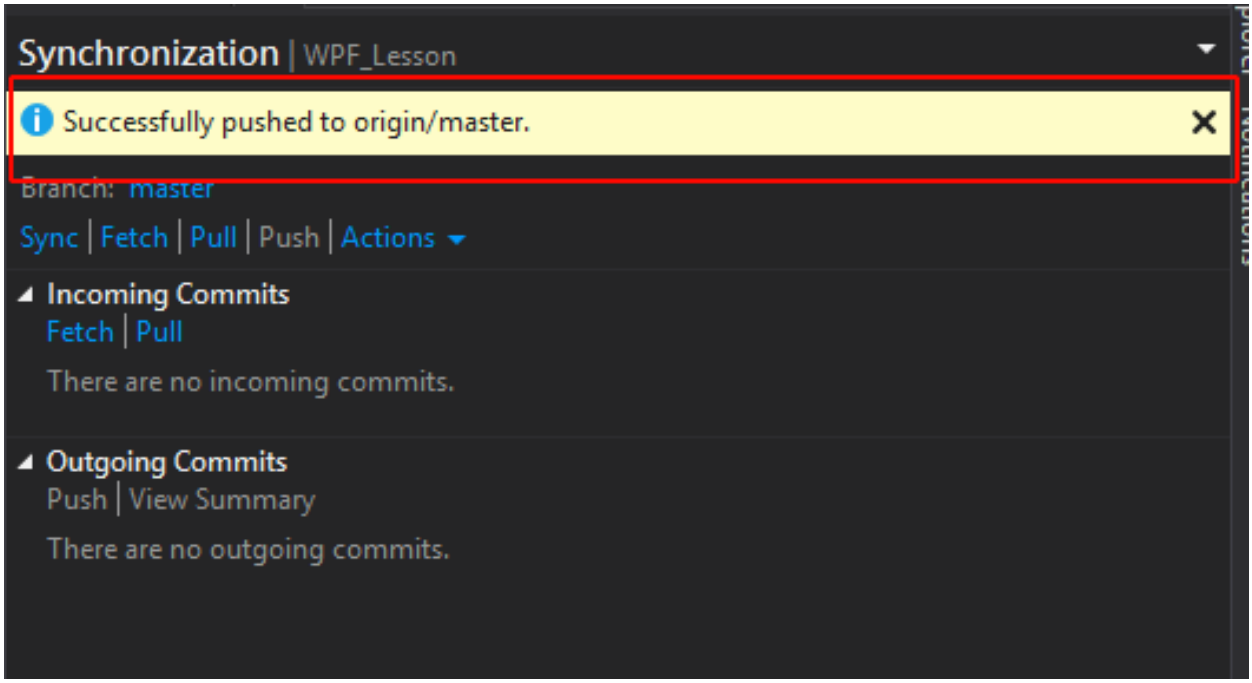
В данном пункте меню, внизу будет отображаться ваш коммит (изменения).



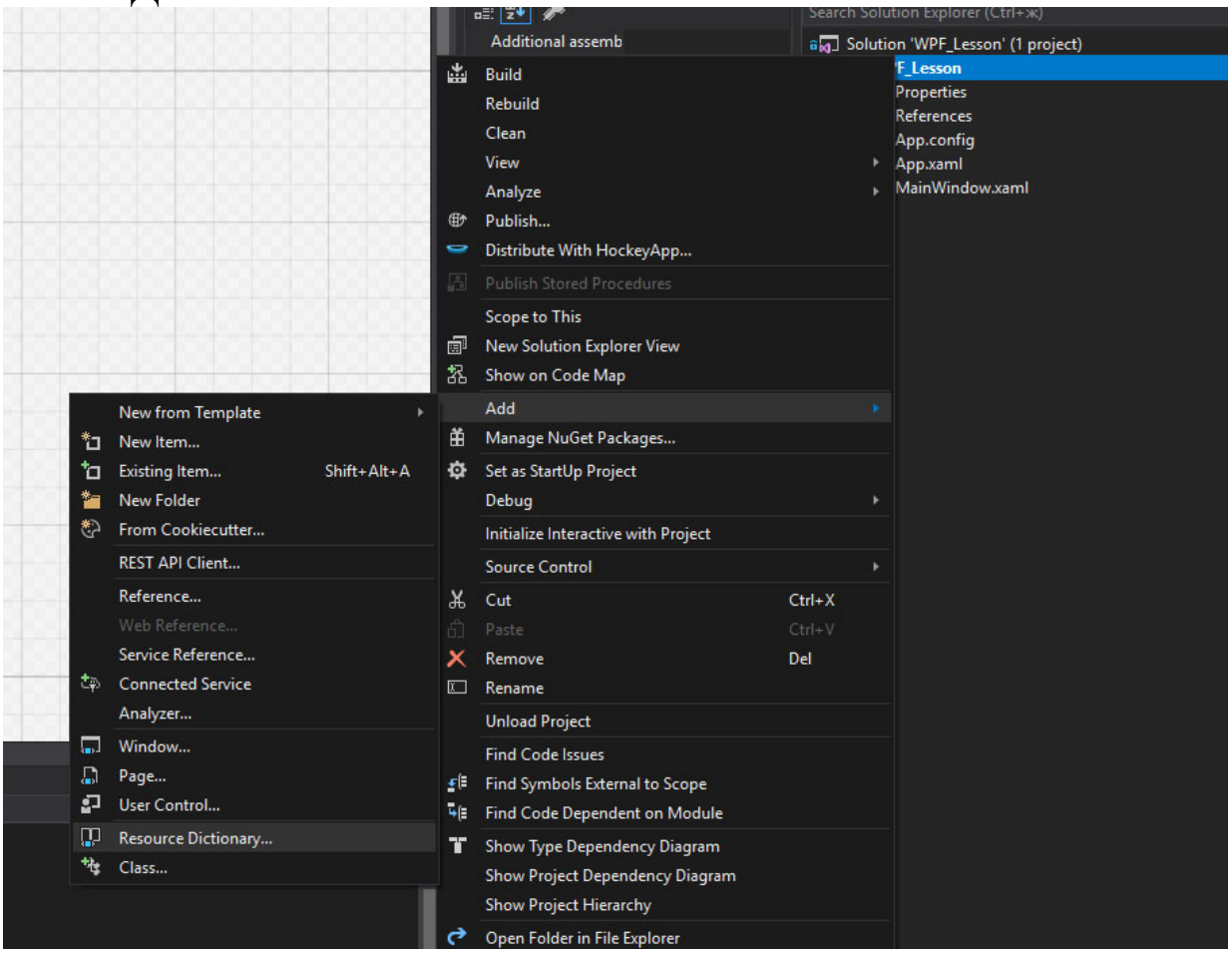
Для того чтобы отправить изменения необходимо нажать на кнопку «Push»



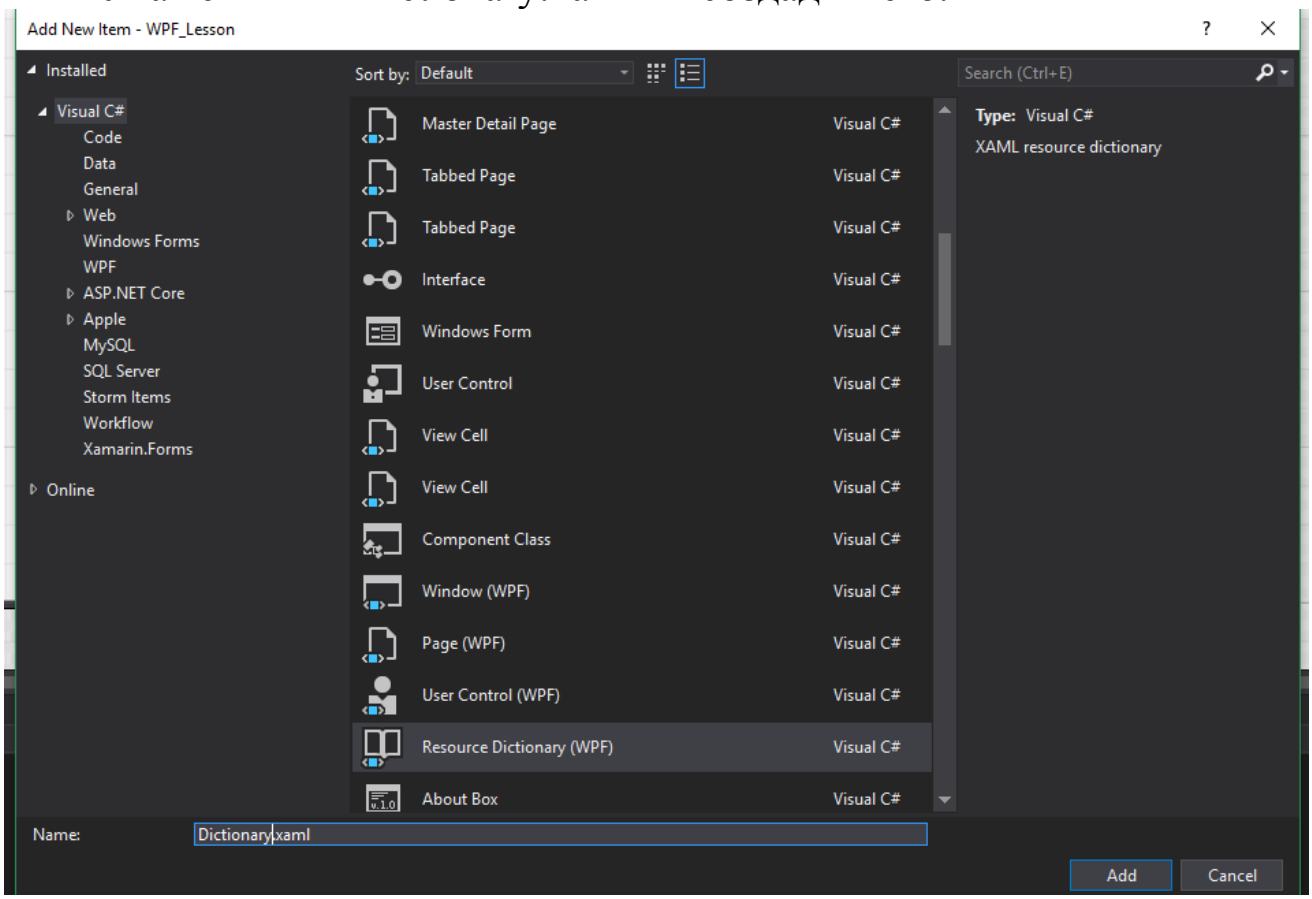
После нажатия кнопку локальный репозиторий будет сохранен на удаленном. И после успешного завершения будет выведено соответствующее сообщение.



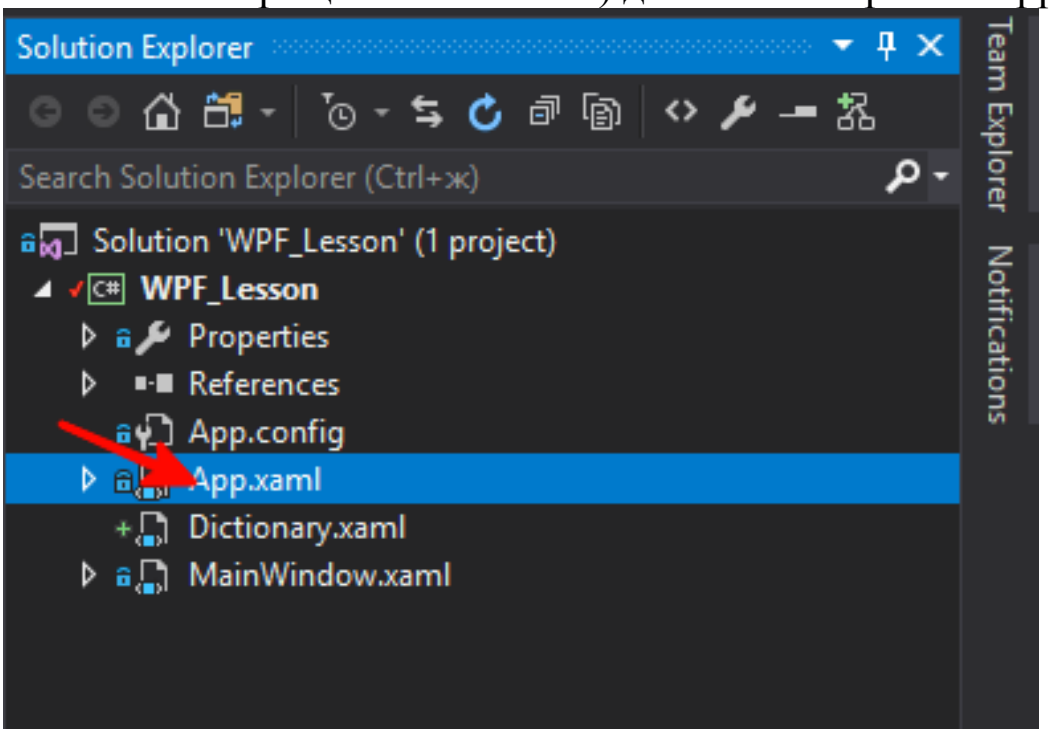
СОЗДАНИЕ СЛОВАРЯ СТИЛЕЙ



Укажем имя «Dictionary.xaml» и создадим его.



Теперь словарь необходимо подключить к проекту (чтобы компоненты могли обращаться к стилям) для этого откроем «App.xaml»



И впишем следующий код для подключения словаря.

```
<Application x:Class="WPF_Lesson.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:WPF_Lesson"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="Dictionary.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>
```

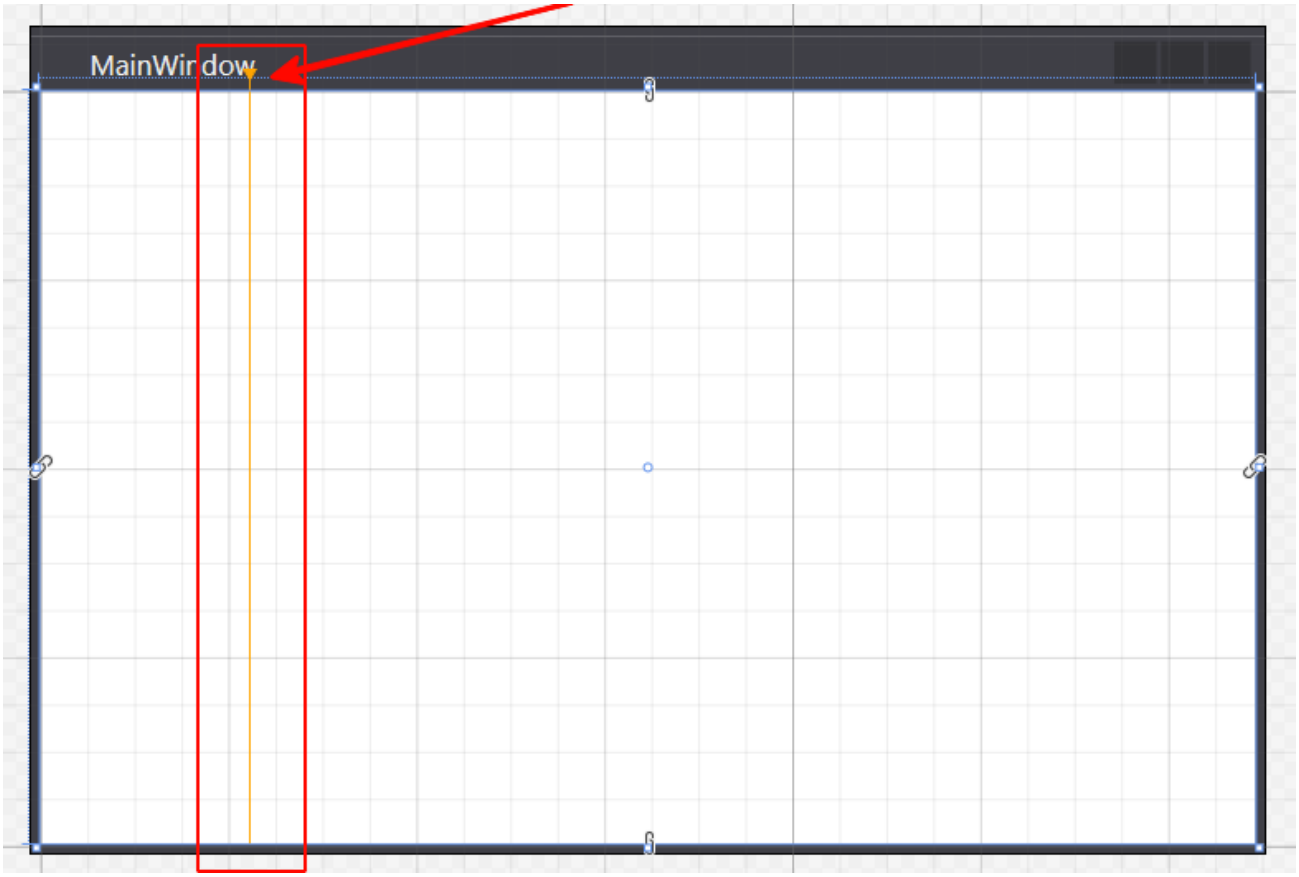
После данных операций можно начинать писать стили и константы. Напишем первый стиль, который будет менять задний фон и внутренний отступ, и он применяться ко всем компонентам типа «Label»

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:local="clr-namespace:WPF_Lesson">
    <Style TargetType="{x:Type Label}">
        <Setter Property="Background" Value="AntiqueWhite"/>
        <Setter Property="Padding" Value="0"/>
    </Style>
</ResourceDictionary>
```

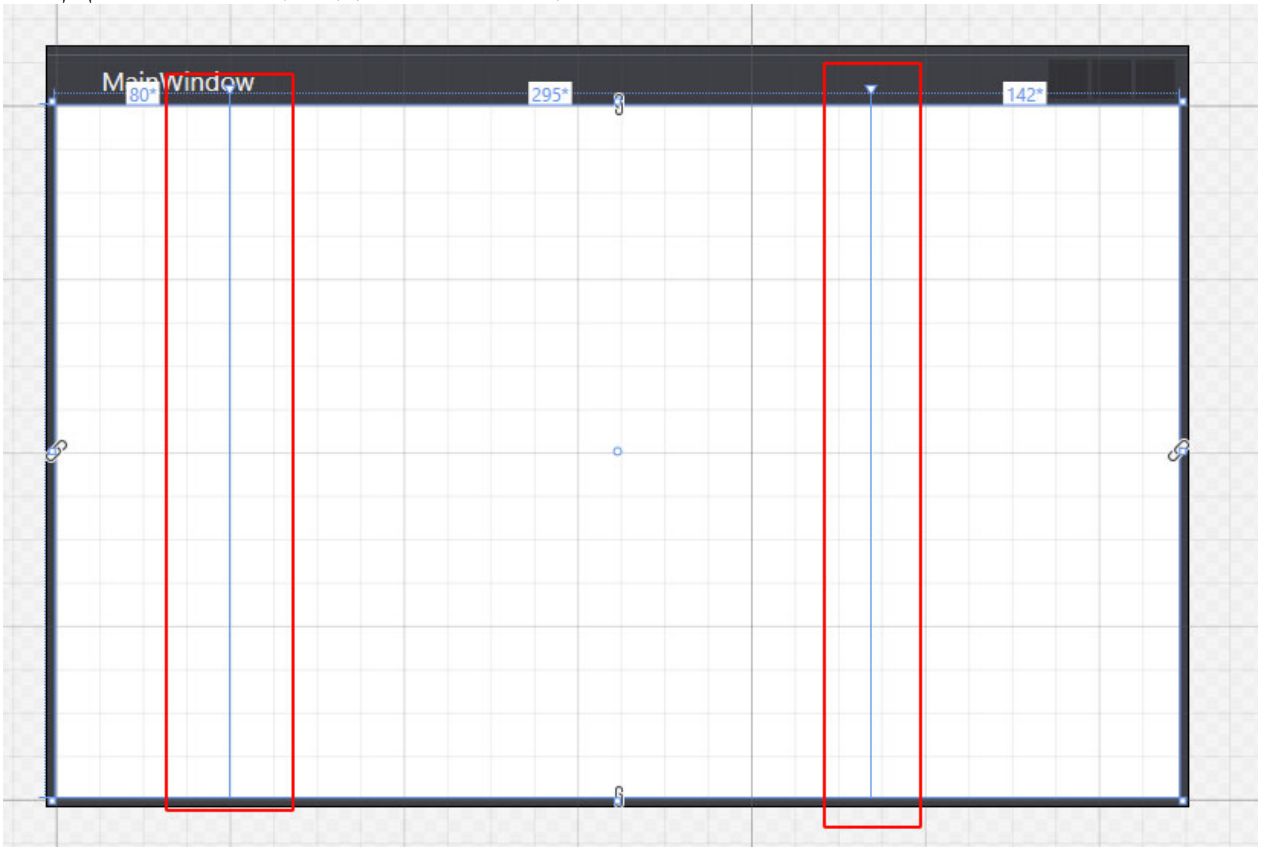
РАЗМЕЩЕНИЕ КОНТЕНТА ПО ЦЕНТРУ ФОРМЫ

Откройте форму или страницу. Выберите компонент Grid, после чего по краям (сверху и слева) появится возможность добавлять строки и столбцы.

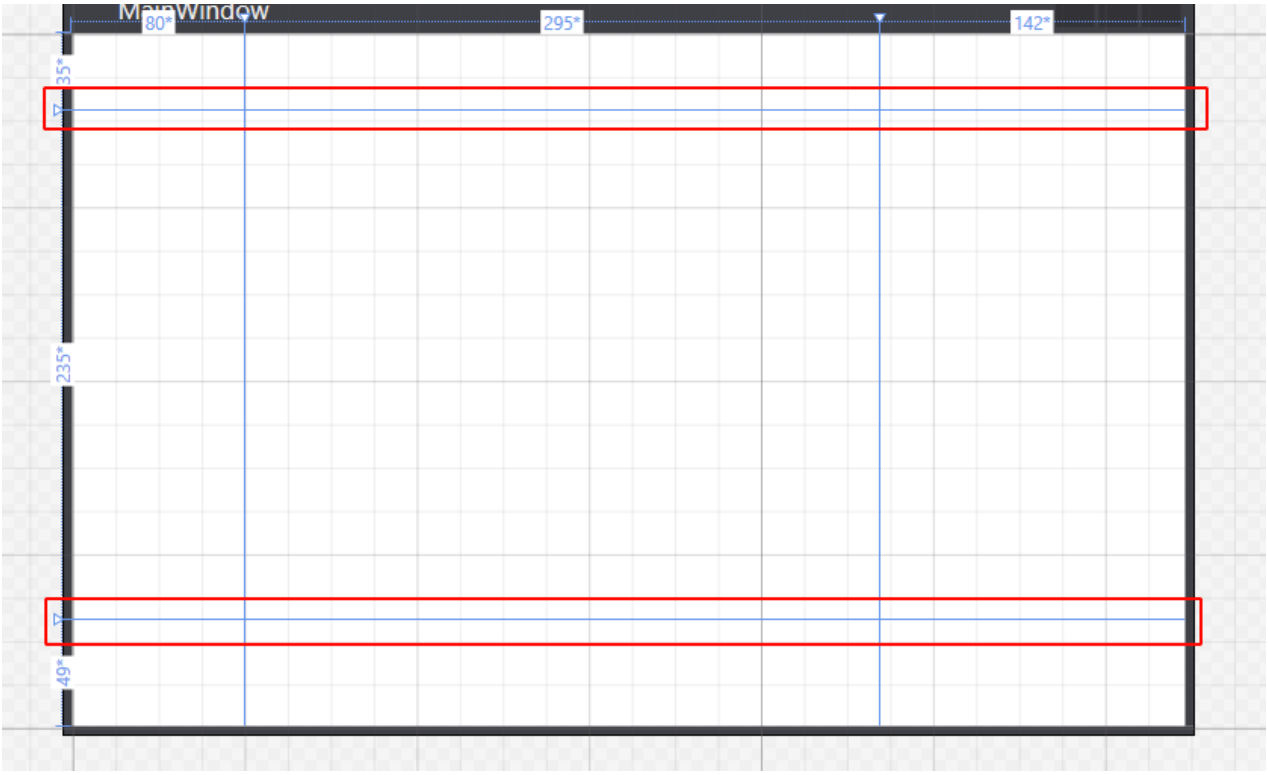
Добавим первый столбец.



Добавим еще один столбец.



Таким же образом добавим две строки.



После всех добавление мышкой, мы можем увидеть следующий код XAML.

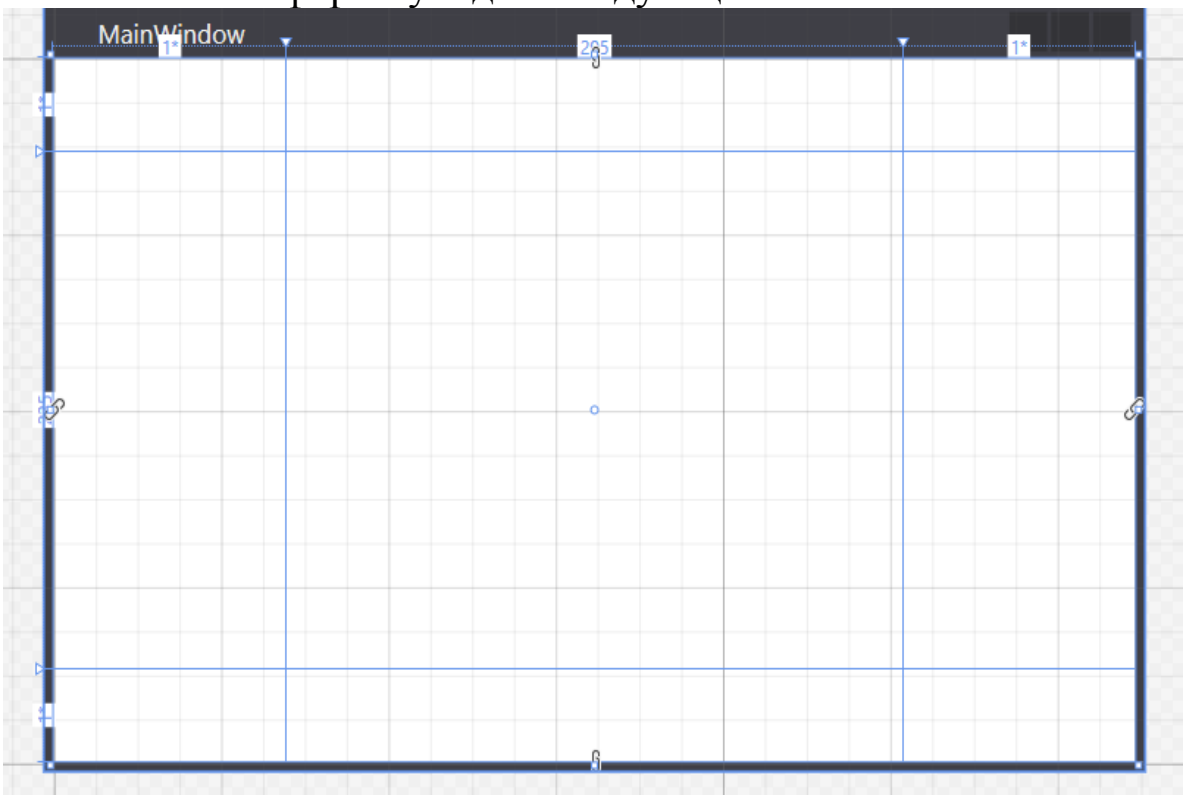
```
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="35*"/>
            <RowDefinition Height="235*"/>
            <RowDefinition Height="49*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="80*"/>
            <ColumnDefinition Width="295*"/>
            <ColumnDefinition Width="142*"/>
        </Grid.ColumnDefinitions>
    </Grid>
</Window>
```

Так же столбцы и строки можно добавлять, редактируя код XAML. Здесь же можно редактировать размер строк и столбцов. Чтобы разместить элементы по центру (которые будут во второй строки и во

втором столбце) необходимо задать размеры строкам и столбцам. Зададим столбцам и строкам следующие размеры.

```
<Grid.RowDefinitions>
    <RowDefinition Height="*"/>
    <RowDefinition Height="235"/>
    <RowDefinition Height="*"/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
    <ColumnDefinition Width="*"/>
    <ColumnDefinition Width="295"/>
    <ColumnDefinition Width="*"/>
</Grid.ColumnDefinitions>
```

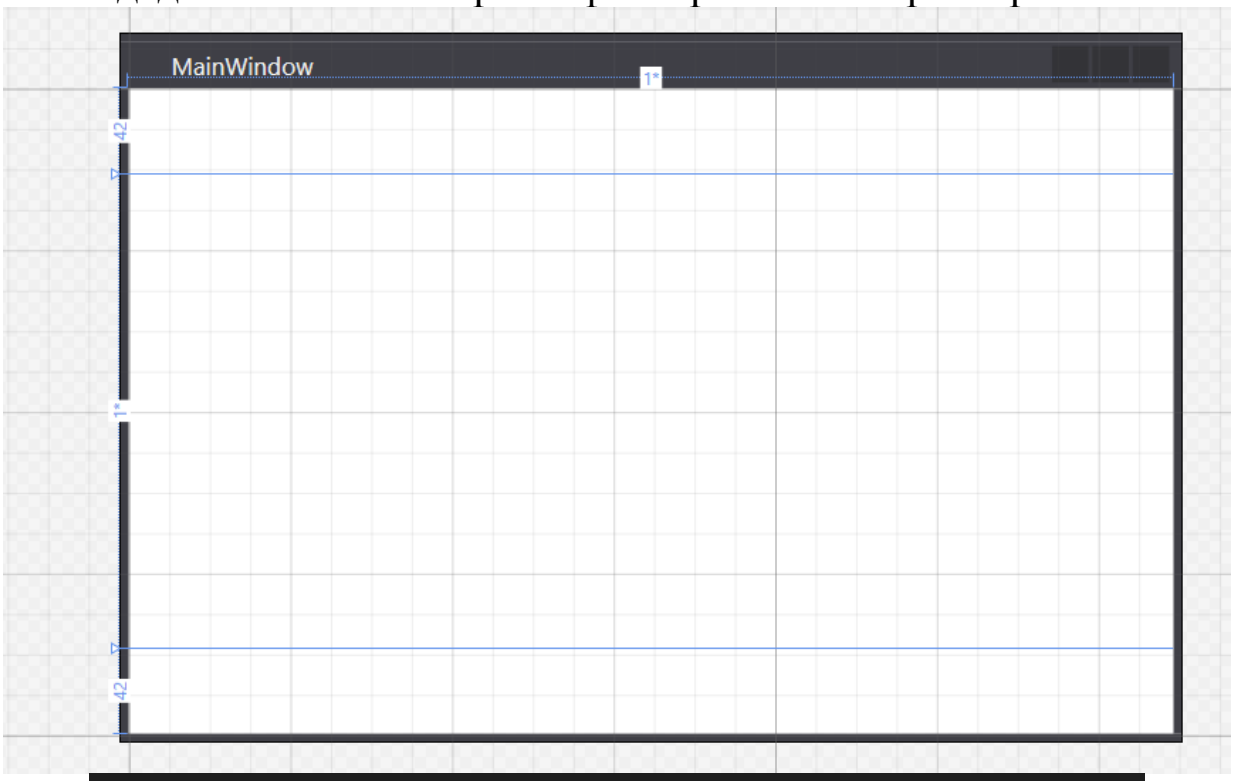
После чего на форме увидим следующее.



В результате средний столбец и средняя строка имеют фиксированный размер, в то время как последние и первые имеют динамический размер (меняется в зависимости от размера контейнера, окна).

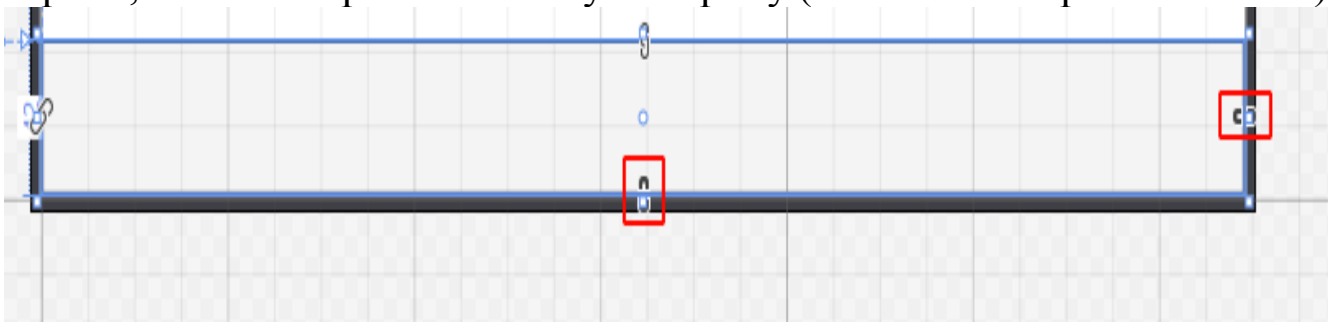
СОЗДАНИЕ БАЗОВОЙ ФОРМЫ

Создадим несколько строк с фиксированными размерами.




```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="42"/>
        <RowDefinition/>
        <RowDefinition Height="42"/>
    </Grid.RowDefinitions>
</Grid>
```

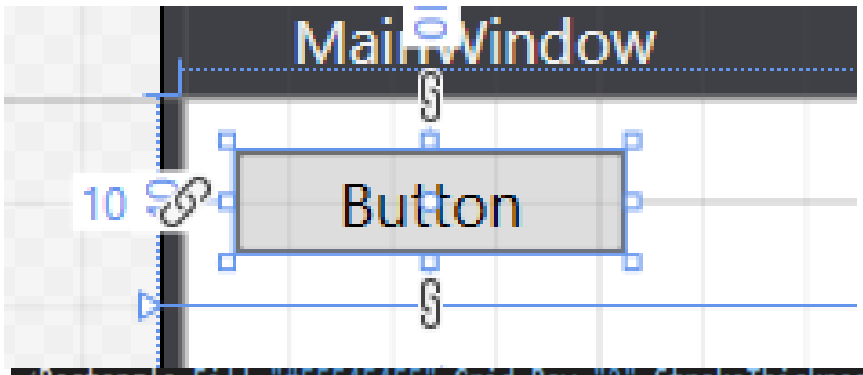
Добавим компонент «Rectangle» на форму и разместим его в 3 строке. Компонент необходимо растянуть на всю ширину и высоту строки, а затем закрепить высоту и ширину (чтобы он мог растягиваться)



Укажем у компонента «Rectangle» свойство «StrokeThickness» равное 0, это задаст размер границы прямоугольника.

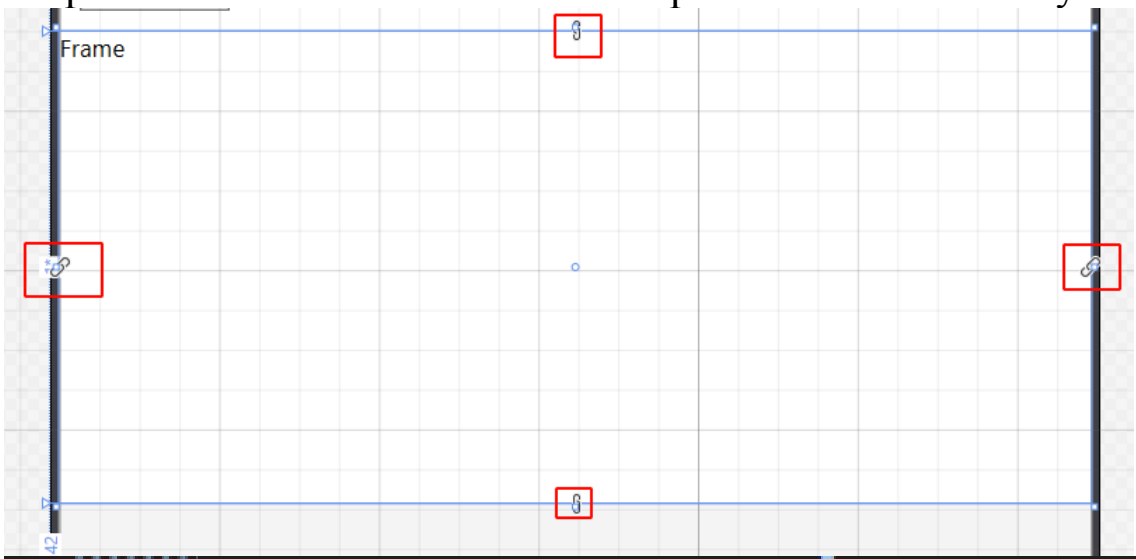
```
</Grid.RowDefinitions>
<Rectangle Fill="#FFF4F4F5" Grid.Row="2" StrokeThickness="0"/>
id>
```

Добавим на форму кнопку и разместим в левом углу первой строки. И укажем имя кнопки «Назад», которая в последствии будет выполнять соответствующею функцию.



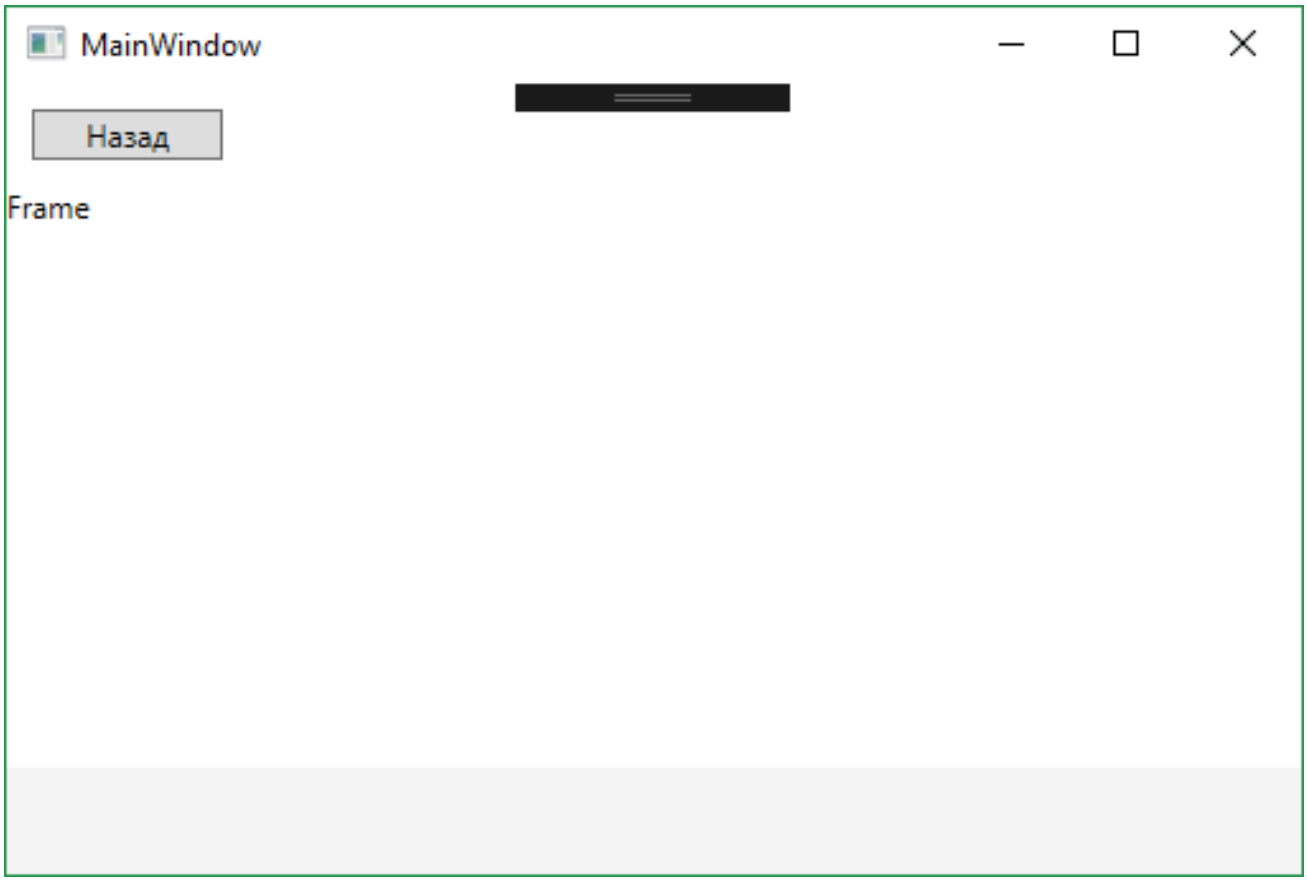

```
<Rectangle Fill="#FFF4F4F5" Grid.Row="2" StrokeThickness="0"/>
<Button Content="Назад" HorizontalAlignment="Left" Margin="10,10,0,10" Width="75"/>
rid>
```

Добавим на форму компонент «Frame» во вторую строку, в него будут загружать страницы (Авторизация, Регистрация и другие). Закрепим компонент по высоте и ширине. Укажем имя ему «MainFrame»




```
<Frame x:Name="MainFrame" Content="Frame" Grid.Row="1"/>
Grid>
```

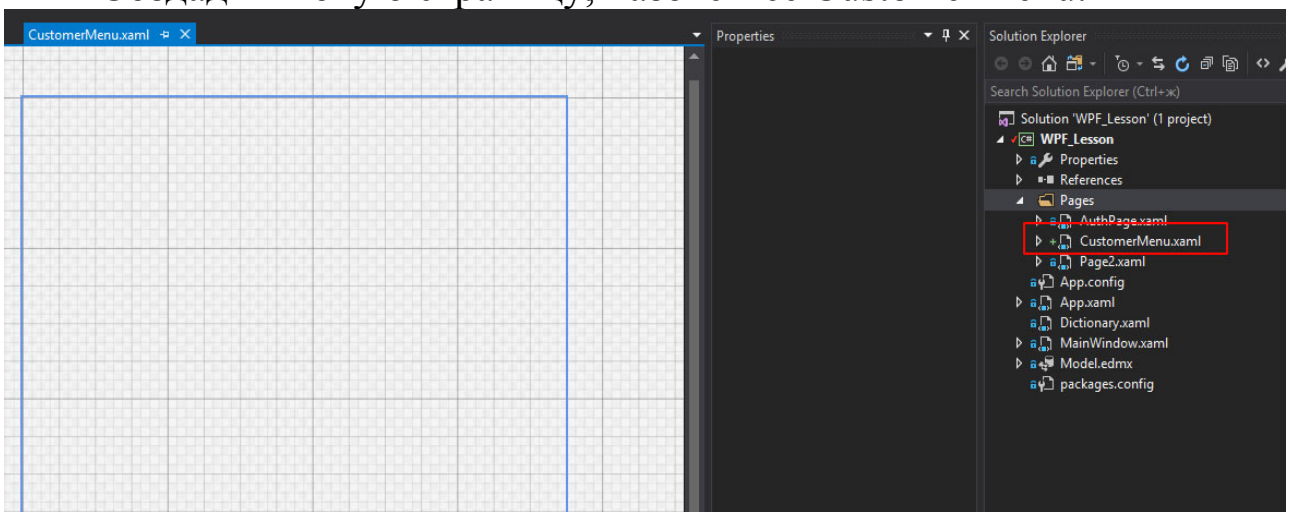
Запустим проект и увидим примерно следующее.



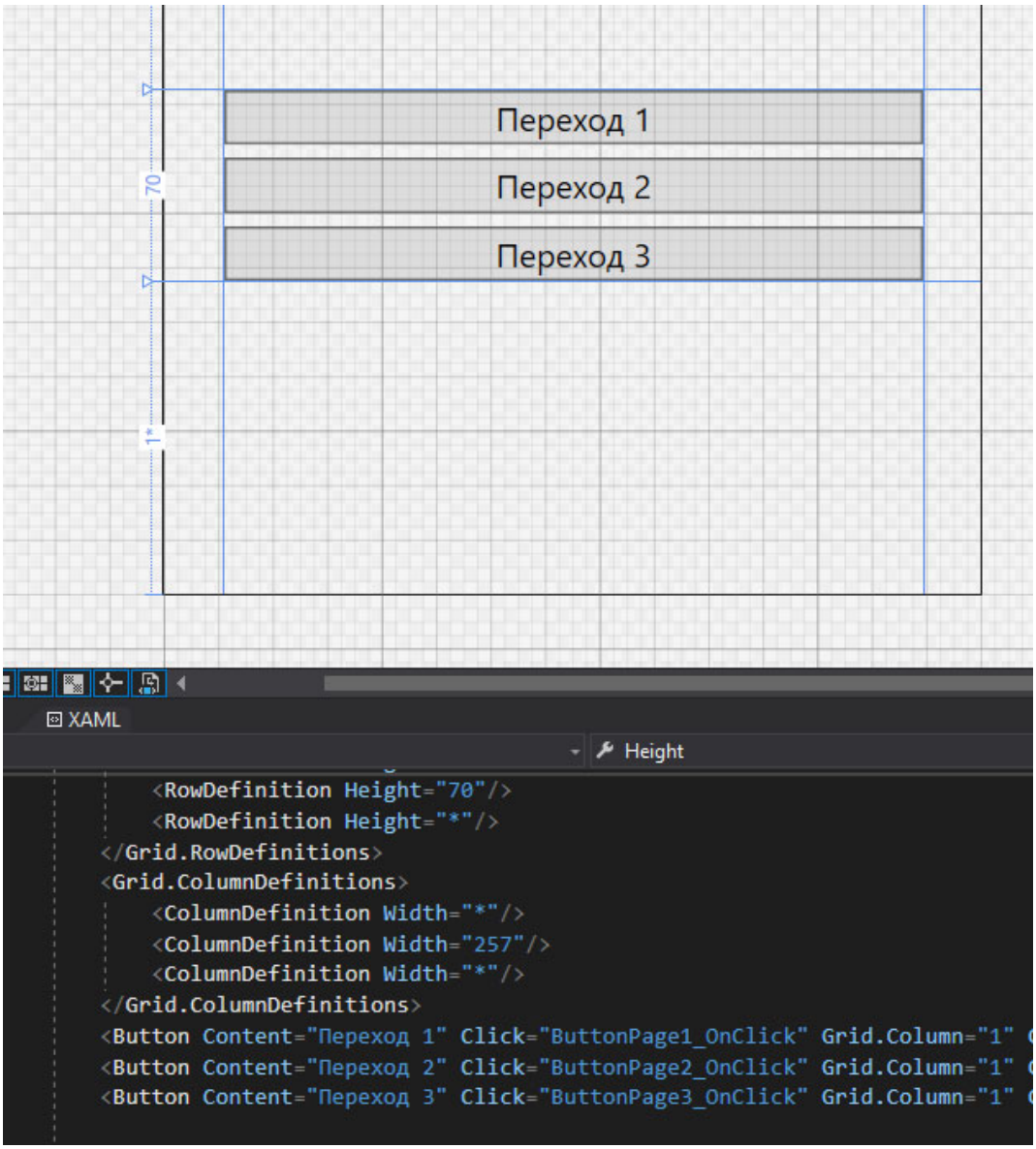
Для выполнения практического задания можно ознакомиться с обучающими видео

СОЗДАНИЕ МЕНЮ ПОЛЬЗОВАТЕЛЯ

Создадим новую страницу, назовем ее CustomerMenu.



Добавим несколько кнопок, разместим их по центру и добавим им обработчик события клик.



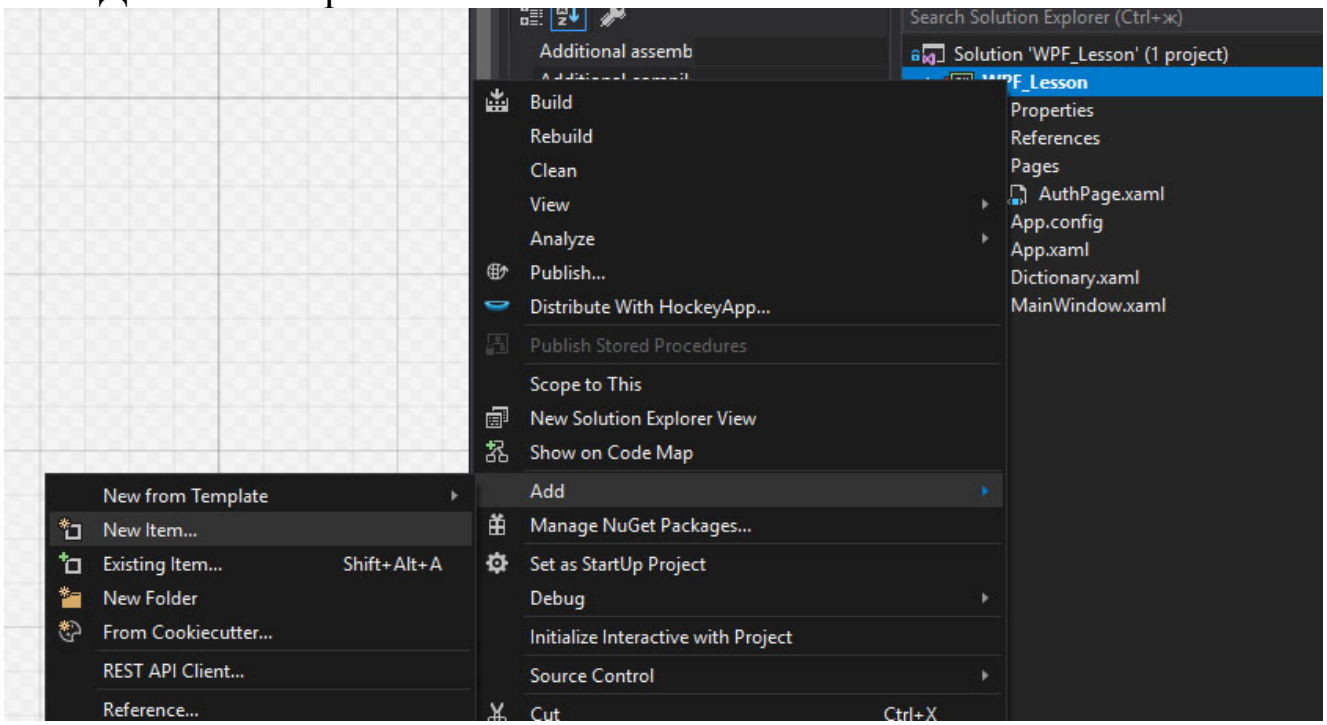
И в коде каждому обработчику укажем свой переход

```
– references | 0 changes | 0 authors, 0 changes
private void ButtonPage1_OnClick(object sender, RoutedEventArgs e)
{
    NavigationService?.Navigate(new Page2());
}

– references | 0 changes | 0 authors, 0 changes
private void ButtonPage2_OnClick(object sender, RoutedEventArgs e)
{
    NavigationService?.Navigate(new AuthPage());
}

