Национальные проекты
Образование

worldskills Russia

РАБОЧАЯ ТЕТРАДЬ

Программа повышения квалификации преподавателей (мастеров производственного обучения)

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

ФИО слушателя

_____

академия

МОСКВА 2020

WORLDSKILLS RUSSIA

«КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА №11»

ЦИКТ ЦЕНТР ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ ГАПОУ КП № 11

Оглавление

МОДУЛЬ 1. СТАНДАРТ ВОРЛДСКИЛЛС И СПЕЦИФИКАЦИЯ СТАНДАРТОВ ВОРЛДСКИЛЛС ПО КОМПЕТЕНЦИИ «ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ БИЗНЕСА». РАЗДЕЛЫ СПЕЦИФИКАЦИИ....6

ТИПОВОЕ КОНКУРСНОЕ ЗАДАНИЕ....28

ФОРМЫ УЧАСТИЯ В КОНКУРСЕ - Индивидуальный конкурс....28

ЗАДАНИЕ ДЛЯ КОНКУРСА....28

Конкурсное задание регионального чемпионата является СЕКРЕТНЫМ (согласно технического описания)....28

Главный эксперт чемпионата получает задание в С-2 по e-mail и проводит ознакомление экспертов с заданием, ресурсами и схемой оценки в течение 1,5-2 часов с последующим оформлением протокола ознакомления. Во время ознакомления запрещено использование любых средств фиксации задания (телефоны, фотоаппараты, блокноты с ручками и т.д.)....28

Конкурсанты должны быть ознакомлены с ОПИСАНИЕМ ПРЕДМЕТНОЙ ОБЛАСТИ И ГИДОМ ПО СТИЛЮ к конкурсному заданию в день С-1 в течение 1-2 часов с последующим оформлением протокола ознакомления....28

Ознакомление конкурсантов с заданием производится только по сессиям в течение 15 минут перед началом соответствующей сессии совместно с экспертом-компатриотом. На момент ознакомления мониторы конкурсантов должны быть выключены. Вести записи конкурсантам или экспертам во время ознакомления запрещено. 28

Содержанием конкурсного задания является разработка информационной системы для определенной предметной области. Участники соревнований получают описание предметной области, необходимые информационные ресурсы для работы с базой данных. Конкурсное задание состоит из нескольких сессий, выполняемых последовательно. 29

Конкурс включает в себя анализ предоставленных данных и проектирование информационной системы, разработку программного обеспечения, тестирование программного обеспечения, разработку сопровождающей документации по разработанной информационной системе, представление разработанного решения заказчику. 29

Аспекты критериев оценки так же являются секретными, ознакомление экспертов с аспектами критериев оценки производится до начала чемпионата (см.выше). Ознакомление конкурсантов с аспектами критериев оценки ЗАПРЕЩЕНО. Оценка производится как по результатам выполнения сессий, так и в отношении процесса выполнения конкурсного задания. 29

Если участник конкурса не выполняет требования техники безопасности, подвергает опасности себя или других конкурсантов, такой участник может быть отстранен от конкурса. 29

Конкурсное задание должно выполняться по сессиям. Оценка также происходит по сессиям. Изменение последовательности сессий или объема задания по сессиям ЗАПРЕЩЕНО. 29

МОДУЛЬ 2. СОВРЕМЕННЫЕ ТЕХНОЛОГИИ В ПРОФЕССИОНАЛЬНОЙ СФЕРЕ 33

МОДУЛЬ 3. ОСОБЕННОСТИ ОБУЧЕНИЯ В СООТВЕТСТВИИ СО СТАНДАРТАМИ ВОРЛДСКИЛЛС И СПЕЦИФИКАЦИЕЙ СТАНДАРТОВ ВОРЛДСКИЛЛС ПО КОМПЕТЕНЦИИ «ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ БИЗНЕСА» 38

Особенности обучения в соответствии со стандартами Ворлдскиллс и спецификацией стандартов Ворлдскиллс по компетенции 40

МОДУЛЬ 4. КУЛЬТУРА БЕЗОПАСНОГО ТРУДА. ОСНОВЫ БЕЗОПАСНОГО ТРУДА И ЭФФЕКТИВНАЯ ОРГАНИЗАЦИЯ РАБОЧЕГО МЕСТА В СООТВЕТСТВИИ СО СТАНДАРТАМИ ВОРЛДСКИЛЛС И СПЕЦИФИКАЦИЕЙ СТАНДАРТОВ ВОРЛДСКИЛЛС ПО КОМПЕТЕНЦИИ 68