– references | 0 changes | 0 authors, 0 changes
private void ButtonPage3_OnClick(object sender, RoutedEventArgs e)
{
    NavigationService?.Navigate(new Menu());
}
```

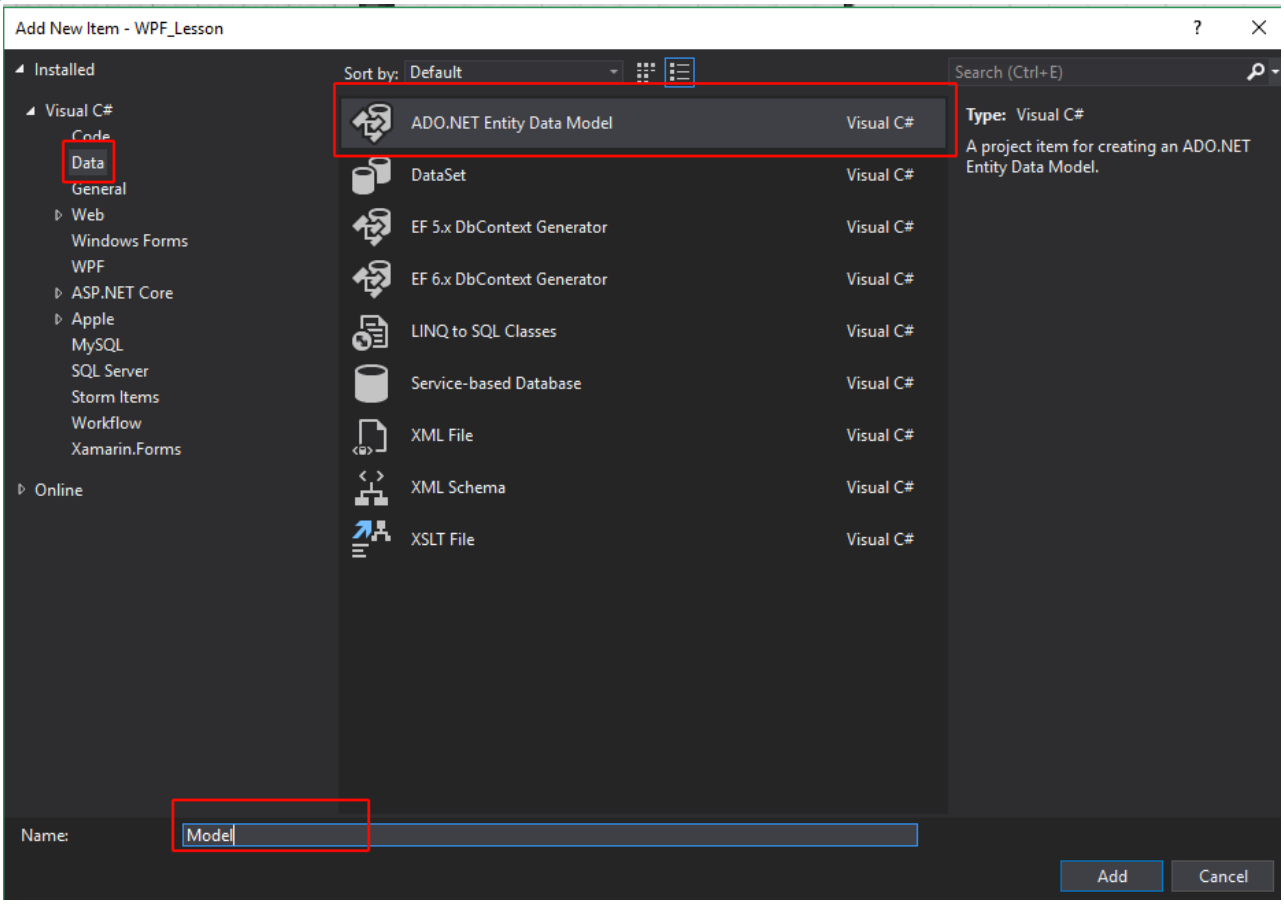
Для выполнения практического задания можно ознакомиться с обучающими видео

СОЗДАНИЕ ПОДКЛЮЧЕНИЯ К БАЗЕ ДАННЫХ

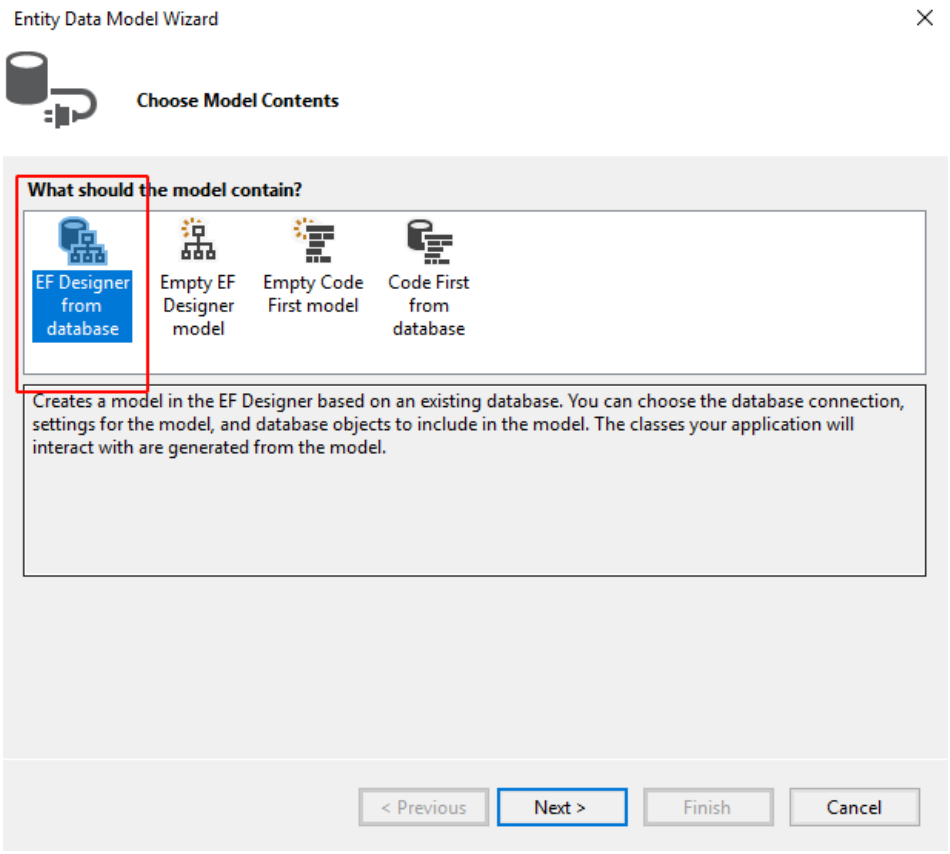
Добавим в проект новый объект



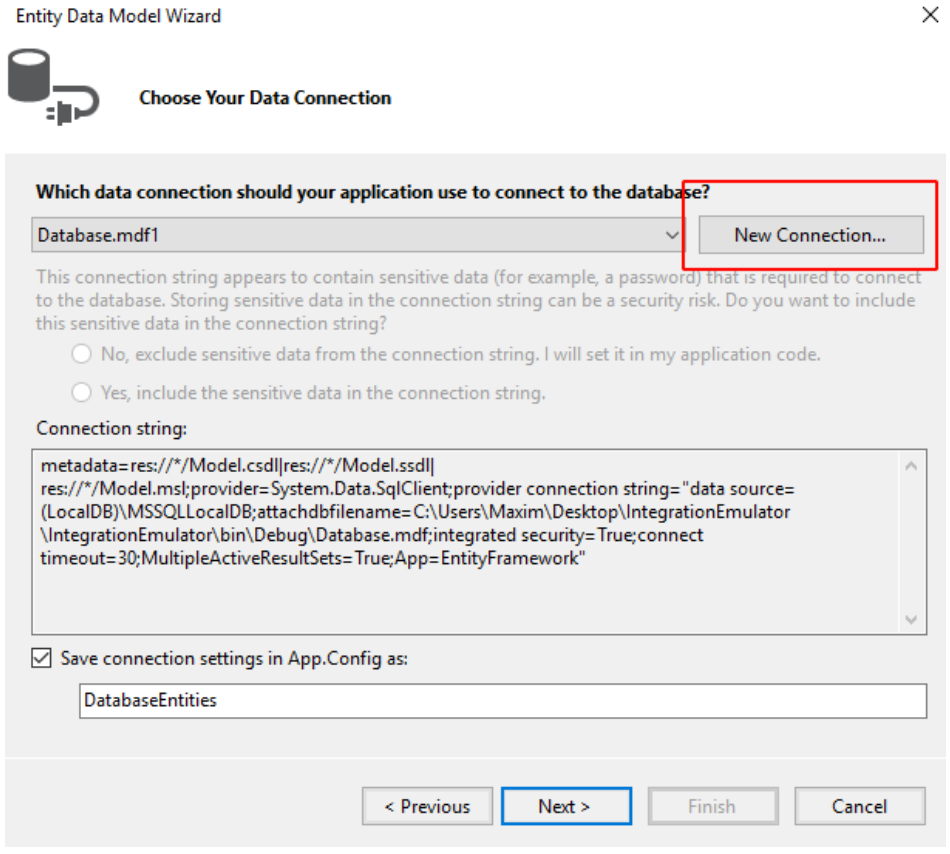
Выберем Data – ADO.NET Entity Data Model, назовем Model и добавим.



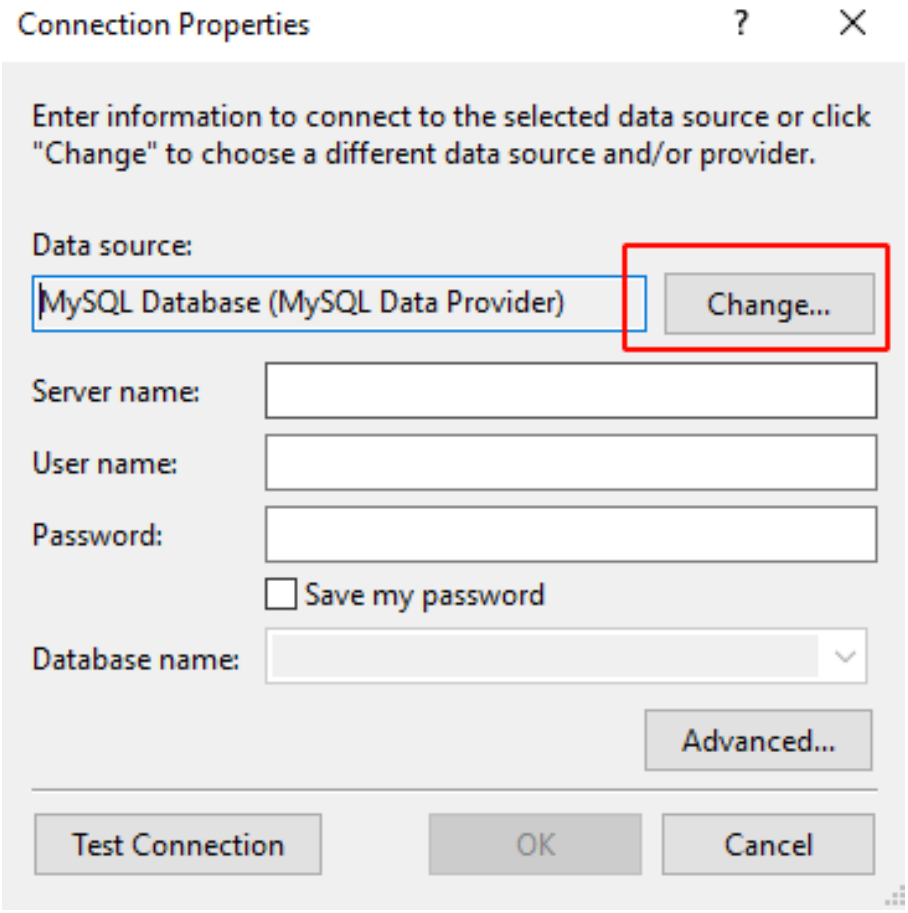
Появится диалоговое окно и выберем пункт «EF Designer from database»



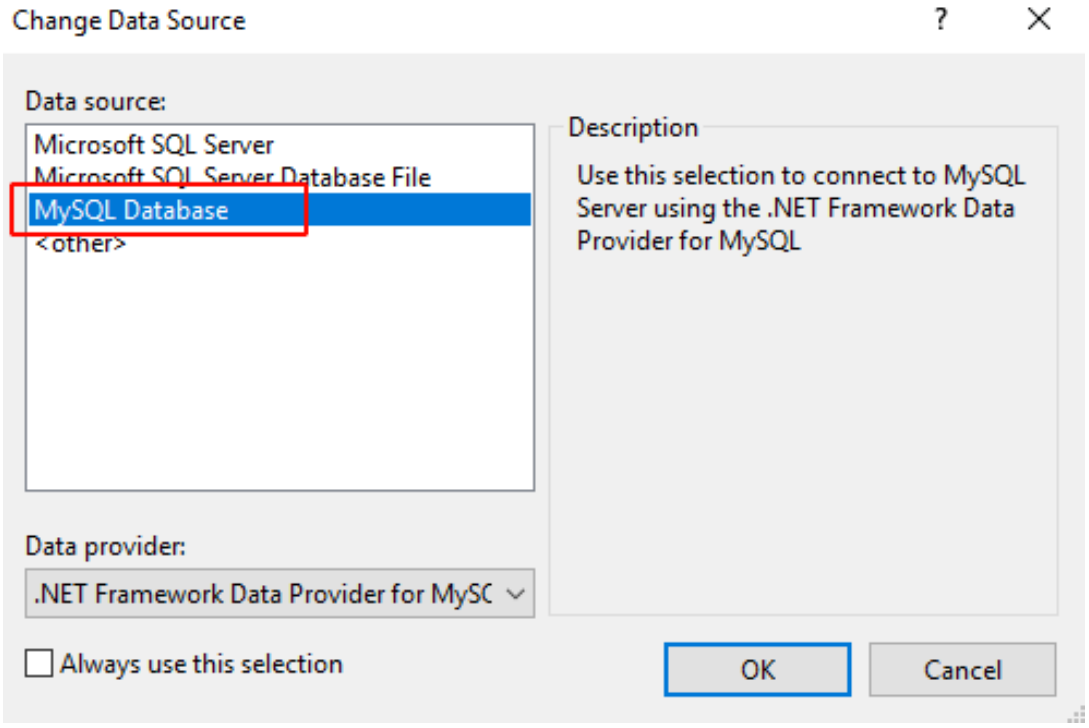
После появится следующее окно. Нажмем «New Connection»



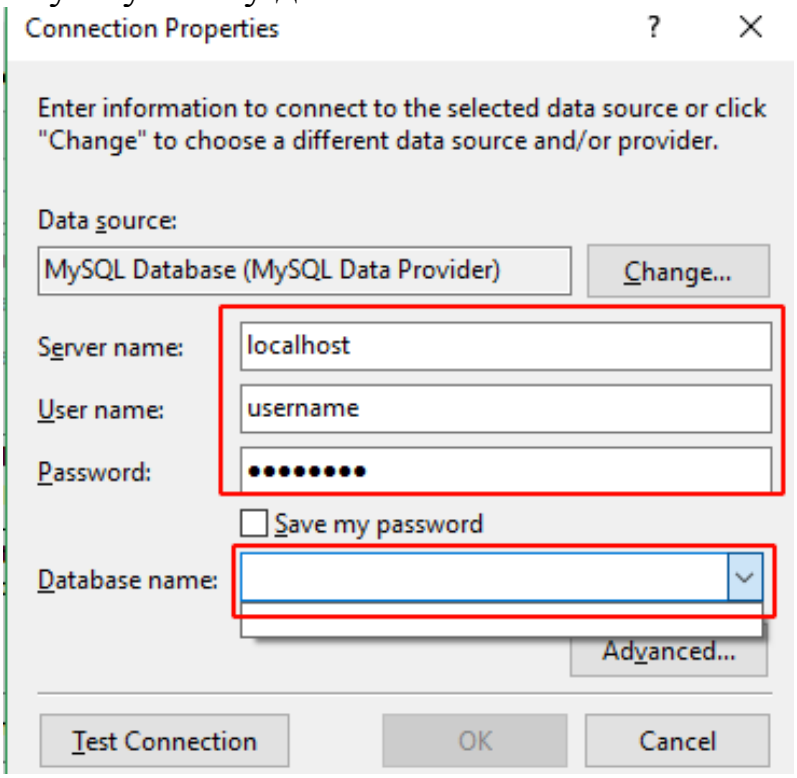
В данном окне выберем «Change».



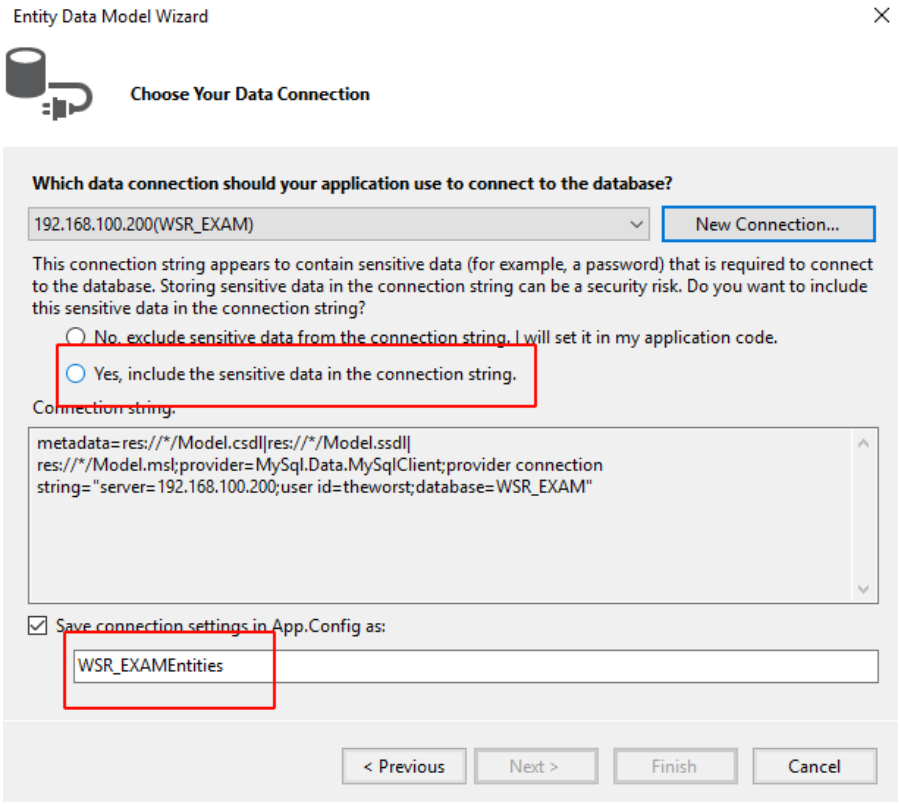
В следующем окне выберем «MySQL Database» и нажмем «ОК».



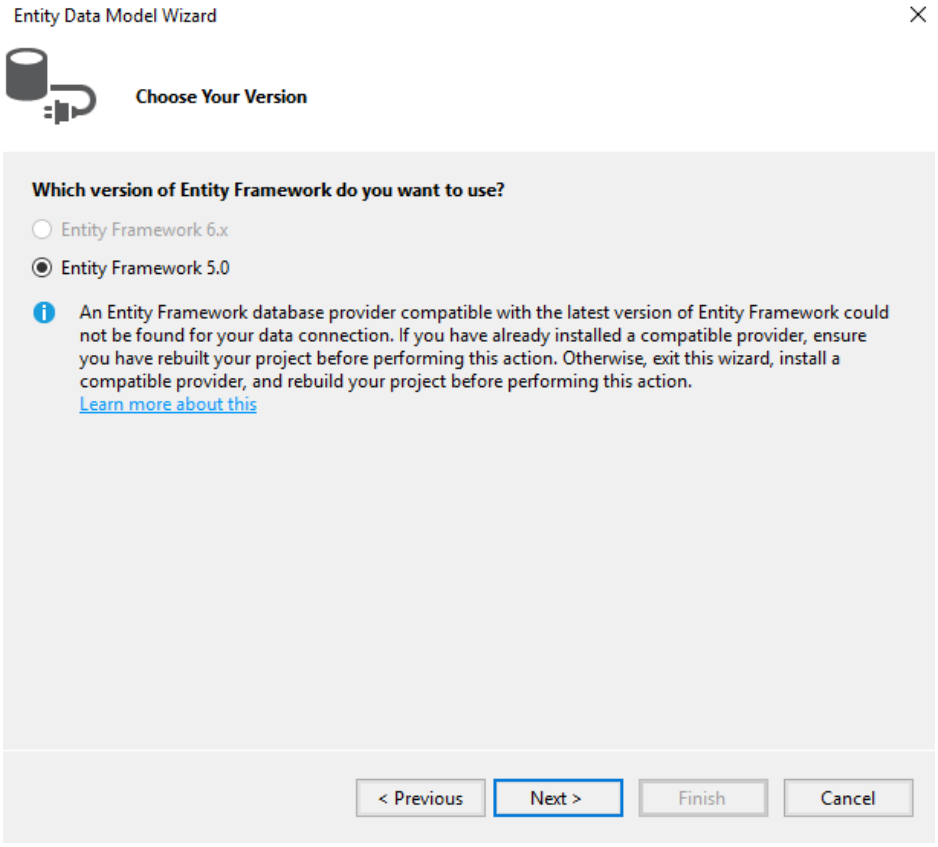
После выбора введем данные сервера: IP адрес, логин и пароль. После выберем нужную базу данных.



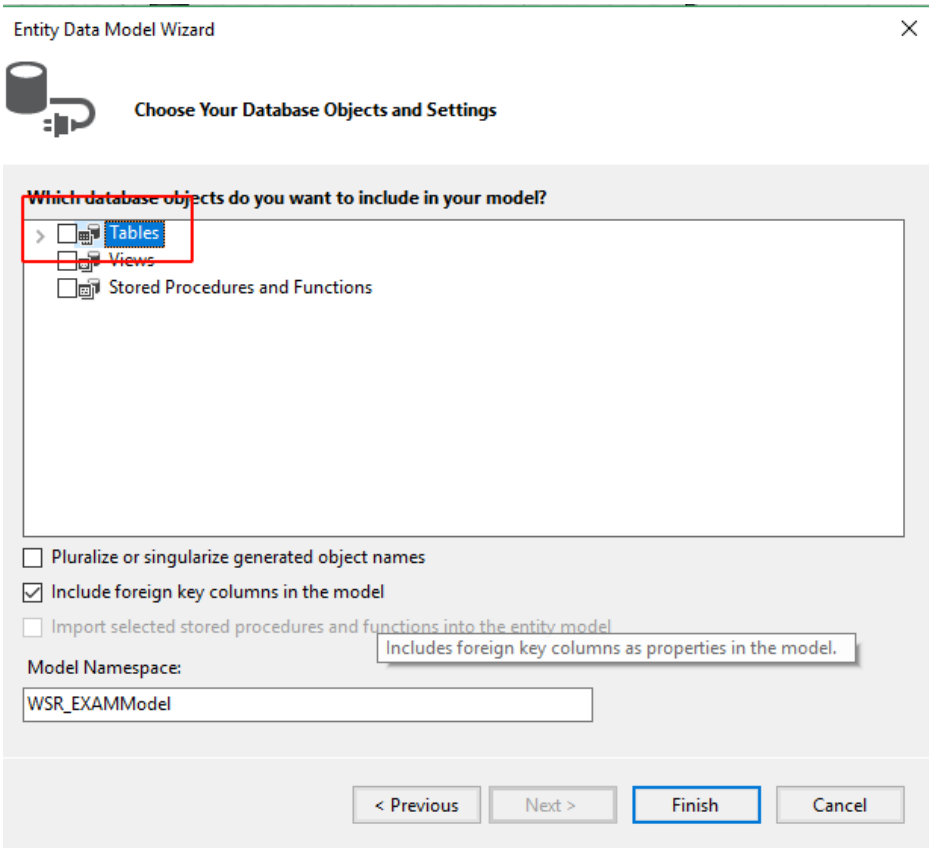
На следующем окне, выберем созданное подключение и поставим «Yes, include the sensitive data in the connection string», а также укажем имя нашего объекта базы данных «Entitites».



В следующем окне оставим по умолчанию:



В этом окне выберем таблицы, которые хотим использовать в проекте:



ПОЛУЧЕНИЕ ДАННЫХ ИЗ БАЗЫ ДАННЫХ

Для подключения к базе данных необходимо создать контекст.

```
using (var db = new Entities())
{

}
```

Теперь загрузим всю таблицу пользователей.

```
using (var db = new Entities())
{
    var users = db.User.AsNoTracking().ToList();
}
```

Получим пользователей по определенному критерию

```
using (var db = new Entities())
{
    var users = db.User.AsNoTracking().Where(u => u.Login.StartsWith("max")).ToList();
}
```

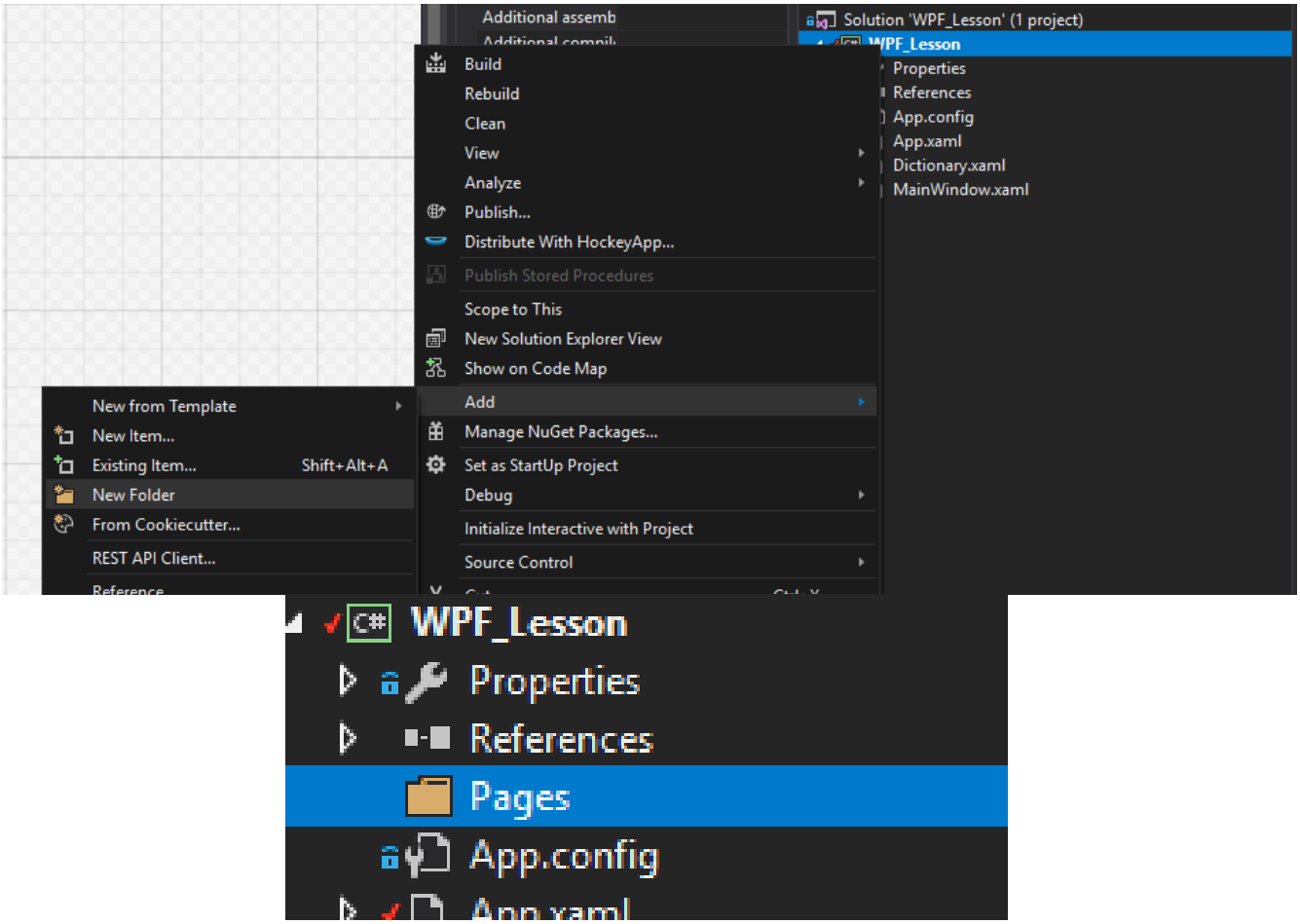
Получим пользователя по определенным критериям

```
using (var db = new Entities())
{
    var user = db.User.AsNoTracking().FirstOrDefault(u => u.Login == "max" && u.Password == "test");
}
```

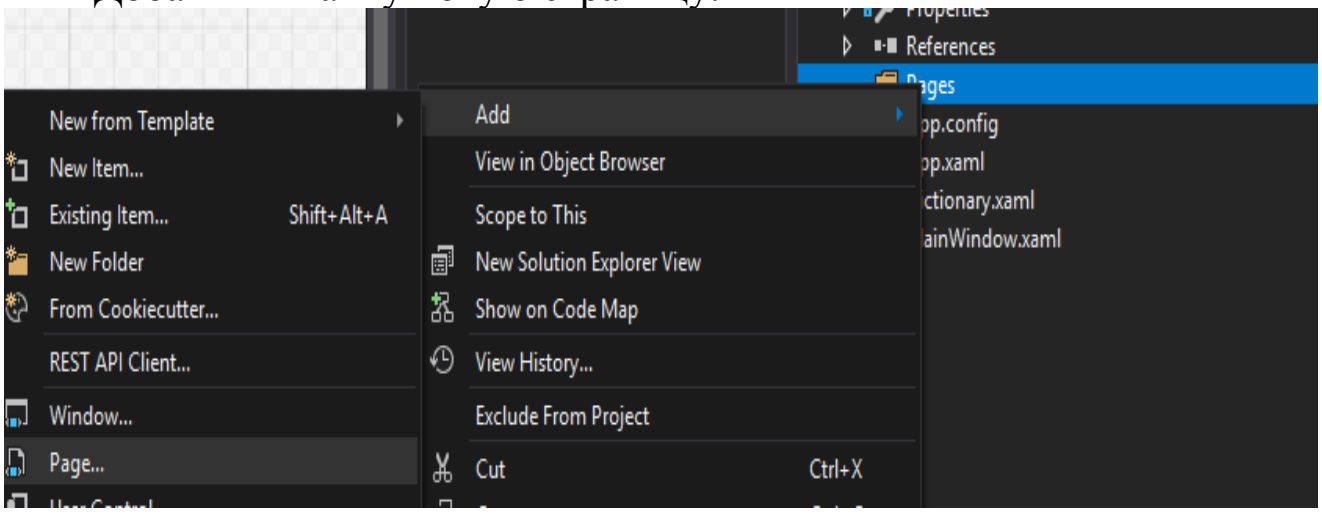
Для выполнения практического задания можно ознакомиться с обучающими видео

Создание формы авторизации

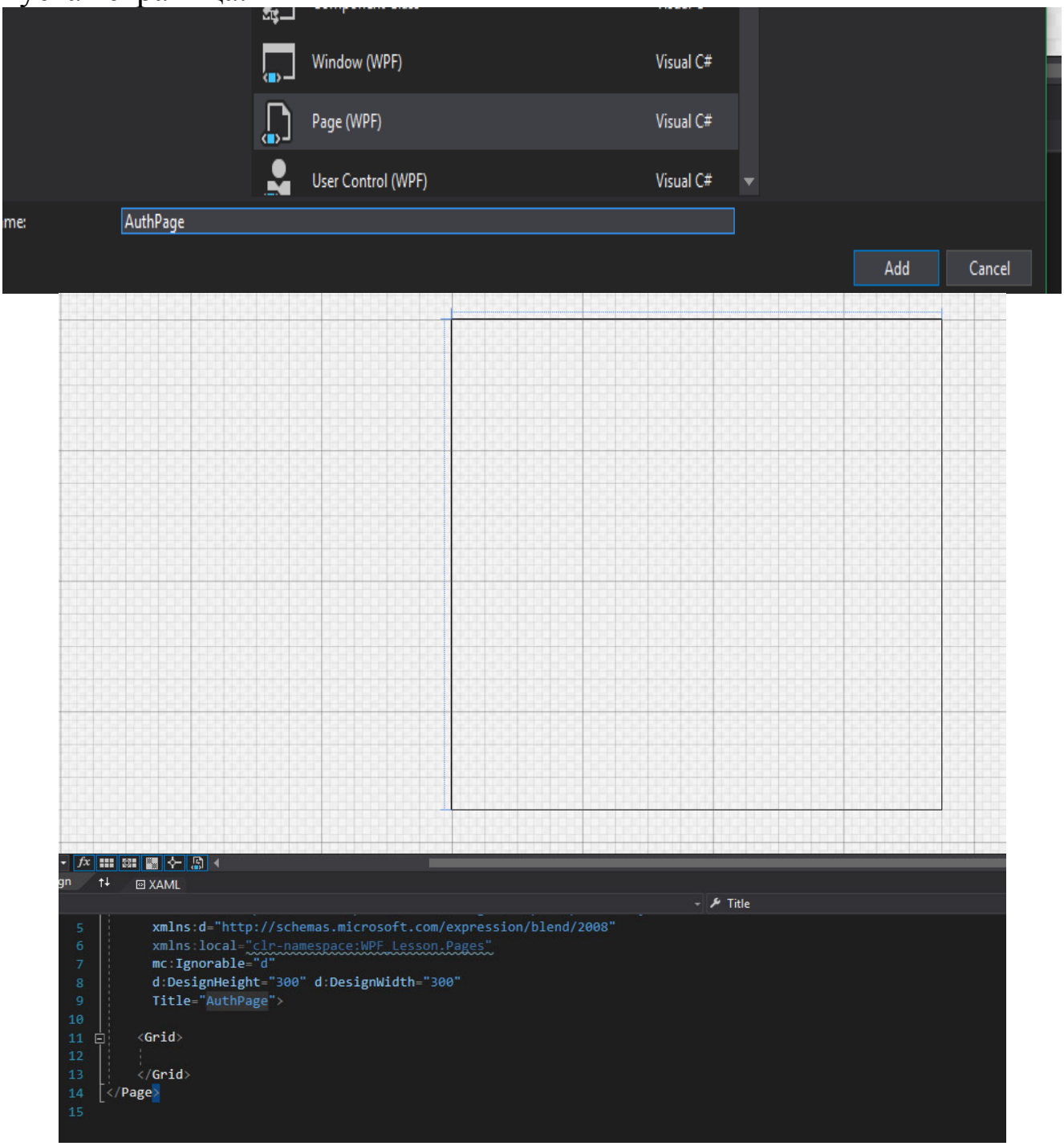
Добавим в проект новую папку и назовем ее «Pages», в этой папке будут находится страницы (авторизация, регистрация и другие)



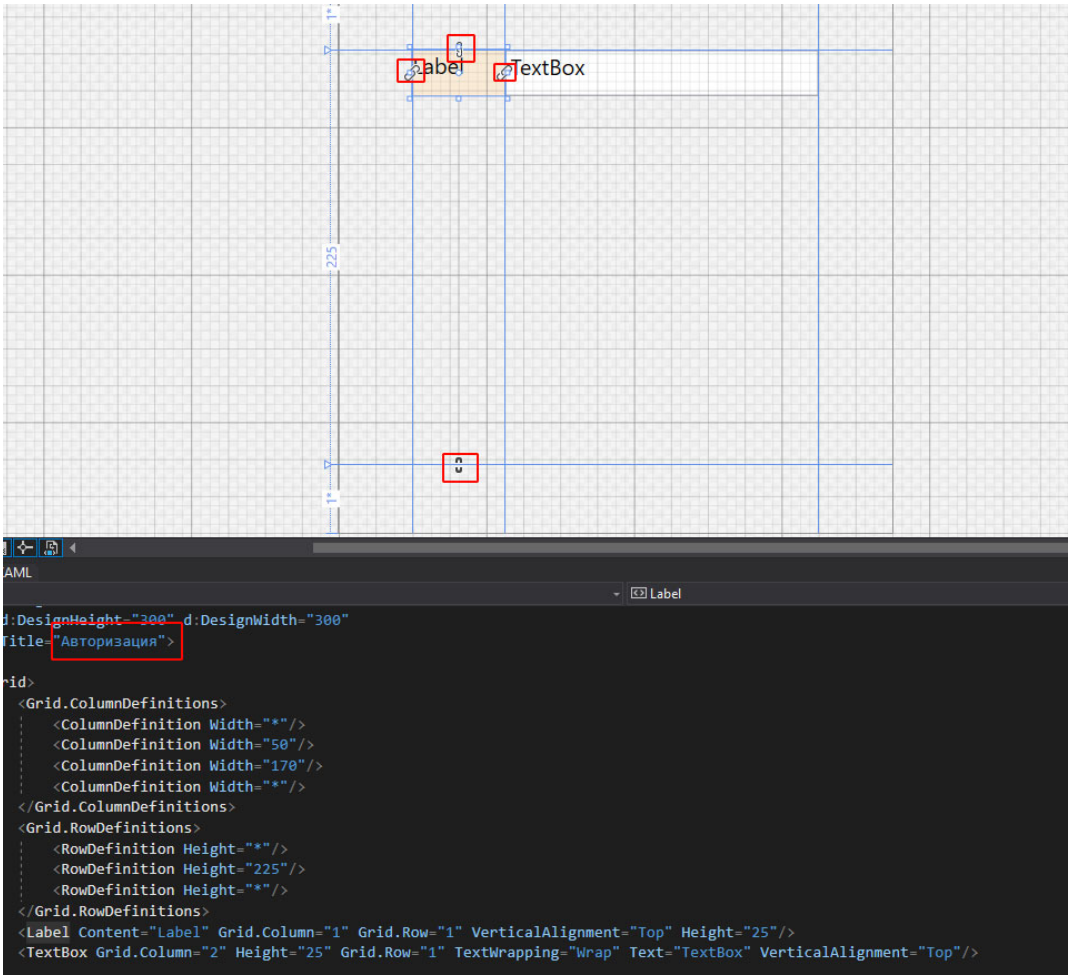
Добавим в папку новую страницу.



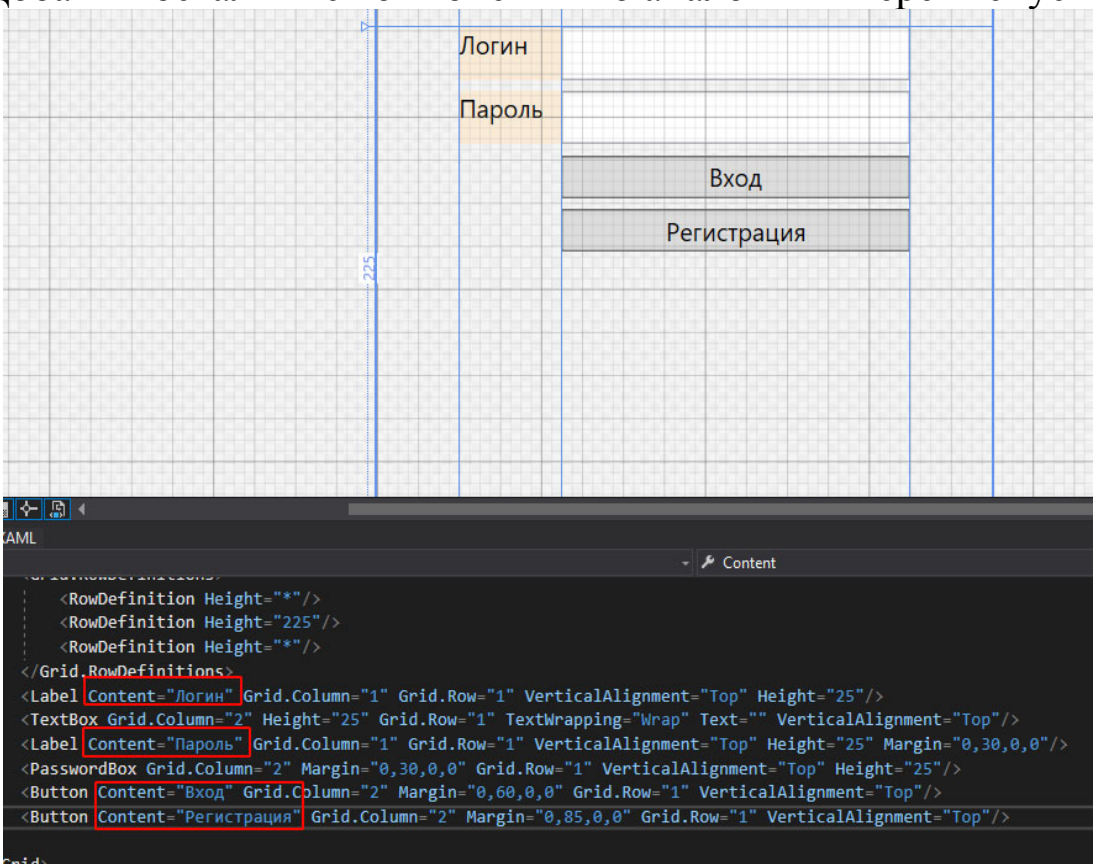
Назовем страницу AuthPage и создадим ее. После чего появится пустая страница.



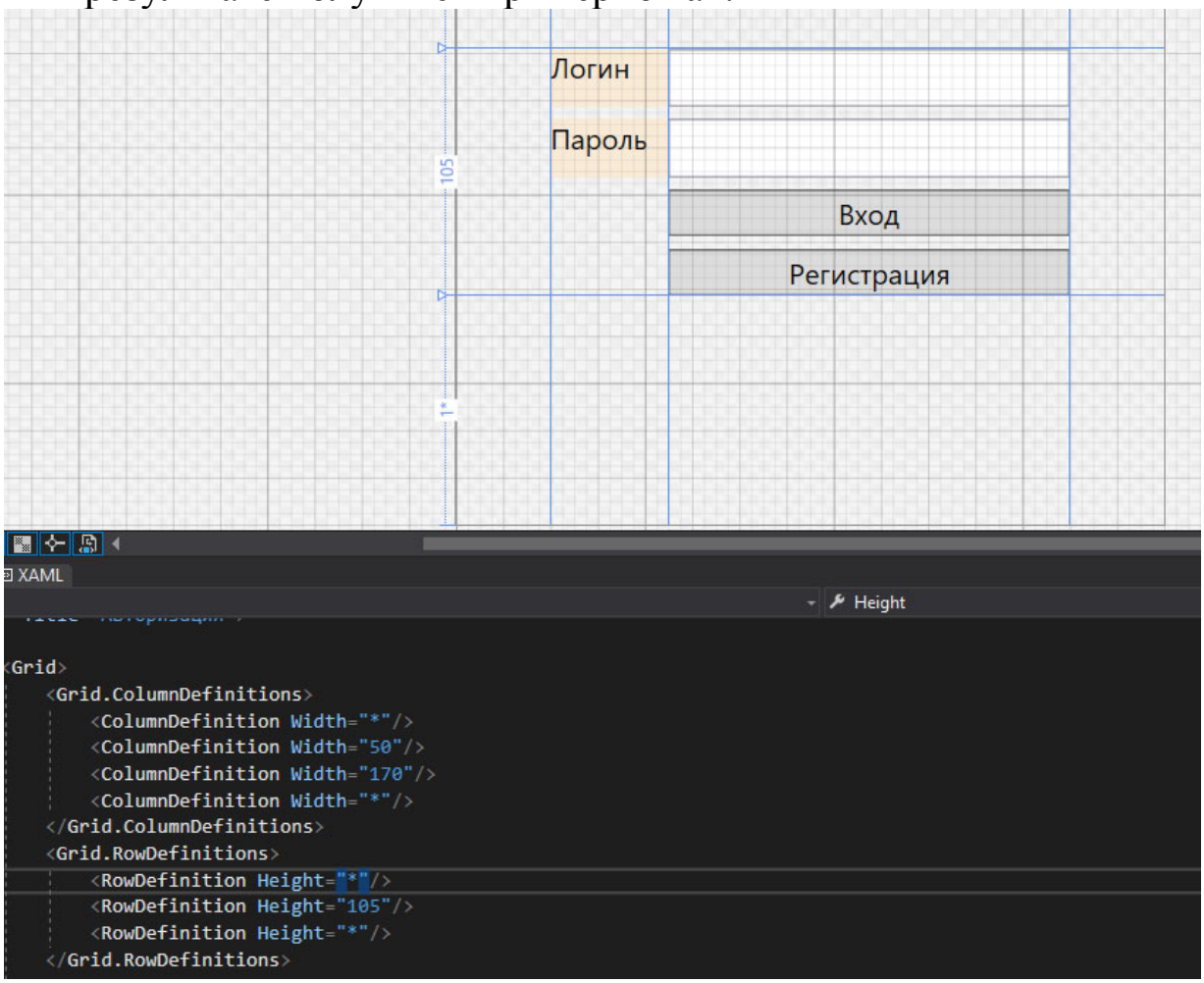
Добавим на форму компоненты «Label» и «TextBox», а затем от центруем их и добавим еще один столбец (для размещения лейблов). Также не забудем переименовать страницу, а затем закрепить лейбл и текстбокс, чтобы они растягивались по ширине.



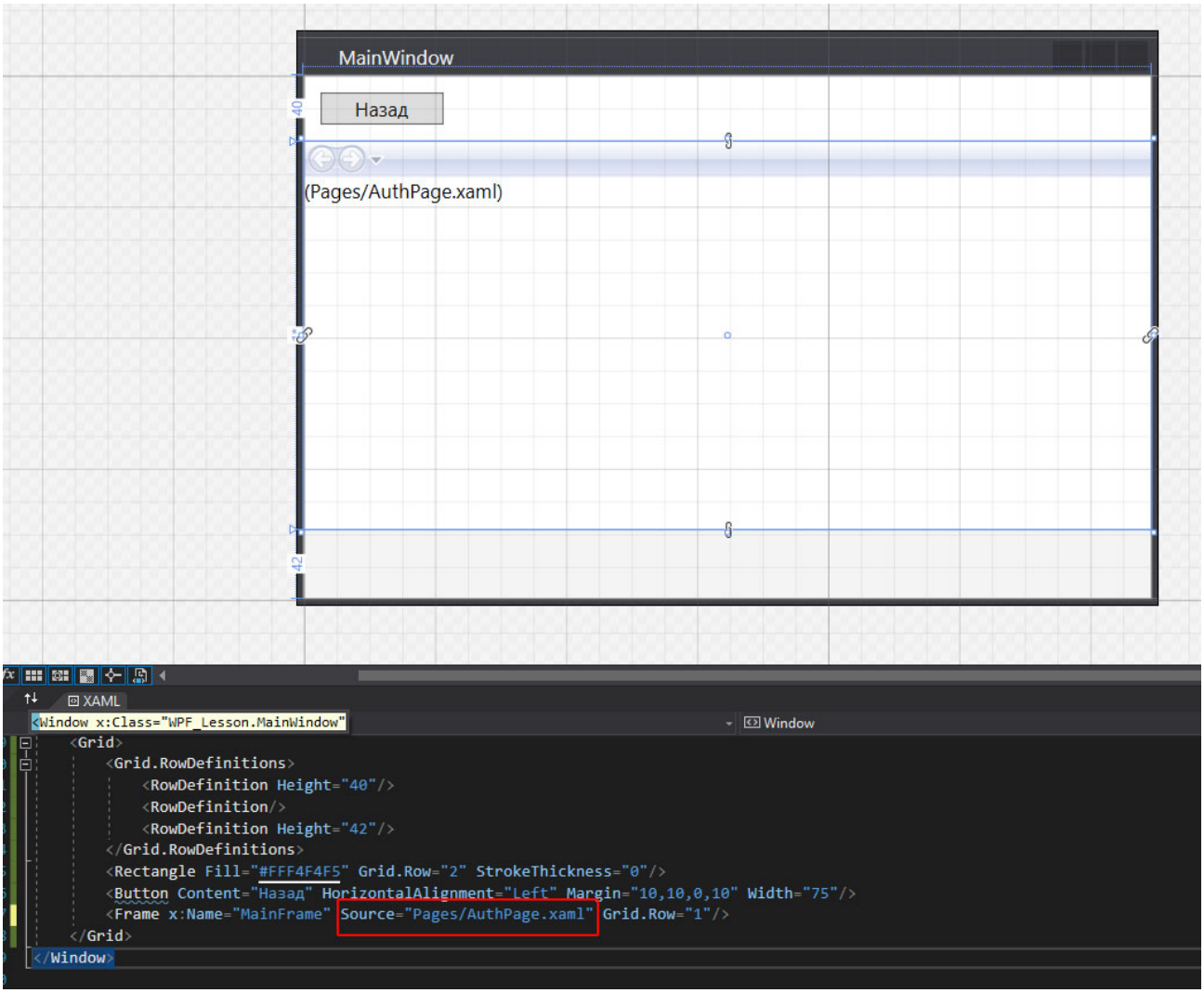
Добавим остальные компоненты по аналогии и переименуем их.



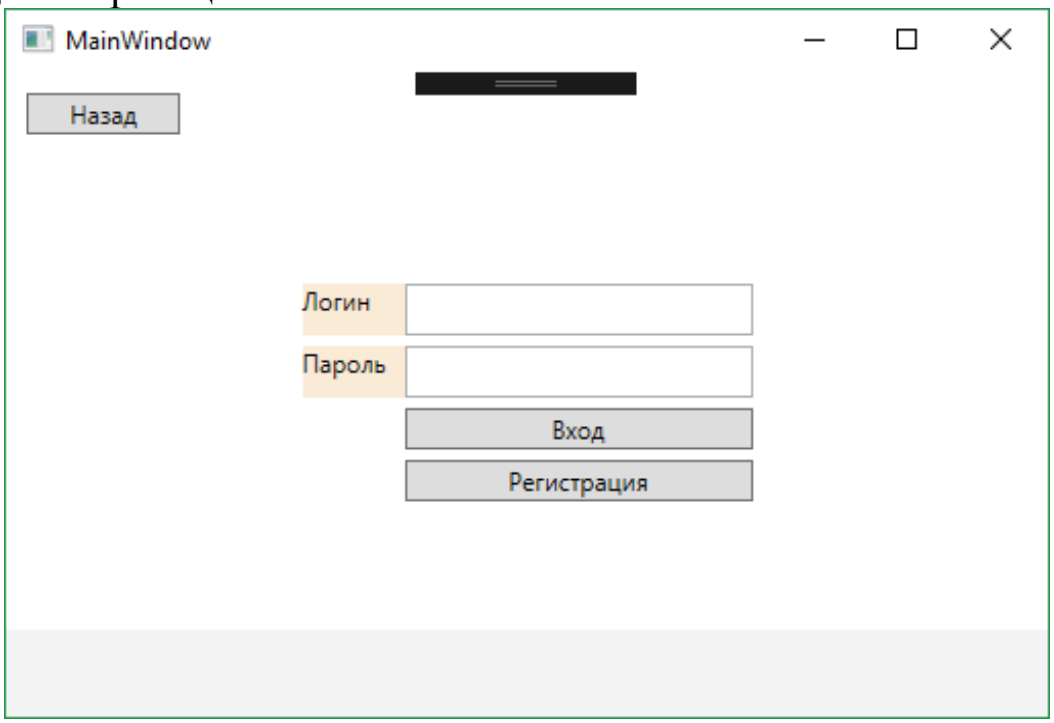
В результате получится примерно так.



Перейдем на главную форму и в компоненте «Frame» укажем в свойстве «Source» нашу страницу «AuthPage.xaml»



Запустим проект и увидим, что при запуске теперь отображается страница авторизации.



Теперь добавим функционал. Добавим обработчик события на кнопку вход.

```
<Button Click="ButtonEnter_OnClick" Content="Вход" Grid.Column="2" Margin="0,60,0,0" Grid.Row="1" VerticalAlignment="Top" Height="20"/>
<Button Content="Регистрация" Grid.Column="2" Margin="0,85,0,0" Grid.Row="1" VerticalAlignment="Top" Height="20"/>

0 references | 0 changes | 0 authors, 0 changes
private void ButtonEnter_OnClick(object sender, RoutedEventArgs e)
{

}
```

Добавим полям имена:

```
    </Grid.RowDefinitions>
    <Label Content="Логин" Grid.Column="1" Grid.Row="1" Ver
    <TextBox x:Name="TextBoxLogin" Grid.Column="2" Height="
    <Label Content="Пароль" Grid.Column="1" Grid.Row="1" Ve
    <PasswordBox x:Name="PasswordBox" Grid.Column="2" Margi
    <Button Click="ButtonEnter_OnClick" Content="Вход" Gric
    <Button Content="Регистрация" Grid.Column="2" Margin="0

</Grid>
ge>
```

Добавим в код базовую проверку

```
private void ButtonEnter_OnClick(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(TextBoxLogin.Text) || string.IsNullOrEmpty(PasswordBox.Password))
    {
        MessageBox.Show("Введите логин и пароль!");
        return;
    }
}
```

Добавим запрос к базе данных:

```
0 references | 0 changes | 0 authors, 0 changes
private void ButtonEnter_OnClick(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(TextBoxLogin.Text) || string.IsNullOrEmpty(PasswordBox.Password))
    {
        MessageBox.Show("Введите логин и пароль!");
        return;
    }

    using (var db = new Entities())
    {
        var user = db.User
            .AsNoTracking()
            .FirstOrDefault(u => u.Login == TextBoxLogin.Text && u.Password == PasswordBox.Password);

        if (user == null)
        {
            MessageBox.Show("Пользователь с такими данными не найден!");
            return;
        }
    }
}
```

И теперь добавим переходы в зависимости от роли на меню пользователя (для этого необходимо создать страницы меню для каждого типа пользователя, CustomerMenu или DirectorMenu и тд)

```
0 references | 0 changes | 0 authors, 0 changes
private void ButtonEnter_OnClick(object sender, RoutedEventArgs e)
{
    if (string.IsNullOrEmpty(TextBoxLogin.Text) || string.IsNullOrEmpty(PasswordBox.Password))
    {
        MessageBox.Show("Введите логин и пароль!");
        return;
    }

    using (var db = new Entities())
    {
        var user = db.User
            .AsNoTracking()
            .FirstOrDefault(u => u.Login == TextBoxLogin.Text && u.Password == PasswordBox.Password);

        if (user == null)
        {
            MessageBox.Show("Пользователь с такими данными не найден!");
            return;
        }

        MessageBox.Show("Пользователь успешно найден!");
        // Переход на меню пользователя в зависимости от роли

        switch (user.Role)
        {
            case "Заказчик":
                NavigationService?.Navigate(new Menu());
                break;
            case "Директор":
                NavigationService?.Navigate(new Menu());
                break;
        }
    }
}
```

Для выполнения практического задания можно ознакомиться с обучающими видео

ПЕРЕХОД МЕЖДУ СТРАНИЦАМИ

Перейдем на базовую форму и создадим обработчик события Navigated у Frame.

```
<Rectangle Fill="#FFF4F4F5" Grid.Row="2" StrokeThickness="0"/>
<Button Content="Назад" HorizontalAlignment="Left" Margin="10,10,0,10" Width="75"/>
<Frame x:Name="MainFrame" Source="Pages/AuthPage.xaml" Grid.Row="1" Navigated="MainFrame_OnNavigated"/>
</Grid>
```

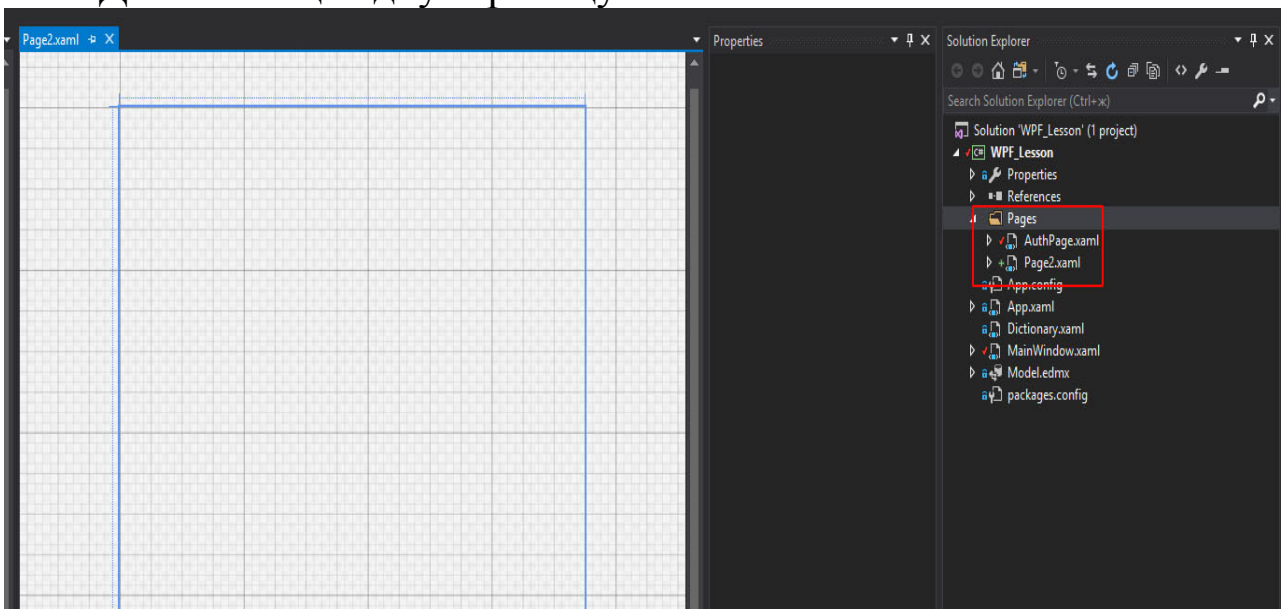
В обработчик события напишем следующий код. Сначала мы проверяем что получили ли мы страницу на вход, затем устанавливаем заголовок формы в соответствии с шаблоном, после в зависимости от страницы отображаем или скрываем кнопку «Назад».

```
– references | 0 changes | 0 authors, 0 changes
private void MainFrame_OnNavigated(object sender, NavigationEventArgs e)
{
    if (!(e.Content is Page page)) return;
    this.Title = $"LESSON - {page.Title}";

    if (page is AuthPage)
    {
        ButtonBack.Visibility = Visibility.Hidden;
    }
    else
    {
        ButtonBack.Visibility = Visibility.Visible;
    }
}
```

А в обработчик события нажатия кнопки «Назад». При нажатии на кнопку будет выполнен переход назад, если такой возможен.

```
private void ButtonBack_OnClick(object sender, RoutedEventArgs e)
{
    if(MainFrame.CanGoBack) MainFrame.GoBack();
}
```

Добавим еще одну страницу.



Теперь выполним переход со страницы AuthPage на Page2. Перейдем на страницу AuthPage и добавим обрабочик события на кнопку «Регистрация»:

```
<Button Content="Регистрация" Grid.Column="2" Margin="0,85,0,0" Grid.Row="1" VerticalAlignment="Top" Height="20" Click="ButtonRegistration_OnClick"/>
```

Напишем следующий код в обработчике:

```
- references | 0 changes | 0 authors, 0 changes
private void ButtonRegistration_OnClick(object sender, RoutedEventArgs e)
{
    NavigationService?.Navigate(new Page2());
}
```

После этого переход по кнопке будет осуществляться на страницу Page2, а по кнопке «Назад» обратно на AuthPage.

Для выполнения практического задания можно ознакомиться с обучающими видео

Задание:

1) Разработать элемент рабочей тетради по теме «Проектирование интерфейса»;
2) Разработать проект интерфейса программного продукта в соответствие с заданием (https://drive.google.com/drive/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-lkn);
3) Разработать лабораторный практикум по теме «Проектирование интерфейсов мобильных приложений».

МОДУЛЬ КОМПЕТЕНЦИИ 3: «ТЕСТИРОВАНИЕ ПРОГРАММНЫХ РЕШЕНИЙ»

Тестирование программных решений

Тестирование ПО – процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом. Тестирование (software testing) — деятельность, выполняемая для оценки и улучшения качества программного обеспечения [1]. Эта деятельность, в общем случае, базируется на обнаружении дефектов и проблем в программных системах.

В соответствие с IEEE Std 829-1998 тестирование – это процесс анализа ПО, направленный на выявление отличий между его реально существующими и требуемыми свойствами (дефект) и на оценку свойств ПО.

По ГОСТ Р ИСО МЭК 12207-99 в жизненном цикле ПО определены среди прочих вспомогательные процессы верификации, аттестации, совместного анализа и аудита.

Процесс верификации является процессом определения того, что программные продукты функционируют в полном соответствии с требованиями или условиями, реализованными в предшествующих работах. Данный процесс может включать анализ, проверку и испытание (тестирование).

Процесс аттестации предусматривает определение полноты соответствия требований и системы их конкретному функциональному назначению. Под аттестацией понимается подтверждение и оценка достоверности проведенного тестирования ИС. Аттестация должна гарантировать полное соответствие спецификациям, требованиям и документации. Аттестацию выполняют путем тестирования во всех возможных ситуациях и используют при этом независимых специалистов.

Процесс совместного анализа является процессом оценки состояний и, при необходимости, результатов работ (продуктов) по проекту.

Процесс аудита является процессом определения соответствия требованиям, планам и условиям договора. В сумме эти процессы и составляют то, что обычно называют тестированием.

Тестирование основывается на тестовых процедурах с конкретными входными данными, начальными условиями и ожидаемым результатом, разработанными для определенной цели, такой, как проверка отдельной программы или верификация соответствия на определенное требование [2]. Тестовые процедуры могут проверять различные аспекты функционирования программы — от правильной работы отдельной функции до адекватного выполнения бизнес-требований.

Основные цели тестирования:

- проверить взаимодействие между объектами;
- проверить корректную интеграцию всех модулей системы;
- проверить, что все требования были корректно реализованы;
- идентифицировать дефекты и убедиться, что они максимально выявлены еще до развертывания системы.

Этапы процесса тестирования

На рисунке 1 представлены этапы процесса тестирования программного обеспечения.

Рисунок 1 - Этапы процесса тестирования

Планирование и подготовка процесса тестирования состоит в анализе требований, предъявляемых к программному продукту, выборе стратегии тестирования, целей и приоритетов.

Создание тест-кейсов. Тест-кейсы должны быть основаны на требованиях к программному продукту, должны покрывать все эти требования и иметь приоритет.

Проверка на критические ошибки, блокирующие процесс тестирования. Если такие ошибки были обнаружены, то программу сразу отправляют на исправление разработчикам. Обычно такие ошибки не заносятся в базу данных дефектов.

Выполнение тест-кейсов – проверка соответствия результатов работы программы ожидаемым результатам.

Описание дефектов. Дефект – выявленное в процессе тестирования несоответствие полученных и ожидаемых результатов.

Проверка и устранение дефектов. Жизненный цикл дефекта представлен на рисунке 2.

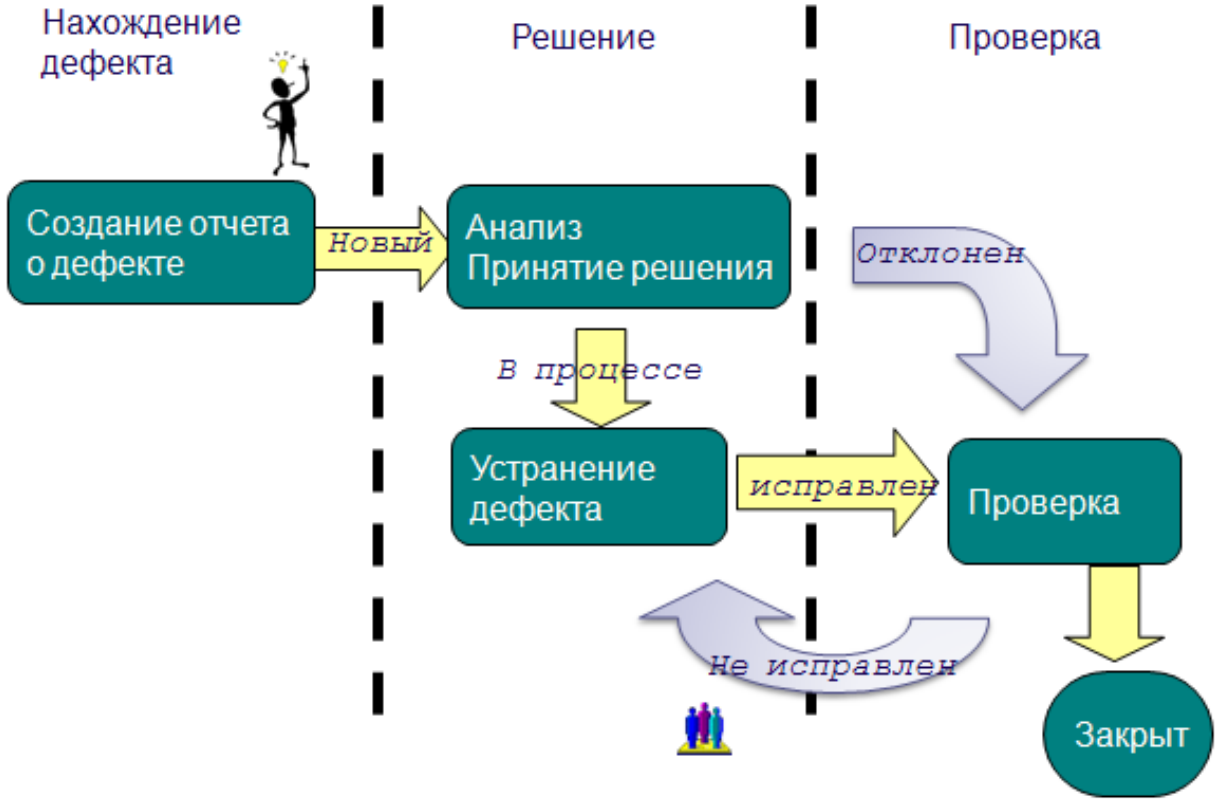


Рисунок 2 - Жизненный цикл дефекта

Автоматизация тестирования выполняется для того, чтобы уже написанные и проверенные один раз тест-кейсы выполнялись автоматически. Повторное выполнение тестов необходимо, чтобы убедиться, что во время исправления дефектов не было внесено новых ошибок. В настоящее время существует много программных продуктов, предназначенных для автоматизации тестирования. Идея этих продуктов заключается в создании централизованного репозитория для хранения,

доступа и управления всеми составляющими компонентами процесса тестирования. Именно с использования такого инструментария, как правило, и начинается переход от тестирования вручную к внедрению автоматизированных средств. Одно из важных требований к инструменту подобного класса — возможность использования обычного браузера в качестве клиентской части, что упрощает установку, настройку и последующую поддержку продукта [3].

Отчет о тестировании. Отчеты о тестировании могут формироваться в различных точках в течение процесса тестирования. Отчеты о тестировании будут суммировать результаты тестирования и документировать любой анализ. Отчет о приемочном тестировании часто является договорным документом, подтверждающим приемку ПО.

Существует несколько признаков, по которым принято производить классификацию видов тестирования. Обычно выделяют следующие признаки [4]:

По объекту тестирования:

- Функциональное тестирование (functional testing);
- Нагрузочное тестирование:
  - Тестирование производительности (perfomance/stress testing);
  - Тестирование стабильности (stability/load testing);
- Тестирование удобства использования (usability testing);
- Тестирование интерфейса пользователя (UI testing);
- Тестирование безопасности (security testing);
- Тестирование локализации (localization testing);
- Тестирование совместимости (compatibility testing).

По знанию системы:

- Тестирование чёрного ящика (black box)

При тестировании чёрного ящика, инженер по тестированию имеет доступ к ПО только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования. Например, тестирующий модуль может виртуально нажимать клавиши или кнопки мыши в тестируемой программе с помощью механизма взаимодействия процессов, с уверенностью в том, все ли идёт правильно, что эти события вызывают тот же отклик, что и реальные нажатия клавиш и кнопок мыши. Как правило, тестирование чёрного ящика ведётся с использованием спецификаций или иных документов, описывающих требования к системе.

- Тестирование белого ящика (white box)

При тестировании белого ящика (англ. *white-box testing*, также говорят — *прозрачного ящика*), разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого ПО. Это типично для юнит-тестирования (англ. *unit testing*), при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции — работоспособны и устойчивы, до определённой степени. При тестировании белого ящика используются метрики покрытия кода.

- Тестирование серого ящика (gray box).

По степени автоматизированности:

- Ручное тестирование (manual testing);
- Автоматизированное тестирование (automated testing);
- Полуавтоматизированное тестирование (semiautomated testing).

По степени изолированности компонентов:

- Компонентное (модульное) тестирование (component/unit testing);
- Интеграционное тестирование (integration testing);
- Системное тестирование (system/end-to-end testing).

По времени проведения тестирования:

- Альфа тестирование (alpha testing):
  - Тестирование при приёмке (smoke testing);
  - Тестирование новых функциональностей (new feature testing);
  - Регрессионное тестирование (regression testing);
  - Тестирование при сдаче (acceptance testing).

В альфа-тестировании программу тестирует разработчик с точки зрения пользователя. Ближе к выходу продукта наступает следующая стадия тестирования – бета-тестирование, когда на тестирование программу отдают реальным, конечным пользователям.

- Бета тестирование (beta testing)

В бета-версии обычно ограничивают функциональность, время использования программы, взамен обратной связи, а также найденных ошибок.

По признаку позитивности сценариев:

- Позитивное тестирование (positive testing)
- Негативное тестирование (negative testing)

По степени подготовленности к тестированию:

- Тестирование по документации (formal testing)

- Эд Хок (интуитивное) тестирование (ad hoc testing)

Статическое и динамическое тестирование

Описанные выше техники — тестирование белого ящика и тестирование чёрного ящика — предполагают, что код исполняется, и разница состоит лишь в той информации, которой владеет инженер по тестированию. В обоих случаях это динамическое тестирование.

При статическом тестировании программный код не выполняется — анализ программы происходит на основе исходного кода, который вычитывается вручную, либо анализируется специальными инструментами. В некоторых случаях, анализируется не исходный, а промежуточный код (такой как байт-код или код на MSIL) [5].

На рисунке 3 представлена классификация техник тестирования программного обеспечения.

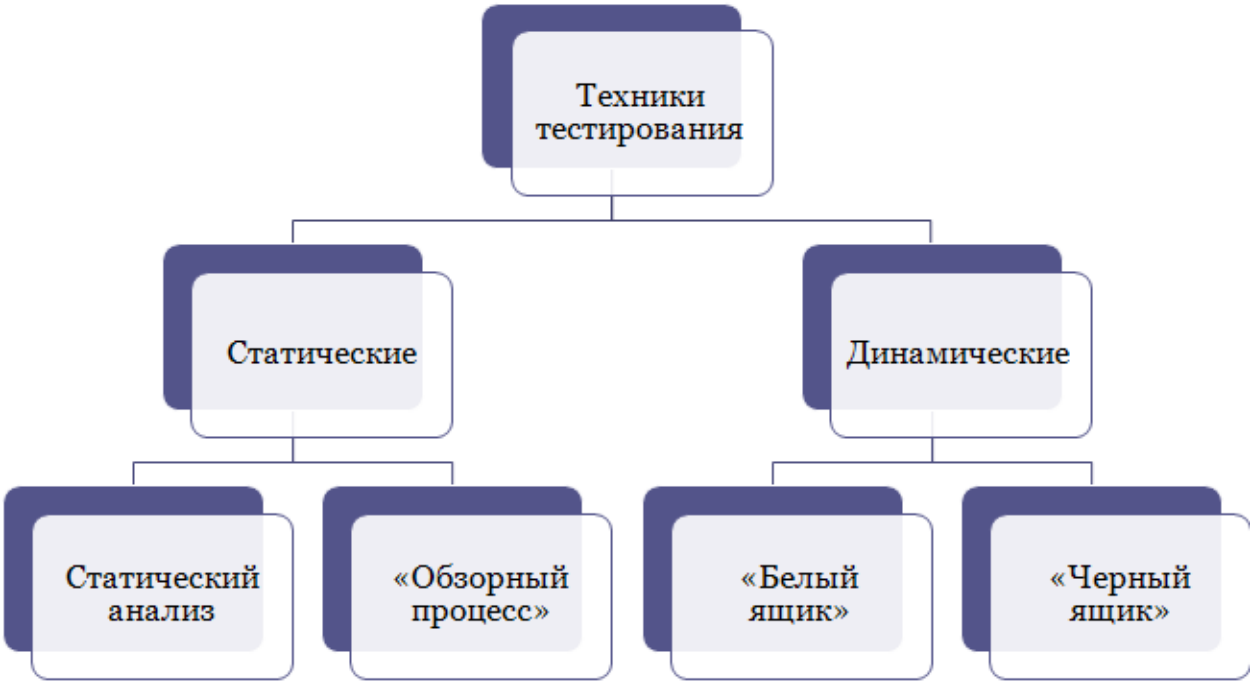


Рисунок 3 - Техники тестирования программного обеспечения

Также к статическому тестированию относят тестирование требований, спецификаций, документации.

Тестирование обычно производится на протяжении всей разработки и сопровождения на разных уровнях. Уровень тестирования определяет «над чем» производятся тесты: над отдельным модулем, группой модулей или системой, в целом. При этом ни один из уровней тестирования не может считаться приоритетным. Важны все уровни тестирования, вне зависимости от используемых моделей и методологий [6].

Модульное тестирование (Unit testing). Этот уровень тестирования позволяет проверить функционирование отдельно взятого элемента системы. Что считать элементом – модулем системы определяется контекстом. Наиболее полно данный вид тестов описан в стандарте IEEE 1008-87 «Standard for Software Unit Testing», задающем интегрированную концепцию систематического и документированного подхода к модульному тестированию.

Интеграционное тестирование (Integration testing). Данный уровень тестирования является процессом проверки взаимодействия между программными компонентами/модулями.

Системное тестирование (System testing). Системное тестирование охватывает целиком всю систему. Большинство функциональных сбоев должно быть идентифицировано еще на уровне модульных и интеграционных тестов. В свою очередь, системное тестирование, обычно фокусируется на нефункциональных требованиях – безопасности, производительности, точности, надежности т.п. На этом уровне также тестируются интерфейсы к внешним приложениям, аппаратному обеспечению, операционной среде и т.д.

Рассмотрим подробнее каждый из уровней тестирования:

Модульное (unit) тестирование [1] – проверка функционирования первого компонента системы, самого элементарного. Обычно берется самый минимально возможный для тестирования компонент, например, одна функция программы.

В данном виде тестирования используется метод «белого ящика». Обычно модульное тестирование выполняется программистами.

Цель этого вида тестирования – изолировать отдельные части программы, протестировать их и показать, что в отдельности они работоспособны.

Преимущество этого вида тестирования в том, что программисты довольно легко идут на изменение программы, не сопротивляясь нововведениям. Это объясняется тем, что протестировать отдельный модуль после изменения достаточно просто. В одном модуле получается достаточно маленький набор вариантов развития событий, и достаточно легко рассмотреть их все.

Этот вид тестирования помогает локализовать ошибку.

Локализовать ошибку – значит определить место, где содержится ошибка, т.е. в каком модуле, в какой функции она произошла. Когда мы находим ошибку, мы пытаемся установить, при каком наборе действий

она возникает, и обобщить его. Затем пытаемся локализовать, т.е. определить в каком компоненте программы происходит сбой.

Юнит-тестирование не решит проблемы производительности, качества, безопасности, надежности. Поэтому остаётся еще целый ряд непроверенных параметров, для которых заданы определенные требования при разработке продукта. Проблемы взаимодействия компонентов этот вид тестирования также не решает.

Этот вид тестирования отдельно никогда не используют, только с другими видами тестирования.

На выходе этого вида тестирования мы получаем протестированные модули программы. Они в свою очередь подаются на вход следующего уровня тестирования – интеграционного тестирования.

Этот вид тестирования проверяет взаимодействие компонентов, взаимодействие модулей, которые общаются между собой. Модули передают информацию друг другу и каким-то образом связаны. Вот на этой стыковке и могут быть сбои в передаче этой информации. Либо она может передаваться не точно, либо вообще не передаваться, либо искажаться при передаче. Поэтому тестировать взаимосвязь компонентов нужно обязательно.

Таким образом, мы получаем на входе протестированные модули, затем объединяем их в группы, тестируем эти группы. И на выходе получаем протестированные группы модулей.

Этот вид тестирования проводится через интерфейс программы - методом «черного ящика».

Также с помощью метода «черного ящика» тестируется следующий уровень.

На вход системного тестирования мы подаем выходные данные интеграционного тестирования, т.е. протестированные группы модулей.

На выходе же системного тестирования мы получаем полностью протестированную программу.

В системном тестировании мы проверяем всю систему целиком.

Какие ошибки обычно выявляются на этом уровне тестирования?

Это ошибки надежности, безопасности, производительности. Также на этом уровне тестируется интерфейс для внешнего окружения, например, доступ к другим программам, доступ к операционной системе, доступ к «железу» компьютера. Знания внутреннего устройства работы программы не требуются.

Таким образом, модульное тестирование - проводим методом «белого ящика», обнаруживаем ошибки функциональности. Затем

проводим интеграционное тестирование - методом «черного ящика», обнаруживаем ошибки взаимодействия модулей программы. И, наконец, системное тестирование – методом «черного ящика», выявляем ошибки производительности, надежности, безопасности и других параметров.

В системном тестировании можно выделить два этапа, которые будут являться стадиями разработки продукта: альфа-тестирование, бета-тестирование.

Выполнение задач процесса тестирования программных комплексов сопровождается разработкой различных артефактов (документов, моделей и других материалов проекта). Обычно разработка артефактов может проводиться в разной форме с разными требованиями к способу выполнения, рецензированию и качеству оформления.

Ниже представлены основные рабочие артефакты тестировщиков, в той или иной форме связанные со сценариями использования. Эти документы необходимо передавать заказчику или группе сопровождения и технической поддержки системы в случае необходимости.

План тестирования (Test plan). План тестирования определяется международным стандартом IEEE 829-1998. В нем должны быть предусмотрены как минимум три раздела содержащие, следующие описания:

- что будет тестироваться (тестовые требования, тестовые варианты);
- какими методами и насколько подробно будет тестироваться система;
- план-график работ и требуемые ресурсы (персонал, техника) (Shedule).

Дополнительно описываются критерии удачного/неудачного завершения тестов, критерии окончания тестирования, риски, непредвиденные ситуации, приводятся ссылки на соответствующие разделы в основных документах проекта - план управления требованиями, план конфигурации [8].

Сценарий тестирования (Test case, тест кейс). Это один из основных документов, с которыми имеет дело тестировщик. По сути, упрощенное описание теста. То есть входной информации, условий и последовательности выполнения действий и ожидаемого выходного результата. Учитывая, что даже успешно прошедшие тесты выполняются неоднократно в ходе регрессионного тестирования, наличие таких описаний необходимо. Однако уровень формальных

требований к их оформлению может меняться в очень широких пределах. Одно дело, если вы собираетесь использовать тесты в ходе приемочных испытаний, проводимых заказчиком, и другое — в ходе внутреннего тестирования коробочного продукта.

Тест скрипт(Test script). Обычно говорят о программной реализации теста, хотя скрипт может описывать и ручные действия, необходимые для выполнения конкретного тест кейса.

Набор тестов(Test set). Как правило, сценарии тестирования объединяются в пакеты или наборы. Во-первых, это просто способ группирования тестов со сходными задачами, а, во-вторых, в такой набор можно включать зависимые тесты, которые должны выполняться в определенном порядке (поскольку последующие тесты используют данные, сформированные в ходе выполнения предыдущих).

Список идей тестов. Использование списка идей тестов для анализа и проектирования системы сценариев использования существенно упрощает задачу разработки необходимого набора тестов. Основной объем тестов строится как проверка различных вариантов выполнения каждого сценария использования. Однако тесты не сводятся к сценариям использования, как и задачи тестирования не сводятся только лишь к проверке функциональных требований к системе. Проверка нефункциональных требований может потребовать использования специальных приемов и подходов. Соответствующие тесты не всегда очевидны. Для таких ситуаций и создается список идей тестов. В него все желающие могут записать «что и как» стоит еще проверить. Этот список является внутренним рабочим документом группы тестирования. Наиболее разумная форма его ведения — электронный документ с минимальными формальными требованиями к оформлению.

Модель нагрузки. Сценарии использования, как правило, описывают взаимодействие с системой одного пользователя, часто этого бывает мало. При тестировании систем необходимо учитывать возможность параллельной работы большого числа пользователей, решающих различные задачи. Модель реальной нагрузки описывает характеристики типового «потока заявок», которые должны использоваться для нагрузочного тестирования, имитирующего работу системы в реальных условиях. Также могут быть созданы стрессовые модели нагрузки для тестирования отказоустойчивости системы.

Дефекты(Deffects). Основополагающие артефакты процесса тестирования – описывают обнаруженные факты несоответствия

системы предъявляемым требованиям [9]. Являются одним из подтипов запросов на изменение, описывающих найденную ошибку или несоответствие на всех этапах тестирования. Хотя базу данных дефектов можно вести в текстовом файле или Excel таблице, предпочтительным является использования специализированного инструментального средства, которое позволяет передавать информацию об обнаруженных дефектах от тестировщиков к разработчикам, а в обратную сторону – сведения об устранении дефектов. А также формировать необходимые отчеты о тенденциях изменения количества обнаруживаемых и устраняемых дефектов.

Журнал тестирования. Каждое выполнение теста должно быть зарегистрировано в журнале тестирования. Журнал тестирования будет содержать записи о том, когда запускался каждый тест, итог выполнения каждого теста и может также содержать важные наблюдения, сделанные при выполнении теста. Зачастую журнал тестирования не ведут для нижних уровней тестирования (тестов компонент и интеграции ПО).

Отчеты о тестировании. Отчеты о тестировании могут формироваться в различных точках в течение процесса тестирования. Отчеты о тестировании будут суммировать результаты тестирования и документировать любой анализ. Отчет о приемочном тестировании часто является договорным документом, подтверждающим приемку ПО.

Функциональность

Функциональное тестирование объекта-тестирования планируется и проводится на основе требований к тестированию, заданных на этапе определения требований. В качестве требований выступают диаграммы use-case, бизнес-функции и бизнес-правила. Цель функциональных тестов состоит в том, чтобы проверить соответствие разработанных графических компонентов установленным требованиям. В основе функционального тестирования лежит методика «черного ящика» [3]. Идея тестирования сводится к тому, что группа тестировщиков проводит тестирование, не имея доступа к исходным текстам тестируемого приложения. При этом во внимание принимается только входящие требования и соответствие им тестируемым приложением.

Цель тестирования:

Убедиться в надлежащем функционировании объекта тестирования. Тестируется правильность навигации по объекту, а также ввод, обработка и вывод данных.

Методика:

Необходимо исполнить (проиграть) каждый из use-case, используя как верные значения, так и заведомо ошибочные, для подтверждения правильного функционирования, по следующим критериям:

- продукт адекватно реагирует на все вводимые данные (выводятся ожидаемые результаты в ответ на правильно вводимые данные);
- продукт адекватно реагирует на неправильно вводимые данные (появляются соответствующие сообщения об ошибках);
- каждое бизнес-правило реализовано надлежащим (установленным) образом.

Критерии Завершения:

Все запланированные действия по тестированию выполнены.

Все найденные дефекты были соответствующим образом обработаны (документированы и помещены в базу дефектов).

Целостность данных и баз данных

Цель Тестирования:

Убедиться в надежности методов доступа к базам данных, в их правильном исполнении, без нарушения целостности данных.

Методика:

Необходимо последовательно испробовать максимально возможное число способов обращения к базе. Используется подход, при котором тест составляется таким образом, чтобы «нагрузить» базу последовательностью, как верных значений, так и заведомо ошибочных.

После этого необходимо оценить правильность внесения данных и убедиться в корректной обработке базой входящих значений.

Критерии Завершения:

Все способы доступа функционируют, в соответствии с требованиями.

Действия скрипта не приводят к потере данных или нарушению целостности базы, либо к другим неадекватным реакциям.

Пользовательский интерфейс

Цель Тестирования:

Проверить правильность навигации по объекту тестирования (в том числе межоконные переходы, переходы между полями, правильность обработки клавиш «enter» и «tab», работа с мышью, функционирование клавиш-акселераторов и полное соответствие индустриальным стандартам);

Проверить объекты и их характеристики (меню, размеры, положения, состояния, фокус ввода и др.) на соответствия общепринятым стандартам на графический интерфейс пользователя.

Методика:

Создаются или дорабатываются тесты для каждого из окон, на предмет соответствия навигации и состояний каждого из объектов.

Критерии завершения:

Каждое окно протестировано и удовлетворяет, базовой линии поведения, требованиям стандартов и не противоречит проектным требованиям. Все выявленные дефекты обработаны и документированы.

Описание процесса тестирования как этапа разработки программного обеспечения

Процесс тестирования - один из основных процессов общего процесса разработки программного обеспечения. Для того чтобы выработать правильный алгоритм процесса тестирования, определить стратегии тестирования, запланировать процесс тестирования заданного программного комплекса необходимо знать место процесса тестирования в общем процессе разработки программного обеспечения. Покажем место тестирования в общем процессе разработки программного обеспечения, определим объекты тестирования, уровни тестирования на карте процесса тестирования, которая приведена на рисунке 4.

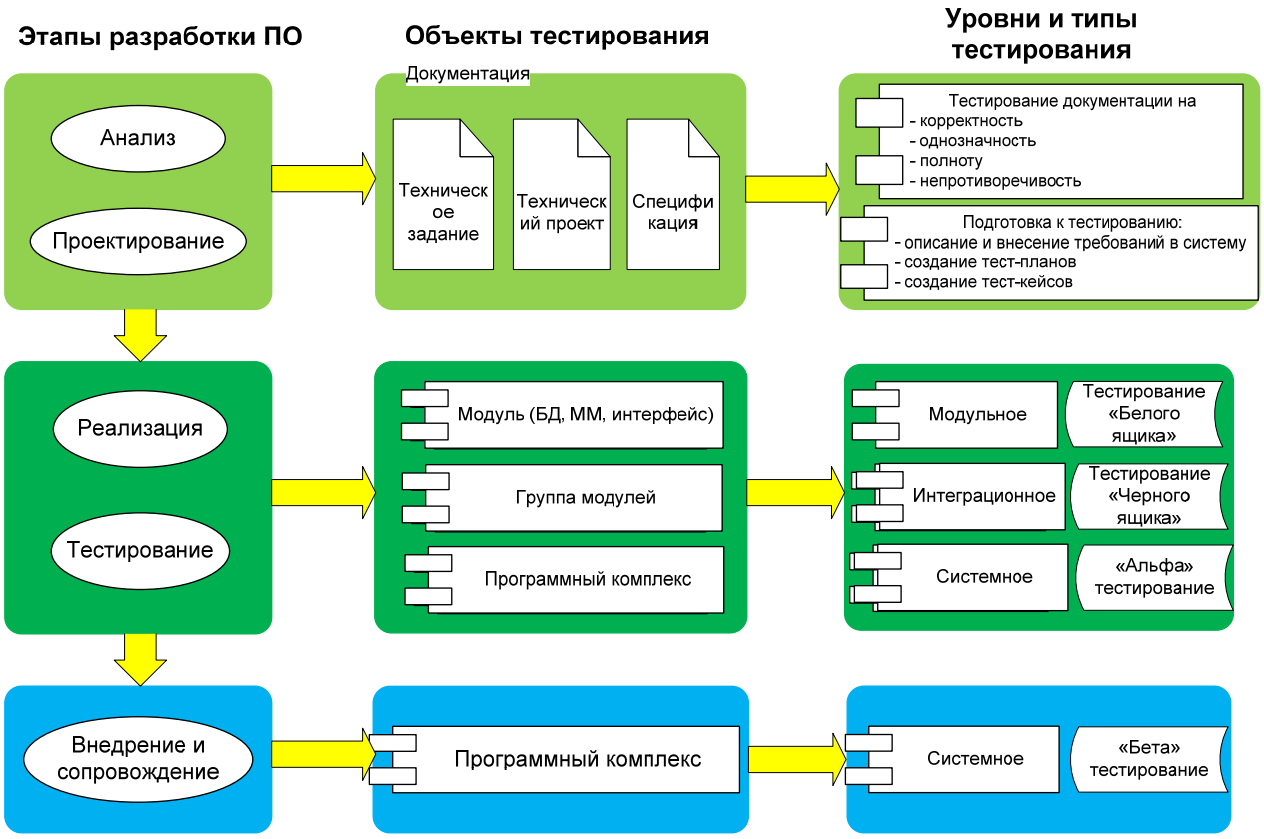


Рисунок 4 – Место тестирования в процессе разработки ПО

Анализируя карту процесса тестирования, мы видим, что процесс тестирования начинается на этапе проектирования и разработки технической документации на программный комплекс. Правильно построенный процесс тестирования не может начинаться на этапе программирования или внедрения и сопровождения. Только таким образом построенный процесс тестирования позволит на ранних стадиях разработки, а именно, на стадии разработки требований к программным комплексам, выявить дефекты, которые в будущем могут негативно сказаться на разрабатываемом программном комплексе. Объектами тестирования на этапе разработки требований к программному комплексу является техническая документация в виде технических заданий, спецификаций на отдельные модули или систему в целом, руководства пользователей. Этап тестирования заключается в регистрации и анализе требований к программному комплексу в системе поддержки процесса тестирования через модуль управления требованиями к ПО. Тестирование документации проводится на корректность, полноту, однозначность, непротиворечивость, тестируемость, упорядоченность, модифицируемость, отслеживаемость. На данном этапе процесса тестирования инженер по тестированию обращается к базе данных документов. В результате тестирования документации и ее анализа необходимо перейти к этапу планирования процесса тестирования конкретного модуля, интеграции модулей или системы в целом. Руководитель группы тестирования составляет план тестирования, определяет стратегии тестирования, при этом он работает с базой данных заданий на тестирование. Группа тестирования начинает проектирование тестов, подготовку тестовых данных, тестовой среды и окружения. Как только группа тестирования получает от разработчиков модуль, группу модулей или программный комплекс начинается этап выполнения тестов и регистрации дефектов. При этом идет пополнение базы данных тестов и дефектов. Руководитель группы тестирования, в свою очередь, работая с базами данных тестов и дефектов через подсистему генерации отчетности, получает информацию о ходе процесса тестирования и анализирует состояние тестируемого программного комплекса. Как только выполняется условие завершения работ (закончилось время, отведенное на тестирование; закончились денежные средства, выделенные на тестирование проекта; найдены дефекты, неисправленными остались лишь незначительные дефекты), руководитель группы тестирования составляет отчет о тестировании и программный комплекс передается в эксплуатацию. Таким образом,

происходит тестирование программного комплекса на всех трех уровнях тестирования: модульном, интеграционном, системном, изменяется лишь объект тестирования.

Модель работы с дефектами

Каждый обнаруженный дефект программного комплекса должен быть зарегистрирован и с ним должна быть проведена работа, которая в конечном итоге должна привести к исправлению дефекта. Работа с дефектами осуществляется по выбранной модели работы с дефектами. Выбор модели работы с дефектами зависит от конкретного проекта, структуры подразделения разработчиков и инженеров по тестированию. Для разработанной системы поддержки процесса тестирования была спроектирована модель работы с дефектами, которая представлена на рисунке 5:

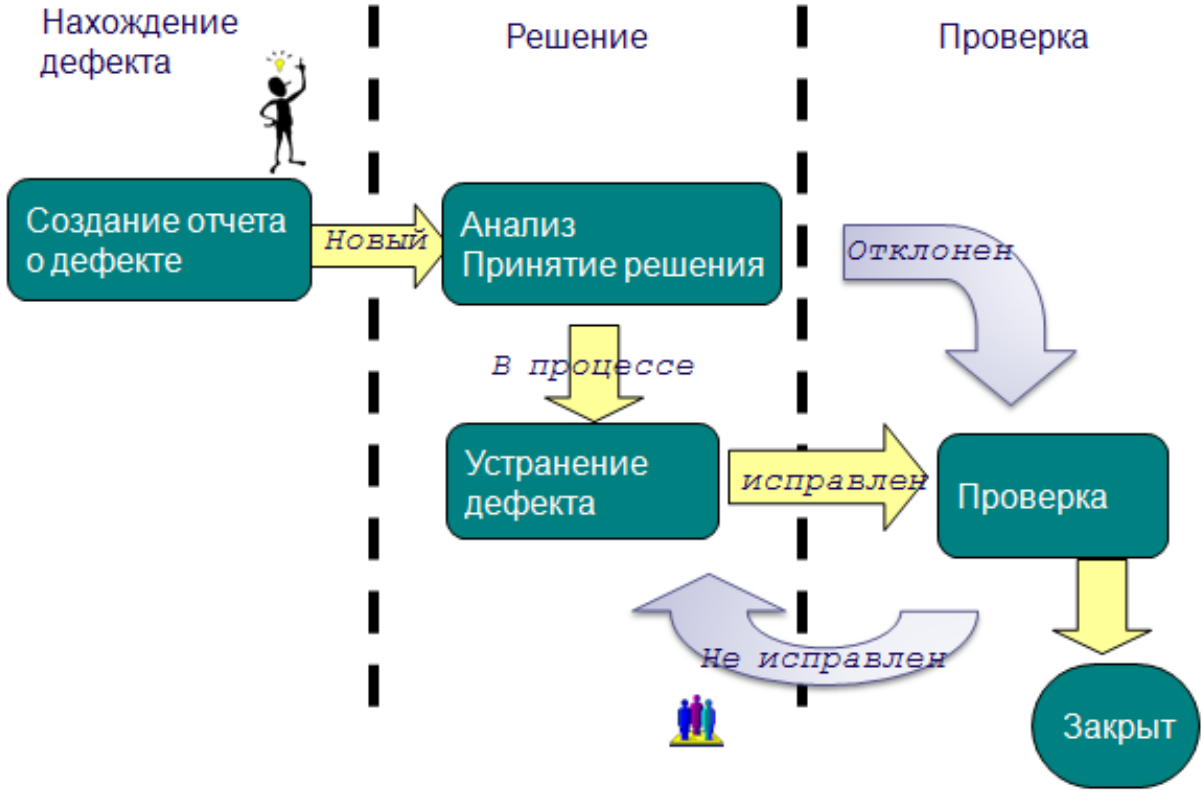


Рисунок 5 – Модель работы с дефектами

Как только дефект обнаружен, тестировщик регистрирует его в подсистеме работы с дефектами, присваивая ему статус «Новый» и направляет его на руководителя группы тестирования. Руководитель группы тестирования подтверждает дефект и направляет его на руководителя группы разработчиков или направляет дефект на инженера по тестированию на доработку. Далее руководитель группы разработчиков направляет дефект на программиста, отвечающего за

дефект, программист изучает суть дефекта и проставляет статус «В процессе». Если в данный момент необходимо отложить исправление дефекта, то разработчик может присвоить дефекту статус «Отложен». После исправления дефекта разработчик присваивает ему статус «Исправлен» и направляет дефект на руководителя группы тестировщиков, а тот направляет дефект на тестировщика. Инженер по тестированию проверяет дефект и если он исправлен, присваивает ему статус «Исправлен». Если дефект не исправлен, то ему присваивается статус «Не исправлен» и направляется на программиста для исправления.

Всем дефектам со статусом «Исправлен» руководитель группы тестирования присваивает статус «Закрыт».

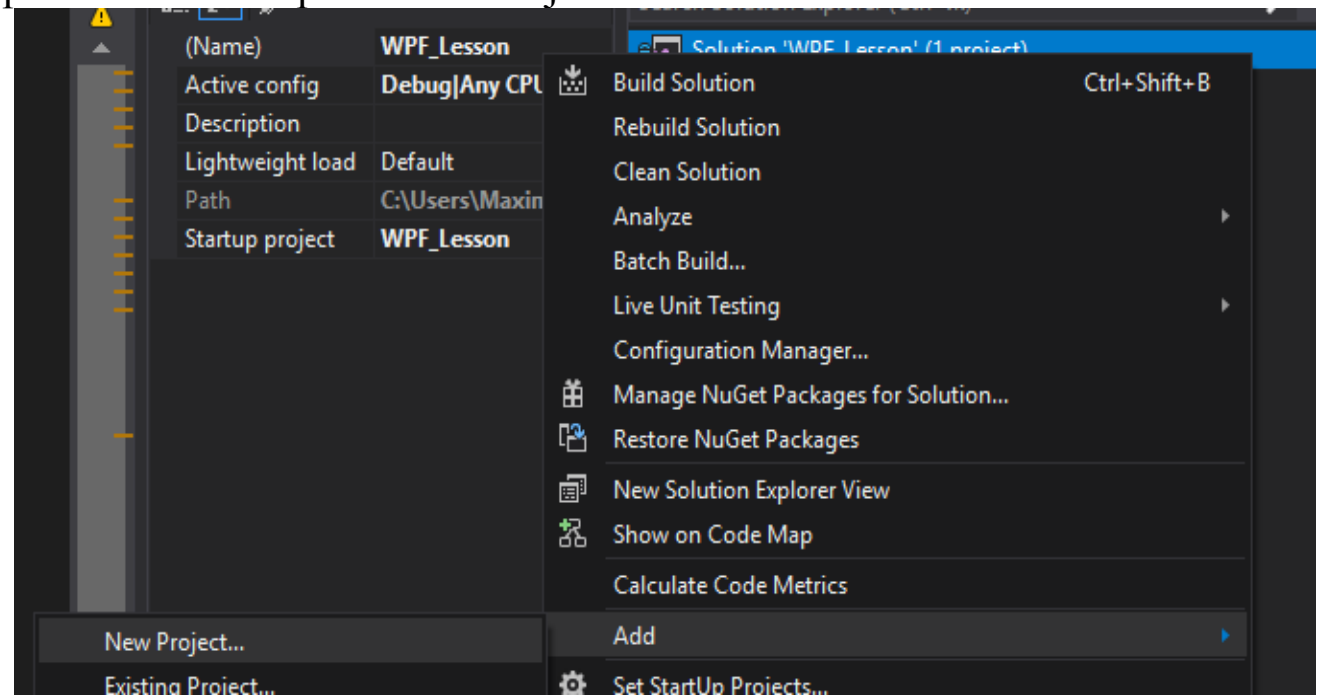
Кроме статусов атрибутами дефекта являются приоритет и важность. Приоритет дефекта показывает насколько быстро нужно исправить дефект, по приоритету определяется очередность выполнения задачи. Важность характеризует критичность дефекта по отношению к тестируемому приложению. Например, по признаку «важность» дефект может быть описан как «блокирующий» - не позволяющий приложению запуститься, «критичный» - некоторые функции приложения недоступны, «крупный» - какая-то функция не работает , «средний» - часть функции не работает и «второстепенный» - не влияющий не работу приложения (например, неправильное написание или неудобное расположение элементов управления). Приоритет в свою очередь может быть «высоким», «средним» и «низким».

Такая модель работы с дефектами позволяет полностью контролировать состояние тестируемого проекта, распределяет роли между участниками процесса тестирования и делает процесс работы с дефектами открытым и отслеживаемым как для тестировщиков, так и для разработчиков.

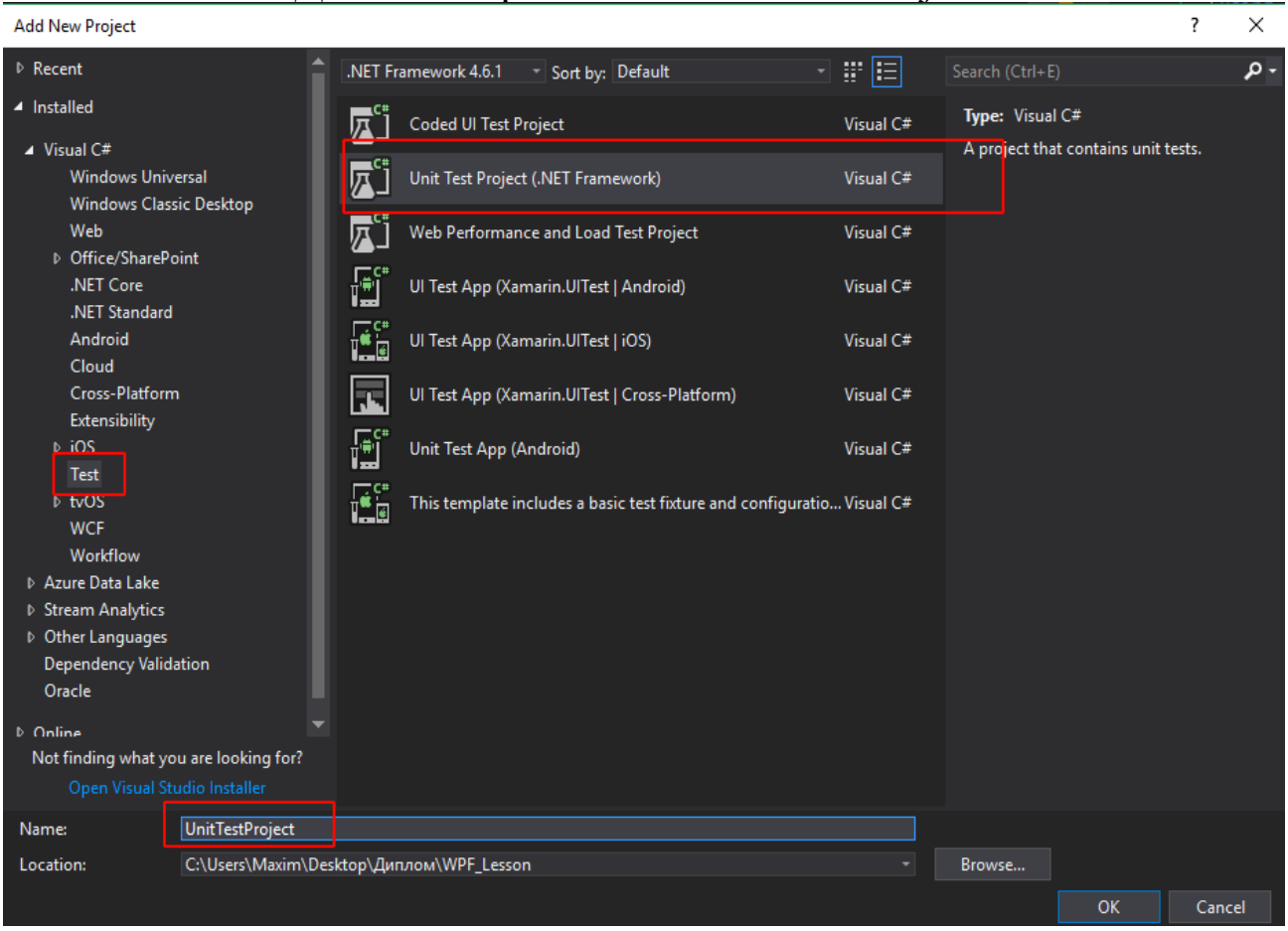
Рассмотрим пример создания unit-test.

СОЗДАНИЕ ПРОЕКТА ТЕСТИРОВАНИЯ

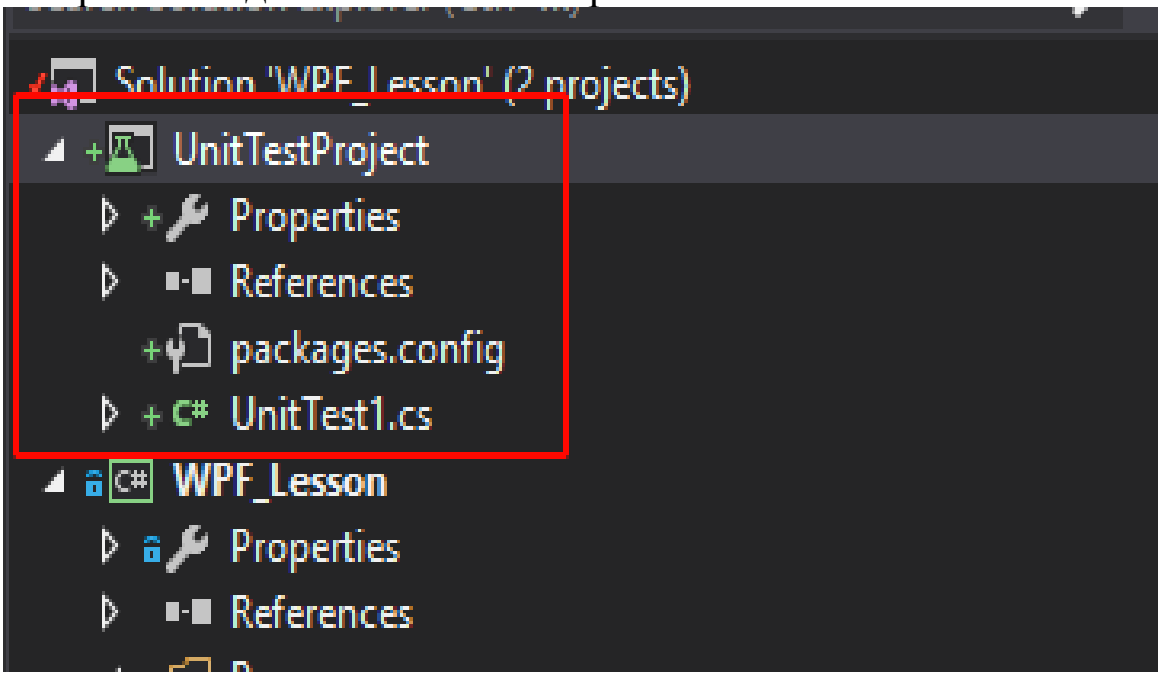
Откроем созданное решение, нажмем правой кнопкой мыши на решении и выберем «New Project»



Далее выберем Test – Unit Test Project.

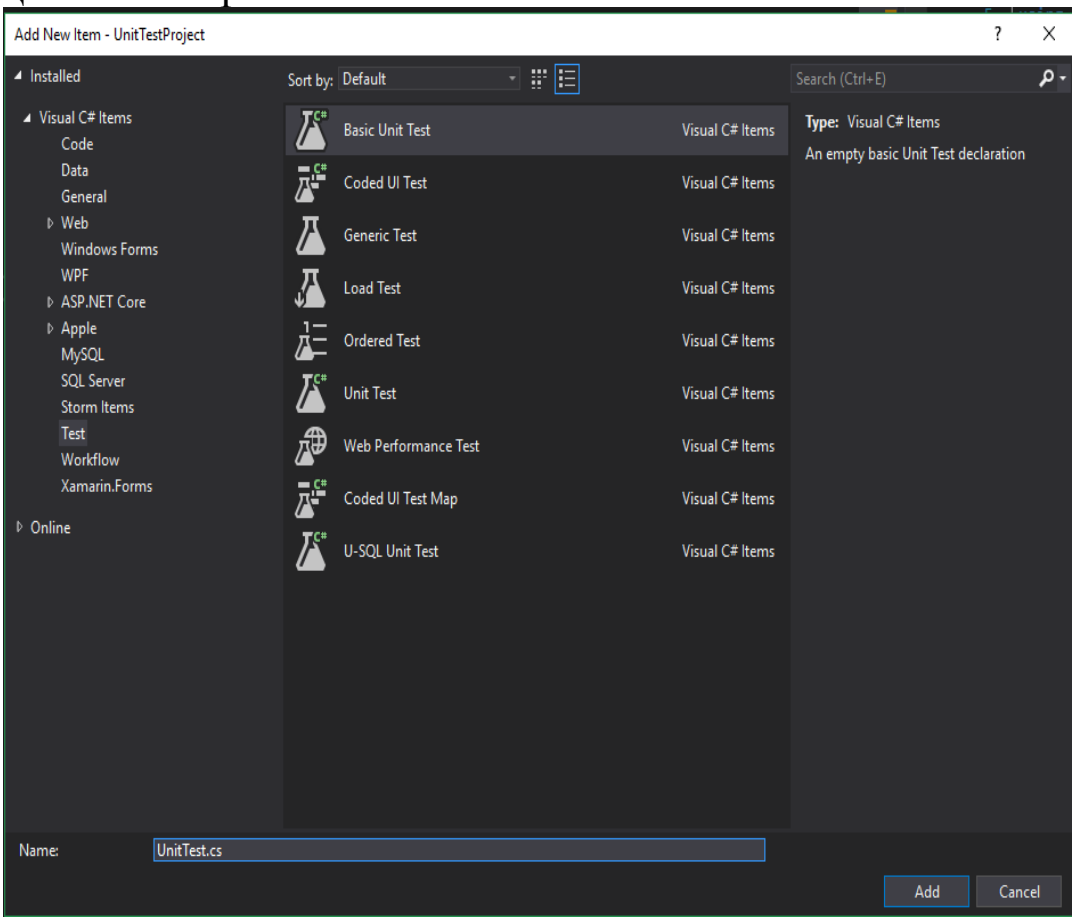


В решение добавится новый проект.



СОЗДАНИЕ ЮНИТ ТЕСТА

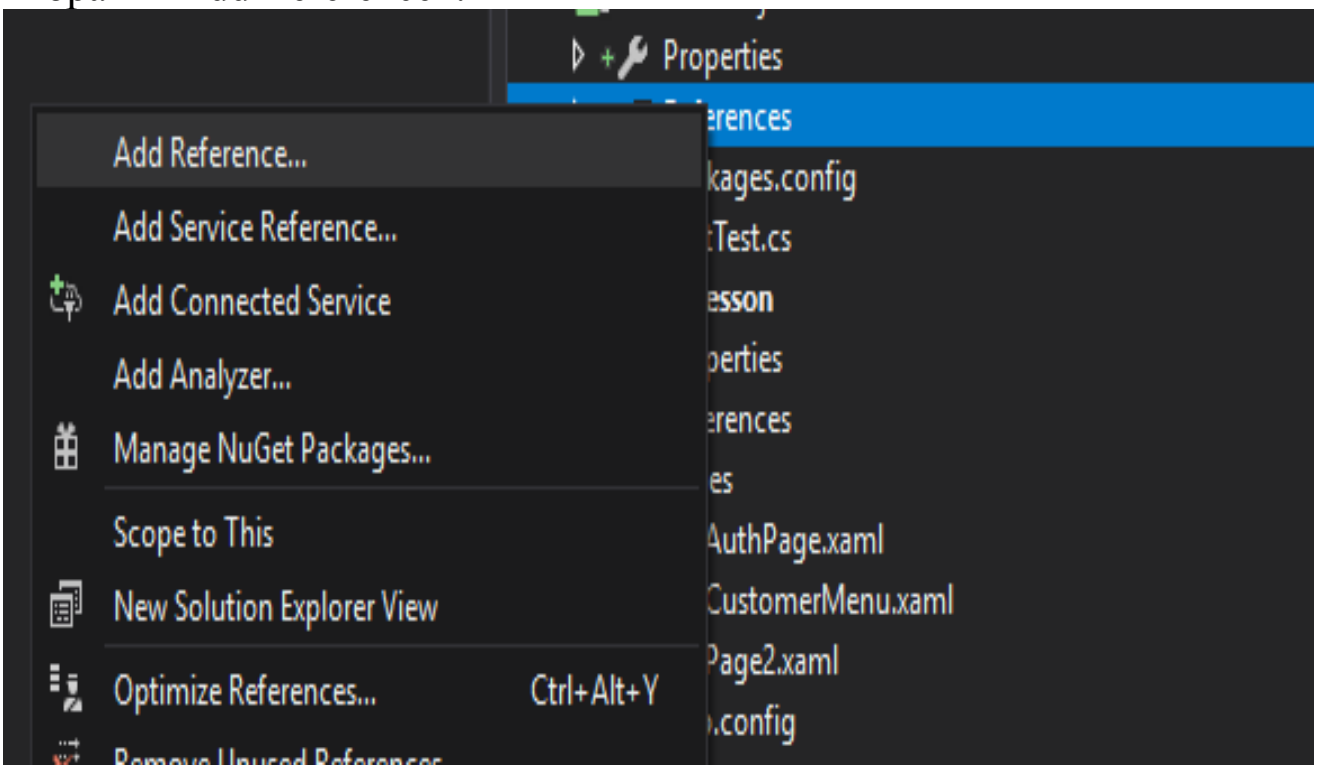
Добавим в проект базовый юнит-тест.



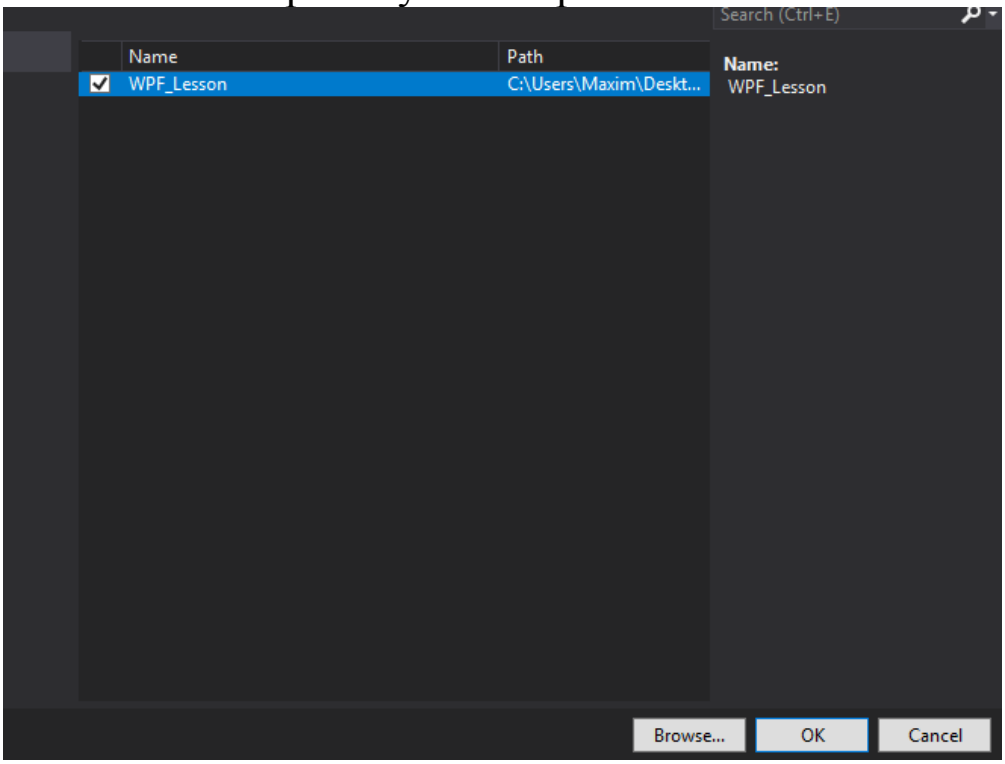
И напишем первый тест. Тест проверяет различные варианты ошибок.

```
[TestClass]
0 references | 0 changes | 0 authors, 0 changes
public class UnitTest
{
    [TestMethod]
    0 references | 0 changes | 0 authors, 0 changes
    public void TestMethod1()
    {
        int res = 2 + 2;

        Assert.AreEqual(res, 4);
        Assert.AreNotEqual(res, 5);
        Assert.IsFalse(res > 5);
        Assert.IsTrue(res < 5);


    }
}
```

Для тестирования проекта, подключим его к тестовому проекту. Для этого необходимо нажать правой кнопкой мыши по «References» и выбрать «Add Reference».



После чего выбрать нужный проект:



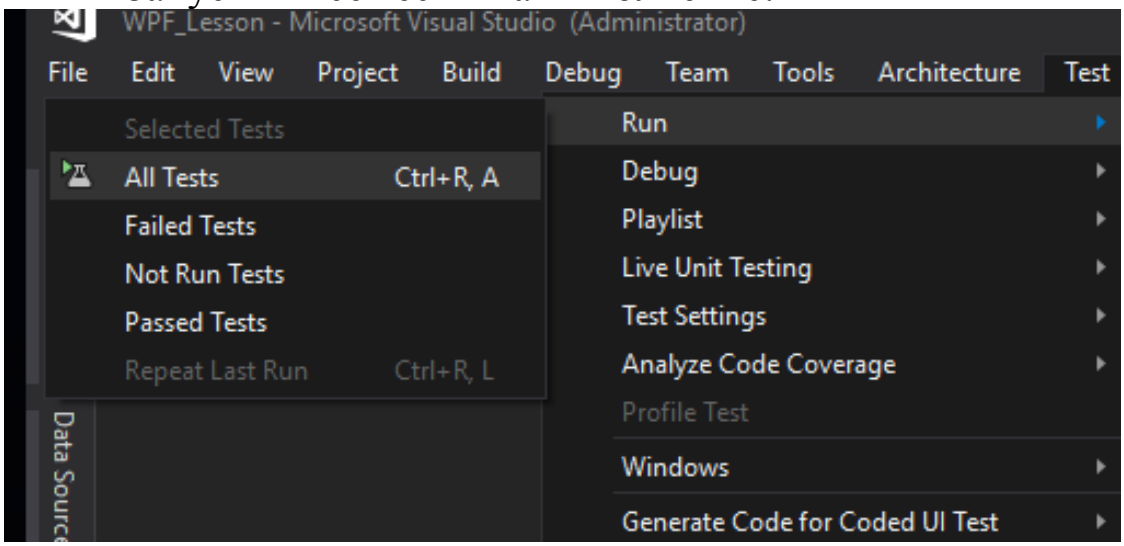
Для тестирования авторизации, нужно немного видоизменить форму авторизации:

```
1 reference | TheWorst, 1 hour ago | 1 author, 1 change
private void ButtonEnter_OnClick(object sender, RoutedEventArgs e)
{
    Auth(TextBoxLogin.Text, PasswordBox.Password);
}

1 reference | 0 changes | 0 authors, 0 changes
private bool Auth(string login, string password)
{
    if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))
    {
        MessageBox.Show("Введите логин и пароль!");
        return false;
    }

    using (var db = new Entities())
    {
        var user = db.User
            .AsNoTracking()
            .FirstOrDefault(u => u.Login == TextBoxLogin.Text && u.Password == PasswordBox.Password);

        if (user == null)
        {
            MessageBox.Show("Пользователь с такими данными не найден!");
            return false;
        }

        MessageBox.Show("Пользователь успешно найден!");
        // Переход на меню пользователя в зависимости от роли

        switch (user.Role)
        {
            case "Заказчик":
                NavigationService?.Navigate(new Menu());
                break;
            case "Директор":
                NavigationService?.Navigate(new Menu());
                break;
        }

        return true;
    }
}
```

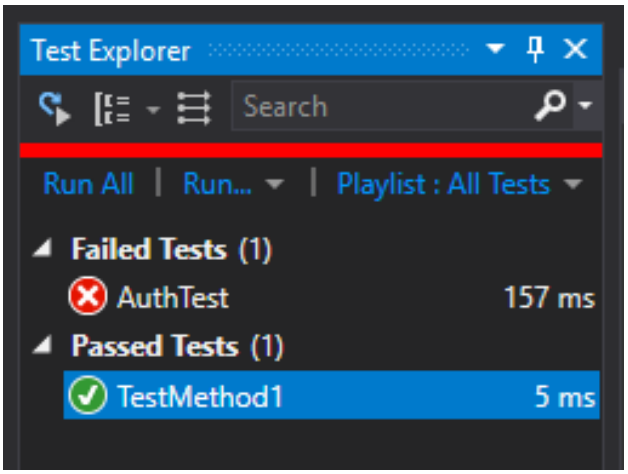
Теперь можно начинать тестировать авторизацию. Создадим новый метод для проверки авторизации и напишем следующий код:

```
[TestMethod]
0 references | 0 changes | 0 authors, 0 changes
public void AuthTest()
{
    var page = new AuthPage();

    Assert.IsTrue(page.Auth("test", "test"));
    Assert.IsFalse(page.Auth("Test", "test"));
    Assert.IsFalse(page.Auth("test", "Test"));
    Assert.IsFalse(page.Auth("test1", "test"));
    Assert.IsFalse(page.Auth("test", "test1"));
    Assert.IsFalse(page.Auth("", ""));
    Assert.IsFalse(page.Auth(" ", ""));
    Assert.IsFalse(page.Auth("", " "));
    Assert.IsFalse(page.Auth("^^", "&&"));
    Assert.IsFalse(page.Auth("^^", "^^"));
    Assert.IsFalse(page.Auth("346456457456745745745745", "12345464567658657"));
}
```

Запустим все тесты на выполнение:



После запуска можем увидеть какие тесты были пройдены, а какие нет.



Задание: Представьте себе, что ваша цель – тестирование приложения или сервиса, указанного в вашем варианте работы. Необходимо указать, какие тесты необходимы для покрытия различных видов, типов и областей тестирования, представленных в таблице 1. При этом нет необходимости перечислять все тесты. Необходимо привести 2-3 конкретных примера тестов (см. пример выполнения работы).

Таблица 1.

Тесты	Пример тестов
Различные виды тестирования	
Функциональное тестирование (Functional testing)	
Тестирование производительности (Performance testing)	
Нагрузочное тестирование (Load testing)	
Тестирование совместимости (Compatibility testing)	
Различные типы тестов	
Позитивные тесты	
Негативные тесты	
Исследовательские тесты	
Различные области тестирования	
Модульное тестирование	
Интеграционное тестирование	
Системное тестирование	

Пример выполнения работы: Тестирования форума тестировщиков (http://software-testing.ru/forum)

Тесты	Пример тестов
Различные виды тестирования	
Функциональные тесты (Functional testing)	Переход по разделам форума. Поиск по сайту. Подписка на рассылку - письма с информацией приходят.

Тесты производительности (Performance testing)	Скорость перехода по вкладкам Скорость поиска по ключевым словам
Нагрузочные тесты (Load testing)	Большое количество пользователей обращаются к разделам форума. Большой апдейт нескольких разделов
Тестирование совместимости (Compatibility testing)	Корректная работа форума в разных браузерах
Различные типы тестов	
Позитивные тесты	Правильность ссылок - ведут куда предполагалось. Правильность поиска по ключевым словам – находят нужные темы.
Негативные тесты	Не авторизованный пользователь не может оставлять комментарии на форуме Не модератор не может закрыть тему на форуме
Исследовательские тесты	Ввод информации символами с диакритическими знаками
Различные области тестирования	
Модульное тестирование	Тестирование каждого раздела в отдельности Тестирование модуля регистрации
Интеграционное тестирование	Авторизация: залогиниться в одном разделе, перейти в другой, система не выкинула – «продолжает» узнавать Правильно ли работают вместе модуль учета статистики и модуль добавления сообщений.
Системное тестирование	Основные сценарии использования форума соответствуют ожиданию. Встроенное видео проигрывается, картинки отображаются, текст отображается корректно.

Варианты:

1) Текстовый редактор Notepad.
2) Почтовый сервис Mail.Ru (www.mail.ru).
3) Графический редактор Paint.
4) Сервис хранения файлов Яндекс.Диск (http://disk.yandex.ru/).
5) Проигрыватель Windows Media Player.
6) Картографический сервис Google-карты (https://maps.google.ru/).
7) Браузер Internet Explorer.
8) Торрент-клиент µTorrent.
9) Архиватор WinRar.
10) Сервис прогноза погоды от Рамблер (http://weather.rambler.ru/)

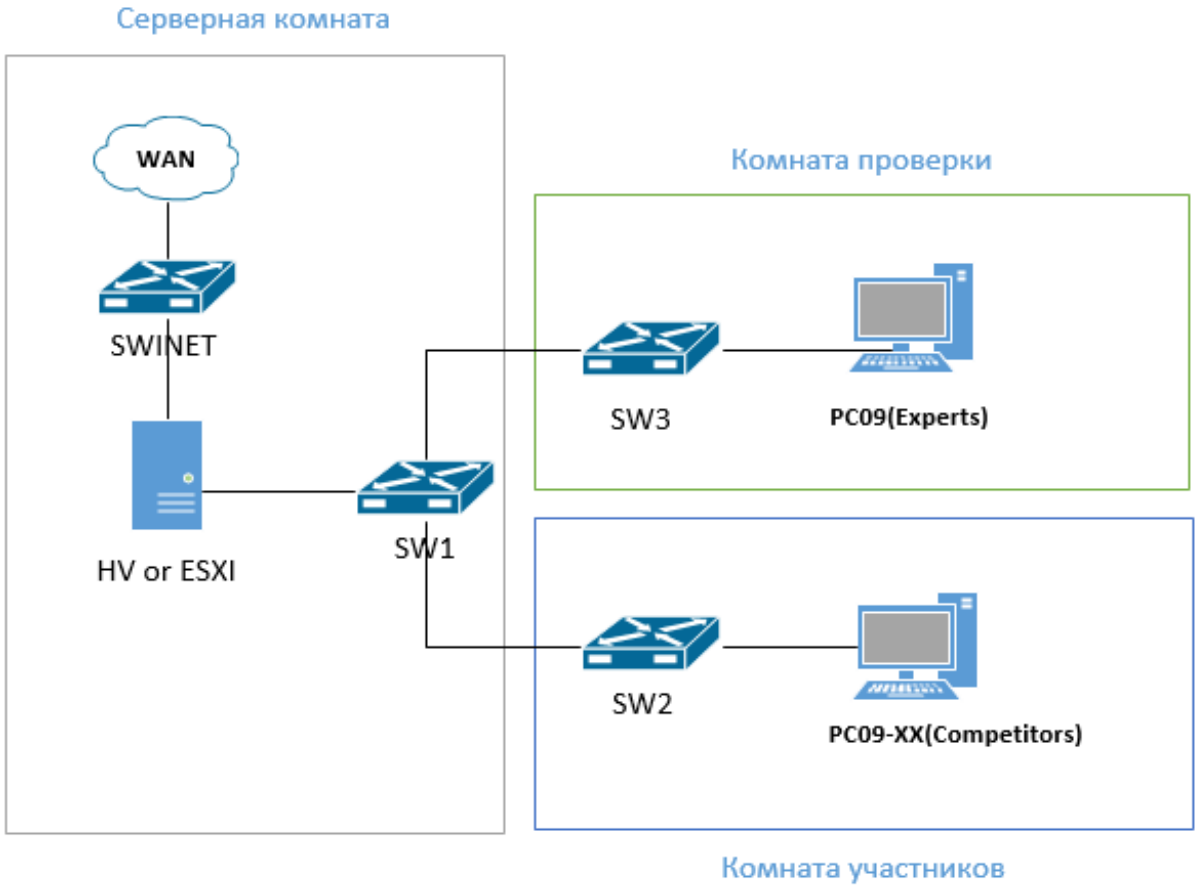
Задание:

1) Изучить шаблоны тестовой документации https://drive.google.com/drive/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-1kn;
2) Выполнить тестирование и оформить тестовую документацию в соответствие с заданием https://drive.google.com/drive/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-1kn.

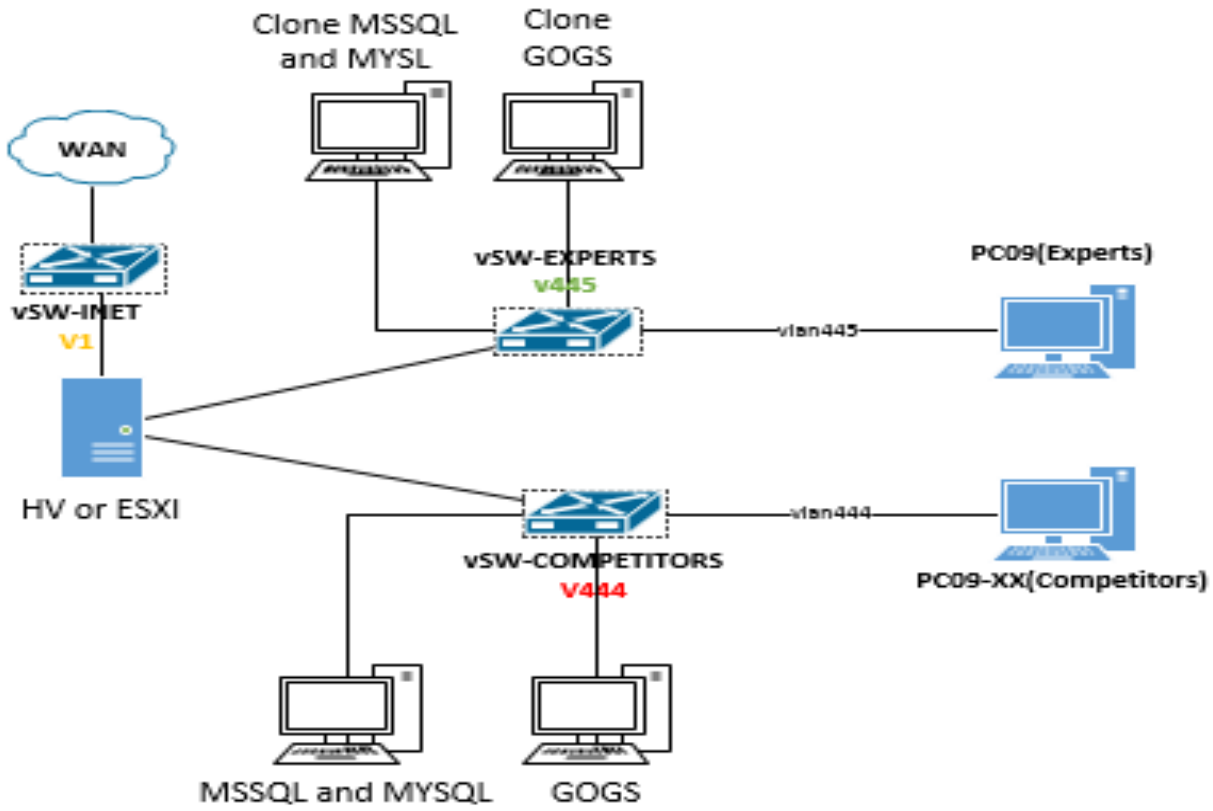
ОРГАНИЗАЦИЯ И ПРОВЕДЕНИЕ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА ПО СТАНДАРТАМ ВОРЛДСКИЛЛС РОССИЯ. ОЦЕНКА КВАЛИФИКАЦИИ СТУДЕНТА (ВЫПУСКНИКА) В ХОДЕ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА. ЗАСТРОЙКА ПЛОЩАДКИ ПРОВЕДЕНИЯ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА

Важную роль в проведении демонстрационного экзамена играет застройка площадки проведения ДЭ.
Порядок застройки площадки и настройка ПО разработаны Оковалковым В.
Общая топология площадки

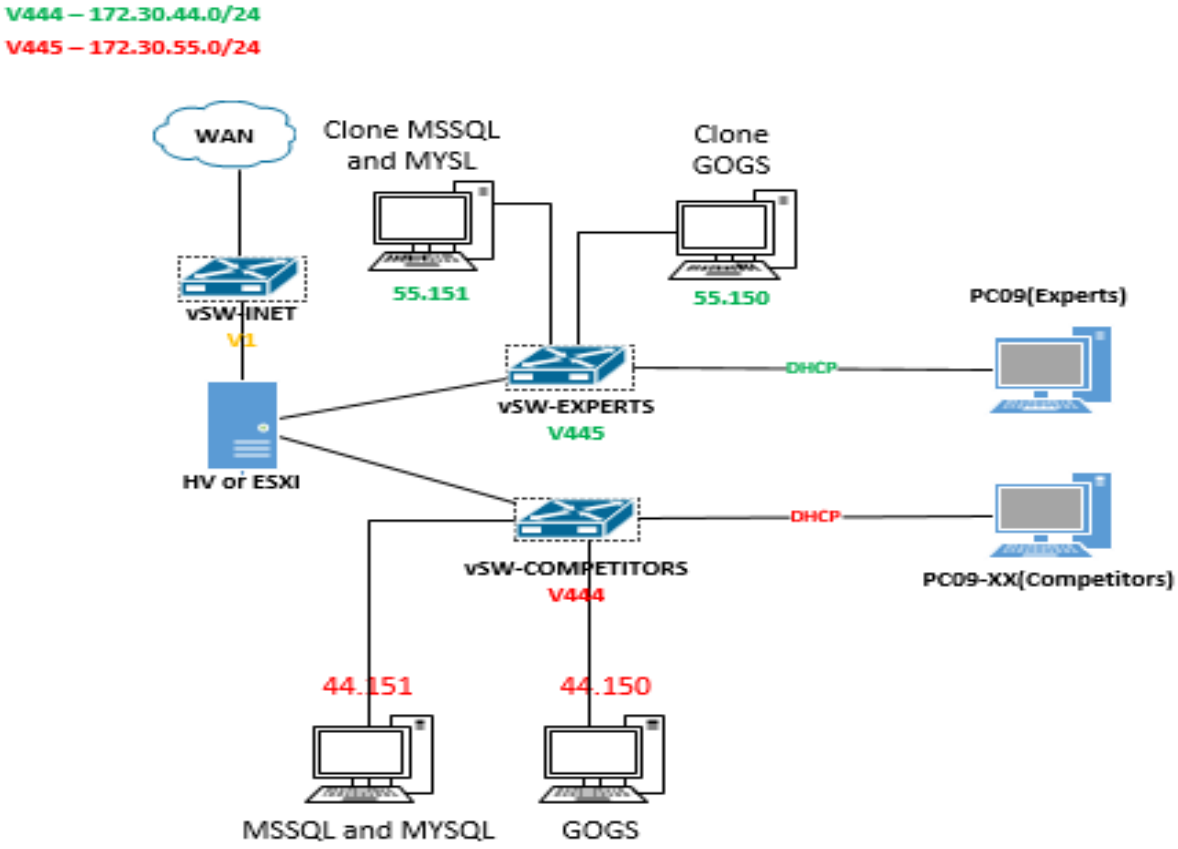
L1



L2

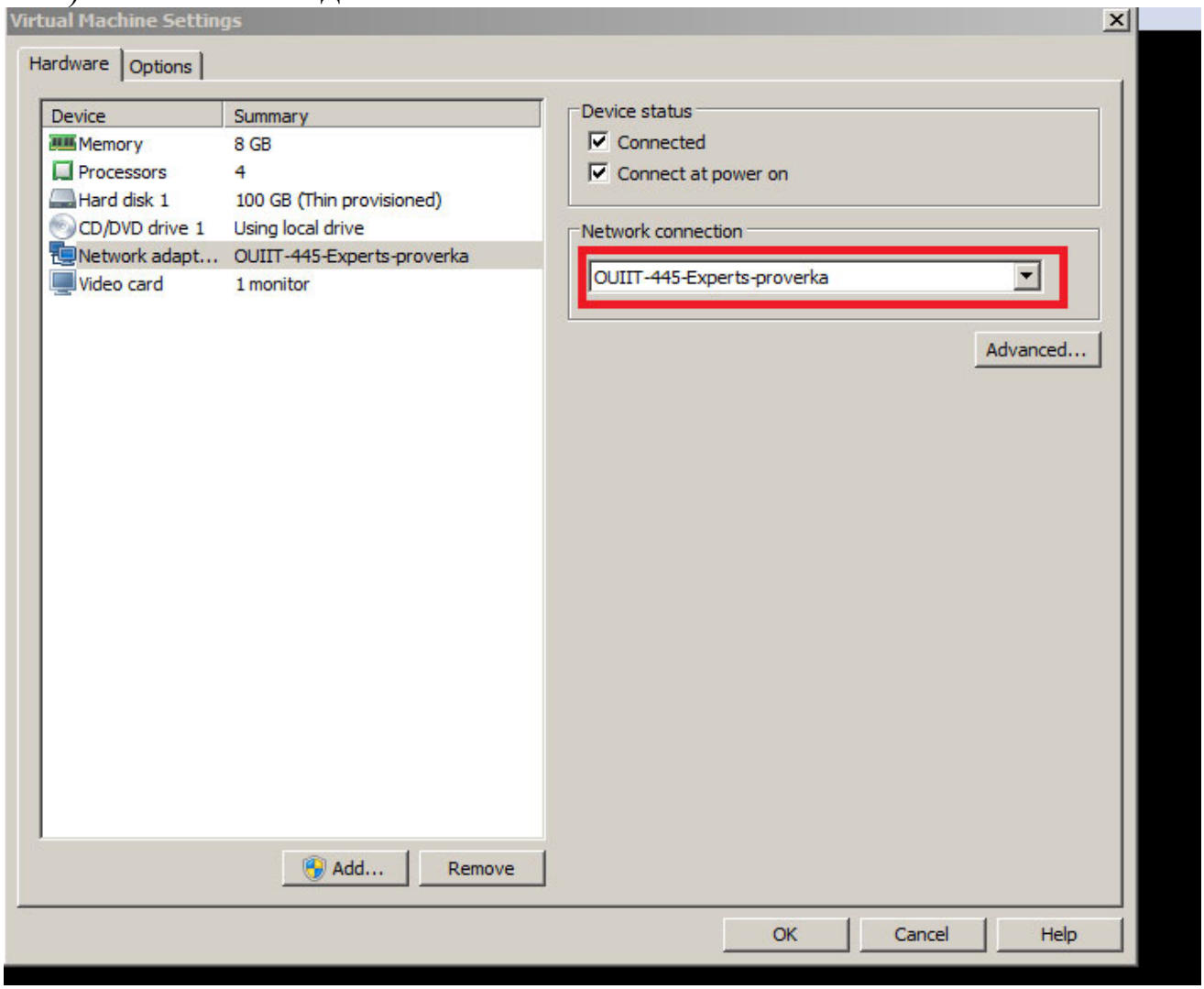

Clone MSSQL and MYSL
Clone GOGS
WAN
vSW-EXPERTS
V445
PC09(Experts)
vSW-INET
V1
vlan445
HV or ESXI
vlan444
vSW-COMPETITORS
V444
PC09-XX(Competitors)
MSSQL and MYSQL
GOGS

L3

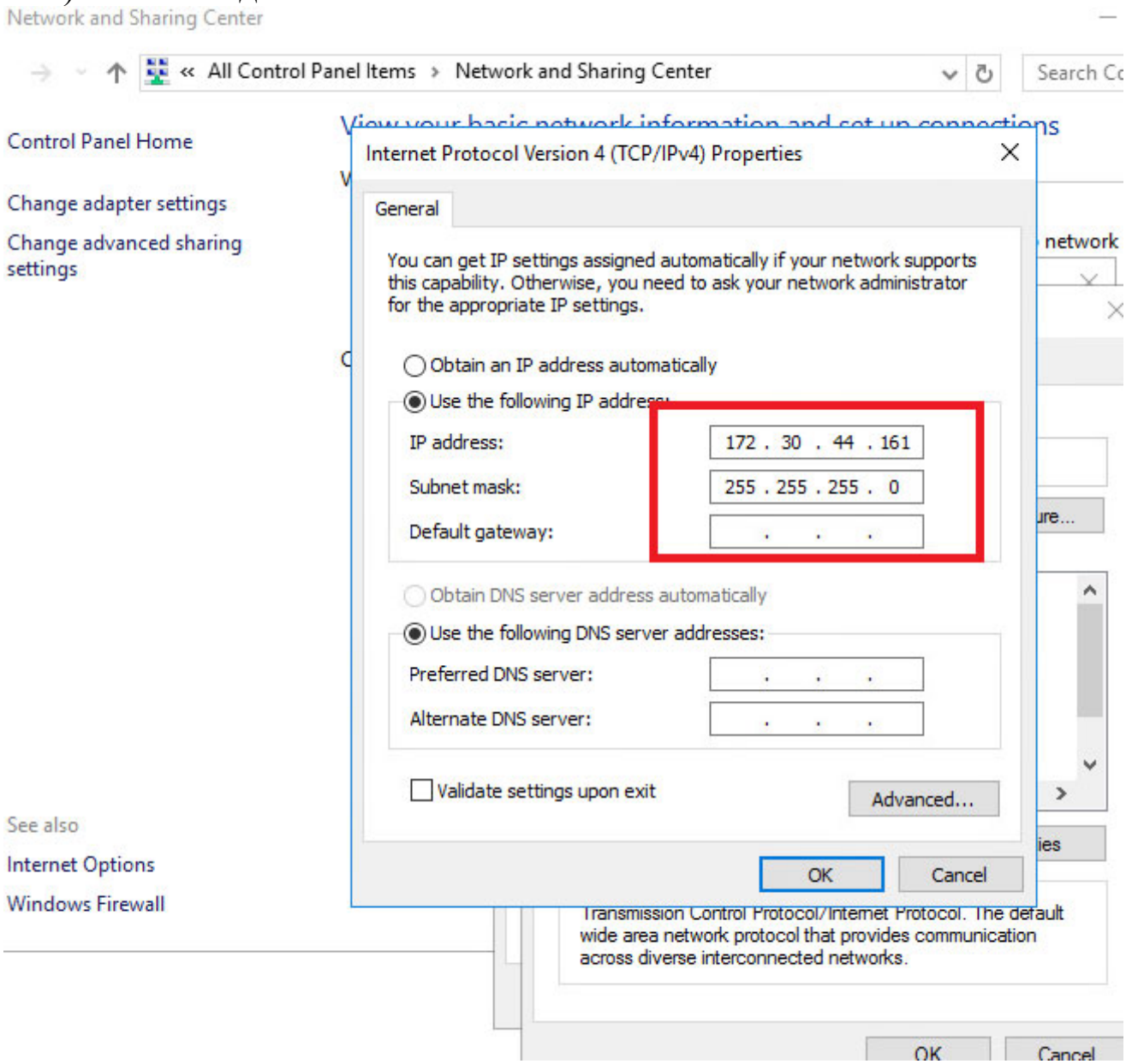

V444 – 172.30.44.0/24
V445 – 172.30.55.0/24
WAN
Clone MSSQL and MYSL
Clone GOGS
55.151
55.150
PC09(Experts)
vSW-INET
V1
DHCP
vSW-EXPERTS
V445
HV or ESXI
DHCP
vSW-COMPETITORS
V444
PC09-XX(Competitors)
44.151
44.150
MSSQL and MYSQL
GOGS

Примечание
В случаи необходимости сохранения ip адреса на клонированных виртуальных машинах необходимо что бы

1) Машина находилась в 445 vlan

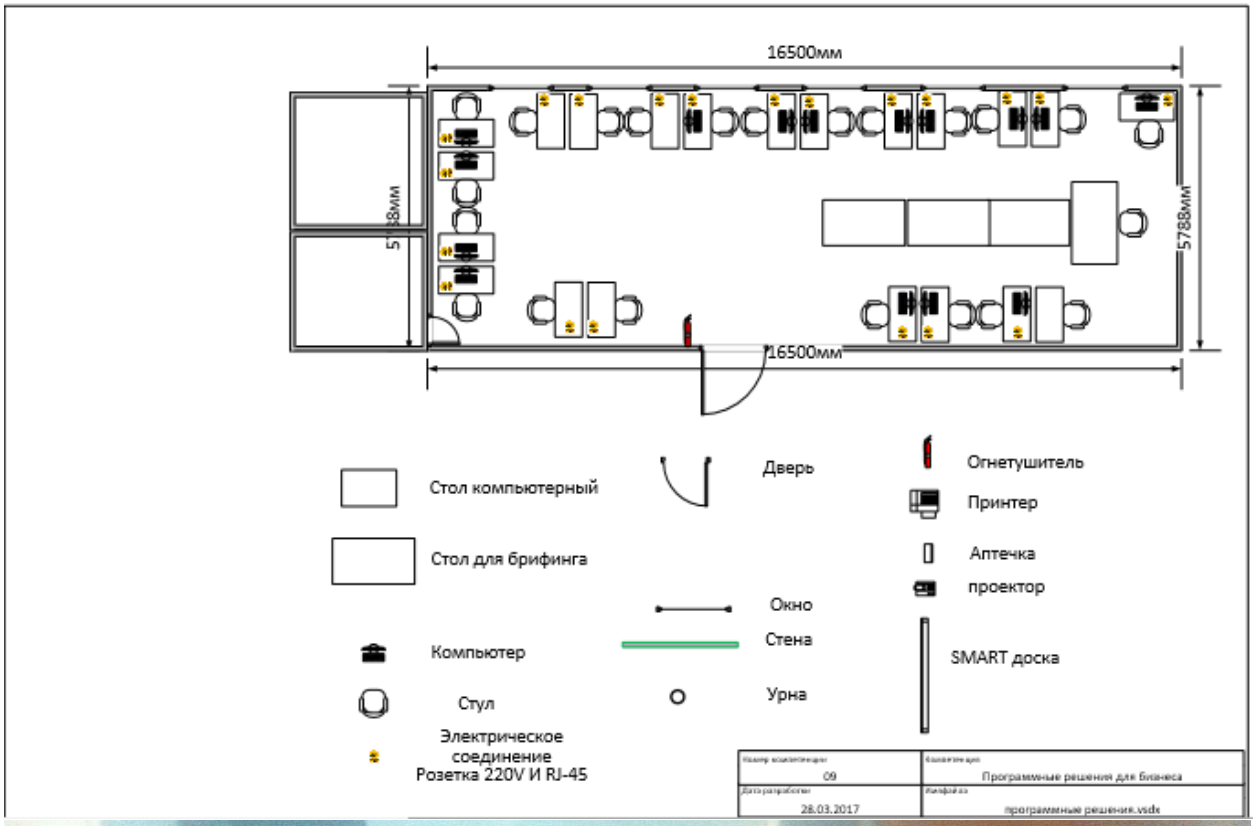


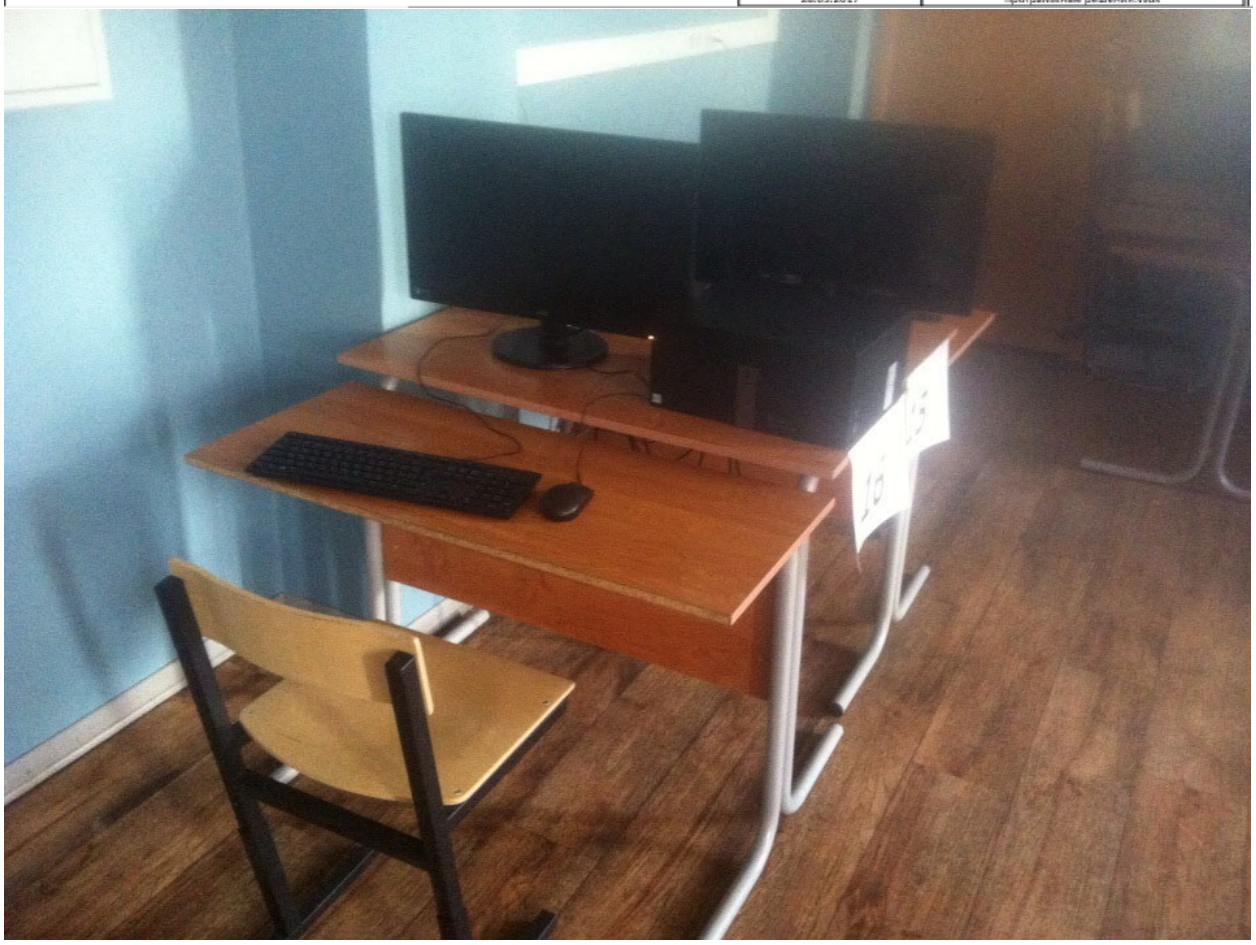
2) Не был задан шлюз



При данной конфигурации вы сохраните адресацию клонированной виртуальной машины для проверки, если это будет необходимо.

План застройки

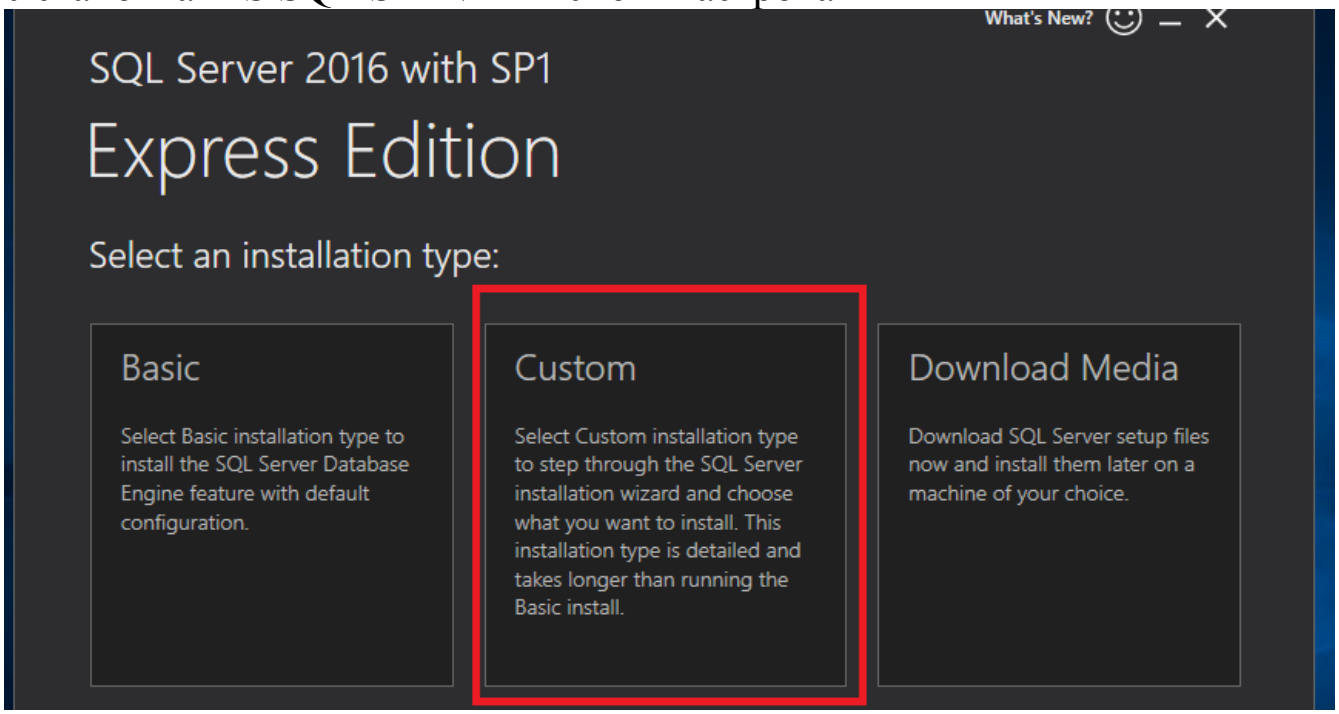


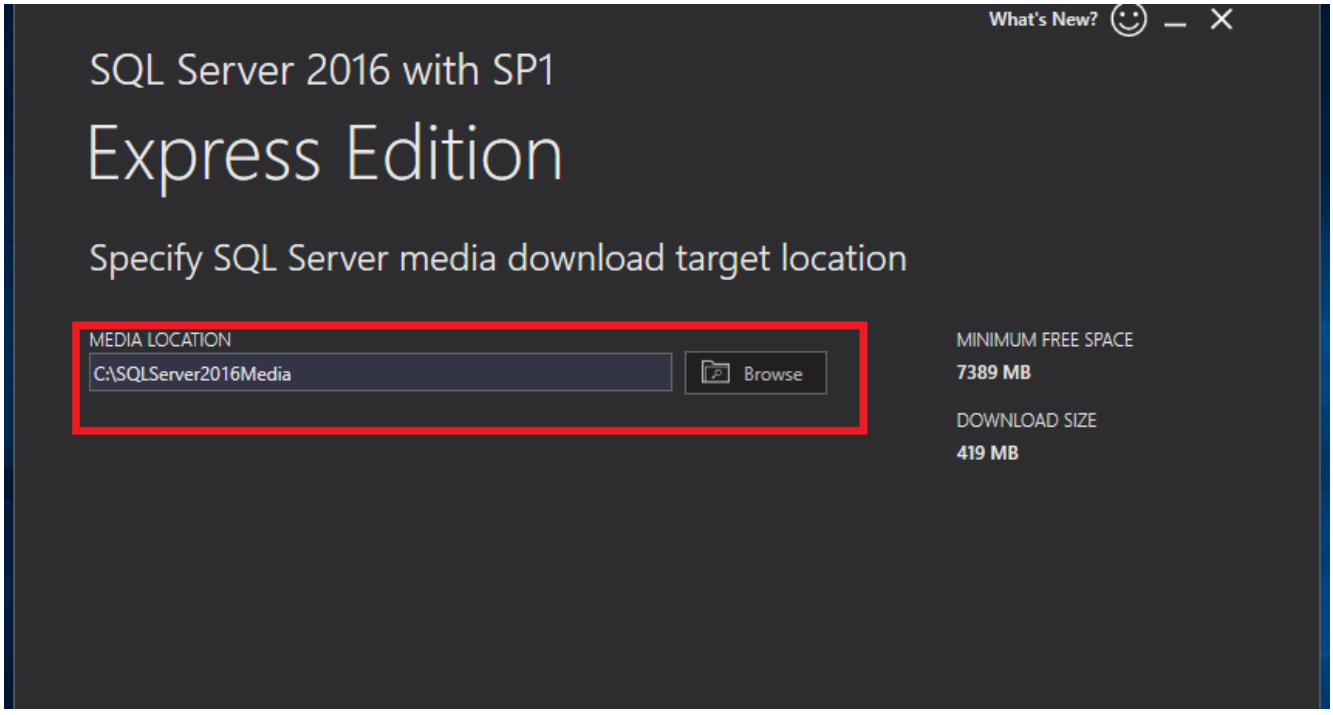


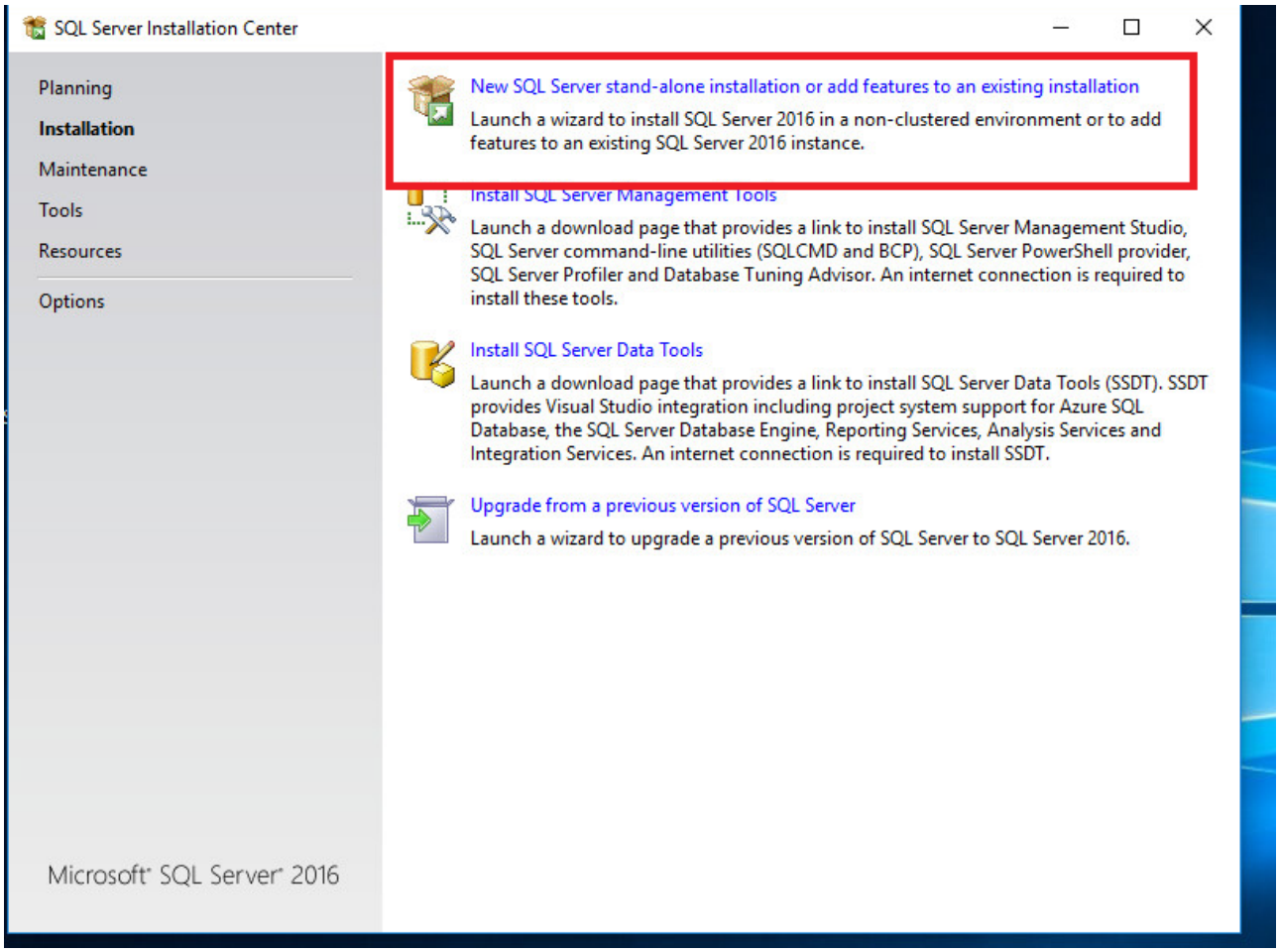

16
15

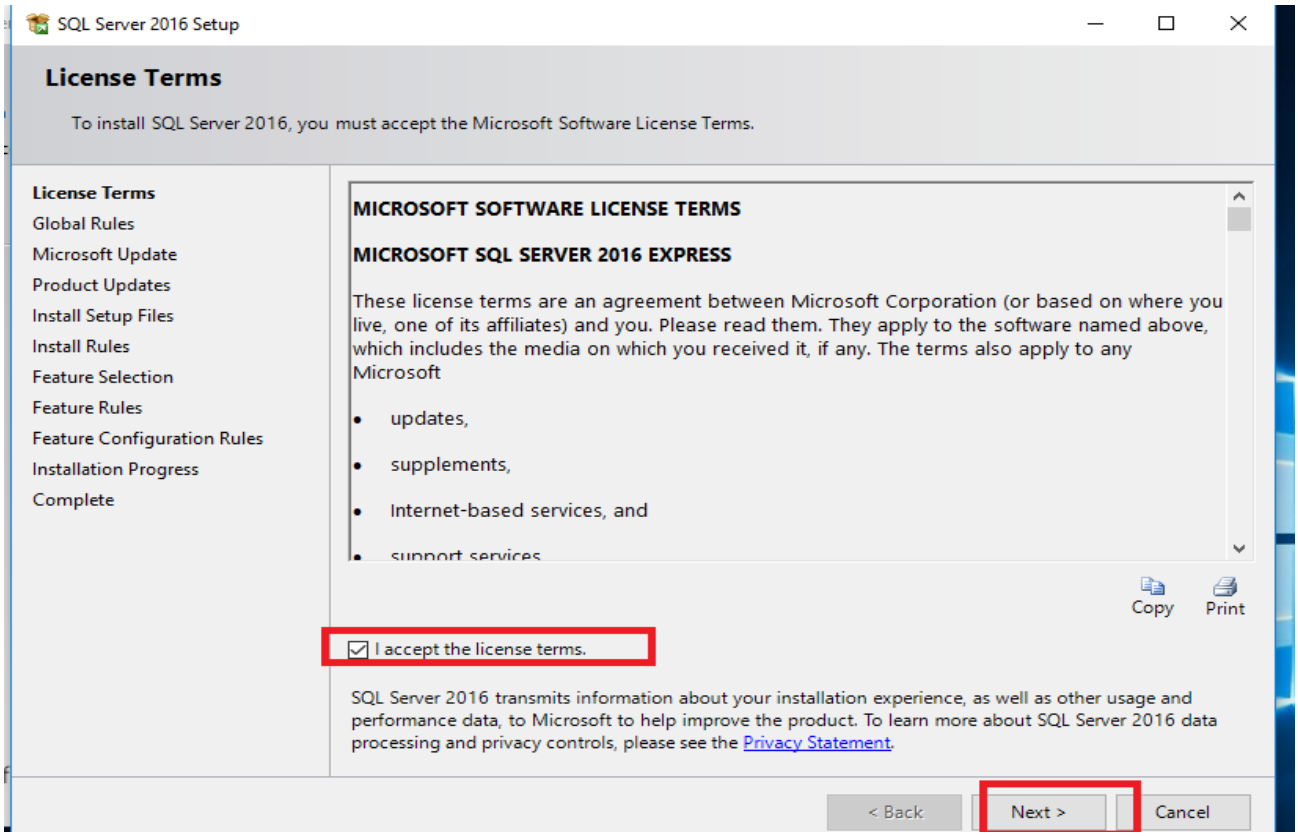

16
15

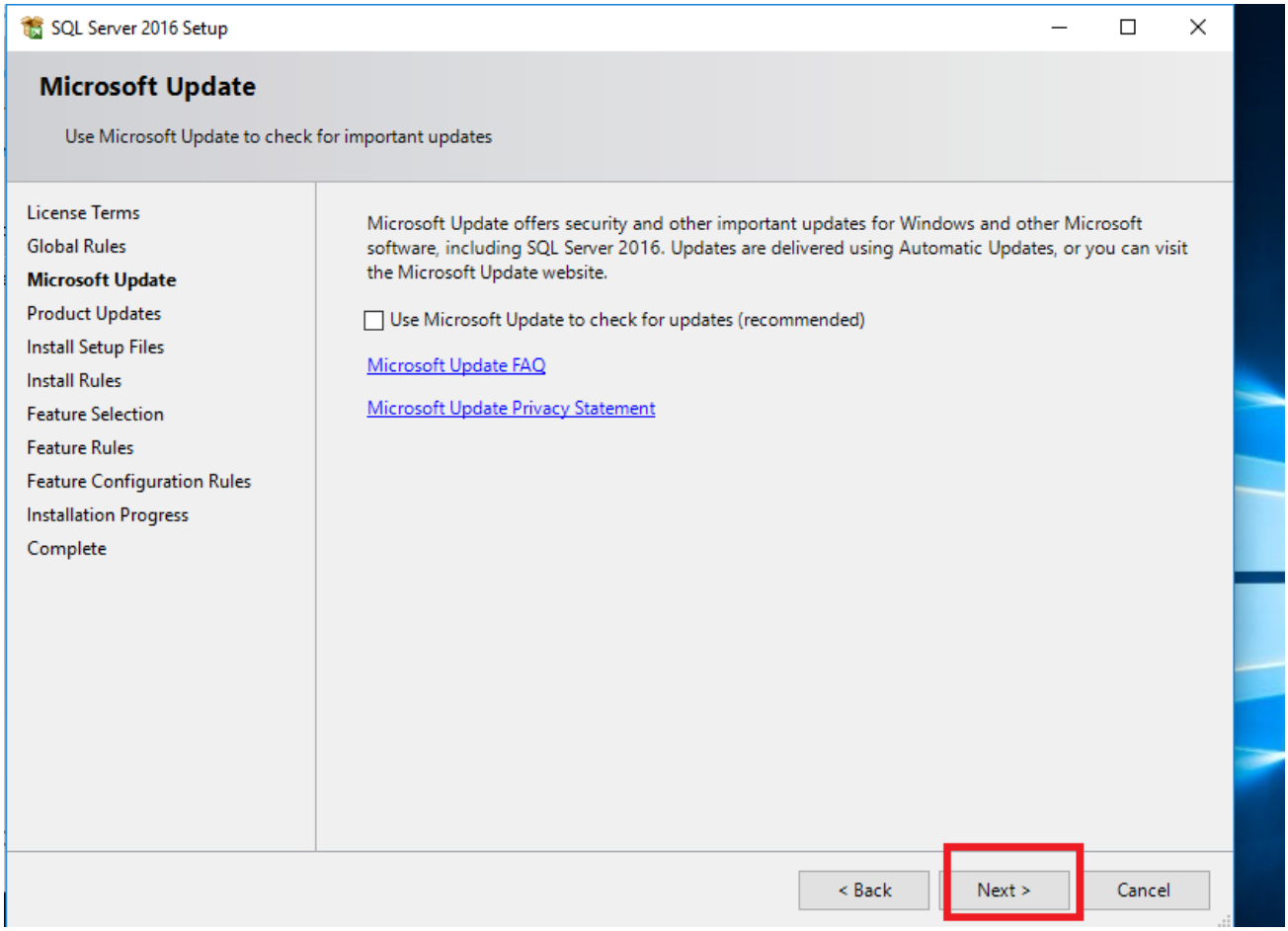
Установка MS SQL SERVER 2016 и настрока

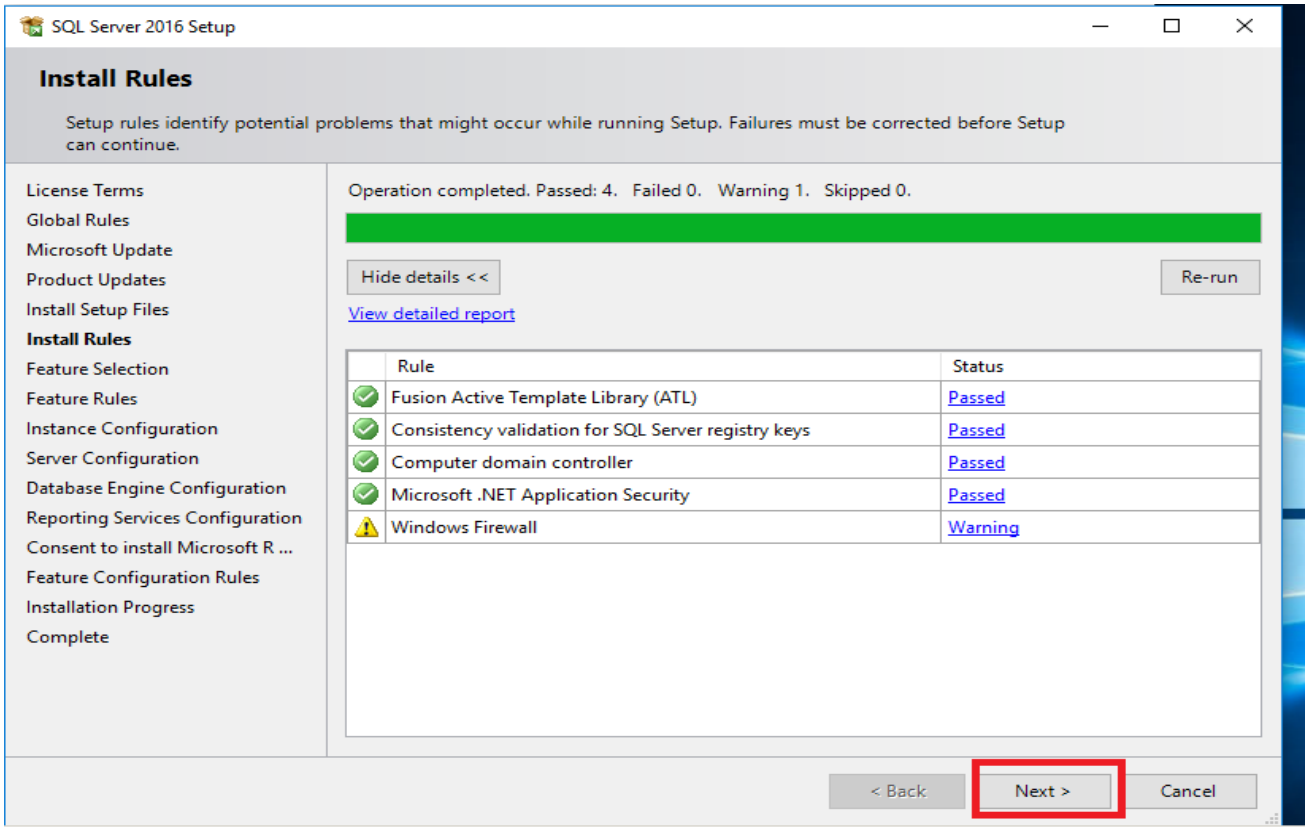


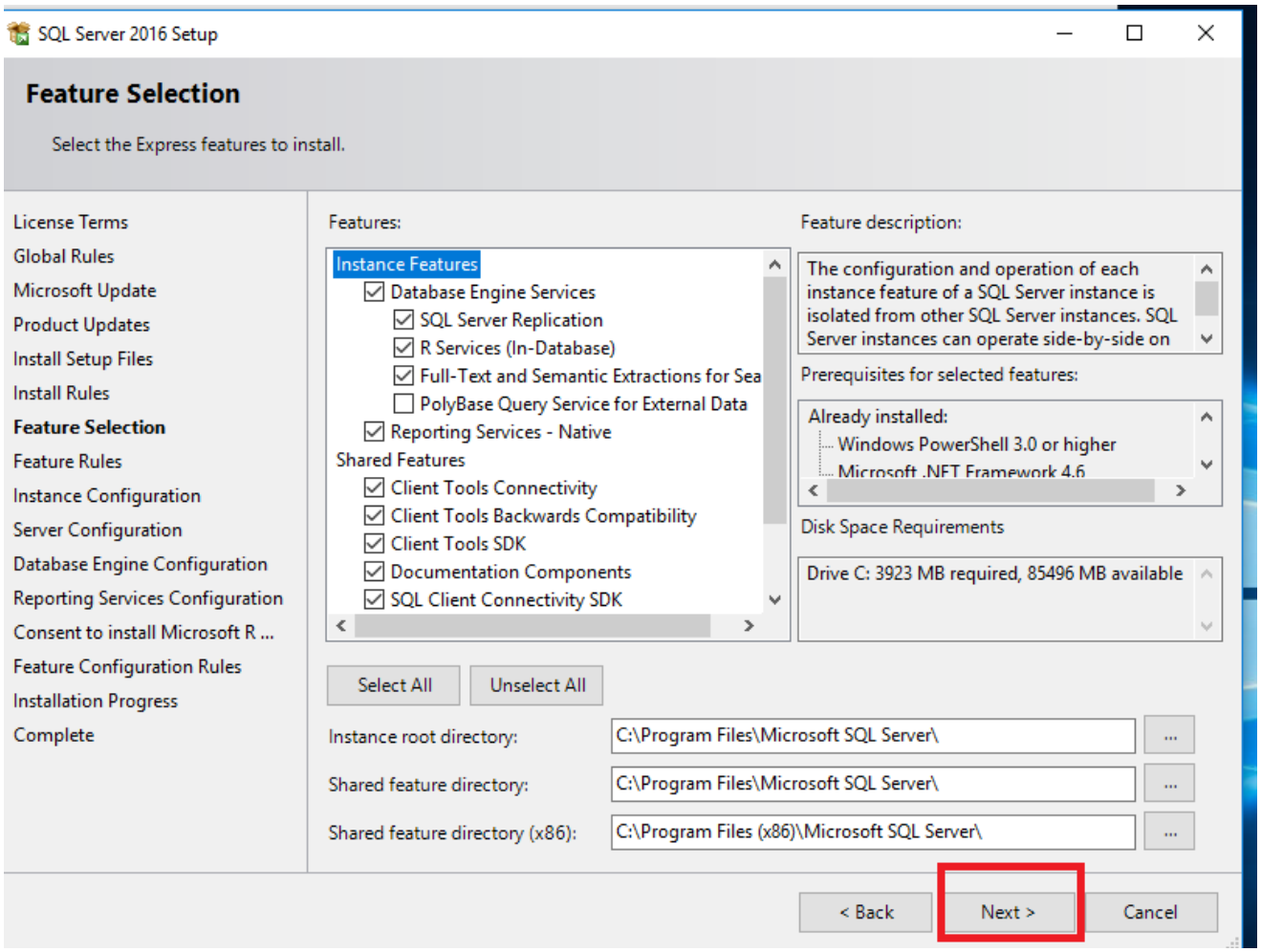


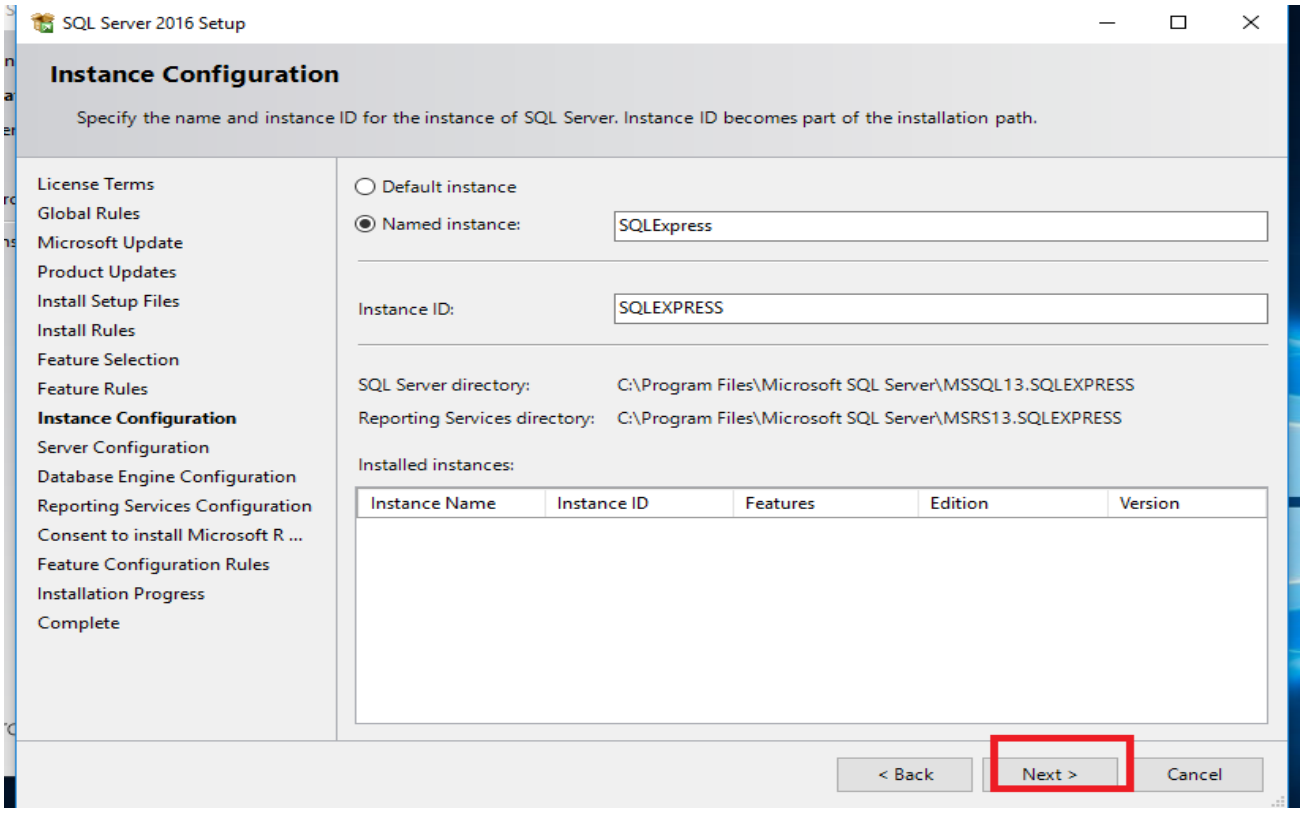

SQL Server Installation Center
Planning
Installation
Maintenance
Tools
Resources
Options
New SQL Server stand-alone installation or add features to an existing installation
Launch a wizard to install SQL Server 2016 in a non-clustered environment or to add features to an existing SQL Server 2016 instance.
Install SQL Server Management Tools
Launch a download page that provides a link to install SQL Server Management Studio, SQL Server command-line utilities (SQLCMD and BCP), SQL Server PowerShell provider, SQL Server Profiler and Database Tuning Advisor. An internet connection is required to install these tools.
Install SQL Server Data Tools
Launch a download page that provides a link to install SQL Server Data Tools (SSDT). SSDT provides Visual Studio integration including project system support for Azure SQL Database, the SQL Server Database Engine, Reporting Services, Analysis Services and Integration Services. An internet connection is required to install SSDT.
Upgrade from a previous version of SQL Server
Launch a wizard to upgrade a previous version of SQL Server to SQL Server 2016.
Microsoft SQL Server 2016

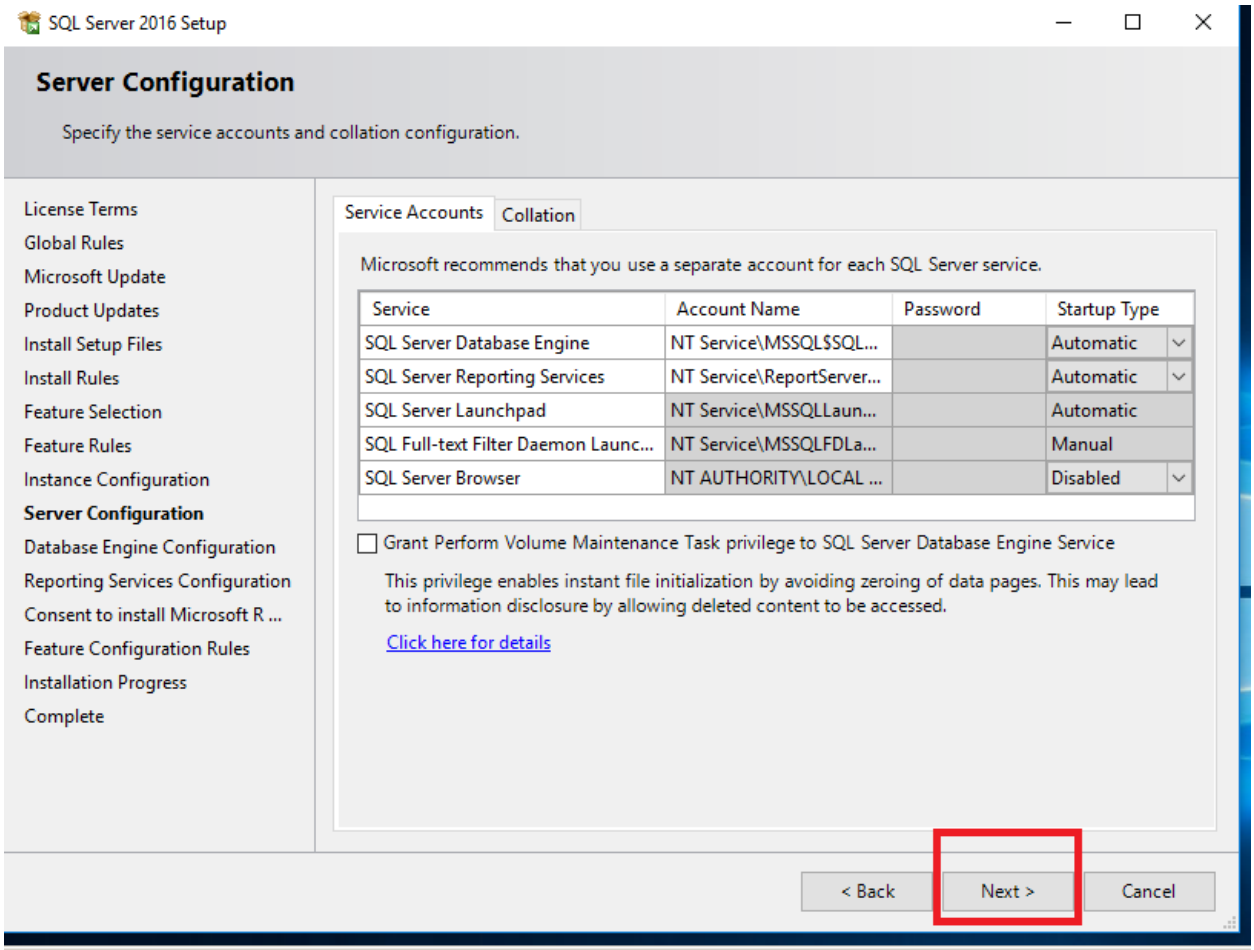

SQL Server 2016 Setup
License Terms
To install SQL Server 2016, you must accept the Microsoft Software License Terms.
License Terms
Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Feature Configuration Rules
Installation Progress
Complete
MICROSOFT SOFTWARE LICENSE TERMS
MICROSOFT SQL SERVER 2016 EXPRESS
These license terms are an agreement between Microsoft Corporation (or based on where you live, one of its affiliates) and you. Please read them. They apply to the software named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft
updates,
supplements,
Internet-based services, and
Copy
Print
I accept the license terms.
SQL Server 2016 transmits information about your installation experience, as well as other usage and performance data, to Microsoft to help improve the product. To learn more about SQL Server 2016 data processing and privacy controls, please see the Privacy Statement.
< Back
Next >
Cancel


SQL Server 2016 Setup
Microsoft Update
Use Microsoft Update to check for important updates
License Terms
Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Feature Configuration Rules
Installation Progress
Complete
Microsoft Update offers security and other important updates for Windows and other Microsoft software, including SQL Server 2016. Updates are delivered using Automatic Updates, or you can visit the Microsoft Update website.
Use Microsoft Update to check for updates (recommended)
Microsoft Update FAQ
Microsoft Update Privacy Statement
< Back
Next >
Cancel

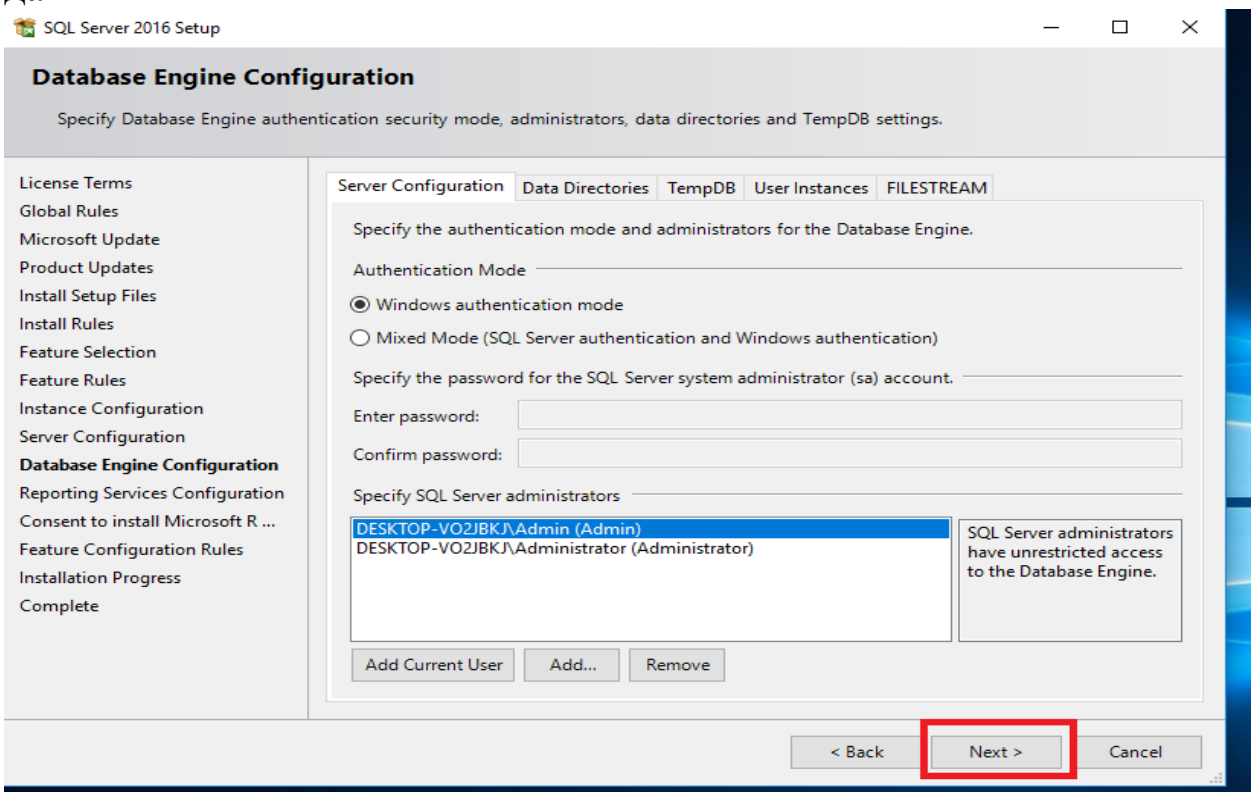

SQL Server 2016 Setup
Install Rules
Setup rules identify potential problems that might occur while running Setup. Failures must be corrected before Setup can continue.
License Terms
Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Reporting Services Configuration
Consent to install Microsoft R ...
Feature Configuration Rules
Installation Progress
Complete
Operation completed. Passed: 4. Failed 0. Warning 1. Skipped 0.
Hide details <<
Re-run
View detailed report
Rule
Status
Fusion Active Template Library (ATL)
Passed
Consistency validation for SQL Server registry keys
Passed
Computer domain controller
Passed
Microsoft .NET Application Security
Passed
Windows Firewall
Warning
< Back
Next >
Cancel

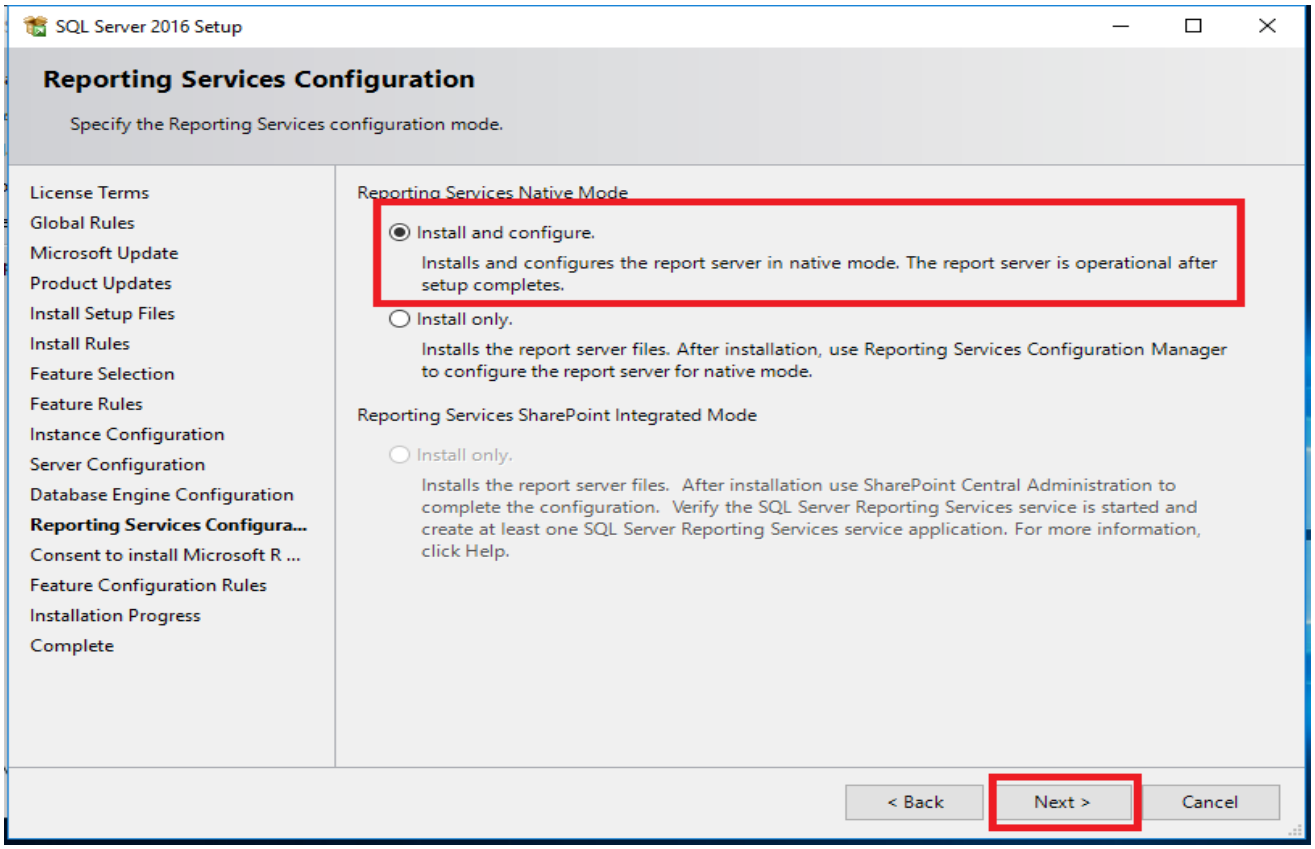

SQL Server 2016 Setup
Feature Selection
Select the Express features to install.
License Terms
Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Reporting Services Configuration
Consent to install Microsoft R ...
Feature Configuration Rules
Installation Progress
Complete
Features:
Instance Features
Database Engine Services
SQL Server Replication
R Services (In-Database)
Full-Text and Semantic Extractions for Sea
PolyBase Query Service for External Data
Reporting Services - Native
Shared Features
Client Tools Connectivity
Client Tools Backwards Compatibility
Client Tools SDK
Documentation Components
SQL Client Connectivity SDK
Feature description:
The configuration and operation of each instance feature of a SQL Server instance is isolated from other SQL Server instances. SQL Server instances can operate side-by-side on
Prerequisites for selected features:
Already installed:
Windows PowerShell 3.0 or higher
Microsoft .NET Framework 4.6
Disk Space Requirements
Drive C: 3923 MB required, 85496 MB available
Select All
Unselect All
Instance root directory:
C:\Program Files\Microsoft SQL Server\
Shared feature directory:
C:\Program Files\Microsoft SQL Server\
Shared feature directory (x86):
C:\Program Files (x86)\Microsoft SQL Server\
< Back
Next >
Cancel

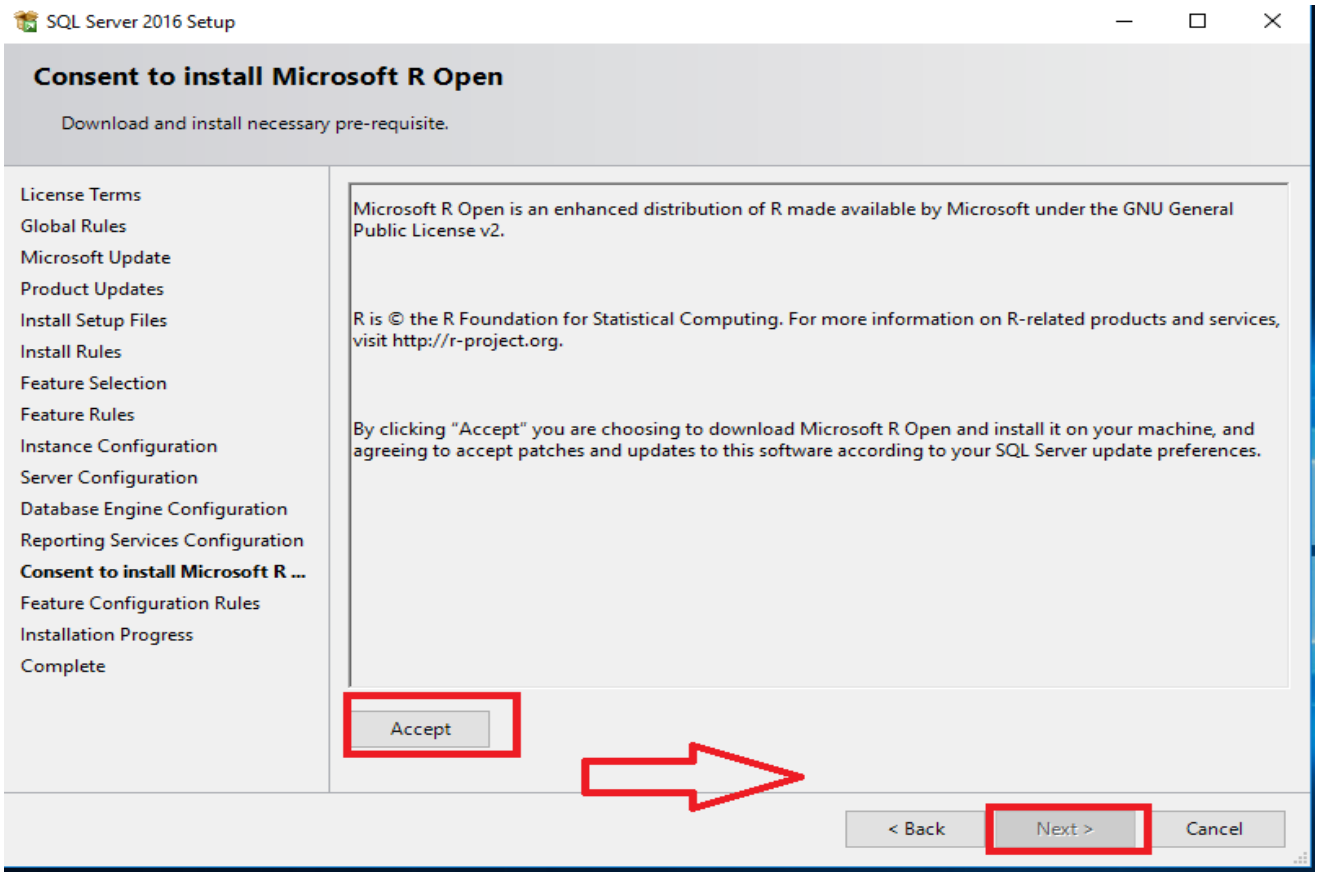

SQL Server 2016 Setup
Instance Configuration
Specify the name and instance ID for the instance of SQL Server. Instance ID becomes part of the installation path.
License Terms
Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Reporting Services Configuration
Consent to install Microsoft R ...
Feature Configuration Rules
Installation Progress
Complete
Default instance
Named instance:
SQLExpress
Instance ID:
SQLEXPRESS
SQL Server directory:
C:\Program Files\Microsoft SQL Server\MSSQL13.SQLEXPRESS
Reporting Services directory:
C:\Program Files\Microsoft SQL Server\MSRS13.SQLEXPRESS
Installed instances:
Instance Name
Instance ID
Features
Edition
Version
< Back
Next >
Cancel

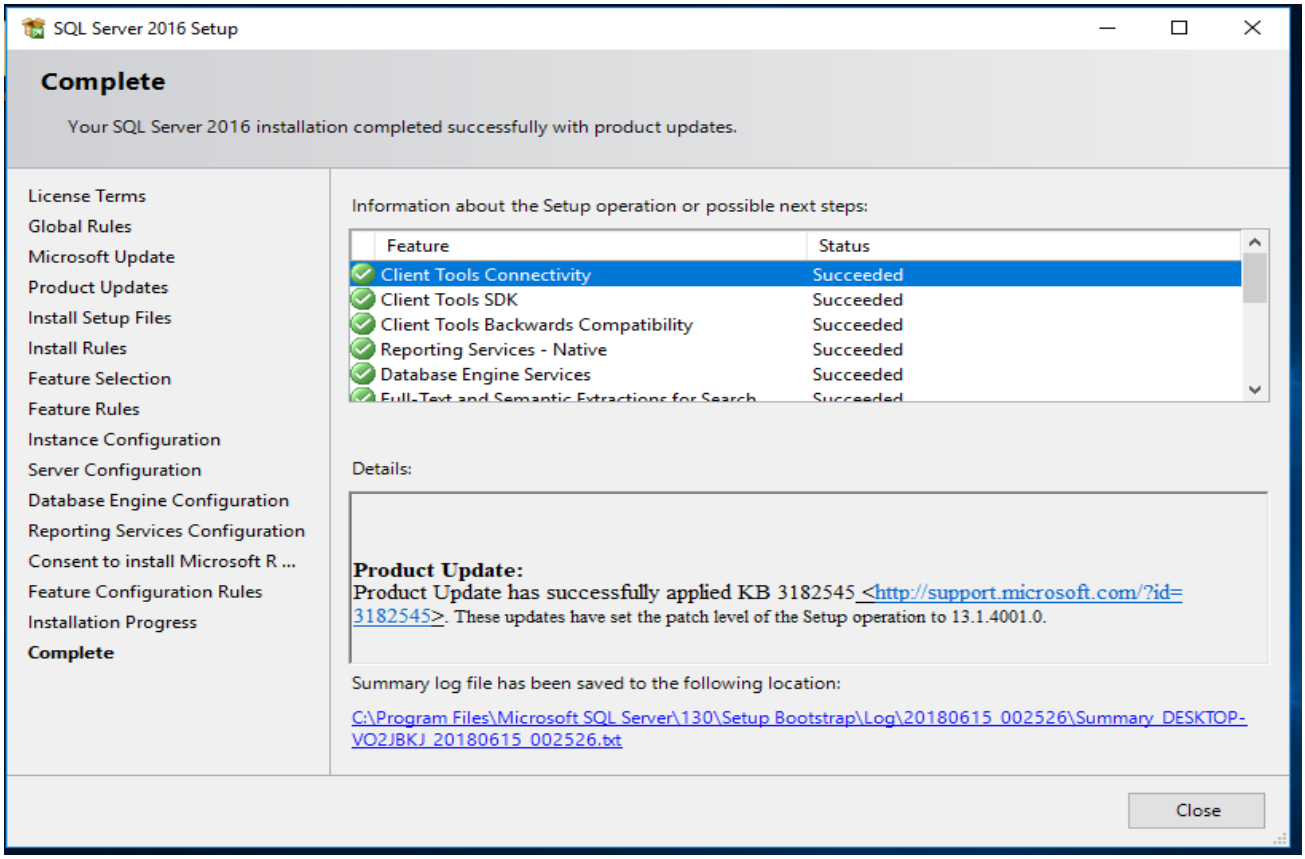


При необходимости можно добавить еще одного администратора базы данных

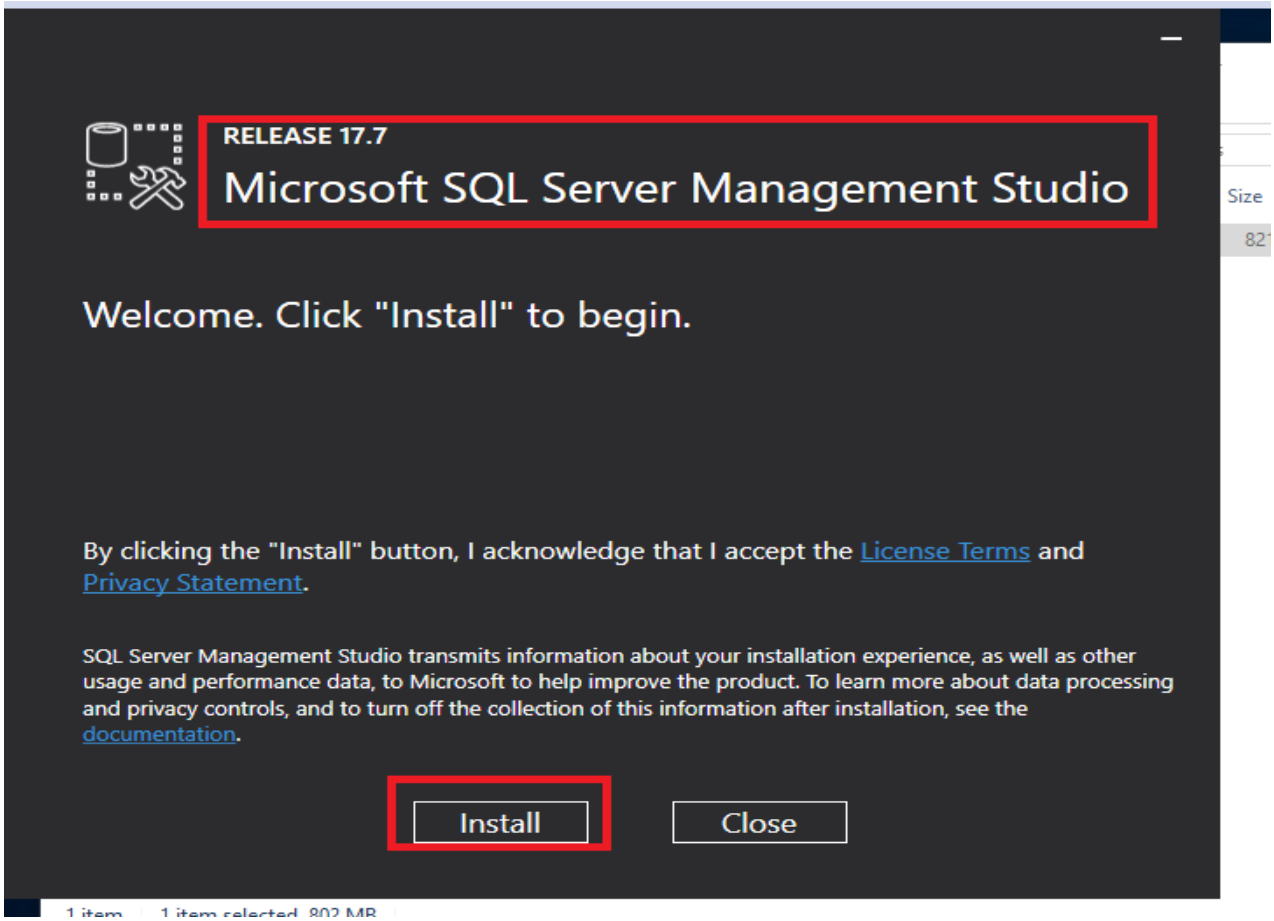



SQL Server 2016 Setup
Reporting Services Configuration
Specify the Reporting Services configuration mode.
License Terms
Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Reporting Services Configura...
Consent to install Microsoft R ...
Feature Configuration Rules
Installation Progress
Complete
Reporting Services Native Mode
Install and configure.
Installs and configures the report server in native mode. The report server is operational after setup completes.
Install only.
Installs the report server files. After installation, use Reporting Services Configuration Manager to configure the report server for native mode.
Reporting Services SharePoint Integrated Mode
Install only.
Installs the report server files. After installation use SharePoint Central Administration to complete the configuration. Verify the SQL Server Reporting Services service is started and create at least one SQL Server Reporting Services service application. For more information, click Help.
< Back
Next >
Cancel


SQL Server 2016 Setup
Consent to install Microsoft R Open
Download and install necessary pre-requisite.
License Terms
Global Rules
Microsoft Update
Product Updates
Install Setup Files
Install Rules
Feature Selection
Feature Rules
Instance Configuration
Server Configuration
Database Engine Configuration
Reporting Services Configuration
Consent to install Microsoft R ...
Feature Configuration Rules
Installation Progress
Complete
Microsoft R Open is an enhanced distribution of R made available by Microsoft under the GNU General Public License v2.
R is © the R Foundation for Statistical Computing. For more information on R-related products and services, visit http://r-project.org.
By clicking "Accept" you are choosing to download Microsoft R Open and install it on your machine, and agreeing to accept patches and updates to this software according to your SQL Server update preferences.
Accept
< Back
Next >
Cancel

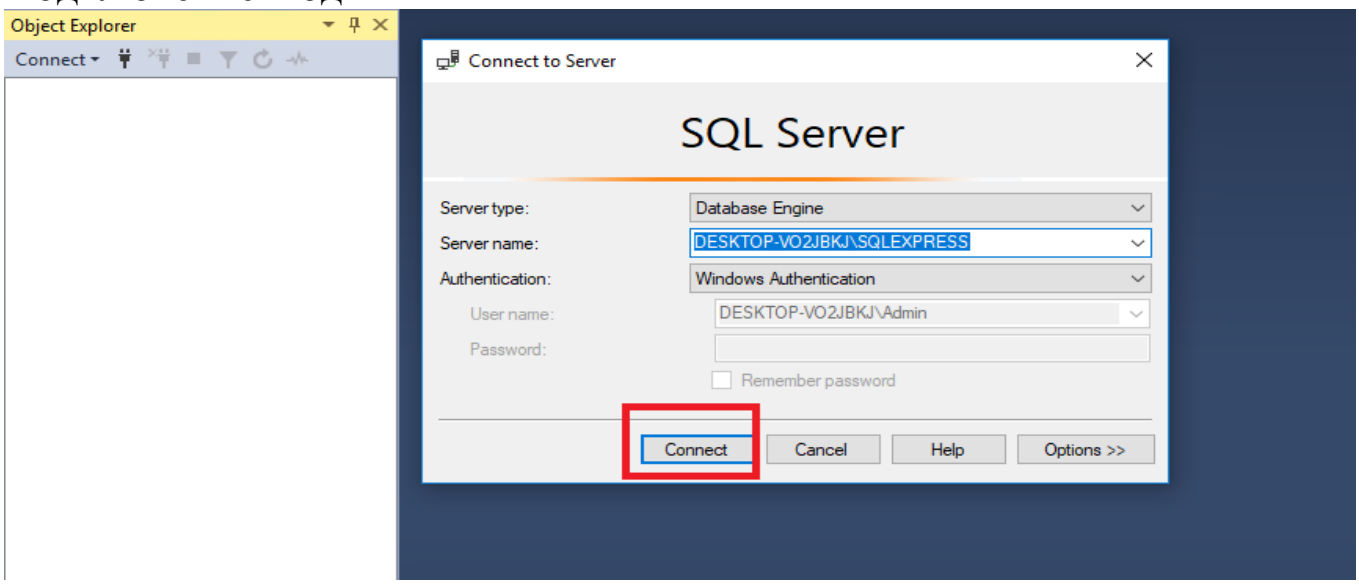


Установка SQL MANAGEMENT STUDIO

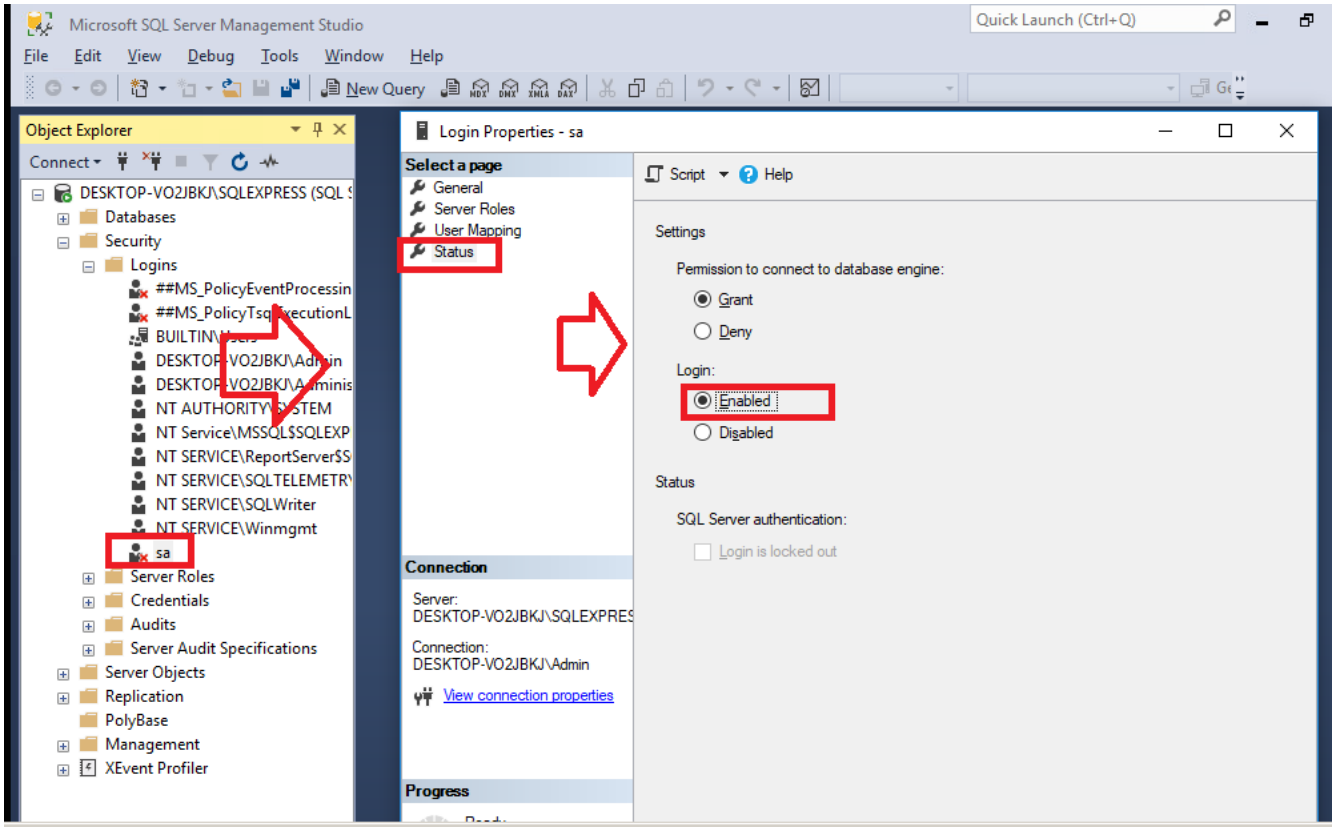


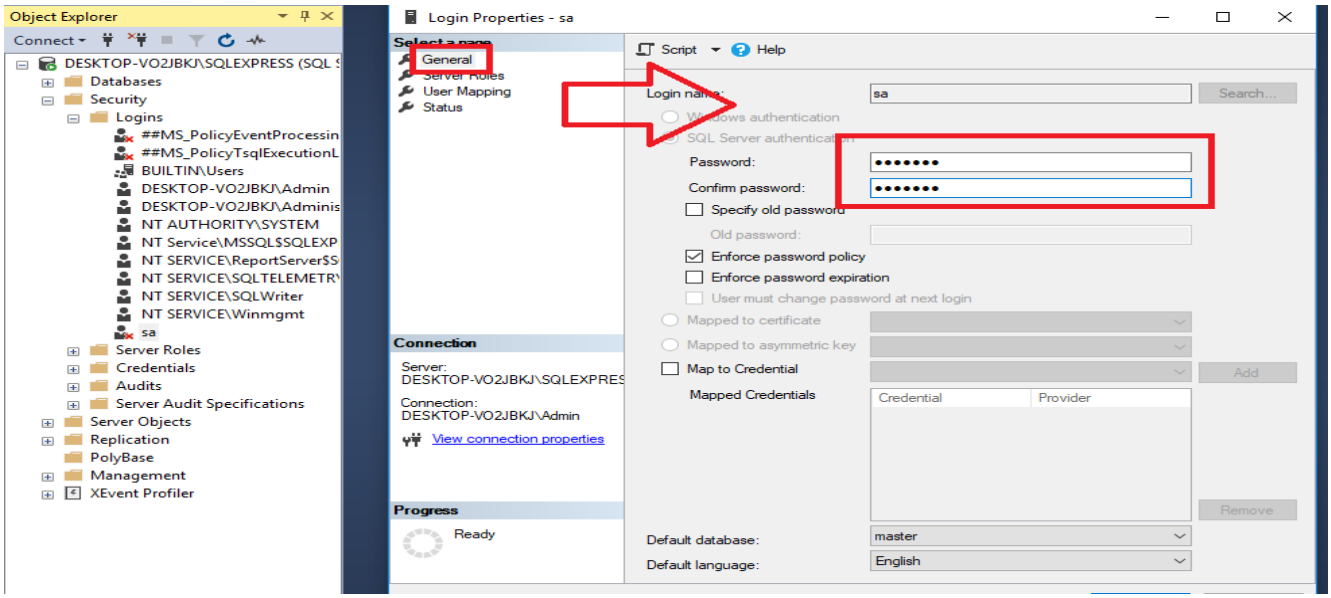
Настройка MSSQL

Подключение к бд

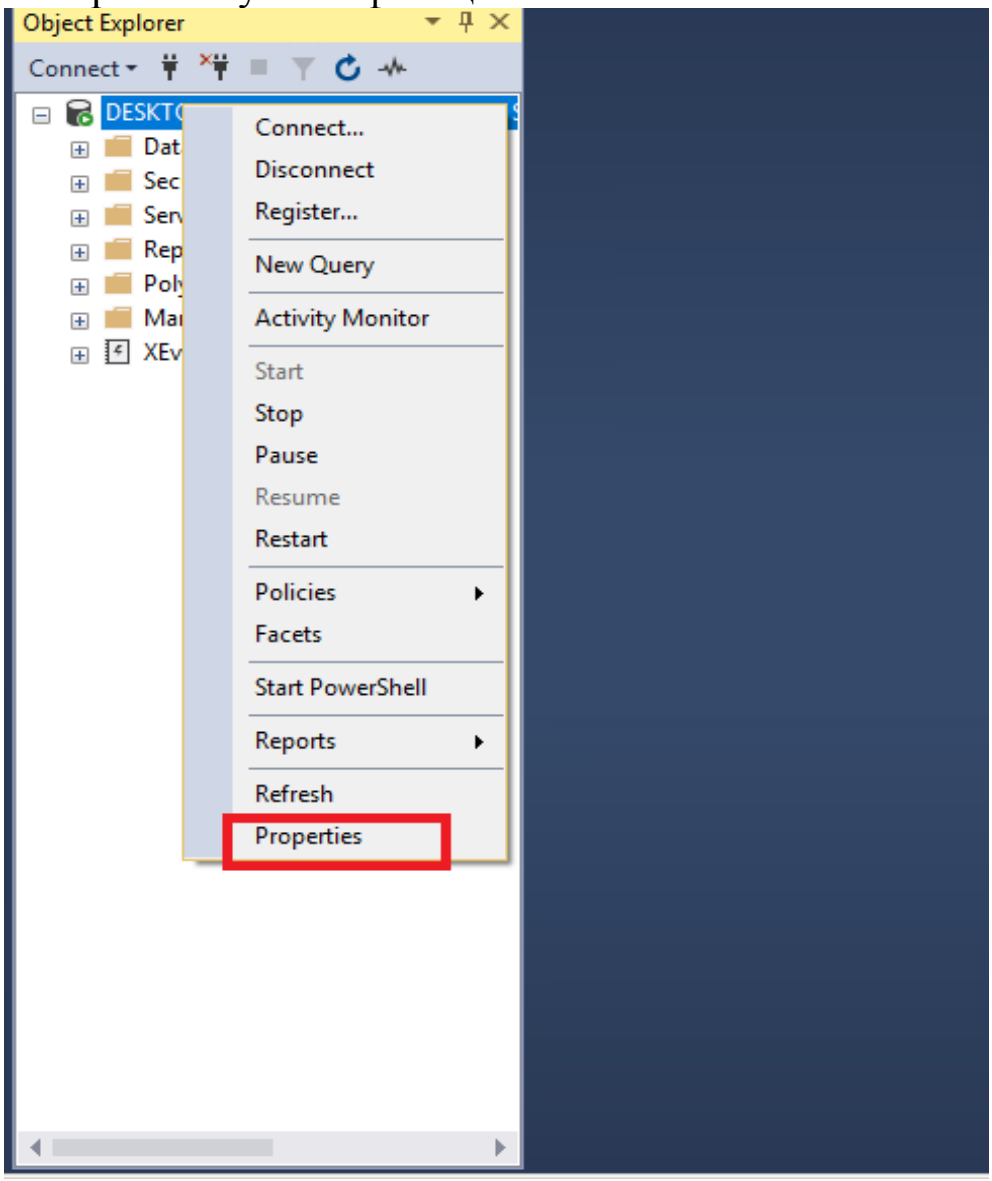


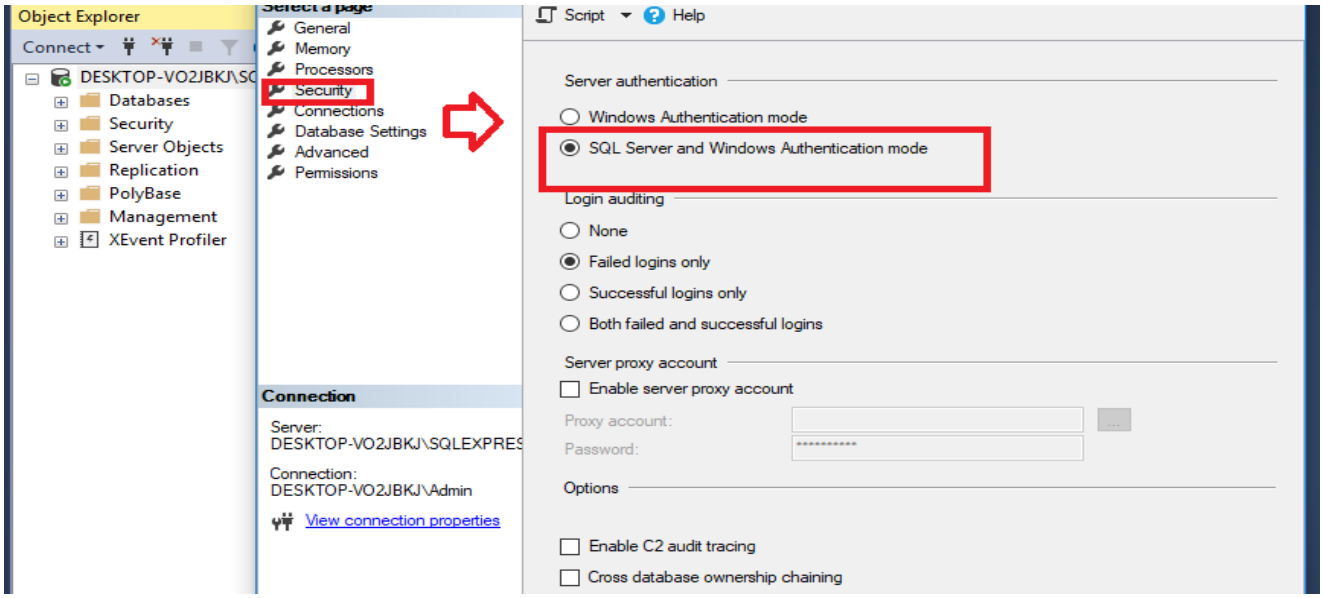
Настройка пользователя sa

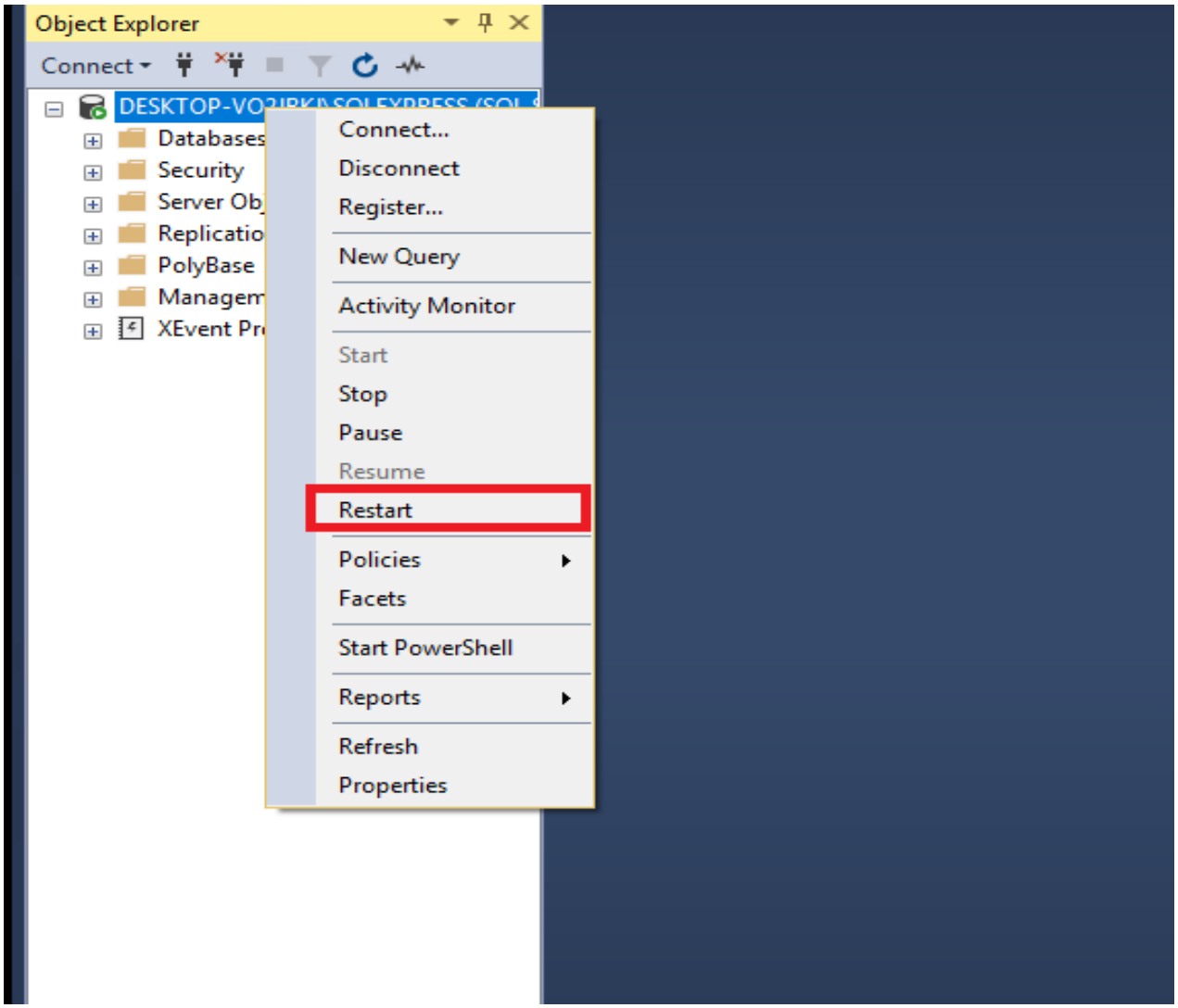




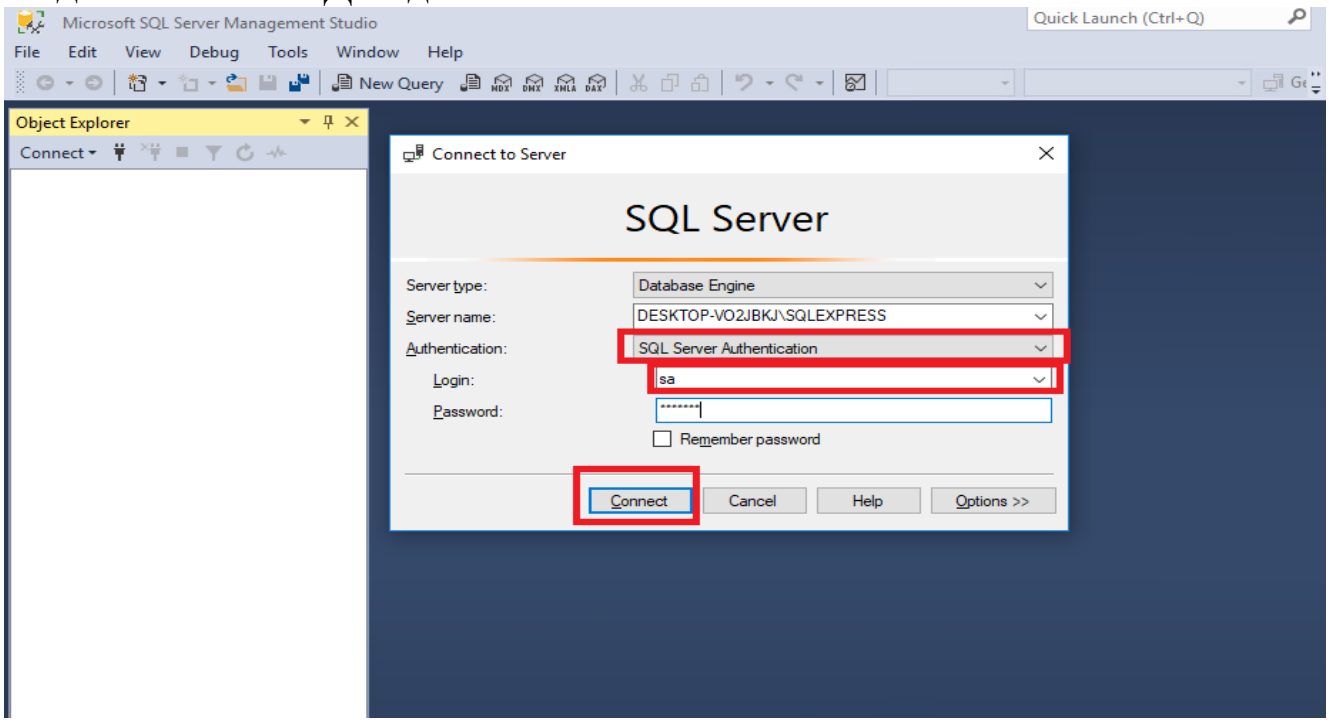
Настройка Аутентификации пользователей



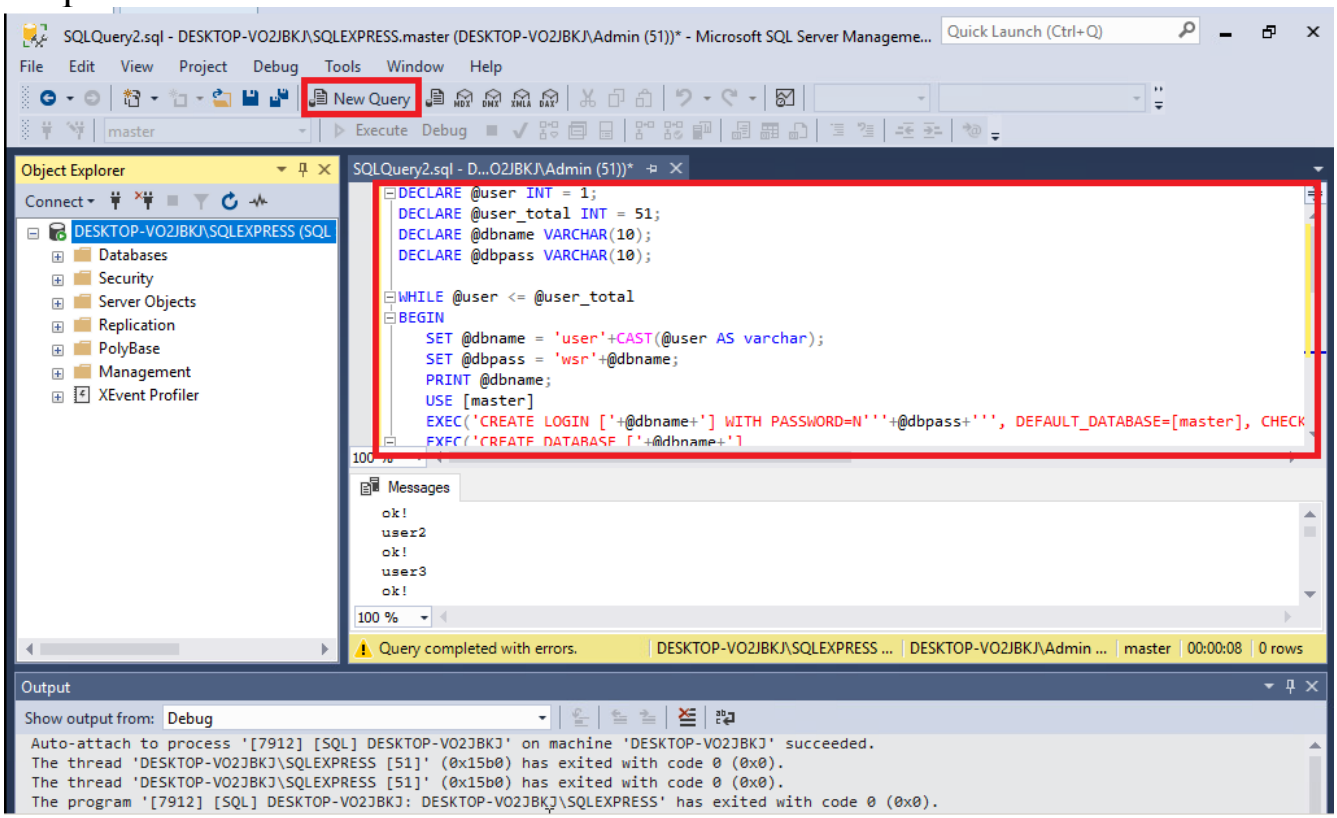

Object Explorer
Connect
DESKTOP-VO2JBKJ\SQ
Databases
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
General
Memory
Processors
Security
Connections
Database Settings
Advanced
Permissions
Script
Help
Server authentication
Windows Authentication mode
SQL Server and Windows Authentication mode
Login auditing
None
Failed logins only
Successful logins only
Both failed and successful logins
Server proxy account
Enable server proxy account
Proxy account:
Password:
Options
Enable C2 audit tracing
Cross database ownership chaining
Connection
Server:
DESKTOP-VO2JBKJ\SQLEXPRES
Connection:
DESKTOP-VO2JBKJ\Admin
View connection properties


Object Explorer
Connect
Databases
Security
Server Ob
Replicatio
PolyBase
Managem
XEvent Pr
Connect...
Disconnect
Register...
New Query
Activity Monitor
Start
Stop
Pause
Resume
Restart
Policies
Facets
Start PowerShell
Reports
Refresh
Properties

Подключение к БД под пользователем SA



Для автоматического создания пользователей и баз данных создаем запрос:

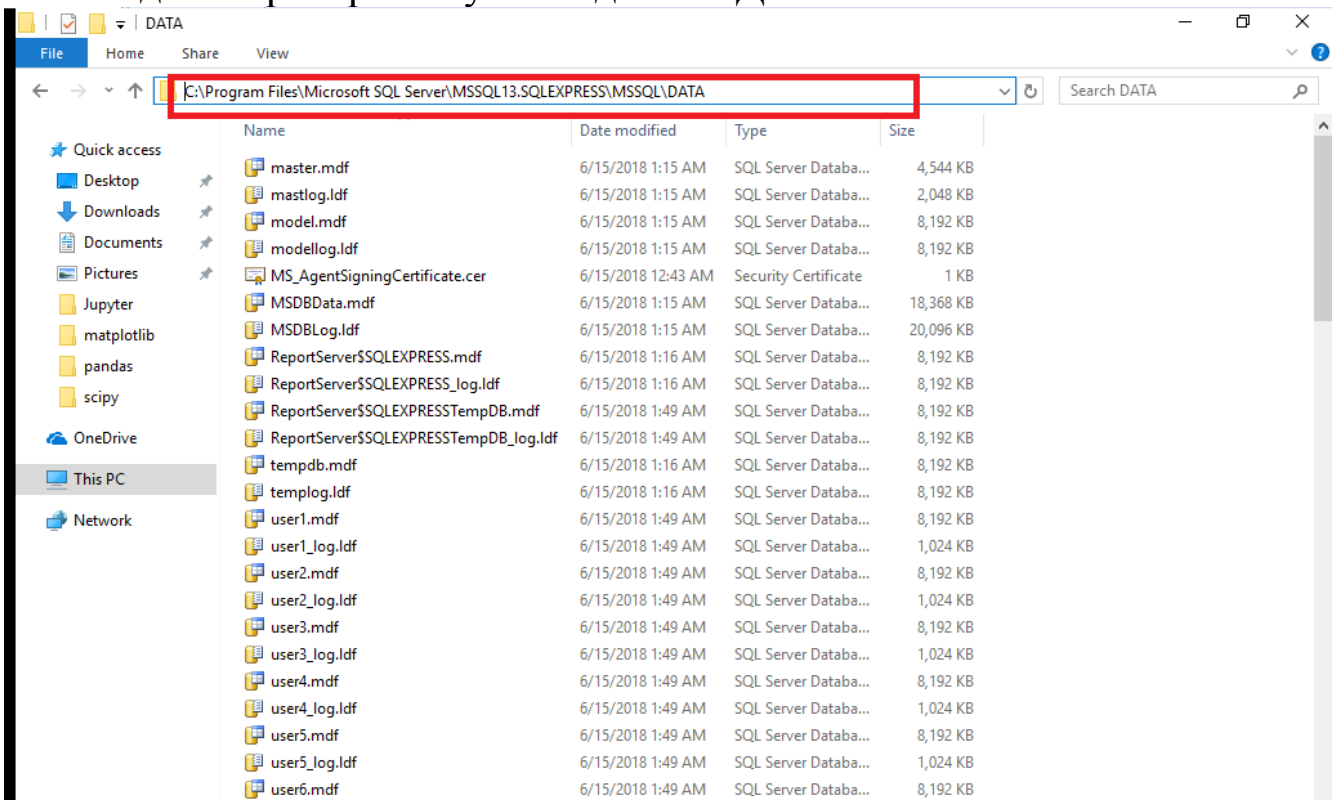


Далее пишем скрипт