Культура безопасного труда 68

МОДУЛЬ 5. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗ ДАННЫХ 78

Создание и редактирование модели данных 79

- Добавление и редактирование таблицы 79
- Управление индексами 80
- Связи между таблицами 80

Наполнение таблицы базовыми данными....81
Создание ER диаграммы (диаграммы "сущность-связь")....82
Импорт существующей схемы данных (из SQL дампа)....83
МОДУЛЬ 6. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ....100
Инструментальные средства для анализа и проектирования программных решений.....100
Платформы для разработки программных решений....104
МОДУЛЬ 7. ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ....155
Модель работы с дефектами....168
ПРОЕКТИРОВАНИЕ СОДЕРЖАНИЯ УЧЕБНО-ПРОИЗВОДСТВЕННОГО ПРОЦЕССА С УЧЕТОМ СПЕЦИФИКАЦИИ СТАНДАРТА ВОРЛДСКИЛЛС....182
ОРГАНИЗАЦИЯ И ПРОВЕДЕНИЕ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА ПО СТАНДАРТАМ ВОРЛДСКИЛЛС РОССИЯ. ОЦЕНКА КВАЛИФИКАЦИИ СТУДЕНТА (ВЫПУСКНИКА) В ХОДЕ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА.....193
Методика организации и проведения демонстрационного экзамена по компетенции «Программные решения для бизнеса»....193

НАЗНАЧЕНИЕ РАБОЧЕЙ ТЕТРАДИ

Рабочая тетрадь предназначена как для работы слушателей во время занятия, так и для самостоятельной работы слушателей. Изложение материала в рабочей тетради соответствует утвержденной рабочей программе курса.

Практические задания и дополнительные материалы по компетенции размещены на дополнительном ресурсе – Google Drive и системе дистанционного обучения.

Рабочая тетрадь предназначена для обучения на базе Государственного автономного профессионального образовательного учреждения города Москвы «Колледж предпринимательства № 11» - Национальный специализированный центр по компетенции «Программные решения для бизнеса».

Ведущие преподаватели программы:

- **сертифицированный эксперт Ворлдскиллс Россия по компетенции «Программные решения для бизнеса» Кривоносова Наталья Викторовна;**
- **эксперт с правом проведения регионального чемпионата по компетенции «Программные решения для бизнеса» Макиевский Станислав Евгеньевич.**

Перед обучением рекомендовано ознакомиться со следующими информационными ресурсами:

- **сайт Союза Ворлдскиллс Россия (https://worldskills.ru);**
- **сайт организации, реализующей программу - (https://www.kp11.ru/);**
- **дистанционный курс Академии Ворлдскиллс Россия «Эксперт демонстрационного экзамена по стандартам Ворлдскиллс» (https://worldskillsacademy.ru/#/programs/8/competences-expert);**
- **дистанционный курс Академии Ворлдскиллс Россия «Навигатор по Future Skills» (https://worldskillsacademy.ru/#/programs/10/competences-expert);**
- **сайт Национальной сборной России по профессиональному мастерству (https://nationalteam.worldskills.ru);**
- **группа Академии Ворлдскиллс Россия в Facebook (https://www.facebook.com/groups/1904334706509962/?ref=share);**
- **сетевой ресурс организации для онлайн- и дистанционного обучения, содержащий материалы по данной программе – (https://academy.kp11.ru/).**

МОДУЛЬ 1. СТАНДАРТ ВОРЛДСКИЛЛС И СПЕЦИФИКАЦИЯ СТАНДАРТОВ ВОРЛДСКИЛЛС ПО КОМПЕТЕНЦИИ «ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ БИЗНЕСА». РАЗДЕЛЫ СПЕЦИФИКАЦИИ

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:

Использовать навыки грамотности для:

- **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.**
- **Использовать навыки устного общения для:**
- **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.**

Использовать навыки письменного общения для:

- **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.**

Использовать коммуникационные навыки при работе в команде для:

- **сотрудничества с другими специалистами для получения желаемых результатов;**
- **успешной работы над групповым решением проблем.**

Использовать навыки управления проектами в:

- **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: **Использовать аналитические навыки для:** • **синтезирования сложной или неоднородной информации;** • **определения функциональных и нефункциональных требований спецификации.** **Использовать навыки исследования и обучения для:** • **понимания пользовательских требований (например, результатов опросов, анкет, поиска и анализа документов, объединенной разработки приложений и наблюдений);** • **независимого исследования возникших проблем.** **Использовать навыки решения проблем для:** • **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: **Анализировать системы с помощью:** • **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).** **Проектировать системы на основе:** • **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**

Задание: проведите анализ спецификаций WSSS. На основе анализа составьте таблицу, в которой отразите примеры задач, которые могут быть включены в конкурсное задание чемпионата или демонстрационного экзамена.