```
--Шаблон создания пользователей, БД и настройка прав
DECLARE @user INT = 1;
DECLARE @user_total INT = 51;
DECLARE @dbname VARCHAR(10);
DECLARE @dbpass VARCHAR(10);

WHILE @user <= @user_total
BEGIN
  SET @dbname = 'user'+CAST(@user AS varchar);
  SET @dbpass = 'wsr'+@dbname;
  PRINT @dbname;
  USE [master]
  EXEC('CREATE LOGIN ['+@dbname+'] WITH
PASSWORD=N'"'+@dbpass+'", DEFAULT_DATABASE=[master],
CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF');
  EXEC('CREATE DATABASE ['+@dbname+']
   ON PRIMARY
   ( NAME = N'"'+@dbname+'", FILENAME = N"C:\Program
Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\'+@dbname+'.mdf" ,
SIZE = 5120KB , FILEGROWTH = 1024KB )
   LOG ON
   ( NAME = N'"'+@dbname+'_log", FILENAME = N"C:\Program
Files\Microsoft SQL
Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\'+@dbname+'_log.ldf"
, SIZE = 1024KB , FILEGROWTH = 10%)');
  EXEC('ALTER LOGIN ['+@dbname+'] WITH
DEFAULT_DATABASE=['+@dbname+'],
DEFAULT_LANGUAGE=[русский], CHECK_EXPIRATION=OFF,
CHECK_POLICY=OFF
  DENY VIEW ANY DATABASE TO ['+@dbname+']');
  EXEC('USE ['+@dbname+']; EXEC dbo.sp_changedbowner @loginame =
'+@dbname+', @map = false;');
  PRINT 'ok!';
  SET @user = @user + 1;
END;
```

ПРИМЕЧАНИЕ

Необходимо проверить пути создания БД в системе.

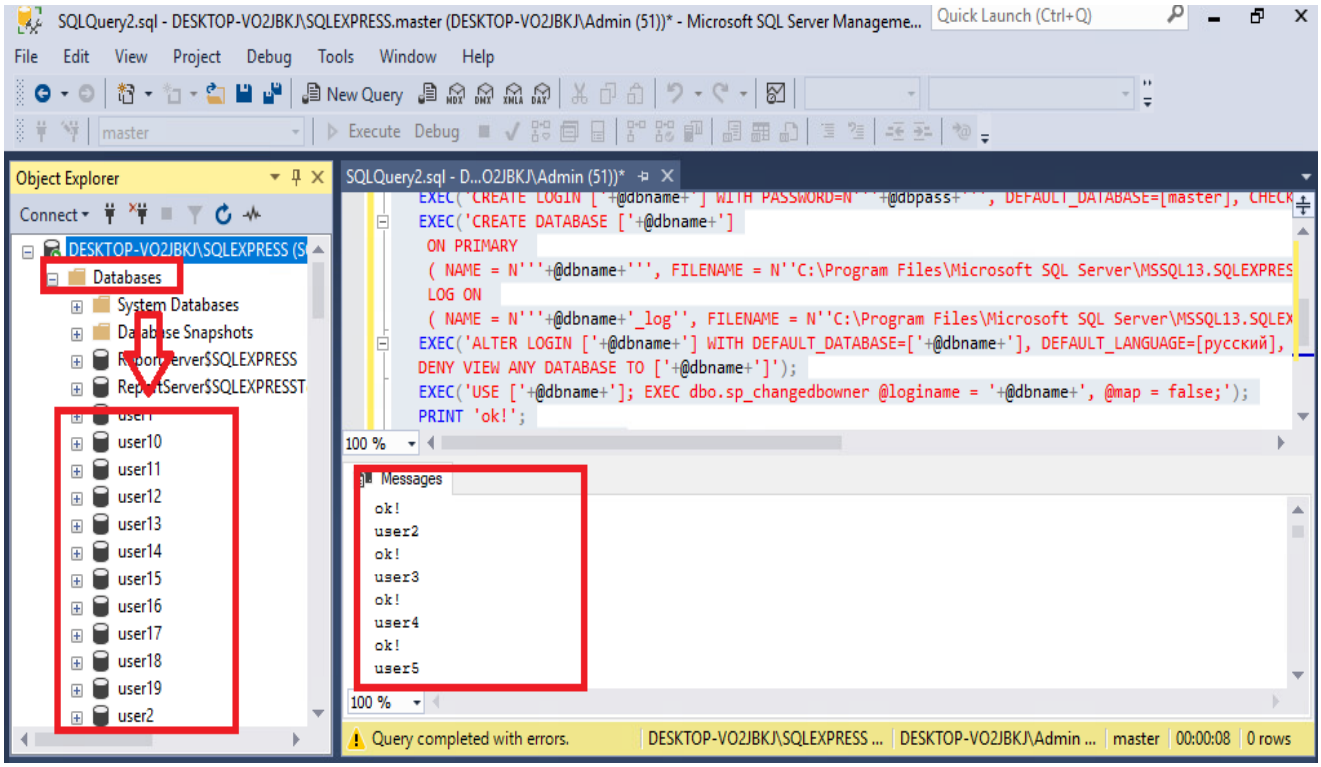


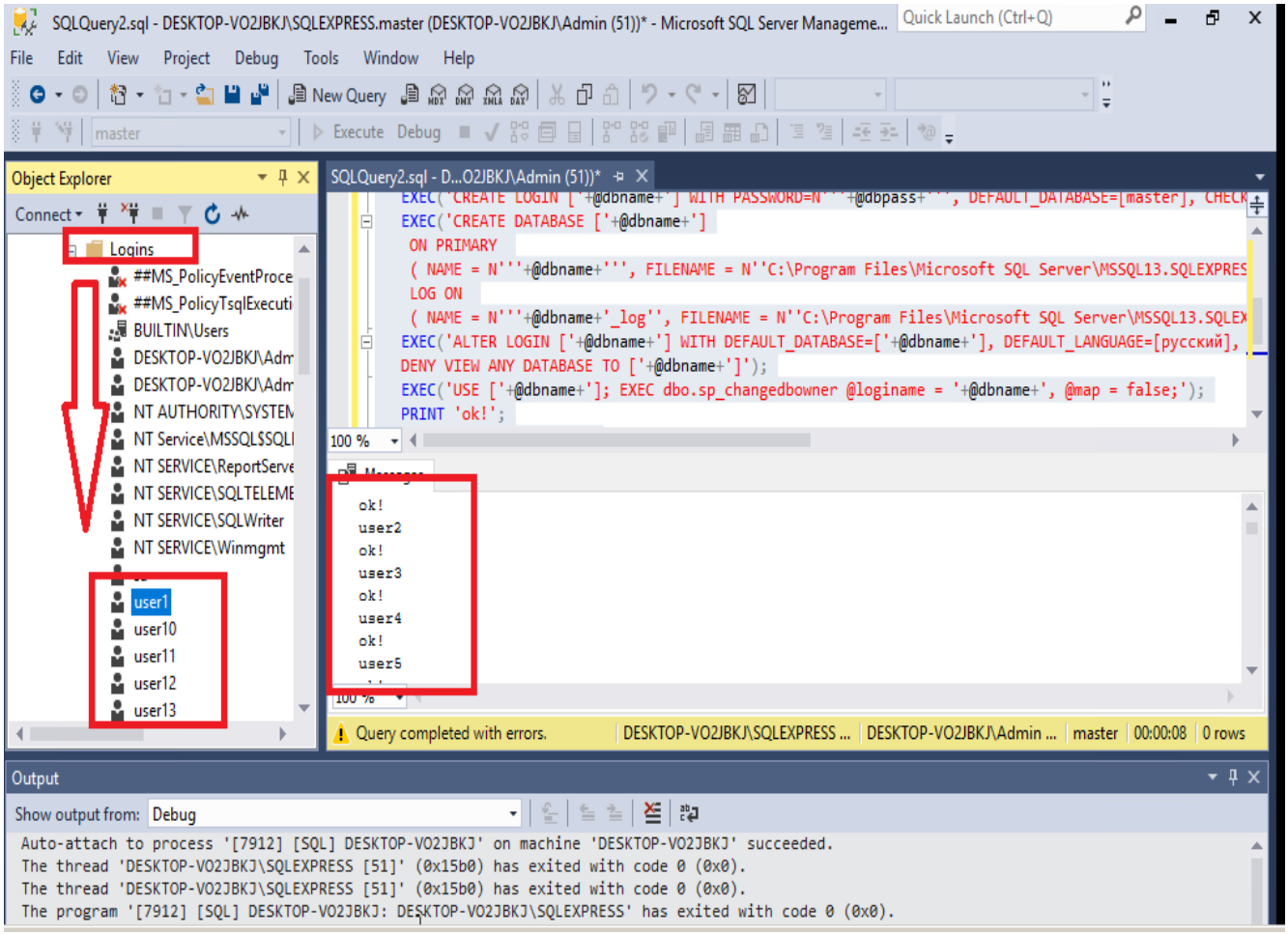
Таким образом скрип в данном случае имеет вид

```
DECLARE @user INT = 1;
DECLARE @user_total INT = 51;
DECLARE @dbname VARCHAR(10);
DECLARE @dbpass VARCHAR(10);

WHILE @user <= @user_total
BEGIN
  SET @dbname = 'user'+CAST(@user AS varchar);
  SET @dbpass = 'wsr'+@dbname;
  PRINT @dbname;
  USE [master]
  EXEC('CREATE LOGIN ['+@dbname+'] WITH
PASSWORD=N'"+@dbpass+"', DEFAULT_DATABASE=[master],
CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF');
  EXEC('CREATE DATABASE ['+@dbname+']
   ON PRIMARY
   ( NAME = N'"+@dbname+"', FILENAME = N'C:\Program
Files\Microsoft SQL
```

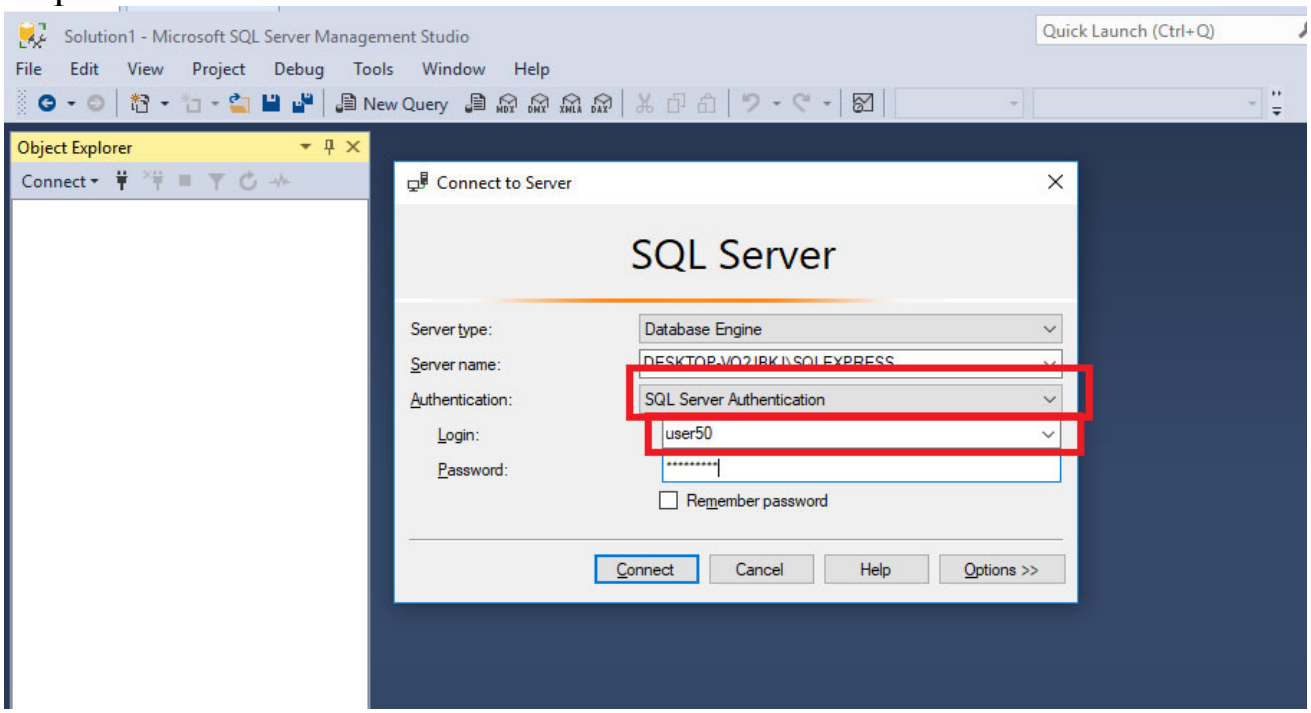
```
Server\MSSQL13.SQLEXPRESS\MSSQL\DATA\'+@dbname+'.mdf" ,
SIZE = 5120KB , FILEGROWTH = 1024KB )
    LOG ON
    ( NAME = N'"+@dbname+'_log", FILENAME = N"C:\Program
Files\Microsoft SQL
Server\MSSQL13.SQLEXPRESS\MSSQL\DATA\'+@dbname+'_log.ldf" ,
SIZE = 1024KB , FILEGROWTH = 10%)');
   EXEC('ALTER LOGIN ['+@dbname+'] WITH
DEFAULT_DATABASE=['+@dbname+'],
DEFAULT_LANGUAGE=[русский], CHECK_EXPIRATION=OFF,
CHECK_POLICY=OFF
   DENY VIEW ANY DATABASE TO ['+@dbname+']');
   EXEC('USE ['+@dbname+']; EXEC dbo.sp_changedbowner @loginame =
'+@dbname+', @map = false;');
   PRINT 'ok!';
   SET @user = @user + 1;
END;
```

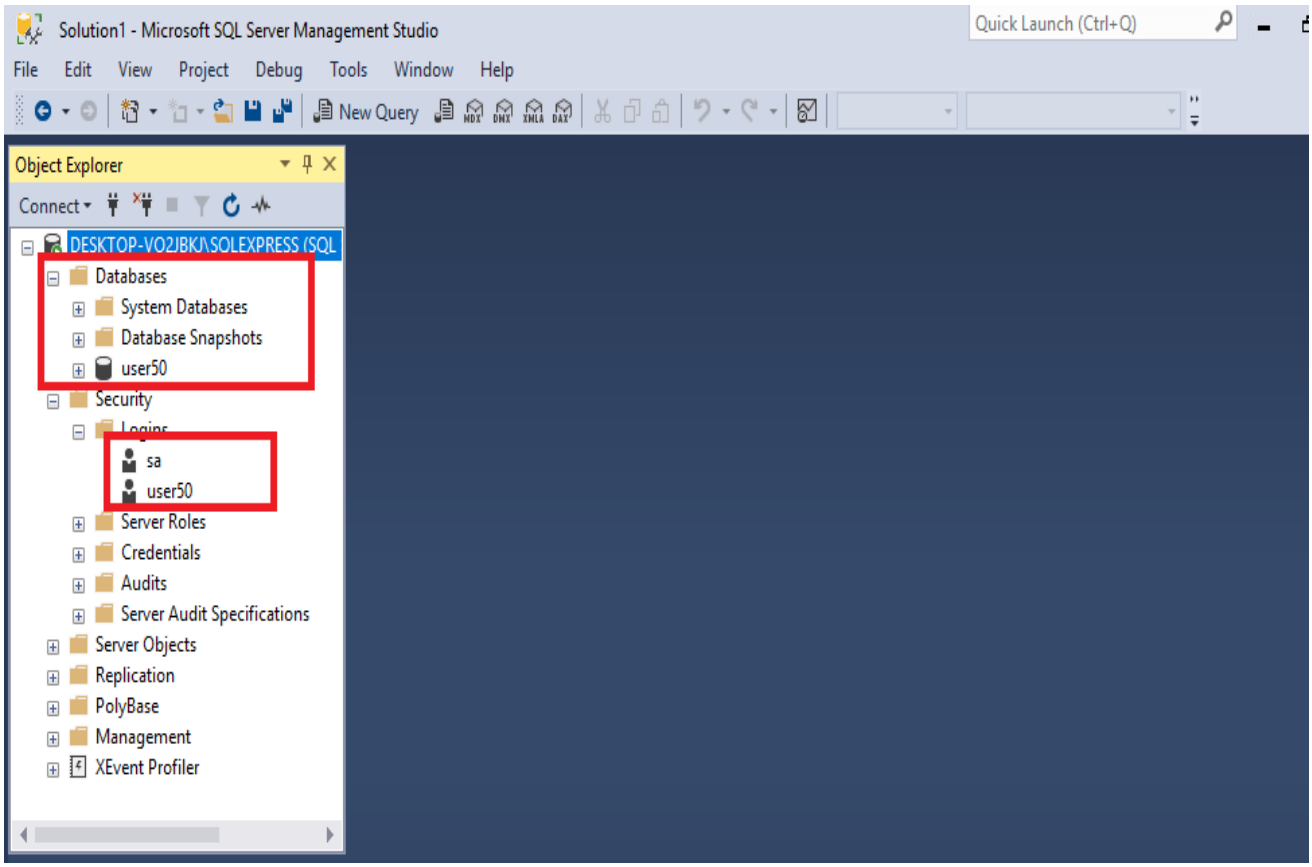
При успешном выполнения скрипта




SQLQuery2.sql - DESKTOP-VO2JBKJ\SQLEXPRESS.master (DESKTOP-VO2JBKJ\Admin (51))* - Microsoft SQL Server Manageme...
Quick Launch (Ctrl+Q)
File Edit View Project Debug Tools Window Help
New Query
Execute Debug
Object Explorer
Connect
Logins
##MS_PolicyEventProce
##MS_PolicyTsqlExecuti
BUILTIN\Users
DESKTOP-VO2JBKJ\Adm
DESKTOP-VO2JBKJ\Adm
NT AUTHORITY\SYSTEM
NT Service\MSSQL$SQLI
NT SERVICE\ReportServe
NT SERVICE\SQLTELEME
NT SERVICE\SQLWriter
NT SERVICE\Winmgmt
user1
user10
user11
user12
user13
SQLQuery2.sql - D...O2JBKJ\Admin (51))*
EXEC('CREATE DATABASE ['+@dbname+']
ON PRIMARY
(NAME = N'''+@dbname+''', FILENAME = N''C:\Program Files\Microsoft SQL Server\MSSQL13.SQLEXPRES
LOG ON
(NAME = N'''+@dbname+'_log'', FILENAME = N''C:\Program Files\Microsoft SQL Server\MSSQL13.SQLEX
EXEC('ALTER LOGIN ['+@dbname+'] WITH DEFAULT_DATABASE=['+@dbname+'], DEFAULT_LANGUAGE=[русский],
DENY VIEW ANY DATABASE TO ['+@dbname+']');
EXEC('USE ['+@dbname+']; EXEC dbo.sp_changedbowner @loginname = '+@dbname+', @map = false;');
PRINT 'ok!';
100 %
ok!
user2
ok!
user3
ok!
user4
ok!
user5
Query completed with errors.
DESKTOP-VO2JBKJ\SQLEXPRESS ...
DESKTOP-VO2JBKJ\Admin ...
master
00:00:08
0 rows
Output
Show output from: Debug
Auto-attach to process '[7912] [SQL] DESKTOP-VO2JBKJ' on machine 'DESKTOP-VO2JBKJ' succeeded.
The thread 'DESKTOP-VO2JBKJ\SQLEXPRESS [51]' (0x15b0) has exited with code 0 (0x0).
The thread 'DESKTOP-VO2JBKJ\SQLEXPRESS [51]' (0x15b0) has exited with code 0 (0x0).
The program '[7912] [SQL] DESKTOP-VO2JBKJ: DESKTOP-VO2JBKJ\SQLEXPRESS' has exited with code 0 (0x0).

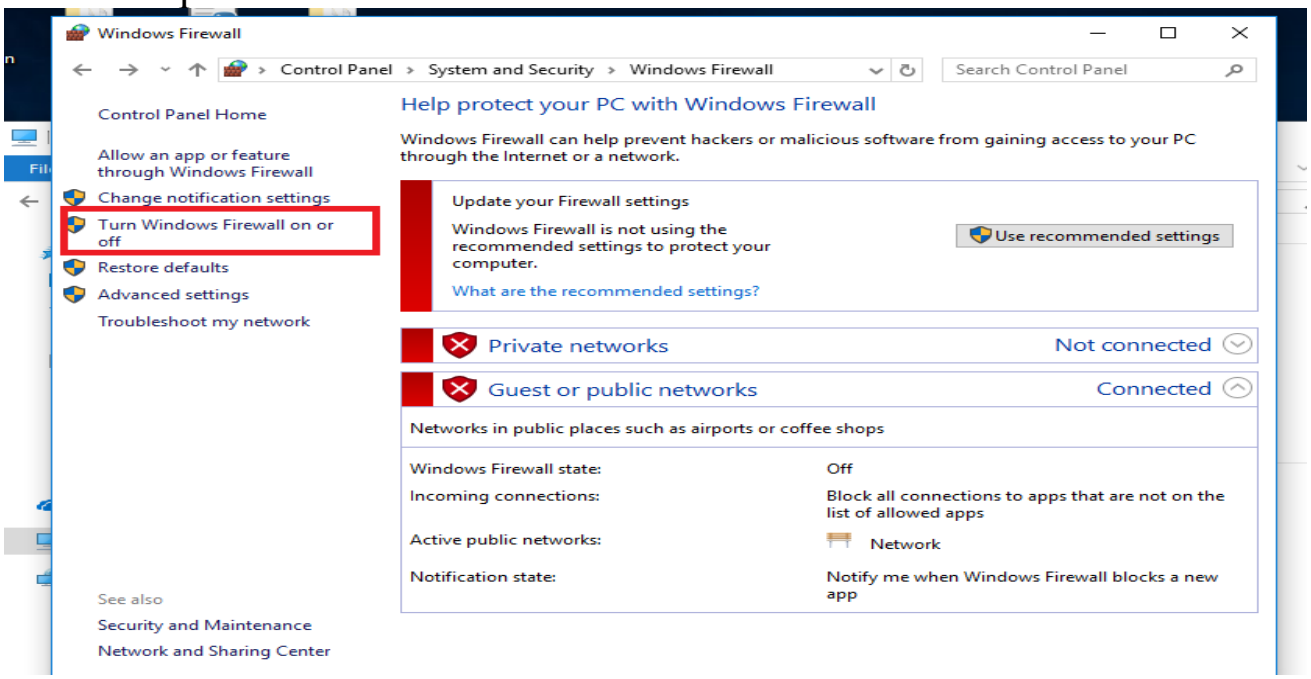
Проверка авторизации пользователя
Логин user50
Пароль wsruser50

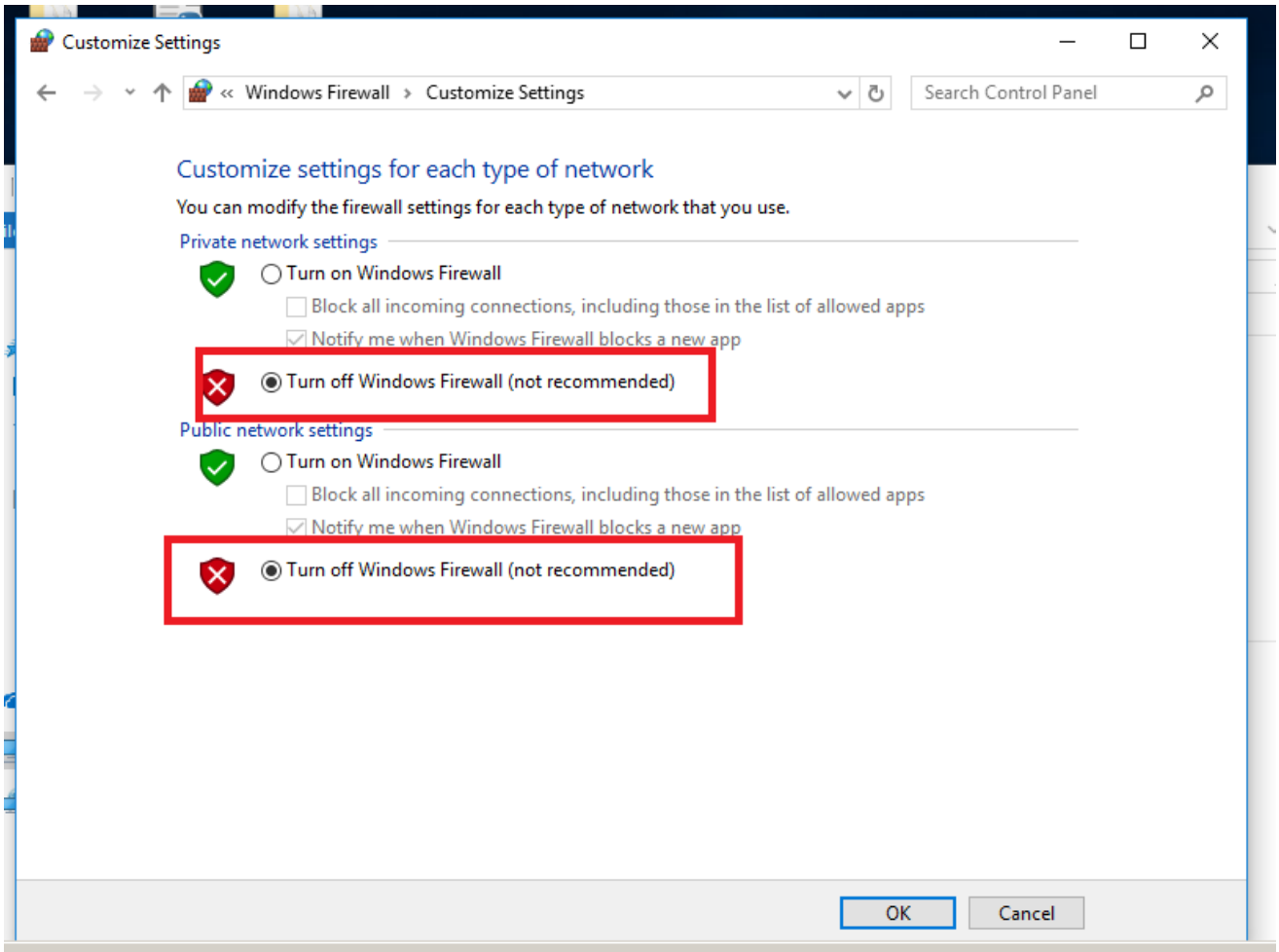




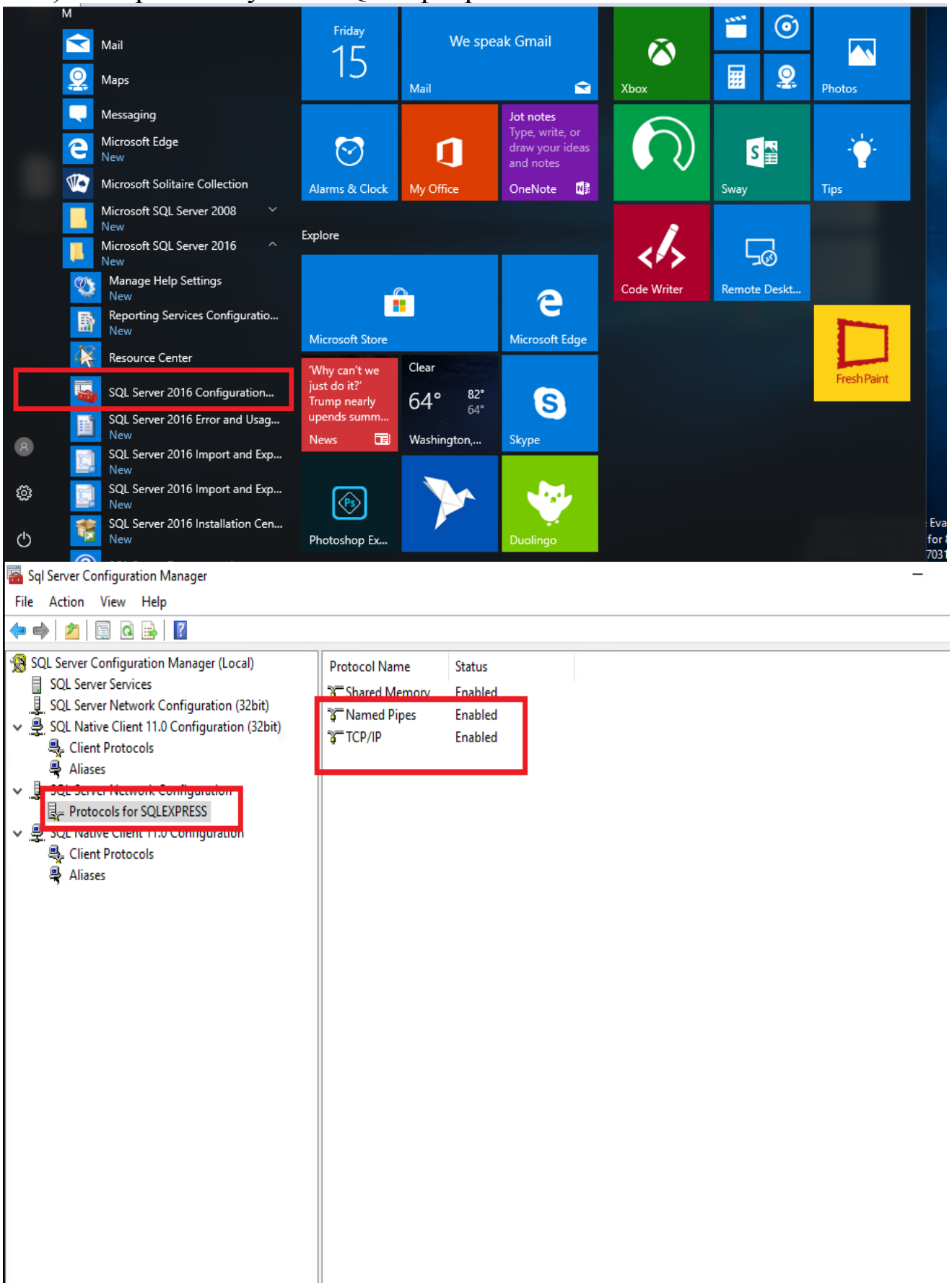
Для разрешения удаленного подключения к серверу SQL необходимо

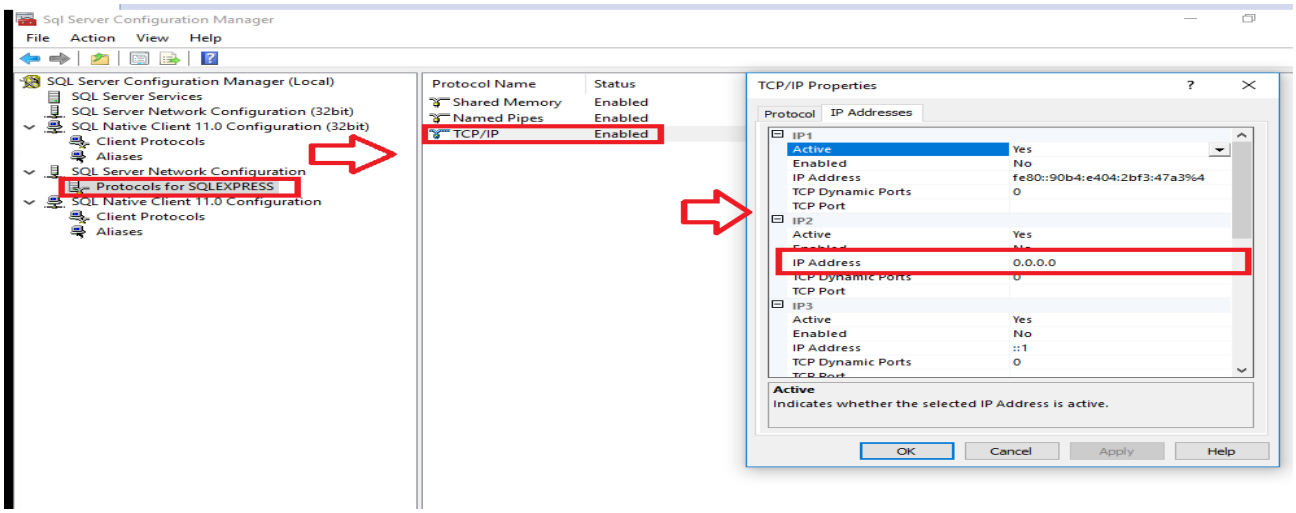
1) Разрешить 1433 порт в настройках firewall или выключить его в настройках

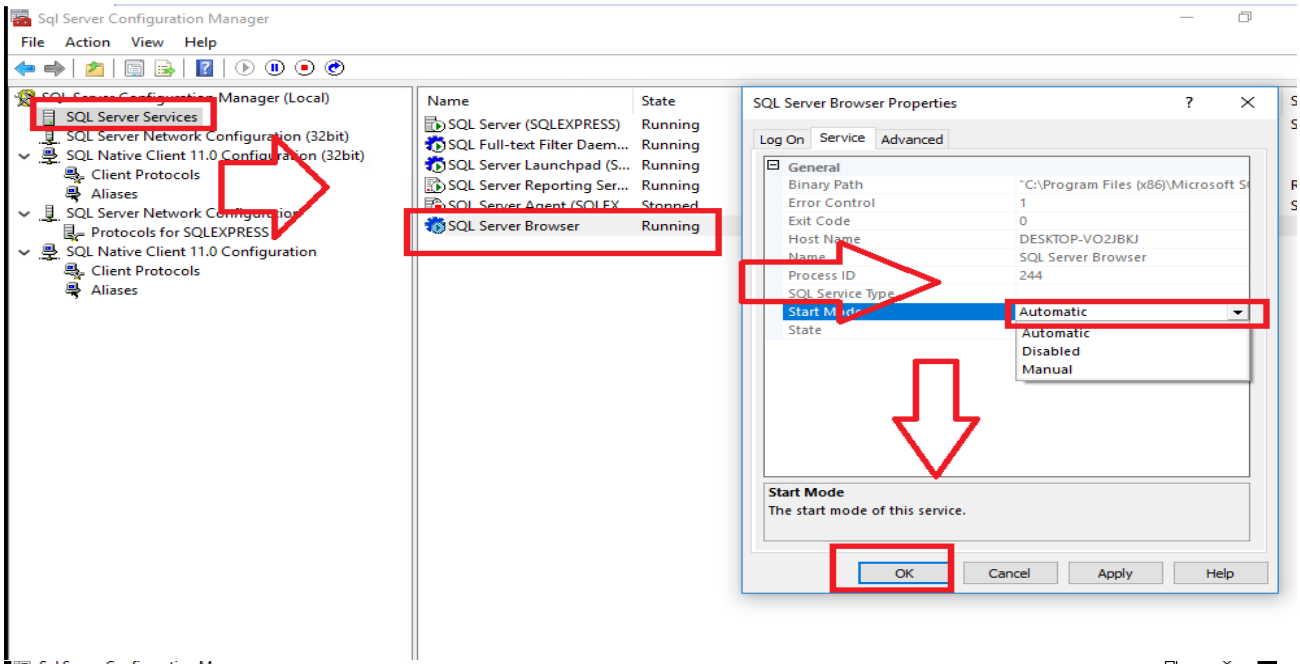


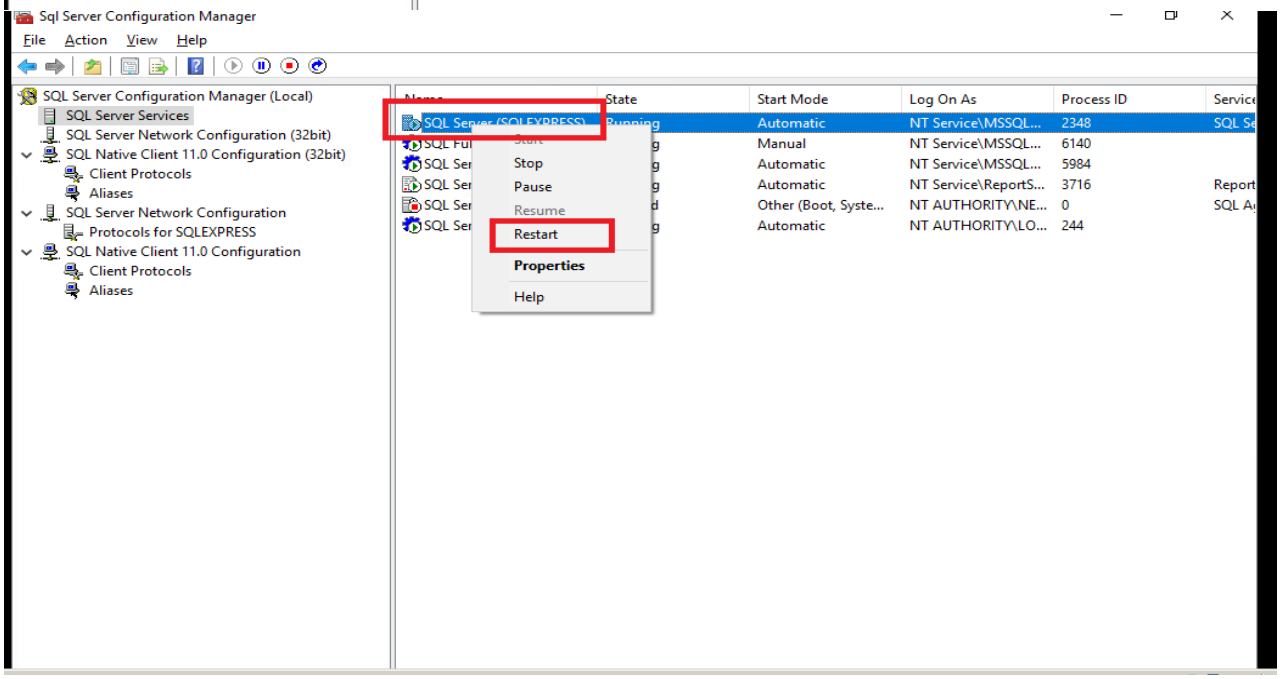


2) Настроить службы SQL сервера

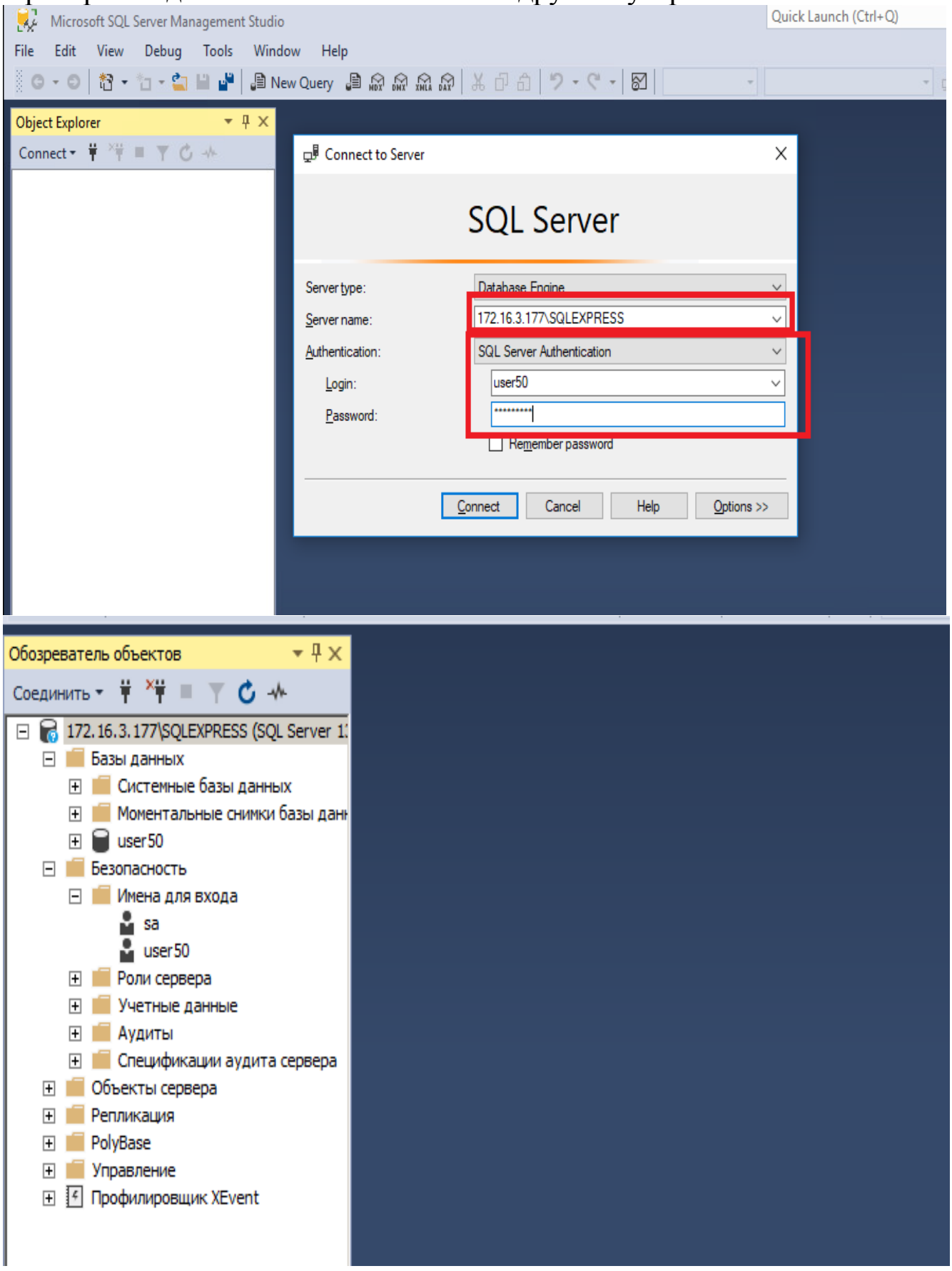



Sql Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 11.0 Configuration (32bit)
Client Protocols
Aliases
SQL Server Network Configuration
Protocols for SQLEXPRESS
SQL Native Client 11.0 Configuration
Client Protocols
Aliases
Protocol Name
Status
Shared Memory
Enabled
Named Pipes
Enabled
TCP/IP
Enabled
TCP/IP Properties
Protocol
IP Addresses
IP1
Active
Yes
Enabled
No
IP Address
fe80::90b4:e404:2bf3:47a3%4
TCP Dynamic Ports
0
TCP Port
IP2
Active
Yes
IP Address
0.0.0.0
TCP Dynamic Ports
0
TCP Port
IP3
Active
Yes
Enabled
No
IP Address
::1
TCP Dynamic Ports
0
Active
Indicates whether the selected IP Address is active.
OK
Cancel
Apply
Help

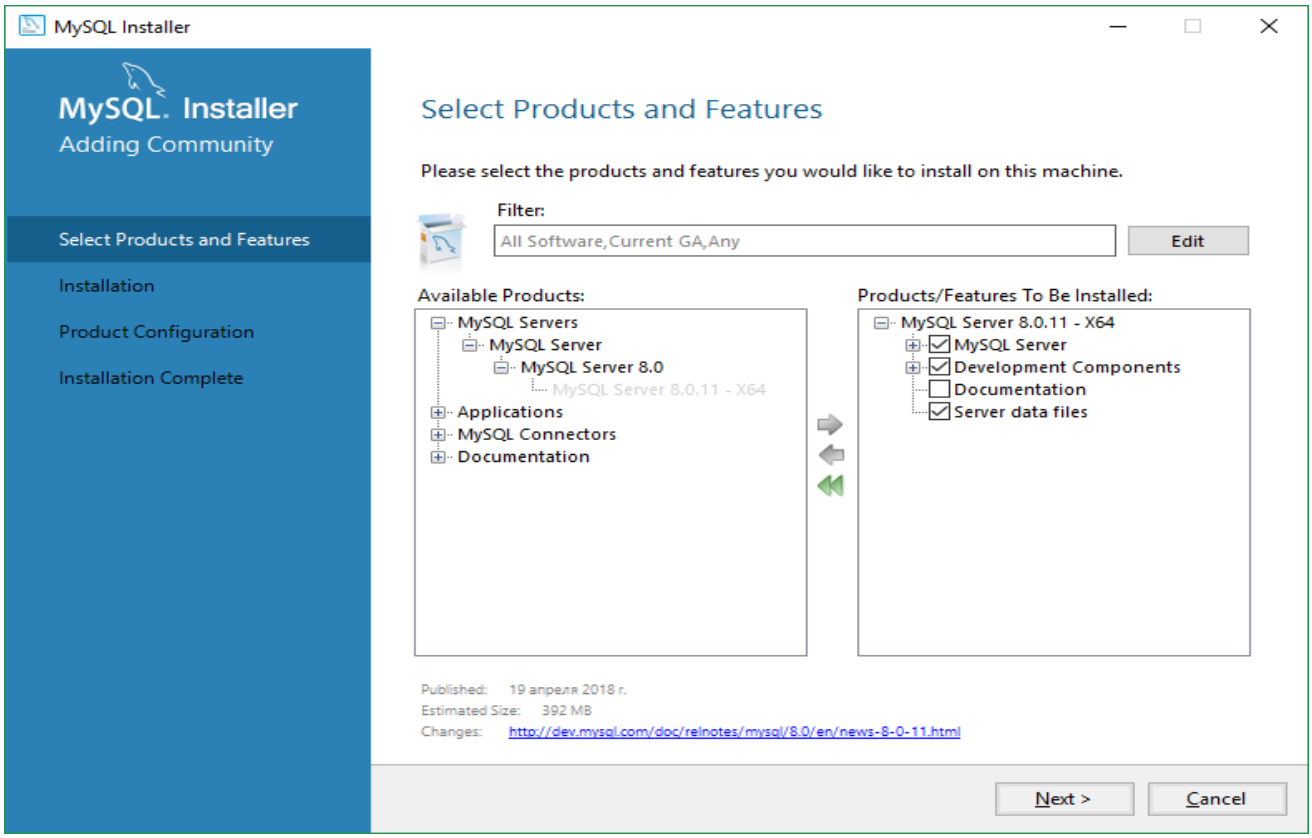

Sql Server Configuration Manager
File Action View Help
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 11.0 Configuration (32bit)
Client Protocols
Aliases
SQL Server Network Configuration
Protocols for SQLEXPRESS
SQL Native Client 11.0 Configuration
Client Protocols
Aliases
Name
State
SQL Server (SQLEXPRESS)
Running
SQL Full-text Filter Daem...
Running
SQL Server Launchpad (S...
Running
SQL Server Reporting Ser...
Running
SQL Server Browser
Running
SQL Server Browser Properties
Log On
Service
Advanced
General
Binary Path
Error Control
1
Exit Code
0
Host Name
DESKTOP-VO2JBKJ
Name
SQL Server Browser
Process ID
244
SQL Service Type
Start Mode
Automatic
State
Automatic
Disabled
Manual
Start Mode
The start mode of this service.
OK
Cancel
Apply
Help

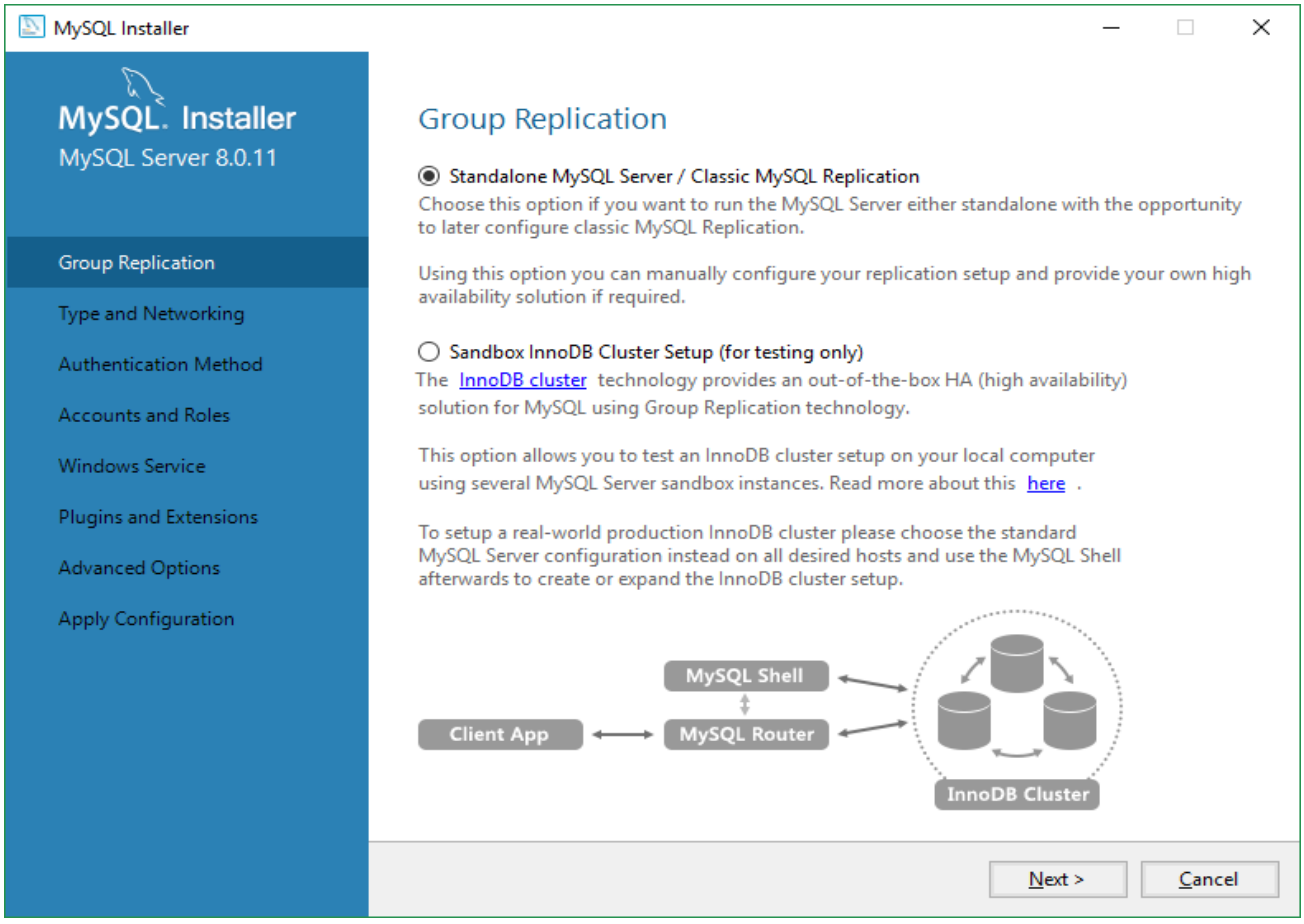

Sql Server Configuration Manager
File Action View Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 11.0 Configuration (32bit)
Client Protocols
Aliases
SQL Server Network Configuration
Protocols for SQLEXPRESS
SQL Native Client 11.0 Configuration
Client Protocols
Aliases
Name
State
Start Mode
Log On As
Process ID
SQL Server (SQLEXPRESS)
Automatic
NT Service\MSSQL...
2348
Manual
NT Service\MSSQL...
6140
Automatic
NT Service\MSSQL...
5984
Automatic
NT Service\ReportS...
3716
Other (Boot, Syste...
0
Automatic
NT AUTHORITY\LO...
244
Stop
Pause
Resume
Restart
Properties
Help

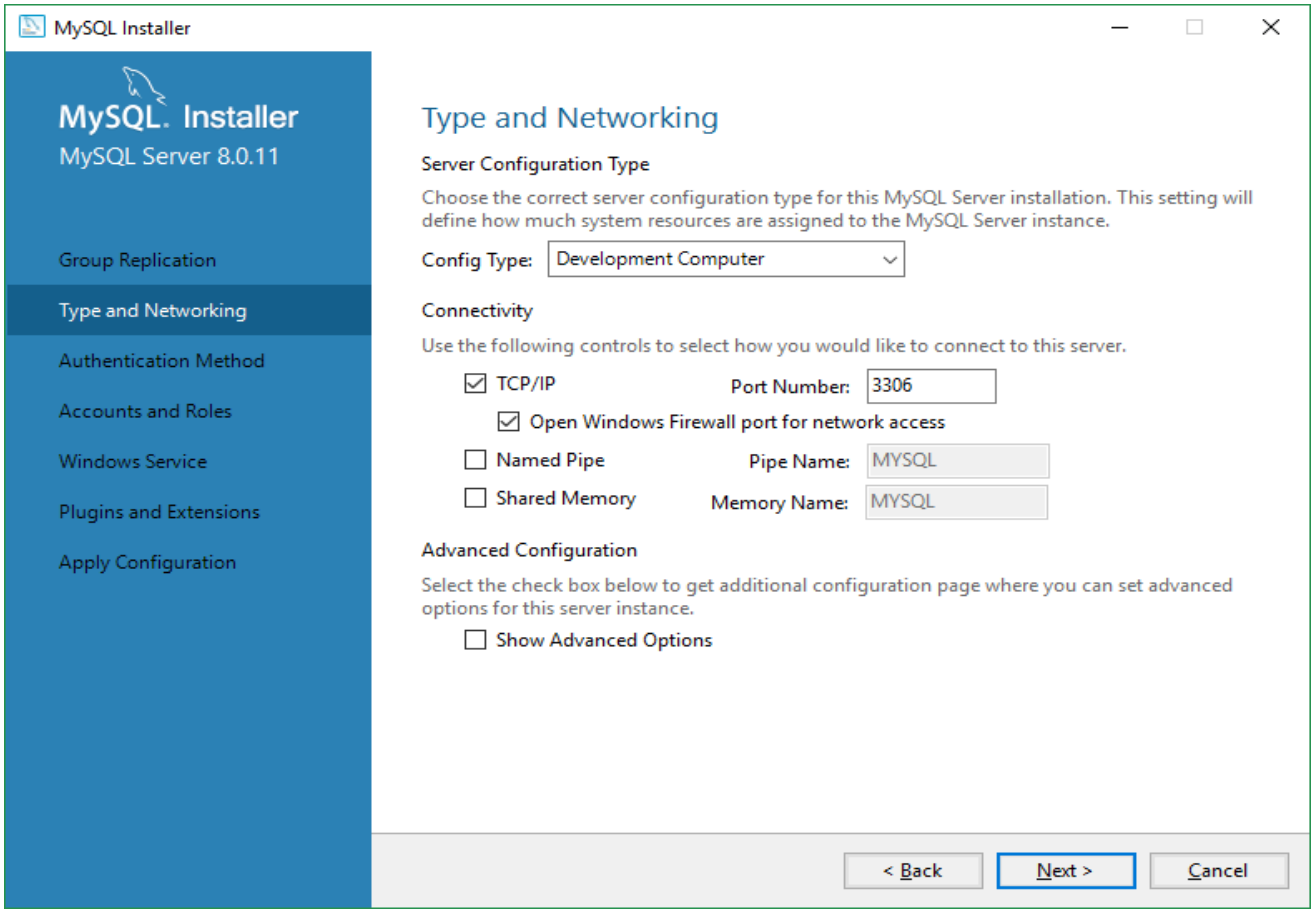
Проверка подключения пользователя с другого устройства

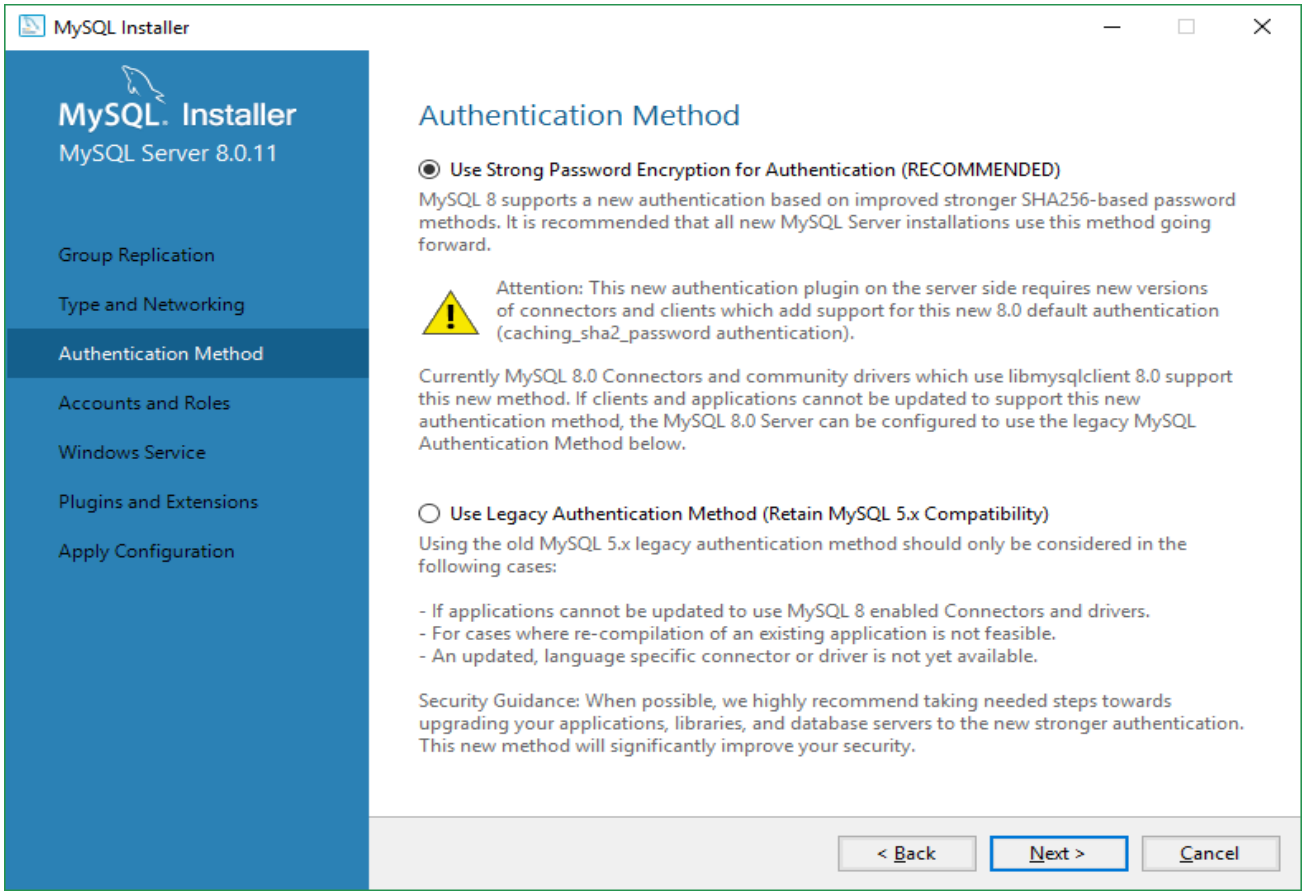


Установка MYSQL и настройка

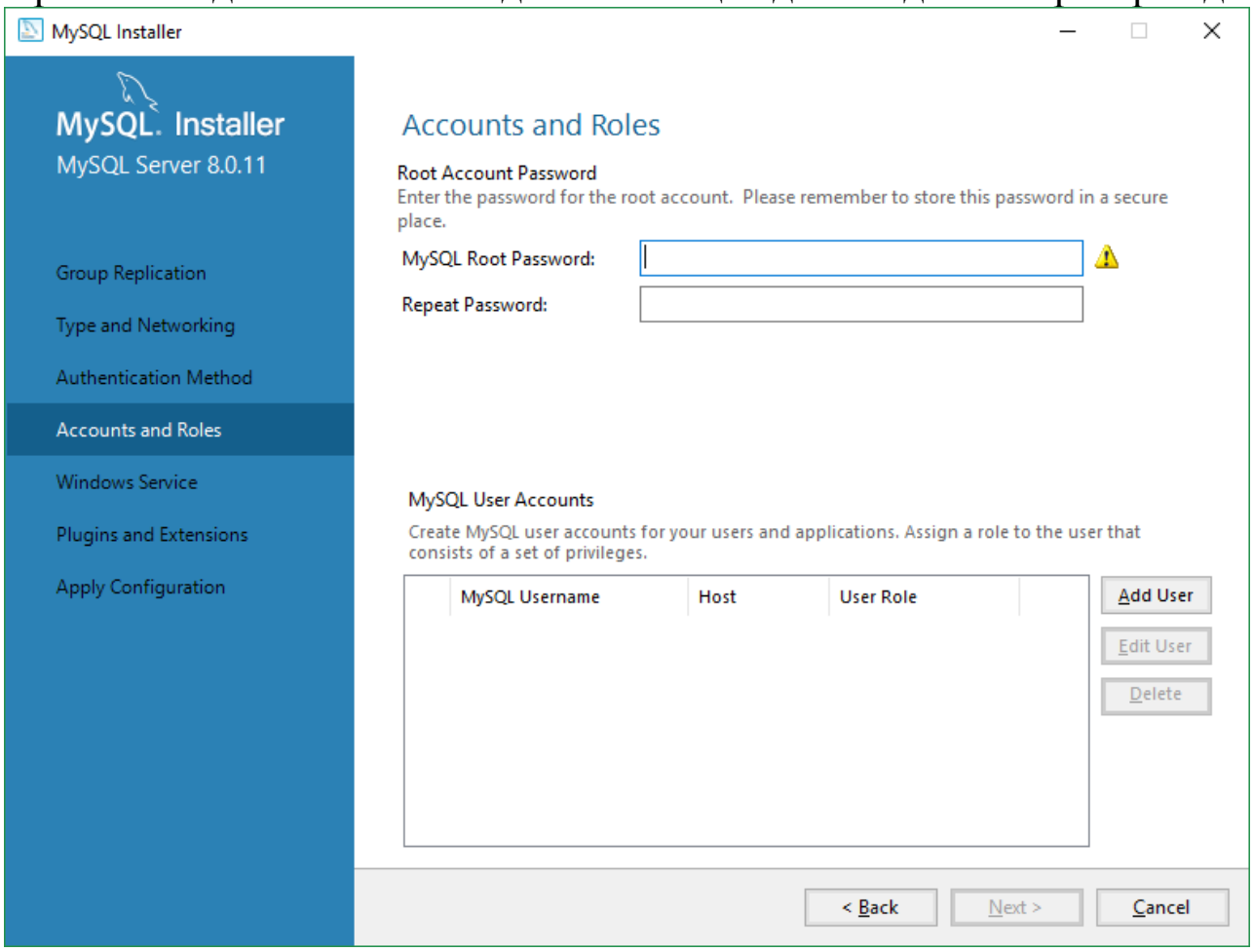


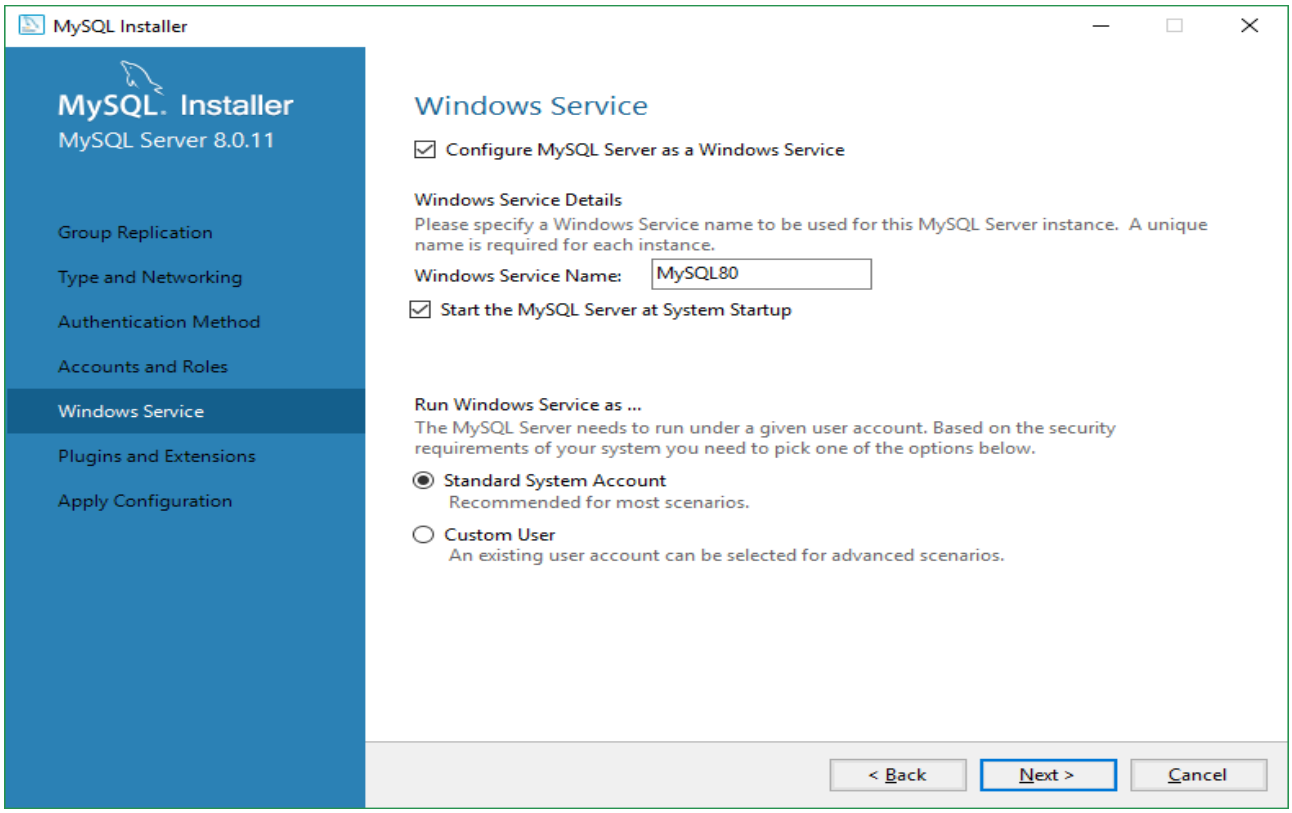



MySQL Installer
MySQL. Installer
MySQL Server 8.0.11
Group Replication
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Plugins and Extensions
Apply Configuration
Type and Networking
Server Configuration Type
Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.
Config Type: Development Computer
Connectivity
Use the following controls to select how you would like to connect to this server.
TCP/IP
Port Number: 3306
Open Windows Firewall port for network access
Named Pipe
Pipe Name: MYSQL
Shared Memory
Memory Name: MYSQL
Advanced Configuration
Select the check box below to get additional configuration page where you can set advanced options for this server instance.
Show Advanced Options
< Back
Next >
Cancel

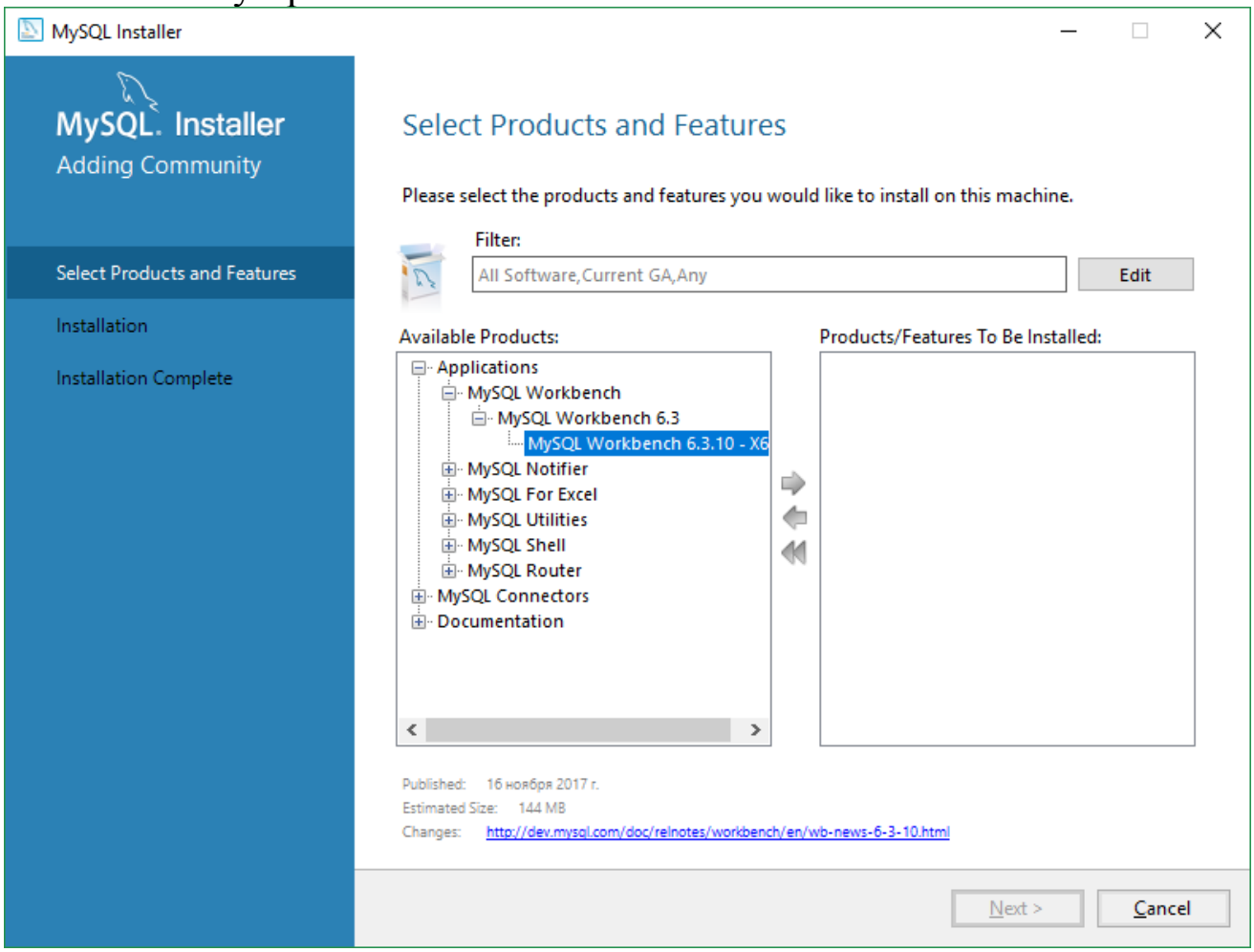

MySQL Installer
MySQL. Installer
MySQL Server 8.0.11
Group Replication
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Plugins and Extensions
Apply Configuration
Authentication Method
Use Strong Password Encryption for Authentication (RECOMMENDED)
MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.
Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).
Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.
Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:
- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.
Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.
< Back
Next >
Cancel