Навык WSSS	Задача для конкурсного задания
внедрять внутрифирменный стандарт (руководство по стилю) для всей системы	**При разработке ПО учитывать шрифт, логотип и правила расположения объектов в интерфейсе**

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**

Задание: приведите примеры отраслевых стандартов по компетенции:

Отраслевой стандарт	Описание отраслевого стандарта, применение при выполнении КЗ

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%.

ЗАДАНИЕ: Опишите методику оценивания приведенных субкритериев и аспектов:

	Субкритерий	Тип аспекта	Аспект	Методика оценивания
B1	Восстановление базы данных из скрипта			
		O	База данных восстановлена с помощью предоставленных скриптов	
B2	Разработка - Список услуг			
		O	Реализован переход в режим администратора с вводом кода	
		O	Обычный режим предоставляет доступ к функциям в соответствии с заданием	
		O	Режим администратора предоставляет расширенный доступ к функциям в соответствии с заданием	
		O	Список услуг загружается из БД	
		O	Список соответствует макету	
		O	Выводятся все необходимые данные об услуге	

		O	Если у услуги снижена стоимость, то услуга подсвечивается зеленым фоном	
		O	Если у услуги снижена стоимость, то основная цена перечеркнута и рядом с ней указана итоговая стоимость	
		O	Вычисляется итоговая стоимость с учетом скидки	
		O	Около каждой услуги должно отображаться основное изображение	
		O	Список услуг можно отсортировать по убыванию стоимости	Не учитывается стандартная сортировка от DataGrid по щелчку на заголовке
		O	Список услуг можно отсортировать по возрастанию стоимости	Не учитывается стандартная сортировка от DataGrid по щелчку на заголовке
		O	Выпадающий список для фильтрации корректно заполнен	
		O	Есть возможность отфильтровать данные по размеру скидки	Диапазоны фильтрации работают правильно
		O	Есть возможность сбросить фильтрацию (показать все)	
		O	Фильтрация работает в реальном времени	

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.

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.

ТИПОВОЕ КОНКУРСНОЕ ЗАДАНИЕ

ФОРМЫ УЧАСТИЯ В КОНКУРСЕ - Индивидуальный конкурс.

ЗАДАНИЕ ДЛЯ КОНКУРСА

Конкурсное задание регионального чемпионата является **СЕКРЕТНЫМ** (согласно технического описания).

Главный эксперт чемпионата получает задание в С-2 по e-mail и проводит ознакомление экспертов с заданием, ресурсами и схемой оценки в течение 1,5-2 часов с последующим оформлением протокола ознакомления. Во время ознакомления запрещено использование любых средств фиксации задания (телефоны, фотоаппараты, блокноты с ручками и т.д.).

Конкурсанты должны быть ознакомлены с **ОПИСАНИЕМ ПРЕДМЕТНОЙ ОБЛАСТИ И ГИДОМ ПО СТИЛЮ** к конкурсному заданию в день С-1 в течение 1-2 часов с последующим оформлением протокола ознакомления.

Ознакомление конкурсантов с заданием производится только по сессиям в течение 15 минут перед началом соответствующей сессии совместно с экспертом-компатриотом. На момент ознакомления мониторы конкурсантов должны быть выключены.

Вести записи конкурсантам или экспертам во время ознакомления запрещено.

Содержанием конкурсного задания является разработка информационной системы для определенной предметной области. Участники соревнований получают описание предметной области, необходимые информационные ресурсы для работы с базой данных. Конкурсное задание состоит из нескольких сессий, выполняемых последовательно.

Конкурс включает в себя анализ предоставленных данных и проектирование информационной системы, разработку программного обеспечения, тестирование программного обеспечения, разработку сопровождающей документации по разработанной информационной системе, представление разработанного решения заказчику.

Аспекты критериев оценки так же являются секретными, ознакомление экспертов с аспектами критериев оценки производится до начала чемпионата (см.выше). Ознакомление конкурсантов с аспектами критериев оценки ЗАПРЕЩЕНО. Оценка производится как по результатам выполнения сессий, так и в отношении процесса выполнения конкурсного задания.