При необходимости можно добавить еще одного администратора бд

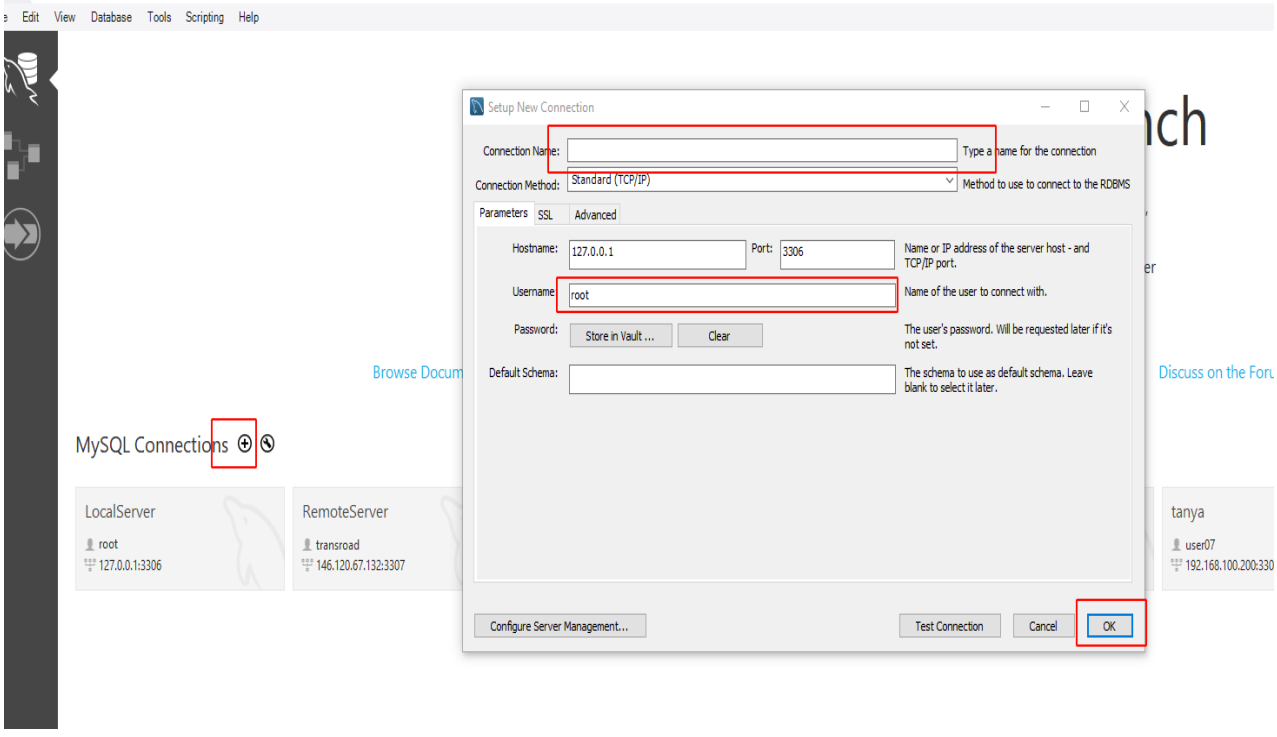




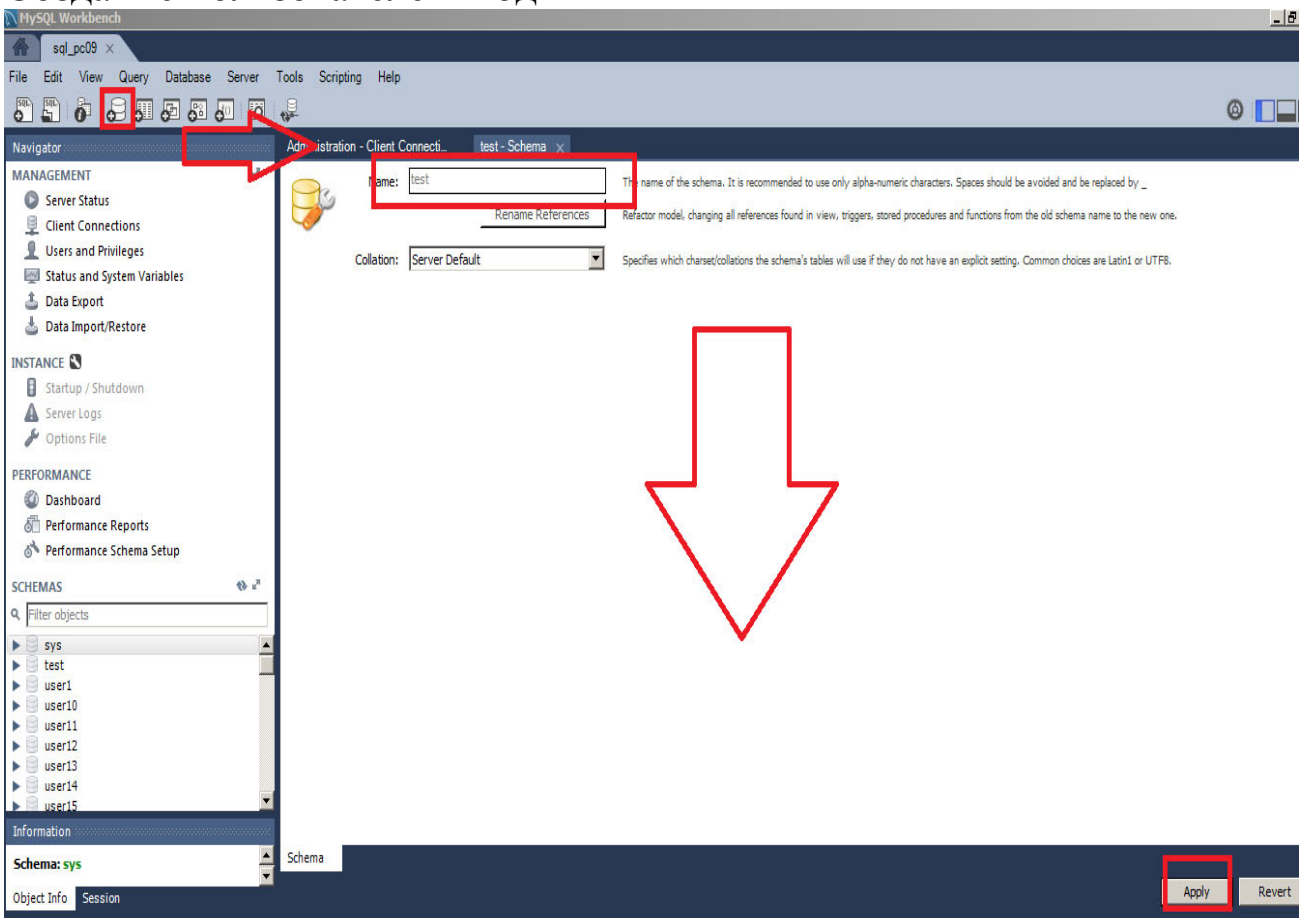
Установка MySql workbench

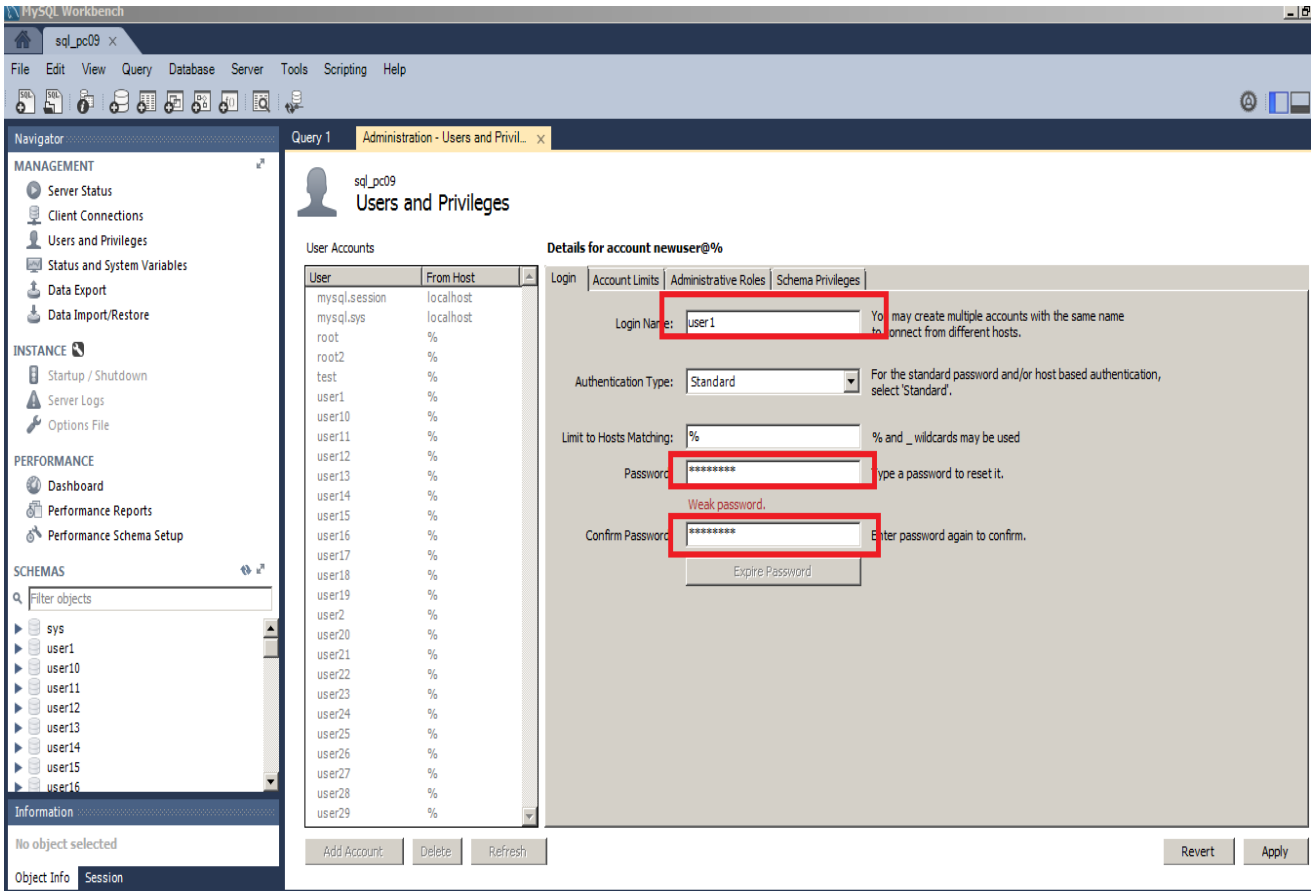


Настройка MYSQL
Подключение к бд

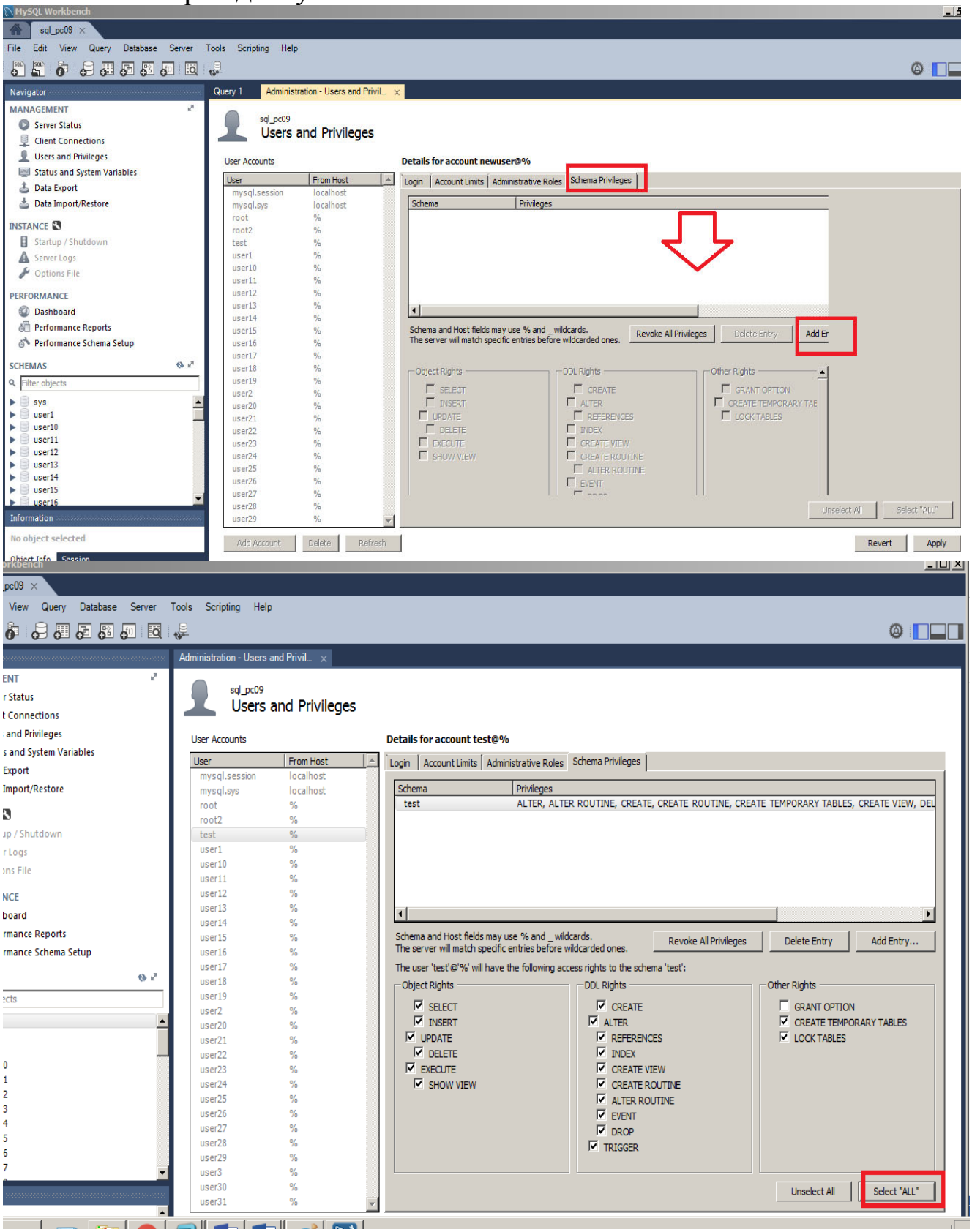


Создание пользователей и бд



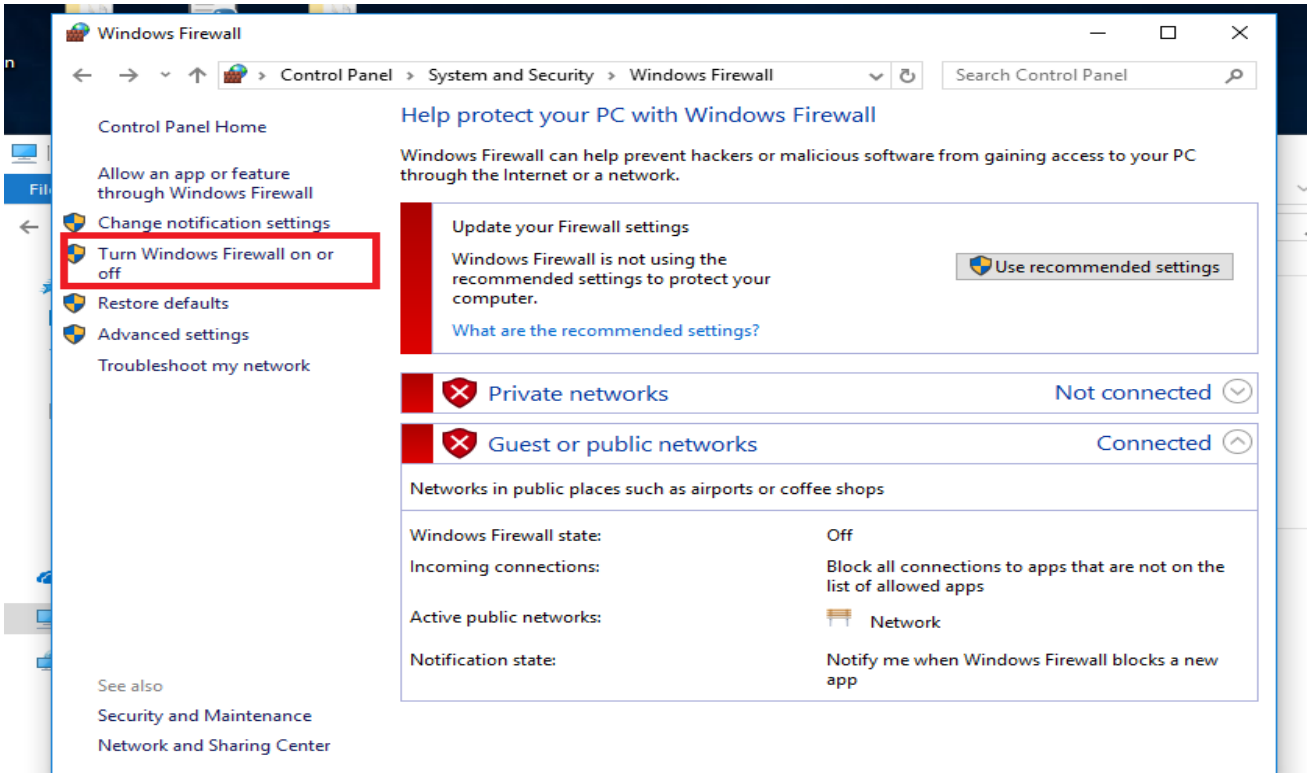


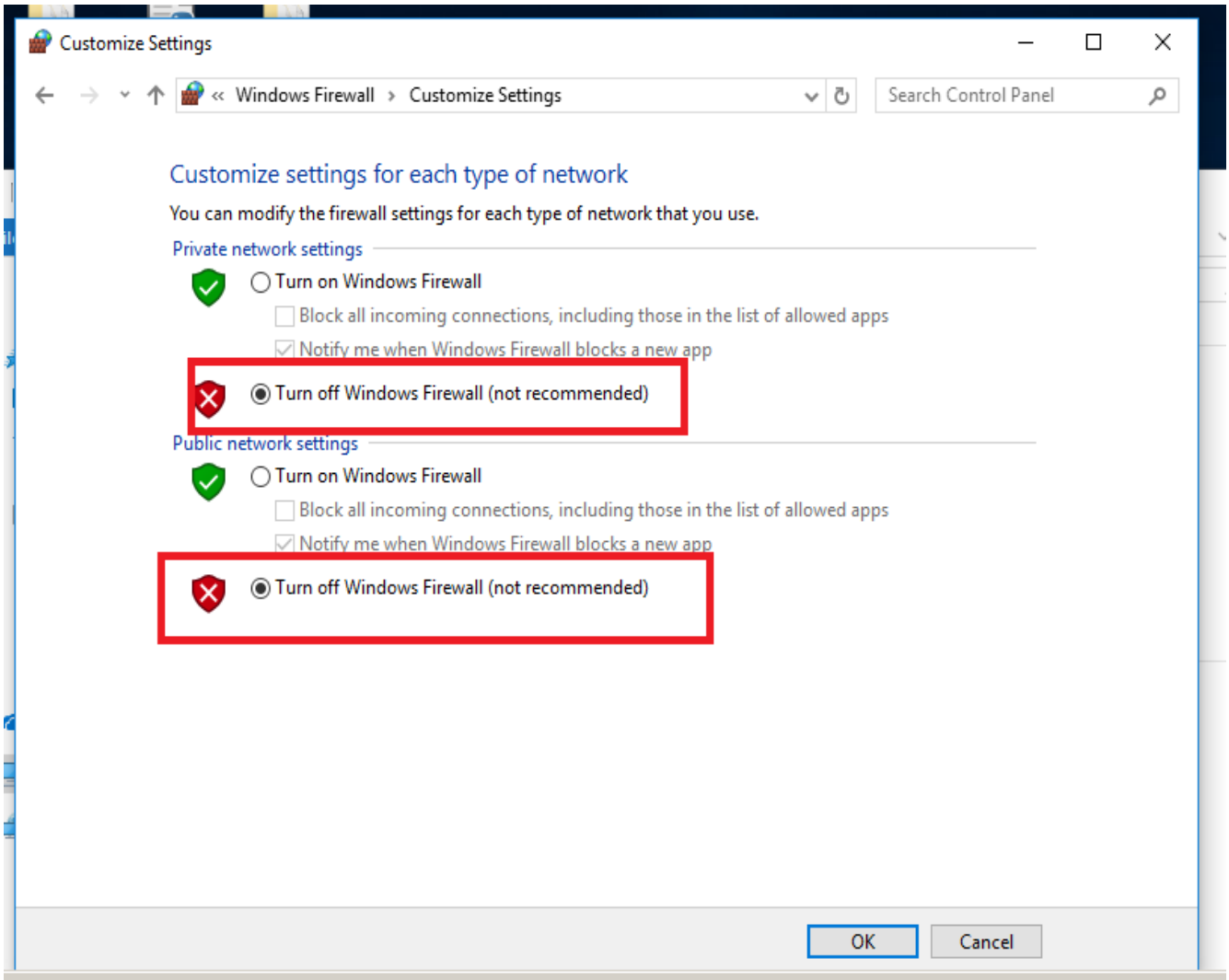
Назначение прав доступа



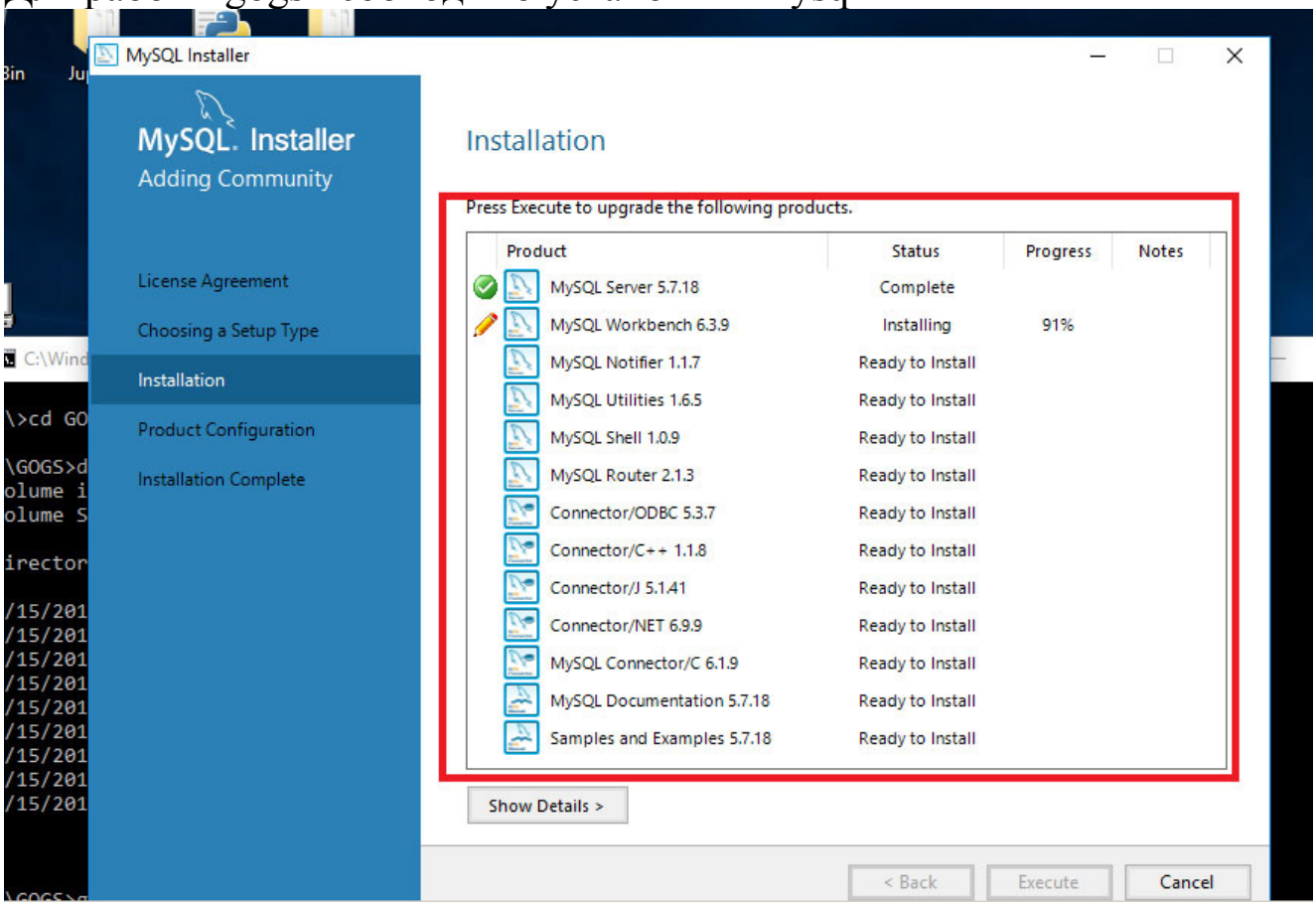
Для разрешения удаленного подключения к серверу My SQL необходимо

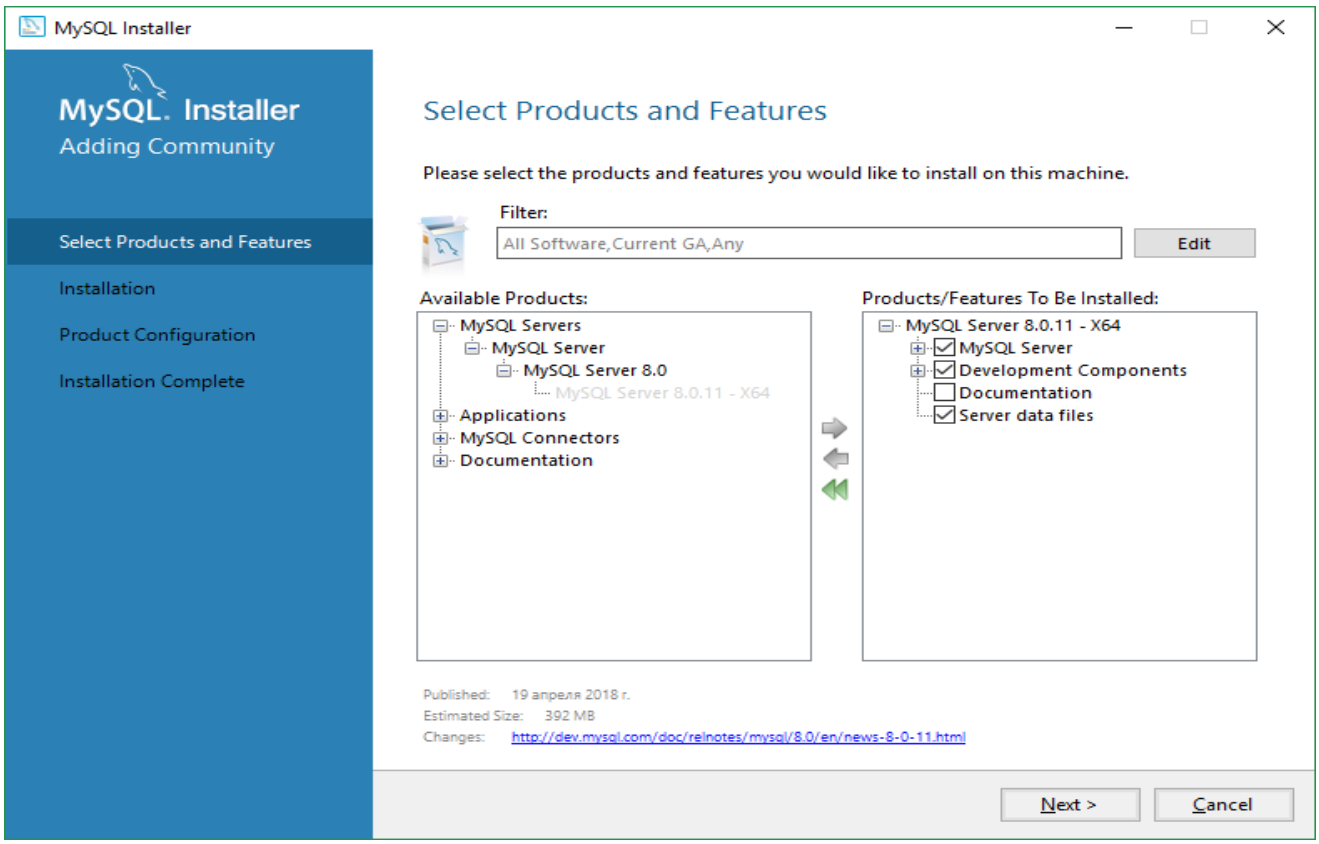
1) Разрешить 3306 порт в настройках firewall или выключить его в настройках

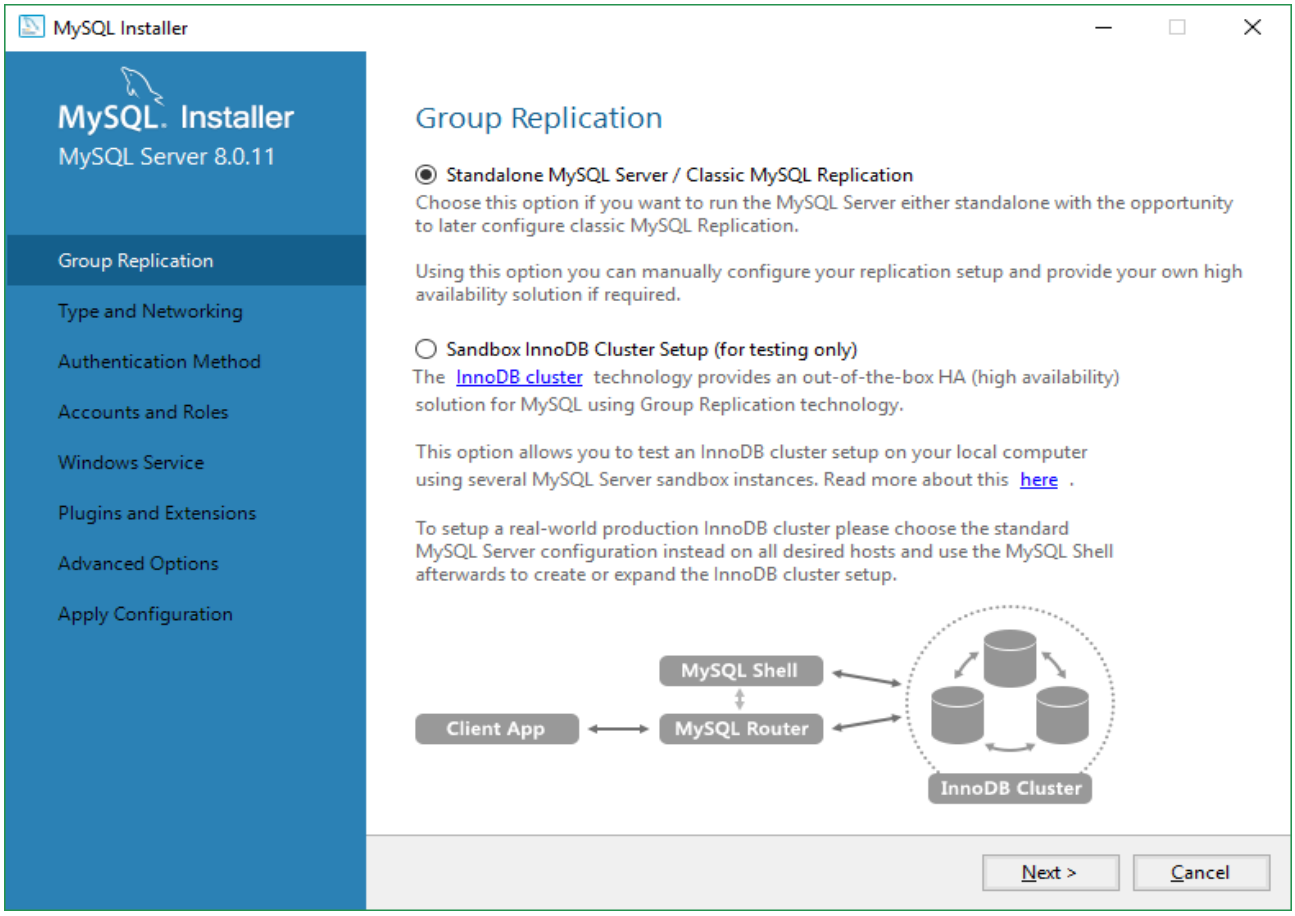

Windows Firewall
Control Panel › System and Security › Windows Firewall
Search Control Panel
Control Panel Home
Allow an app or feature through Windows Firewall
Change notification settings
Turn Windows Firewall on or off
Restore defaults
Advanced settings
Troubleshoot my network
Help protect your PC with Windows Firewall
Windows Firewall can help prevent hackers or malicious software from gaining access to your PC through the Internet or a network.
Update your Firewall settings
Windows Firewall is not using the recommended settings to protect your computer.
What are the recommended settings?
Use recommended settings
Private networks
Not connected
Guest or public networks
Connected
Networks in public places such as airports or coffee shops
Windows Firewall state:
Off
Incoming connections:
Block all connections to apps that are not on the list of allowed apps
Active public networks:
Network
Notification state:
Notify me when Windows Firewall blocks a new app
See also
Security and Maintenance
Network and Sharing Center

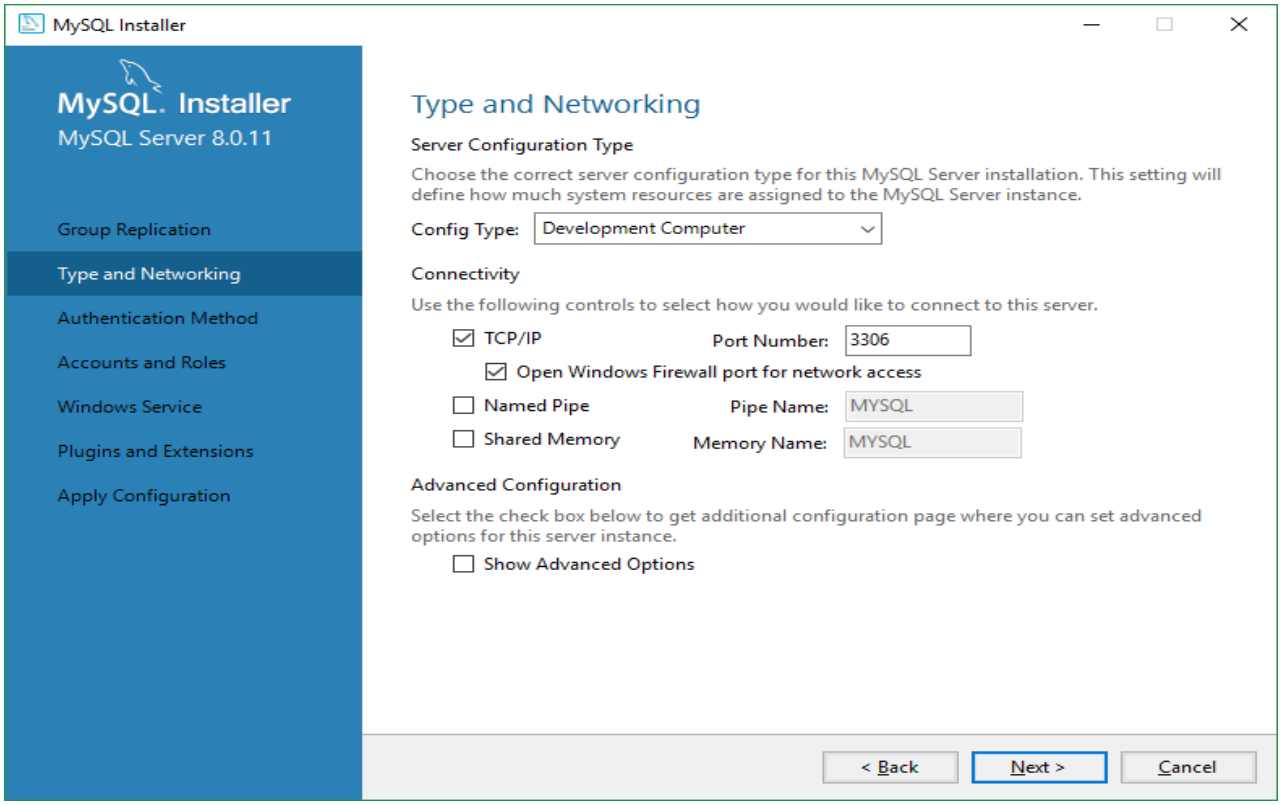

Customize Settings
« Windows Firewall › Customize Settings
Search Control Panel
Customize settings for each type of network
You can modify the firewall settings for each type of network that you use.
Private network settings
Turn on Windows Firewall
Block all incoming connections, including those in the list of allowed apps
Notify me when Windows Firewall blocks a new app
Turn off Windows Firewall (not recommended)
Public network settings
Turn on Windows Firewall
Block all incoming connections, including those in the list of allowed apps
Notify me when Windows Firewall blocks a new app
Turn off Windows Firewall (not recommended)
OK
Cancel

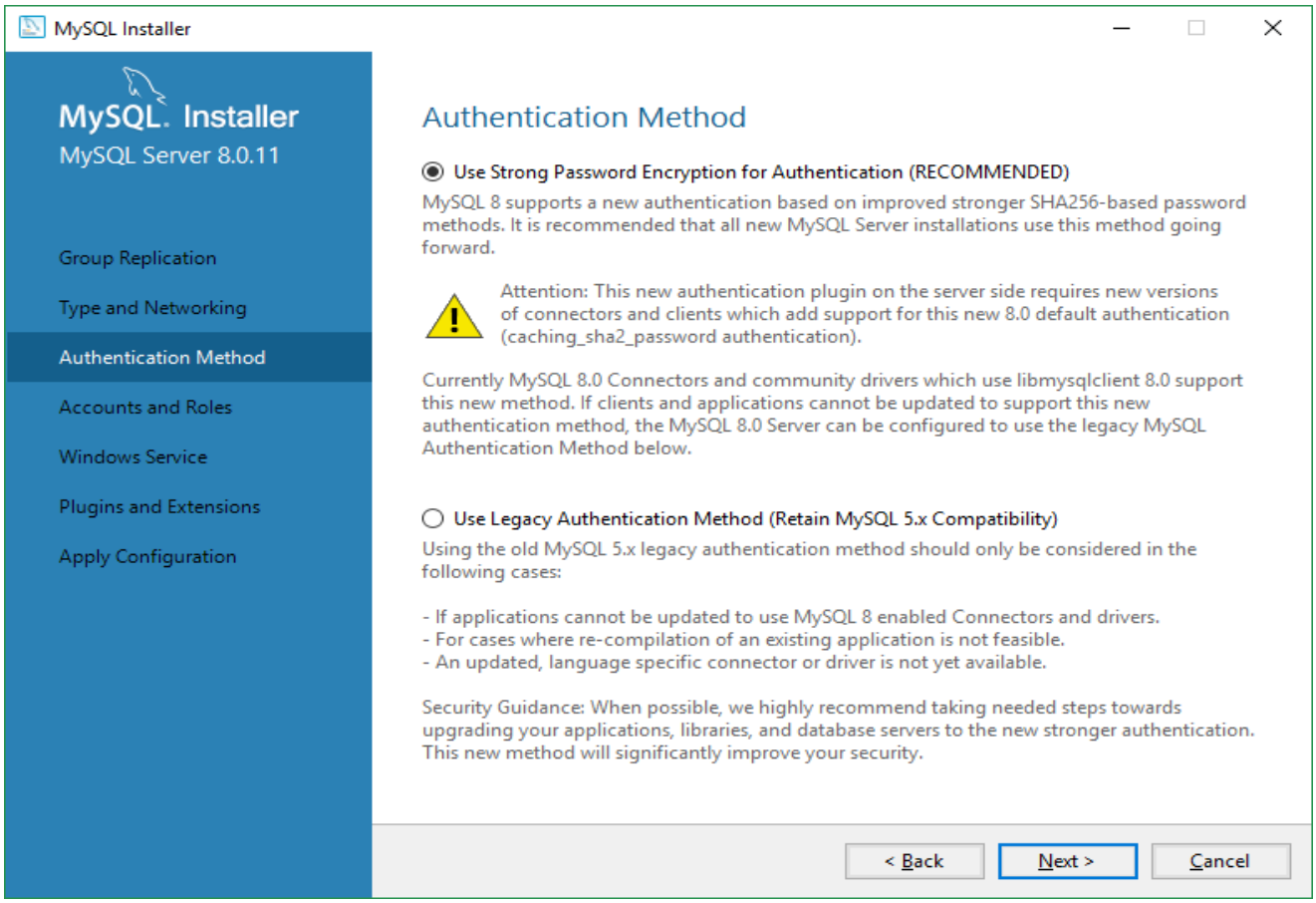
Установка GOGS и настройка
Для работы gogs необходимо установить mysql



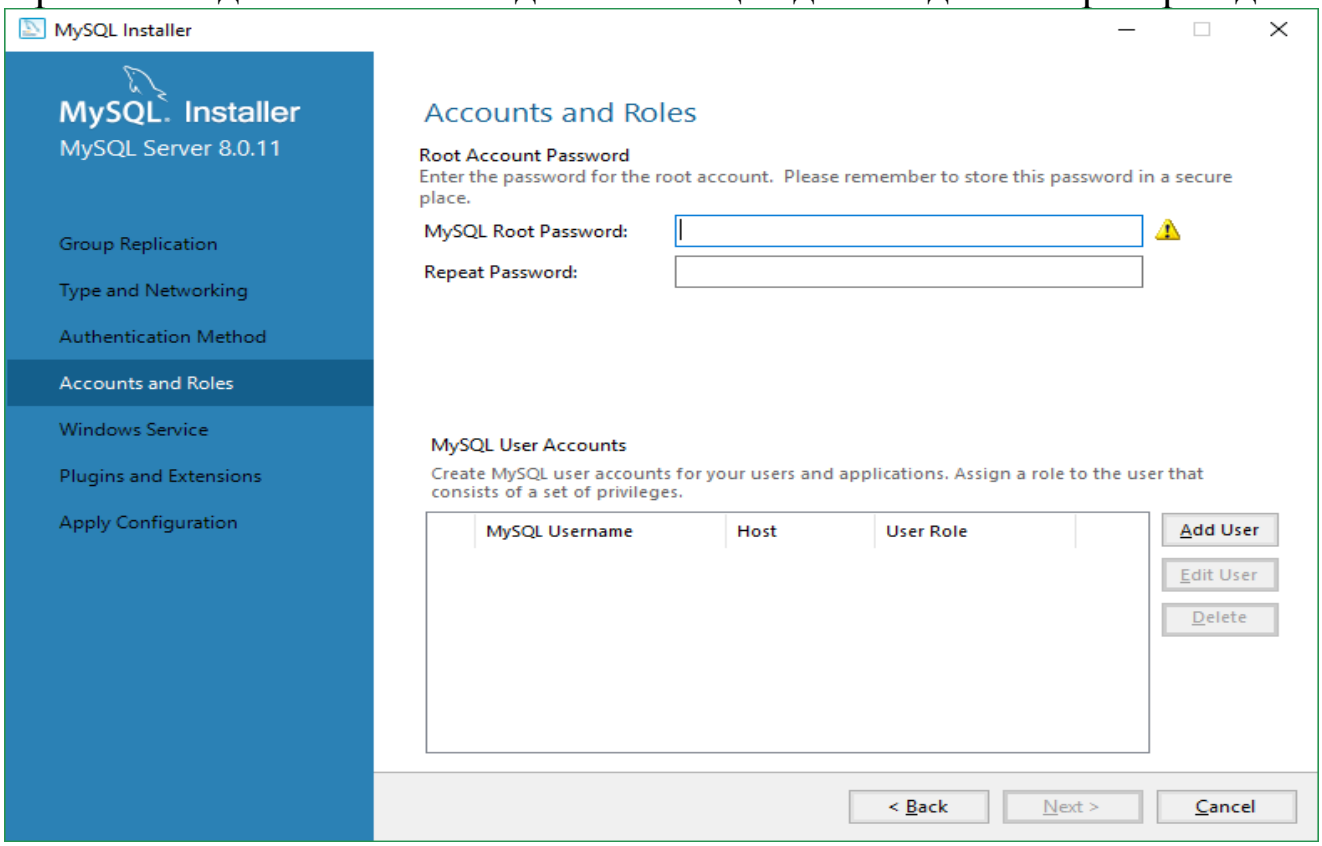


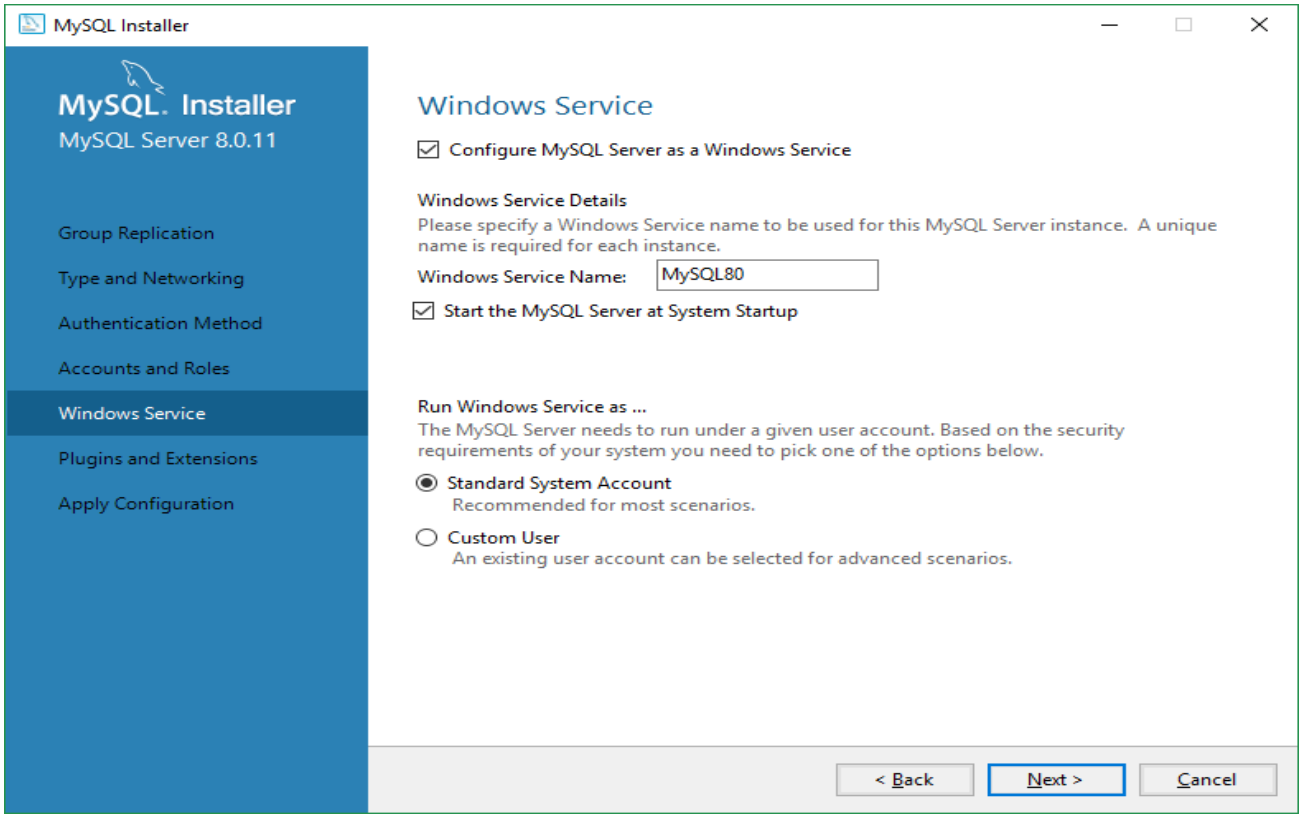

MySQL Installer
MySQL. Installer
MySQL Server 8.0.11
Group Replication
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Plugins and Extensions
Advanced Options
Apply Configuration
Group Replication
Standalone MySQL Server / Classic MySQL Replication
Choose this option if you want to run the MySQL Server either standalone with the opportunity to later configure classic MySQL Replication.
Using this option you can manually configure your replication setup and provide your own high availability solution if required.
Sandbox InnoDB Cluster Setup (for testing only)
The InnoDB cluster technology provides an out-of-the-box HA (high availability) solution for MySQL using Group Replication technology.
This option allows you to test an InnoDB cluster setup on your local computer using several MySQL Server sandbox instances. Read more about this here .
To setup a real-world production InnoDB cluster please choose the standard MySQL Server configuration instead on all desired hosts and use the MySQL Shell afterwards to create or expand the InnoDB cluster setup.
MySQL Shell
Client App
MySQL Router
InnoDB Cluster
Next >
Cancel


MySQL Installer
MySQL. Installer
MySQL Server 8.0.11
Group Replication
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Plugins and Extensions
Apply Configuration
Type and Networking
Server Configuration Type
Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.
Config Type: Development Computer
Connectivity
Use the following controls to select how you would like to connect to this server.
TCP/IP
Port Number: 3306
Open Windows Firewall port for network access
Named Pipe
Pipe Name: MYSQL
Shared Memory
Memory Name: MYSQL
Advanced Configuration
Select the check box below to get additional configuration page where you can set advanced options for this server instance.
Show Advanced Options
< Back
Next >
Cancel

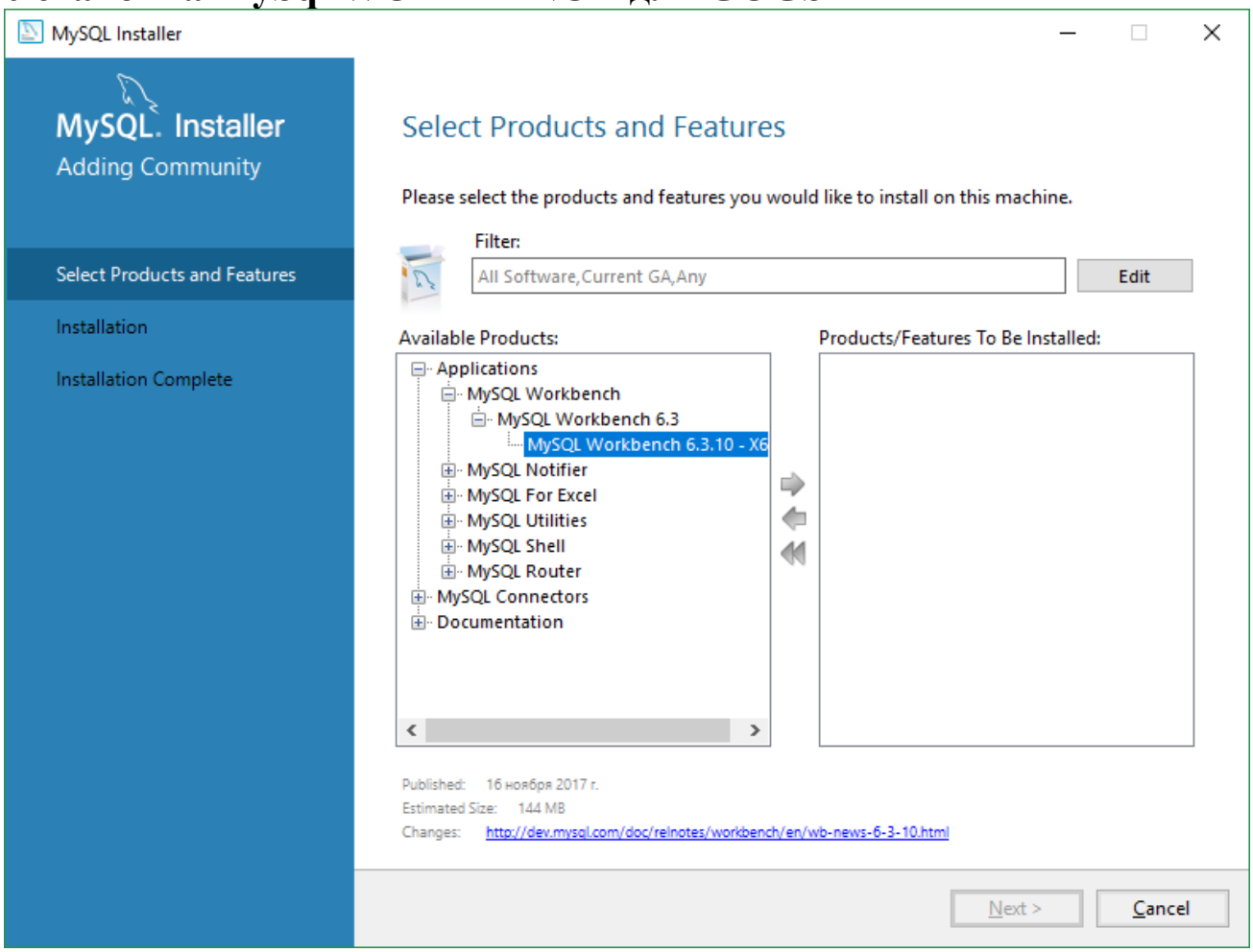


При необходимости можно добавить еще одного администратора бд

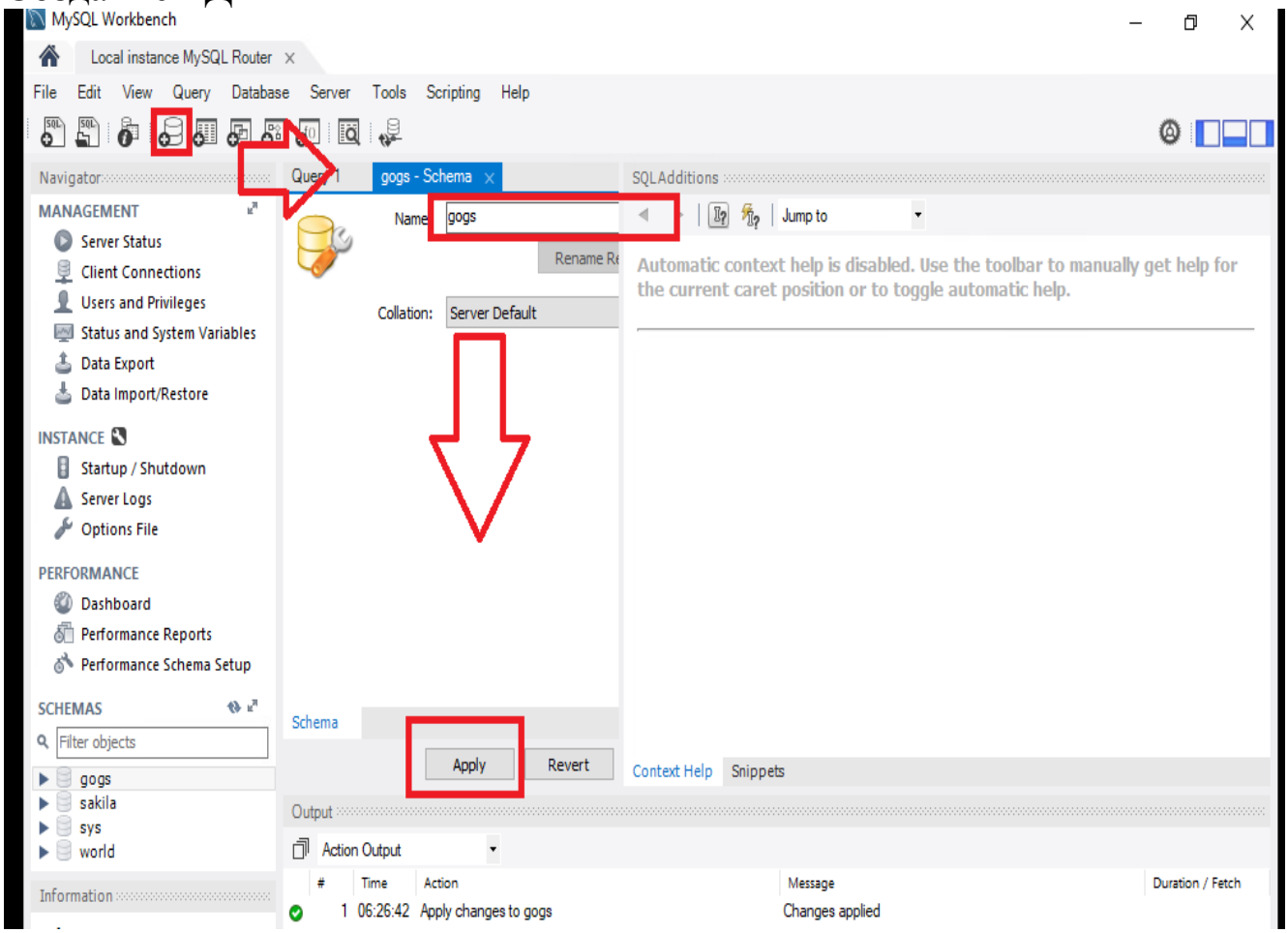




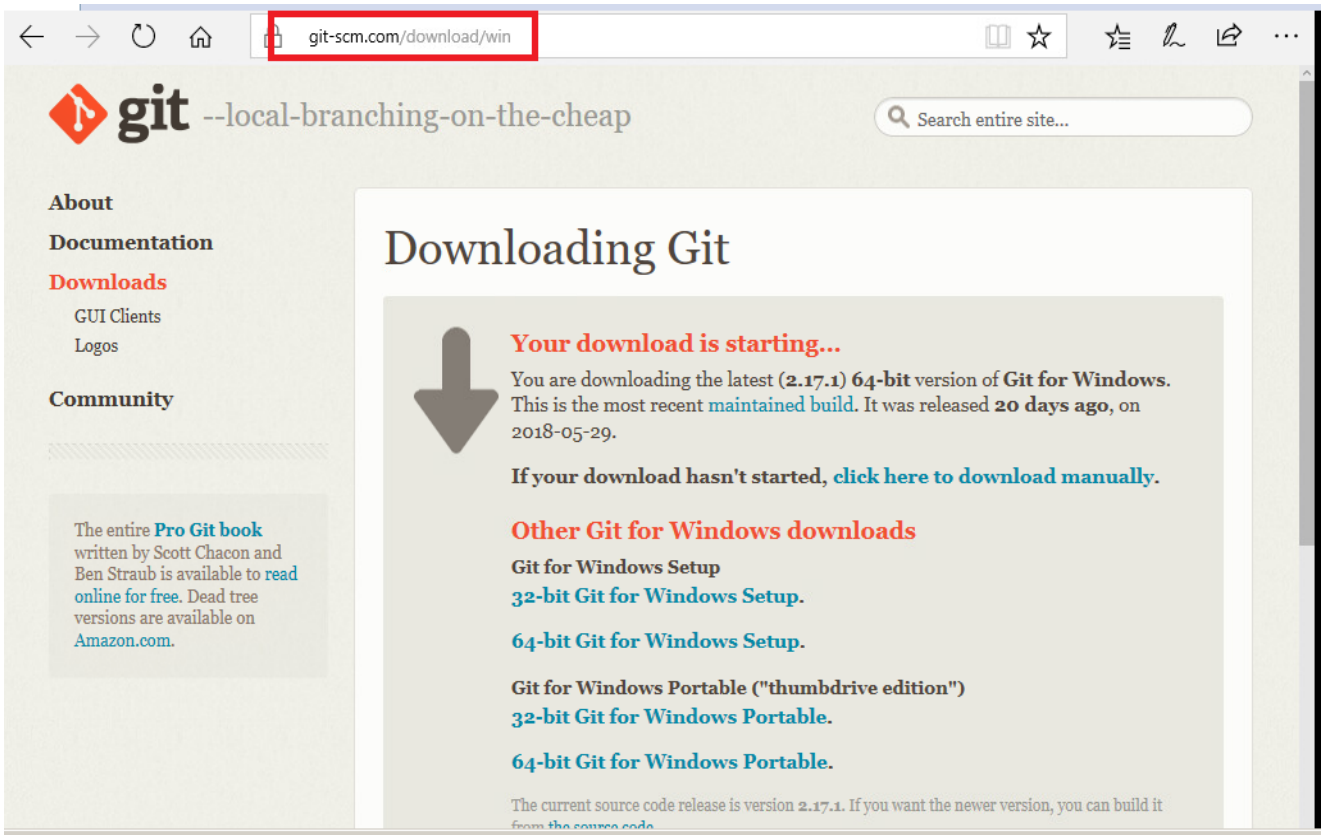
Установка MySql WORKBENCH для GOGS



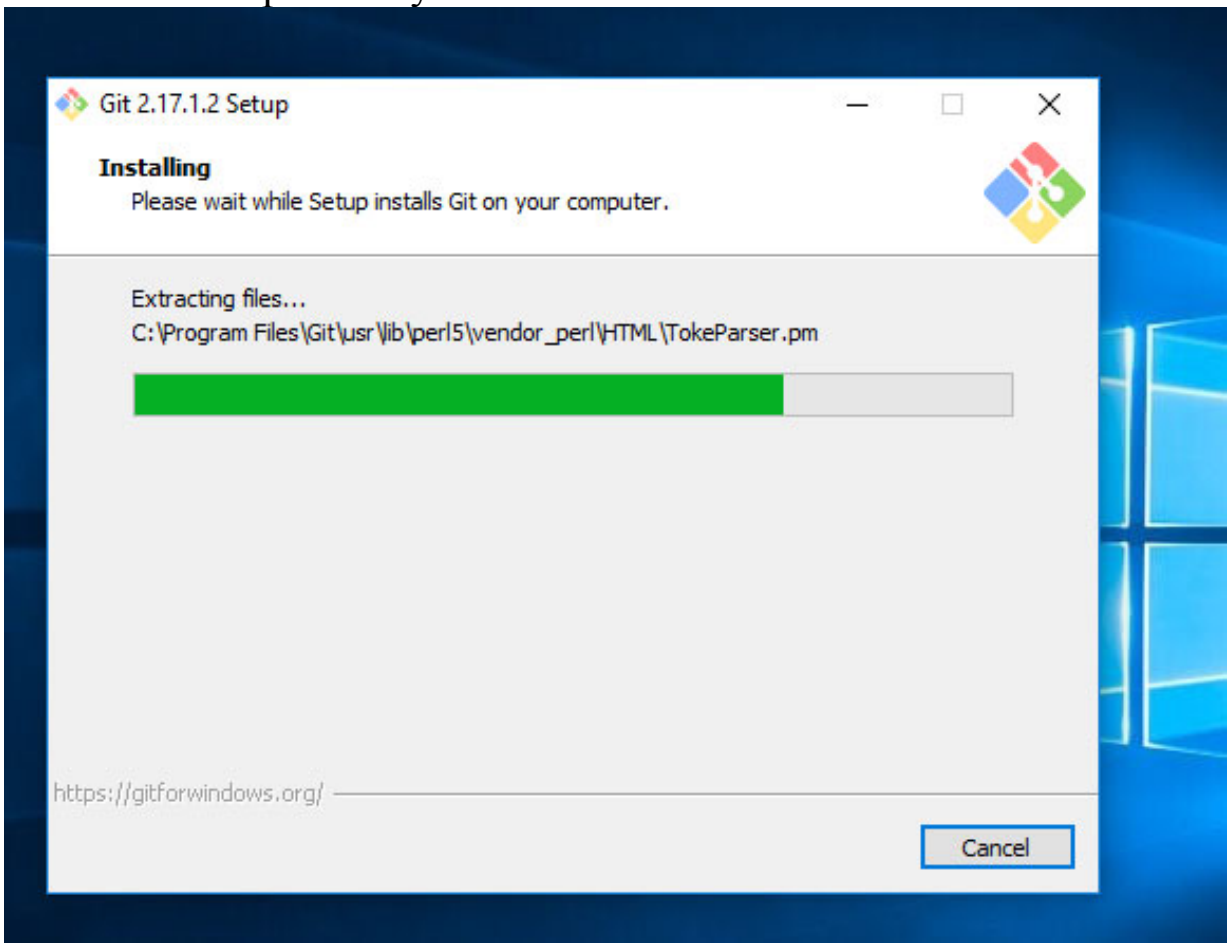
Создание БД



Установка SCM GIT

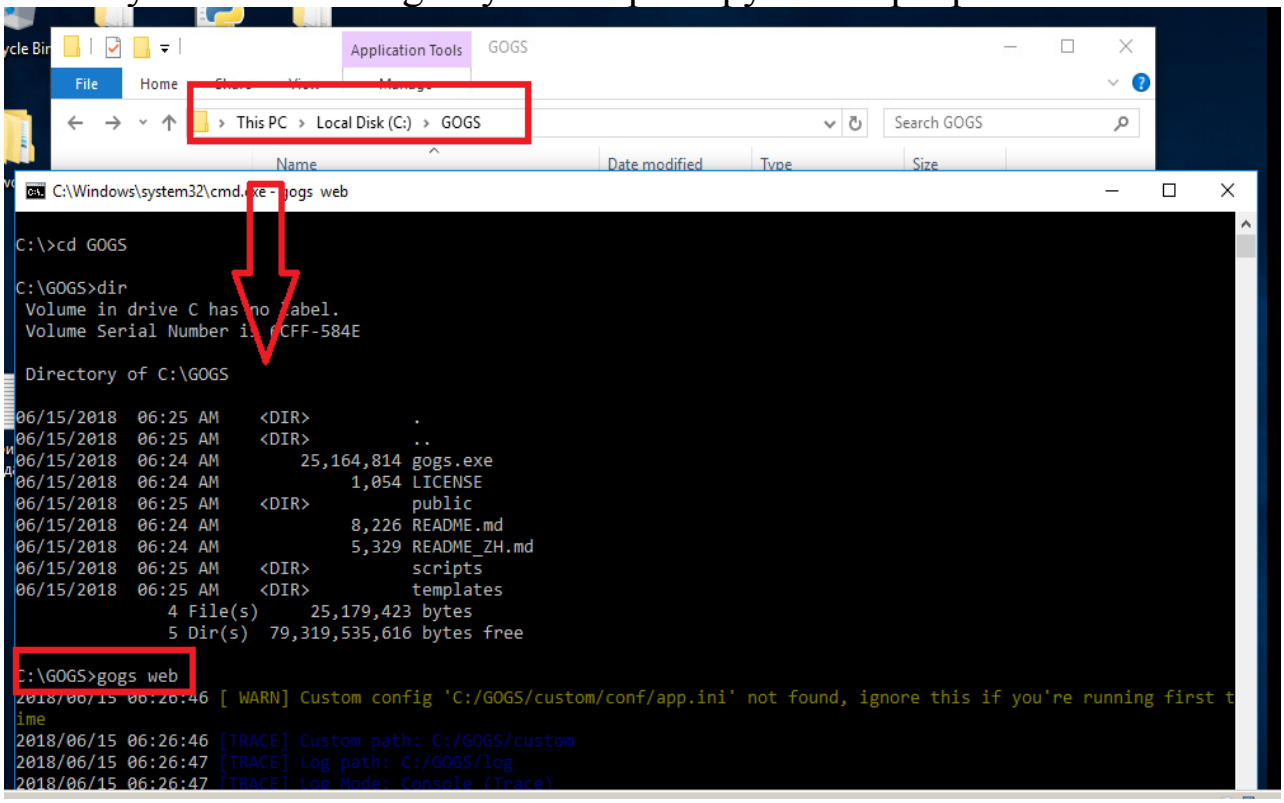


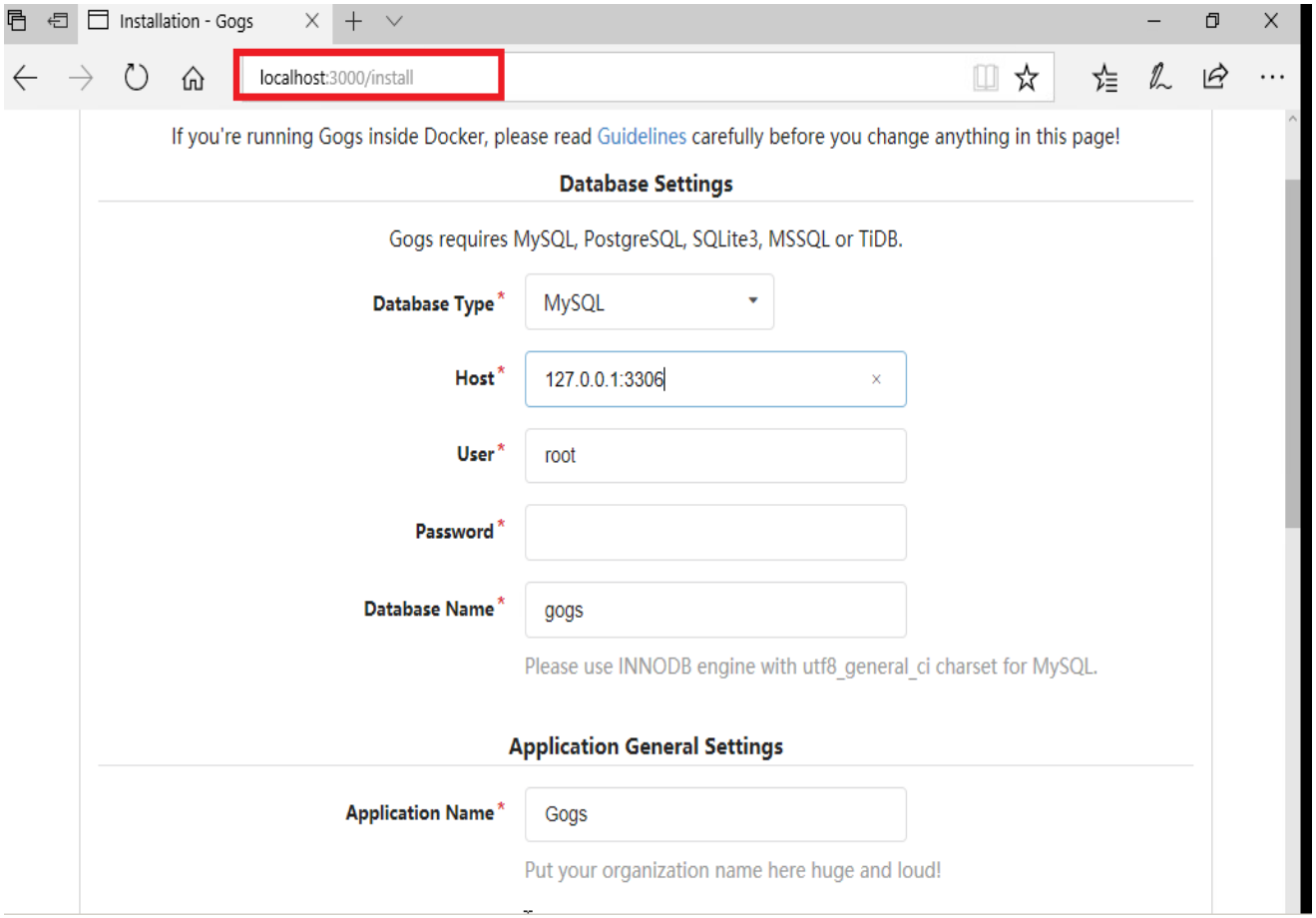
Не меняя настроек по умолчанию:

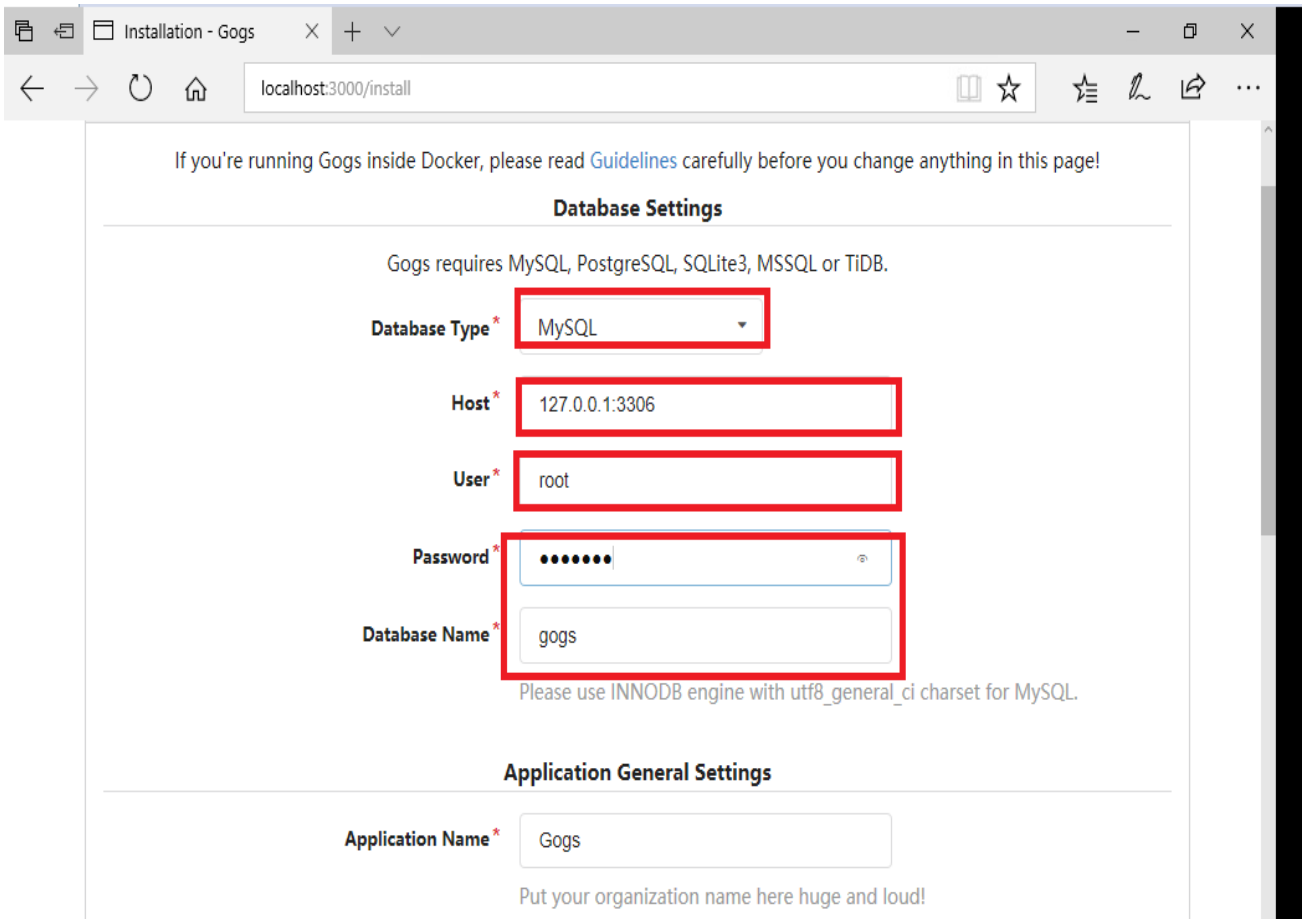


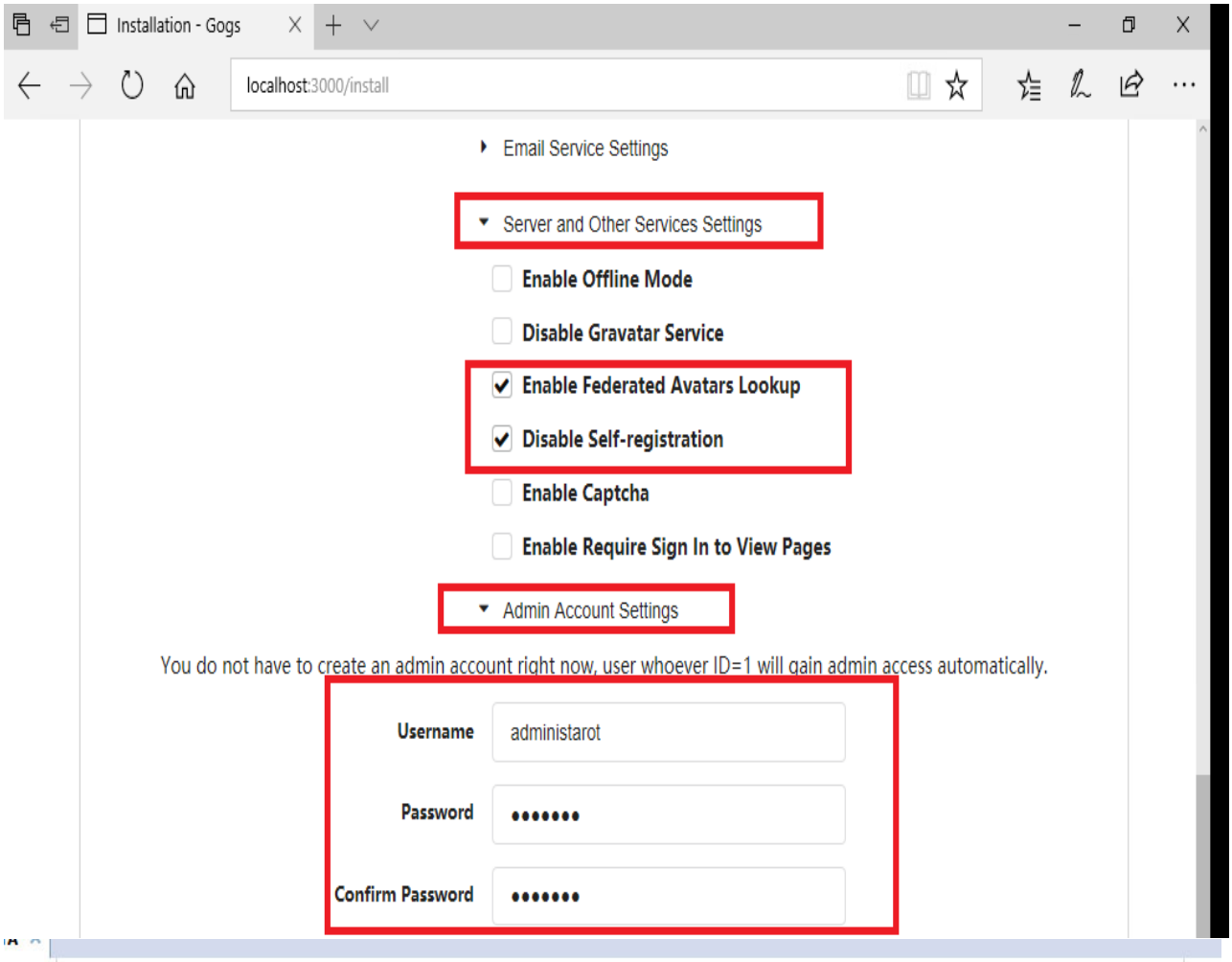
Установка gogs

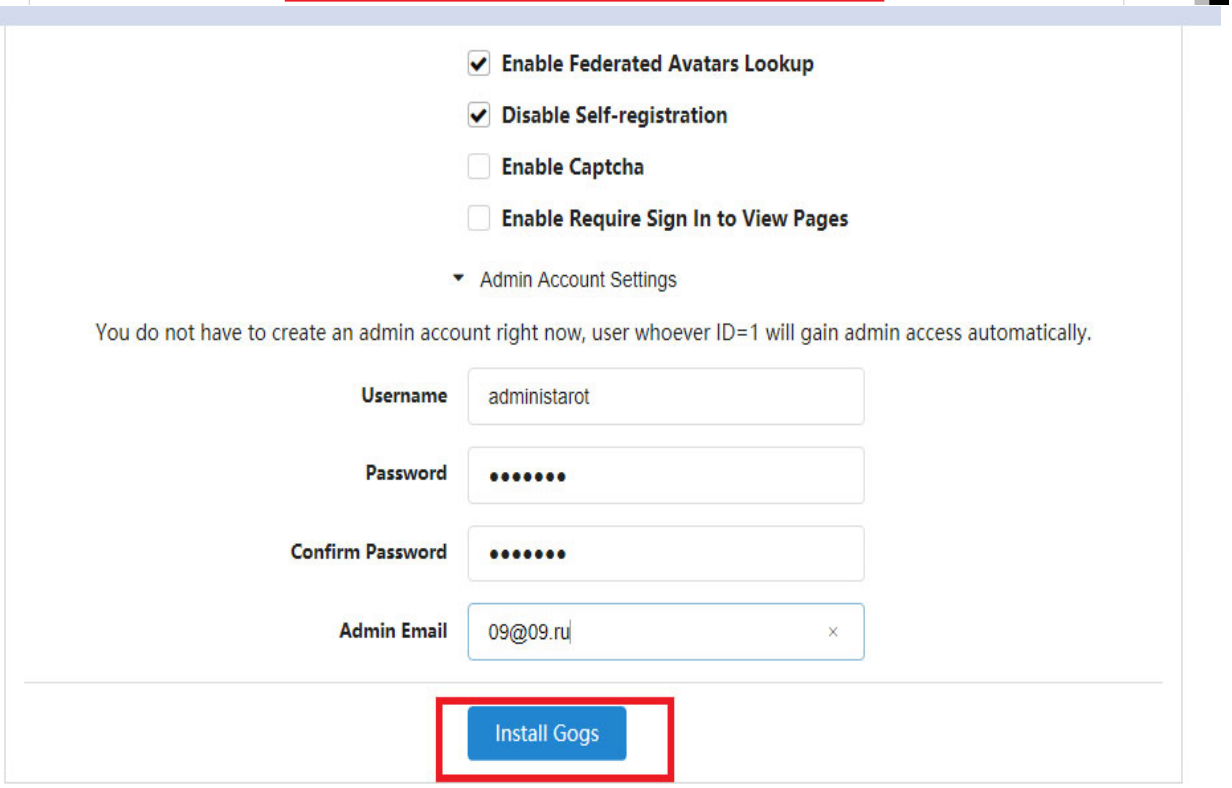
После установки scm git нужно перезагрузить сервер




Installation - Gogs
localhost:3000/install
If you're running Gogs inside Docker, please read Guidelines carefully before you change anything in this page!
Database Settings
Gogs requires MySQL, PostgreSQL, SQLite3, MSSQL or TiDB.
Database Type*
MySQL
Host*
127.0.0.1:3306
User*
root
Password*
Database Name*
gogs
Please use INNODB engine with utf8_general_ci charset for MySQL.
Application General Settings
Application Name*
Gogs
Put your organization name here huge and loud!

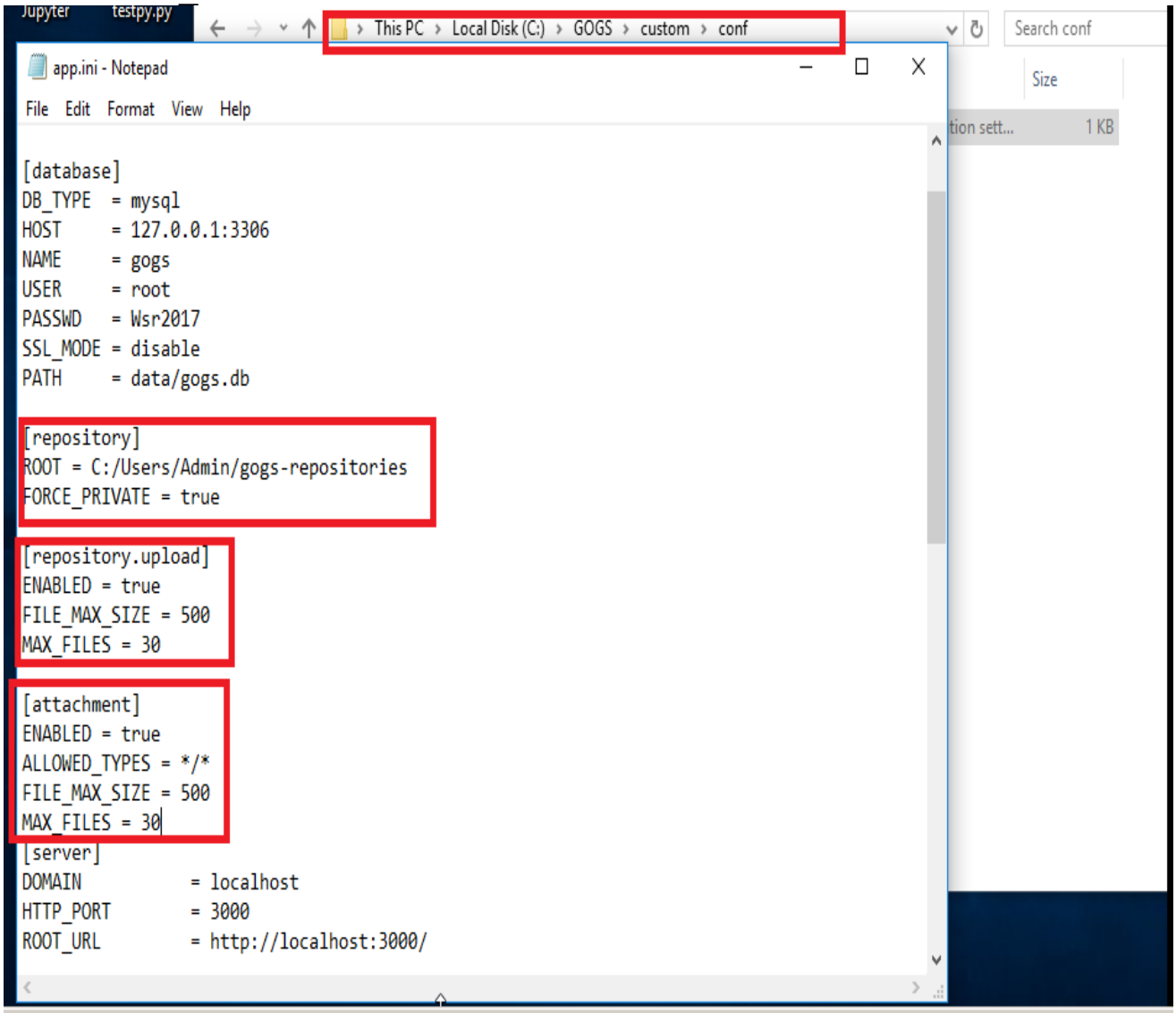

Installation - Gogs
localhost:3000/install
If you're running Gogs inside Docker, please read Guidelines carefully before you change anything in this page!
Database Settings
Gogs requires MySQL, PostgreSQL, SQLite3, MSSQL or TiDB.
Database Type*
MySQL
Host*
127.0.0.1:3306
User*
root
Password*
Database Name*
gogs
Please use INNODB engine with utf8_general_ci charset for MySQL.
Application General Settings
Application Name*
Gogs
Put your organization name here huge and loud!


Installation - Gogs
localhost:3000/install
Email Service Settings
Server and Other Services Settings
Enable Offline Mode
Disable Gravatar Service
Enable Federated Avatars Lookup
Disable Self-registration
Enable Captcha
Enable Require Sign In to View Pages
Admin Account Settings
You do not have to create an admin account right now, user whoever ID=1 will gain admin access automatically.
Username
administarot
Password
Confirm Password

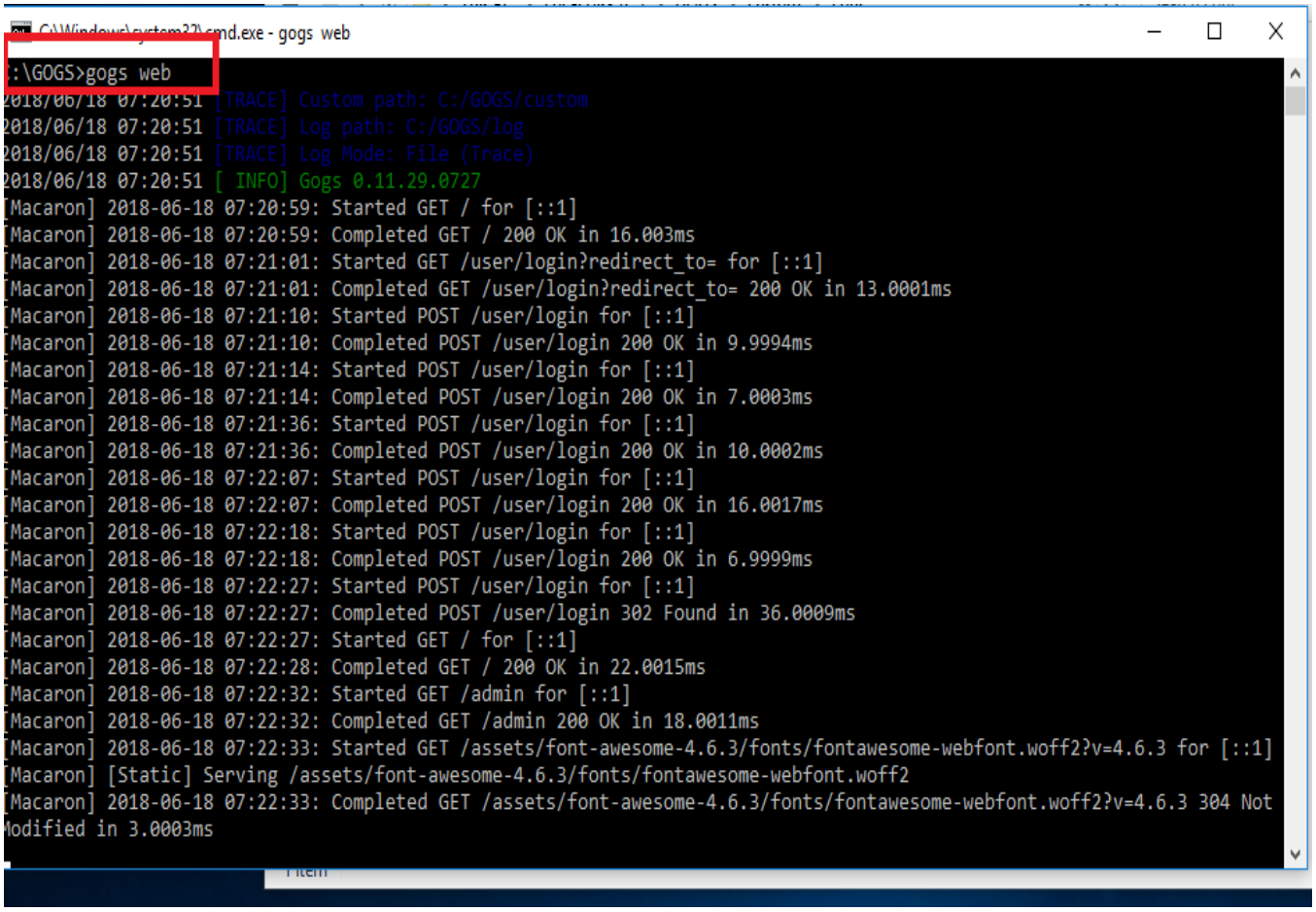

Enable Federated Avatars Lookup
Disable Self-registration
Enable Captcha
Enable Require Sign In to View Pages
Admin Account Settings
You do not have to create an admin account right now, user whoever ID=1 will gain admin access automatically.
Username
administarot
Password
Confirm Password
Admin Email
09@09.ru
Install Gogs

Затем открыть файл app.ini

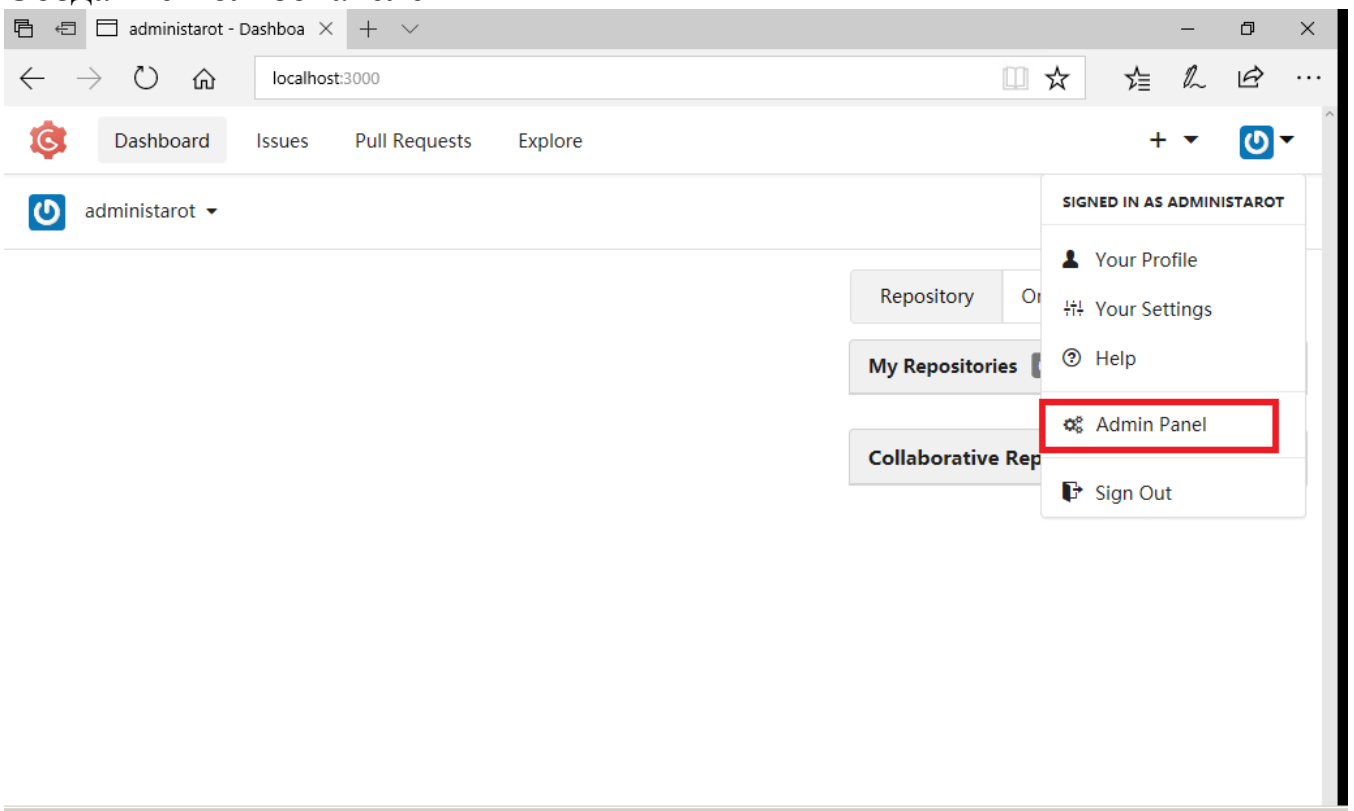
```
[repository]
FORCE_PRIVATE = true

[repository.upload]
ENABLED = true
FILE_MAX_SIZE = 500
MAX_FILES = 30

[attachment]
ENABLED = true
ALLOWED_TYPES = */*
MAX_SIZE = 500
MAX_FILES = 30
```

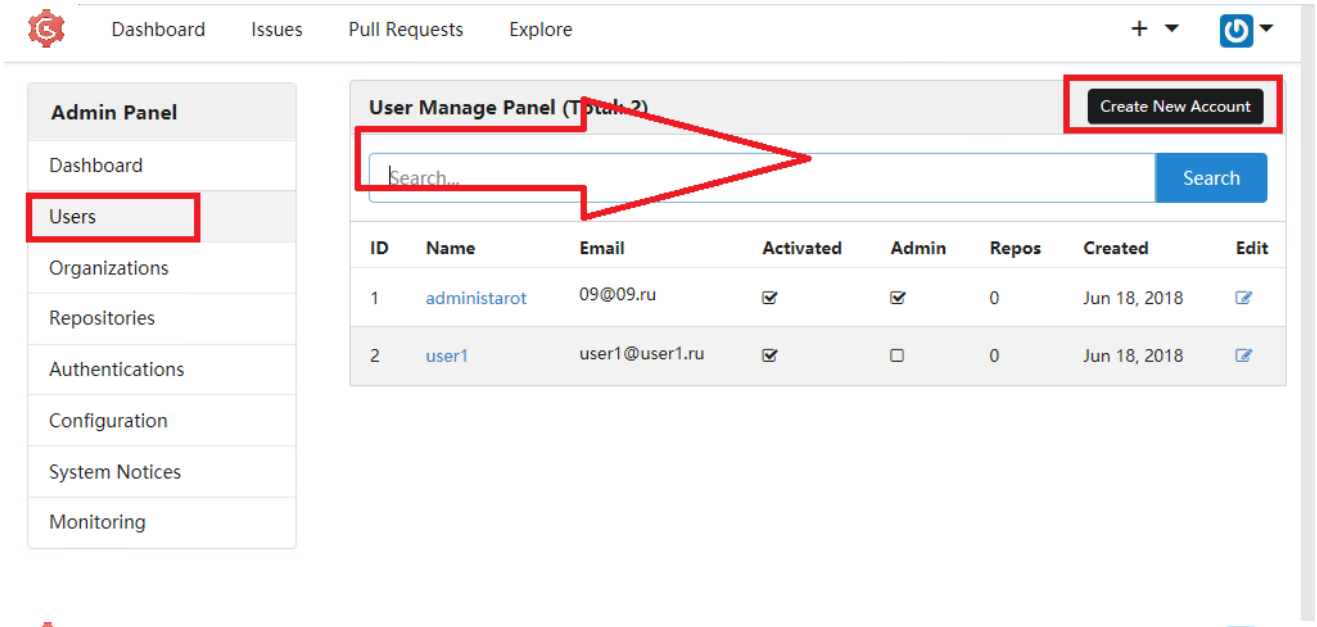


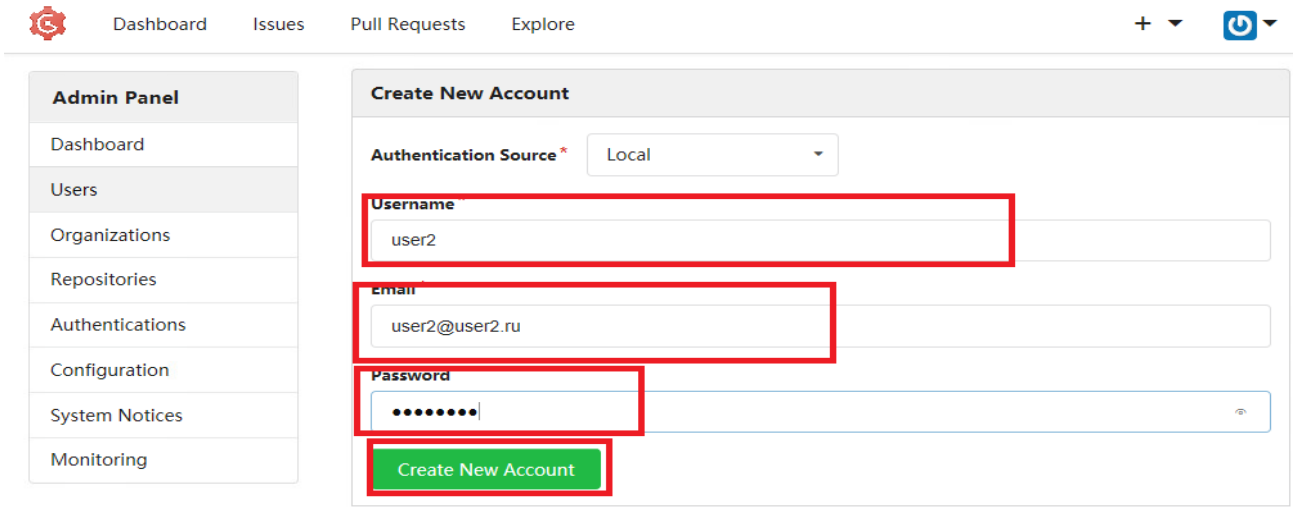
После чего необходимо перезапустить gogs

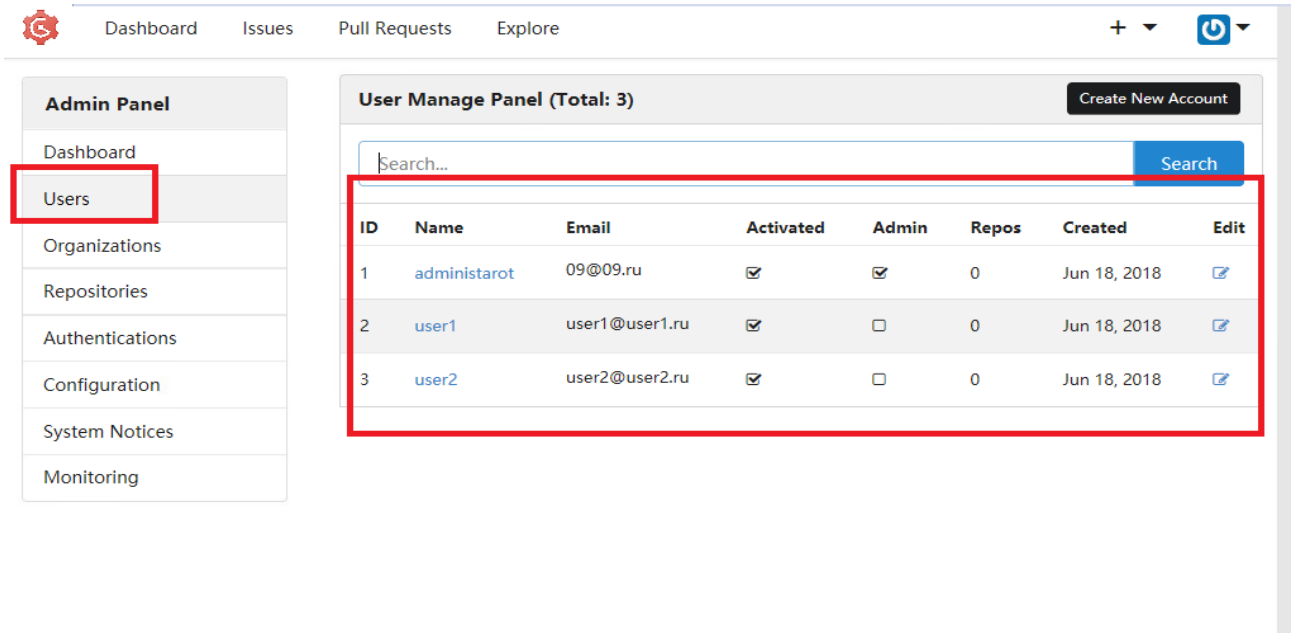


Создание пользователей

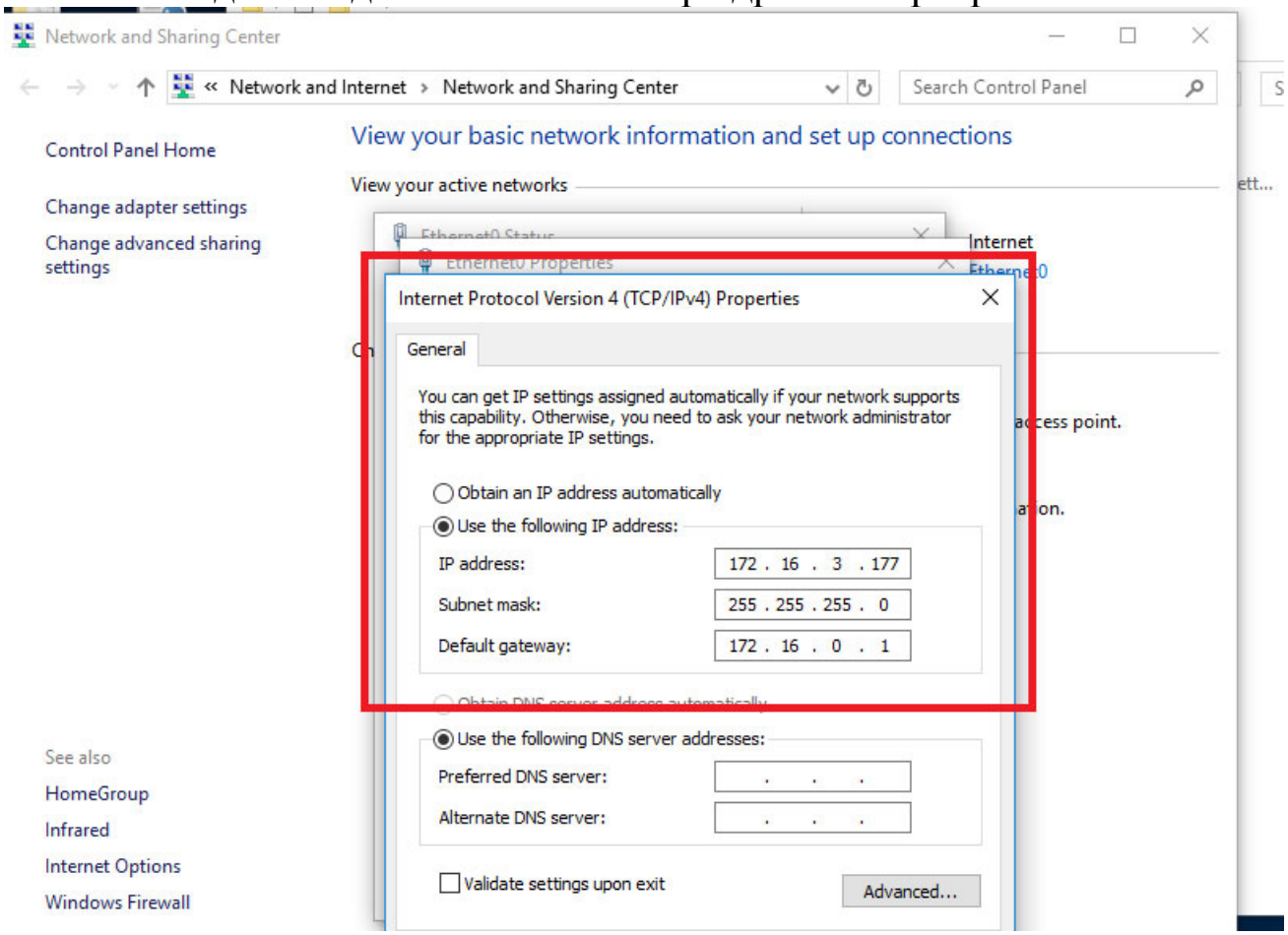



Dashboard
Issues
Pull Requests
Explore
Admin Panel
Dashboard
Users
Organizations
Repositories
Authentications
Configuration
System Notices
Monitoring
User Manage Panel (Total: 2)
Create New Account
Search...
Search
ID
Name
Email
Activated
Admin
Repos
Created
Edit
1
administarot
09@09.ru
0
Jun 18, 2018
2
user1
user1@user1.ru
0
Jun 18, 2018

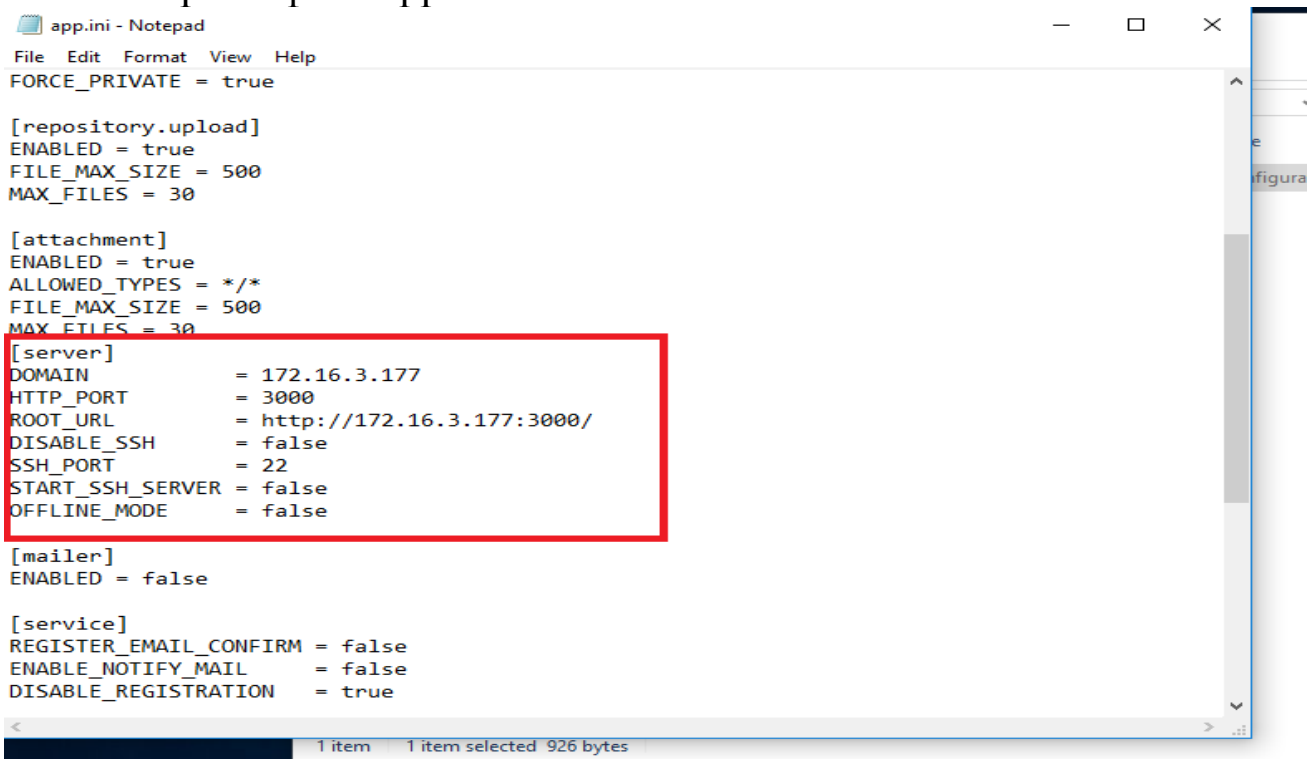

Dashboard
Issues
Pull Requests
Explore
Admin Panel
Dashboard
Users
Organizations
Repositories
Authentications
Configuration
System Notices
Monitoring
Create New Account
Authentication Source *
Local
Username
user2
Email *
user2@user2.ru
Password
Create New Account


Dashboard
Issues
Pull Requests
Explore
Admin Panel
Dashboard
Users
Organizations
Repositories
Authentications
Configuration
System Notices
Monitoring
User Manage Panel (Total: 3)
Create New Account
Search...
Search
ID
Name
Email
Activated
Admin
Repos
Created
Edit
1
administarot
09@09.ru
0
Jun 18, 2018
2
user1
user1@user1.ru
0
Jun 18, 2018
3
user2
user2@user2.ru
0
Jun 18, 2018

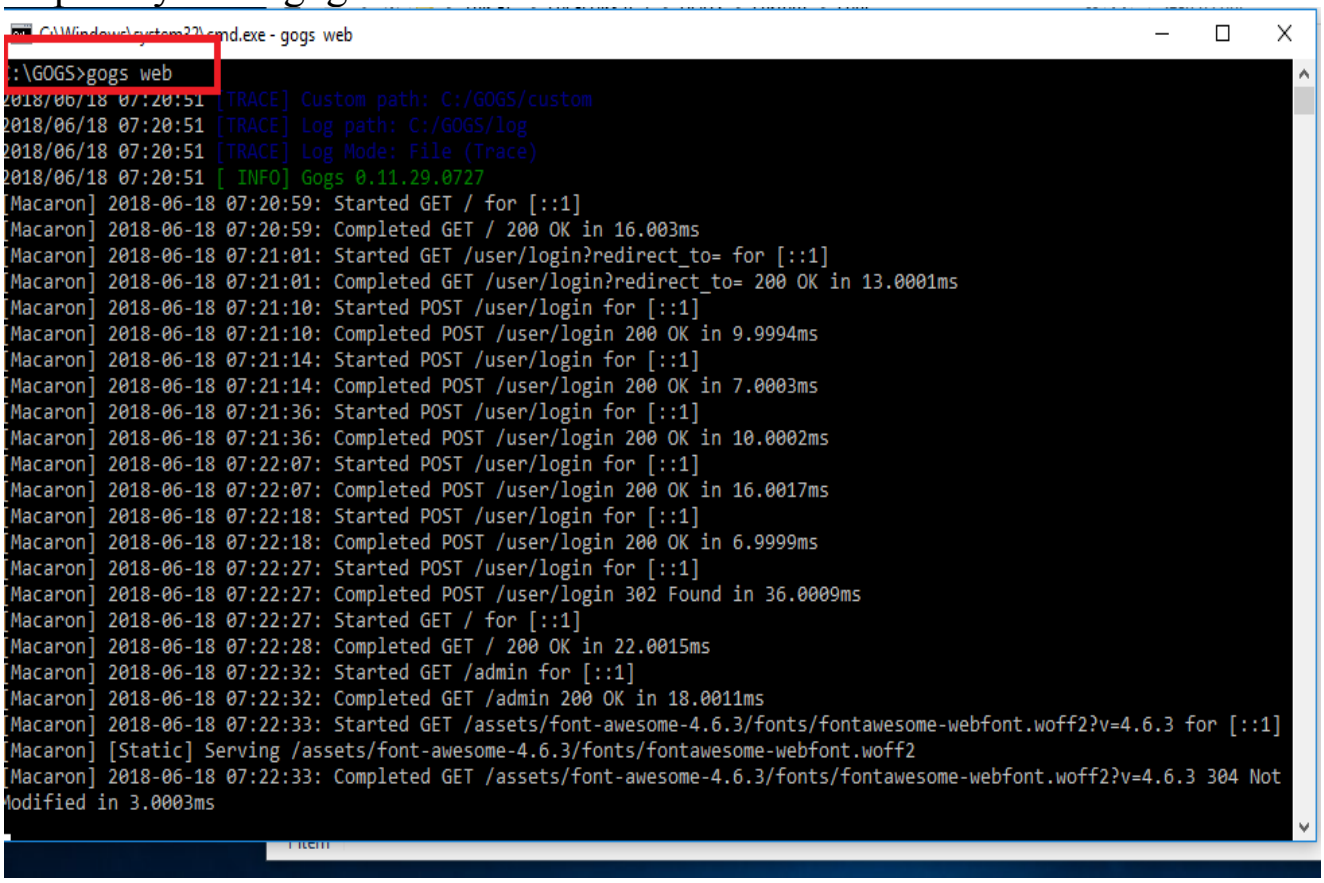
Настройка ip на сервере gogs
Вам необходимо задать статический ip адрес на сервере



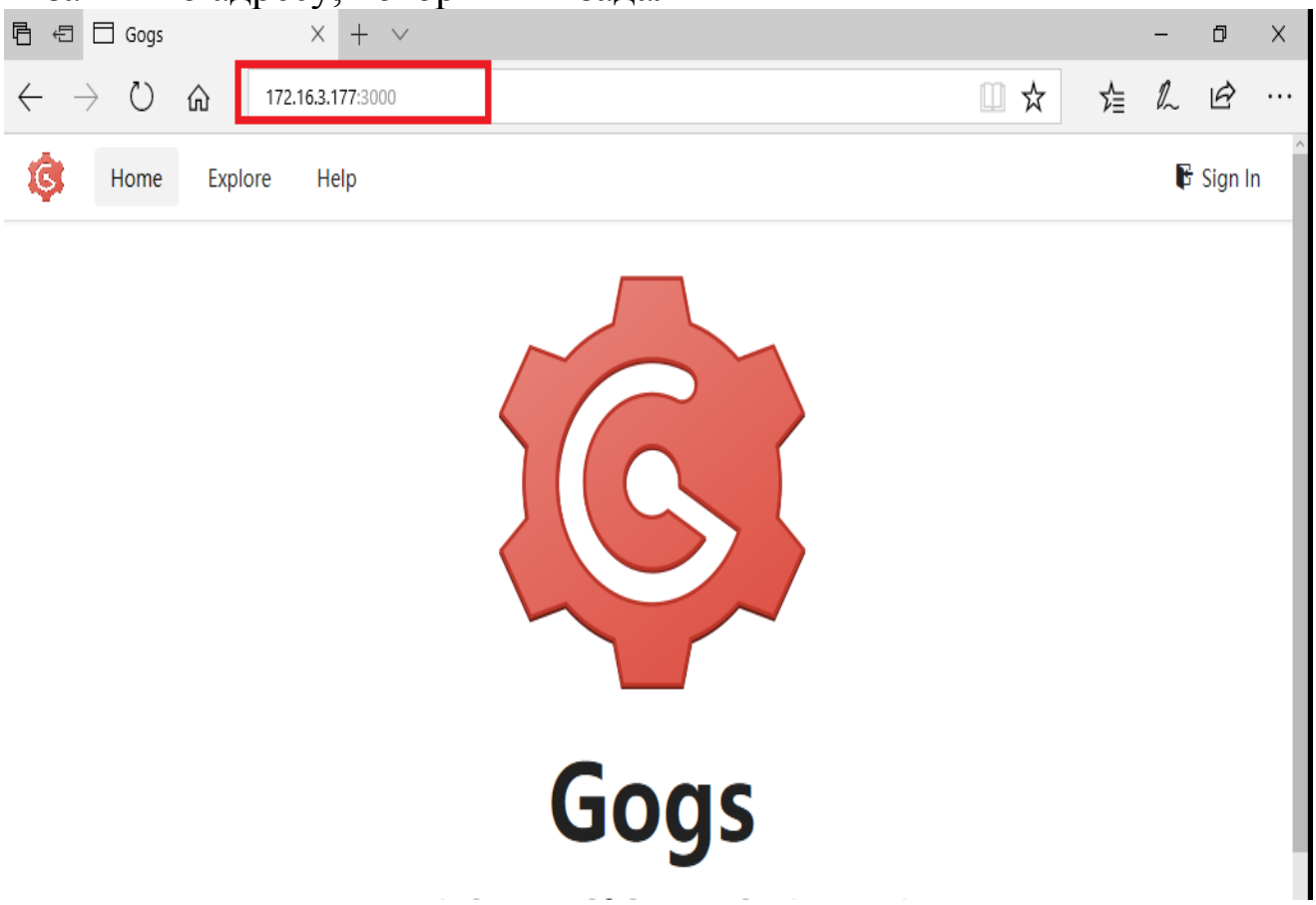
Затем открыть файл app.ini



Перезапустить gogs



И зайти по адресу, который вы задали



План застройка площадки по компетенции № 09 «Программные решения для бизнеса» (версионность ПО и количество рабочих мест должно определять инфраструктурным листом конкретного чемпионата или демонстрационного экзамена)

Подготовка комнаты экспертов (рабочих мест 5 шт.)

Раздел 1. Выполнение застройки

Этап 1.1. Установка столов в количестве 5 шт.

Этап 12. Расстановка стульев в количестве 5 шт.

Этап 1.3. Установка МФУ M5035 1 шт.

Этап 1.4. Установка компьютеров на столы 5 шт.

Этап 1.5. Подключение компьютеров к ЛВС и сети электропитания 5 шт.

Раздел 2. Настройка рабочих мест экспертов

Этап 2.1. Настройка управляемого коммутатора 1 шт.

Этап 2.2. Создание изолированной сети для экспертов.

Этап 2.3. Настройка сервера дубликации рабочих мест экспертов.

Этап 2.4. Установка 0S Windows 10

Этап 2.5. Установка и настройка ПО Eclipse IDE for java EE developers

Этап 2.6. Установка и настройка ПО Framework.net

Этап 2.7. Установка и настройка ПО JDk 8

Этап 2.8. Установка и настройка ПО Microsoft SQL Server 2014 Express

Этап 2.9. Установка и настройка ПО Microsoft Visio Professional 2013

Этап 2.10. Установка и настройка ПО Microsoft Visual Studio Ultimate 2013

Этап 2.11. Установка и настройка ПО MySQL Community Server

Этап 2.12. Установка и настройка ПО MySQL Connector/J

Этап 2.13. Установка и настройка ПО MySQL Connector/NET

Этап 2.14. Установка и настройка ПО MySQL Workbench

Этап 2.15. Установка и настройка ПО Netbeans

Этап 2.16. Установка и активация ПО Microsoft Office 2013

Этап 2.17. Установка и настройка ПО SQL Server Management Studio 2014 Express

Раздел 3. Сопровождение работы экспертных групп

Этап 3. 1. Разворачивание дубликатов программных проектов рабочих мест участников каждой сессии демонстрационного экзамена

Этап 3.2. Разворачивание дубликатов баз данных рабочих мест участников каждой сессии демонстрационного экзамена

Этап 3.3. Оперативное решение проблем, критических для работоспособности площадки, возникающих при работе экспертных групп

Подготовка комнаты участников (рабочих мест 15 шт.)

Раздел 4. Выполнение застройки

Этап 4.1. Установка столов в количестве 15 шт.

Этап 4.2. Расстановка стульев в количестве 15 шт.

Этап 4.3. Установка компьютеров на столы 15 шт.

Этап 4.4. Установка управляемого коммутатора 1 шт.

Этап 4.5. Подключение компьютеров к ЛВС и сети электропитания 15 шт.

Этап 4.6. Установка веб-камеры 1 шт.

Раздел 5. Настройка рабочих мест участников

Этап 5.1. Настройка управляемого коммутатора 1 шт.

Этап 5.2. Создание изолированной сети для участников.

Этап 5.3. Настройка сервера дубликации рабочих мест участников.

Этап 5.4. Установка 0S Windows 10

Этап 5.5. Установка и настройка ПО Eclipse IDE for java EE developers

Этап 5.6. Установка и настройка ПО Framework.net

Этап 5.7. Установка и настройка ПО JDk 8

Этап 5.8. Установка и настройка ПО MicrosoR SQL Server 2014 Express

Этап 5.9. Установка и активация ПО Microsoft Office 2013

Этап 5.10. Установка и настройка ПО Microsoft Visio Professional 2013 Этап

5.11. Установка и настройка ПО Microsoft Visual Studio Ultimate 2013

Этап 5.12. Установка и настройка ПО MySQL Community Server

Этап 5.13. Установка и настройка ПО MySQL Connector/J

Этап 5.14. Установка и настройка ПО MySQL Connector/NET Этап 5.15. Установка и настройка ПО MySQL Workbench

Этап 5.16. Установка и настройка ПО Netbeans

Этап 5.17. Установка и настройка ПО SQL Server Management Studio
2014 Express
Этап 5.18. Установка драйверов и настройка веб-камеры
Этап 5.19. Авторизация канала на youtube.com для обеспечения трансляции
Этап 5.20. Создание дубликатов рабочих мест участников на каждую сессию
Этап 5.21. Настройка и сопровождение таймера
Этап 5.22. Изготовление патч-кордов больше 3 метров 4 шт.
Этап 5.23. Замена комплектующих.

Раздел 6. Сопровождение работы участников

Этап 6.1. Разворачивание дубликатов рабочих мест участников каждой группы демонстрационного экзамена
Этап 6.2. Оперативное решение проблем, критических для работоспособности площадки, возникающих при работе участников
Этап 6.3. Ресурсообеспечение деятельности участников (в части расходных материалов)
Этап 6.4. Инструктаж участников по охране труда и технике безопасности при работе с ПЭВМ
Этап 6.5. Проведение жеребьевки участников
Этап 9.6. Инструктаж участников по работе с системой версионного контроля программного обеспечения

Методика организации и проведения демонстрационного экзамена по компетенции «Программные решения для бизнеса»

При подготовке к демострационному экзамену (далее – ДЭ) следует руководствоваться актуальной документацией https://www.esat.worldskills.ru/.
Основные направления деятельности при подготовке демонстрационного экзамена:

1. Выбрать компетенцию для сдачи ДЭ на основе анализа комплектов оценочных материалов, разработанных Союзом «Молодые профессионалы (Ворлдскиллс Россия)» (далее – Союз);
2. Определить форму участия в пилотной апробации: ДЭ в рамках промежуточной аттестации или в рамках ГИА;

3. Определить центр приема ДЭ (далее – ЦПДЭ) для участия (на базе своей образовательной организации или на базе другого ЦПДЭ региона);
4. Подготовить пакет документов от образовательной организации согласно Методики организации и проведения демонстрационного экзамена по стандартам Ворлдскиллс Россия, утвержденной Союзом (далее – Методика).

В случае получения статуса ЦПДЭ важно соблюдать сроки подачи документов и получения электронного аттестата ЦПДЭ, установленные Методикой. Для получения статуса ЦПДЭ в образовательной организаци должно быть оборудование, соответствующее утвержденному инфраструктурному листу компетенции. При подаче документов образовательная организация должна разработать индивидуальный инфраструктурный лист, указав точные технические характеристики всех ключевых позиций. Технические характеристики не могут быть ниже заявленных в утвержденном инфраструктурном листе.

При формировании плана застроки необходимо учесть минимальные требования к застройке, указанными в техническом описании компетенции, - площадка проведения ДЭ, комната экспертов для проверки работ, серверная. Проверка работ конкурсантов производится удаленно. Проверка работ на рабочих местах участников строго запрещена.

Для организации работы участников обязательно развертывание сервера баз данных и системы контроля версий. Использование локальных баз данных конкурсантами запрещено. Проверка работ экспертами производится на копиях серверов, чтобы не мешать работе участников.

При проведении демонтсрационного экзамена необходимо ориентироваться на SMP, утвержденный главным экспертом на форуме forum.worldskills.ru.

Задание ДЭ является секретным, распространения информации о задании ДЭ строго запрещено в соответствие с Кодексом этики.

Программа подготовки обучающихся к демонстрационному экзамену

WSSS	Наименование	Навык
Анализ и проектирование программных решений	UML	Диаграмма прецедентов
		Описание прецедентов
		Описание действующих субъектов
		Диаграмма вариантов использования
		Объект класса
		Диаграммы класса домена
		Схемы последовательности
		Схемы взаимодействия
		Диаграмма состояний
		Диаграмма деятельности
		Диаграмма сущностей и связей
		Нормализации
		Словарь данных
	MVC	Создание формы
		Создание Модели
		Создание контроллера
	Шаблоны проектиро-вания	Шаблон одиночка
		Шаблон конечный автомат
Разработка программных решений	Руководство по стилю	Создание базовой формы
		Использование базовой формы
	Управление каталогами и файлами	Создание каталога
		Создание файла
		Чтение из файла
		Запись в файл
	Управления версиями	Создание репозитория
		Подключение удаленного репозитория
		Получение изменений
		Коммит
	Подключение к БД	Создание подключения к MySQL через EF5
		Создание подключения к MySQL

		Создание подключения к MSSQL через EF5
		Создание подключения к MSSQL
		Выгрузка данных через EF5
		Выгрузка данных через класс
		Добавление данных через EF5
		Обновление данных через класс
		Обновление данных через EF5
		Добавление данных через класс
		Загрузка изображений
		Выгрузка изображений
	Библиотеки и фреймворки	EntityFramework
	Разработка мобильного интерфейса на основе серверной системы	Создание Activity
		Переход между Activity
		Доступ к данным через REST API
		Создание списка
		Обработчик нажатия кнопки
Тестирование программных решений	Тестирование	Модульное тестирование
		Объемное испытания
		Интеграционное тестирование
		Приемочные испытания
	Документирование	Составление плана тестирования
		Разработка тест-кейсов
		Проверка результатов тест-кейсов
		Составление отчета о процессе тестирования
Документирование программных решений	Документирование	Документация пользователя
		Техническая документация
		Презентация

Задание:

1) Составить план подготовки участников ДЭ на основании предложенного шаблона (https://drive.google.com/drive/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-1kn);
2) Подготовить задание для демонстрационного экзамена на основании предложенного шаблона в соответствие с представленной предметной областью.

Итоговая аттестация слушателей: выполнение задания демонстрационного экзамена по компетенции «Программные решения для бизнеса».