Если участник конкурса не выполняет требования техники безопасности, подвергает опасности себя или других конкурсантов, такой участник может быть отстранен от конкурса.

Конкурсное задание должно выполняться по сессиям. Оценка также происходит по сессиям. Изменение последовательности сессий или объема задания по сессиям ЗАПРЕЩЕНО.

МОДУЛИ ЗАДАНИЯ И НЕОБХОДИМОЕ ВРЕМЯ

Модули и время (примерное) сведены в таблице 1.

Таблица 1.

№ п/п	Наименование модуля	Рабочее время (по конкурсным дням)	Время на выполнение
1	Модуль 1: Проектирование структуры данных	C1	120 минут
2	Модуль 2: Импорт данных	C1	10 минут
3	Модуль 3: Программирование	C1, C2	320 минут
4	Модуль 4: Реализация отчетов	C2	20 минут
5	Модуль 5: Проектирование архитектуры	C1	30 минут
6	Модуль 6: Тестирование	C3	30 минут
7	Модуль 7: Разработка мобильного приложения	C2,C3	90 минут
8	Модуль 10: Документация	C3	30 минут
9	Модуль 11: Общий профессионализм решения	C1, C2, C3	60 минут

ОПИСАНИЕ МОДУЛЕЙ КОНКУРСНОГО ЗАДАНИЯ

Модуль 1: Проектирование структуры данных

Анализ исходных файлов данных, спроектировать на их основе структуру данных.

Модуль 2: Импорт данных

Приведение исходных файлов данных к виду, подходящему для импорта. Импортировать данные в базу данных.

Модуль 3: Программирование

Создание настольного приложения, различных окон, таблиц, форм для заполнения, чтение и запись в базу данных. Работа с API

Модуль 4: Реализация отчетов

Разработка и реализация отчетов, необходимых пользователям приложений, с графиками и возможностью вывода на печать.

Модуль 5: Проектирование архитектуры

Создание ERD на основе анализа предоставленных документов, проектирование архитектуры программного продукта

Модуль 6: Тестирование

Интеграционное тестирование, модульное тестирование. Разработка тест-кейсов

Модуль 7: Разработка мобильного приложения

Разработка мобильного приложения под ОС Android(API).

Модуль 10: Документация

Создание пакета сопровождающей документации по разрабатываемой информационной системе.

Модуль 11: Общий профессионализм решения

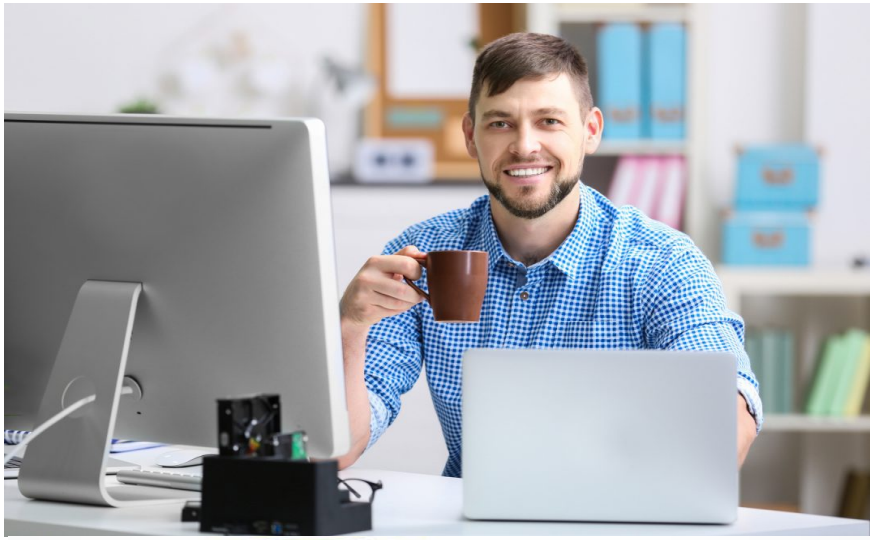
В общем профессионализме решения учитывается возможность развития информационной системы другими разработчиками, соответствие руководству по стилю заказчика, обратная связь системы с пользователем, стабильная работа всех разработанных программ, стиль кода на протяжении разработки всей системы.

Проверь себя!

1. **Какой тип тулбокса в компетенции?**
2. **Какие позитии в типовом ИЛ могут быть убраны для регионального/вузовского чемпионата?**
3. **Как ведется работа с КЗ на чемпионате/на ДЭ?**

МОДУЛЬ 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.

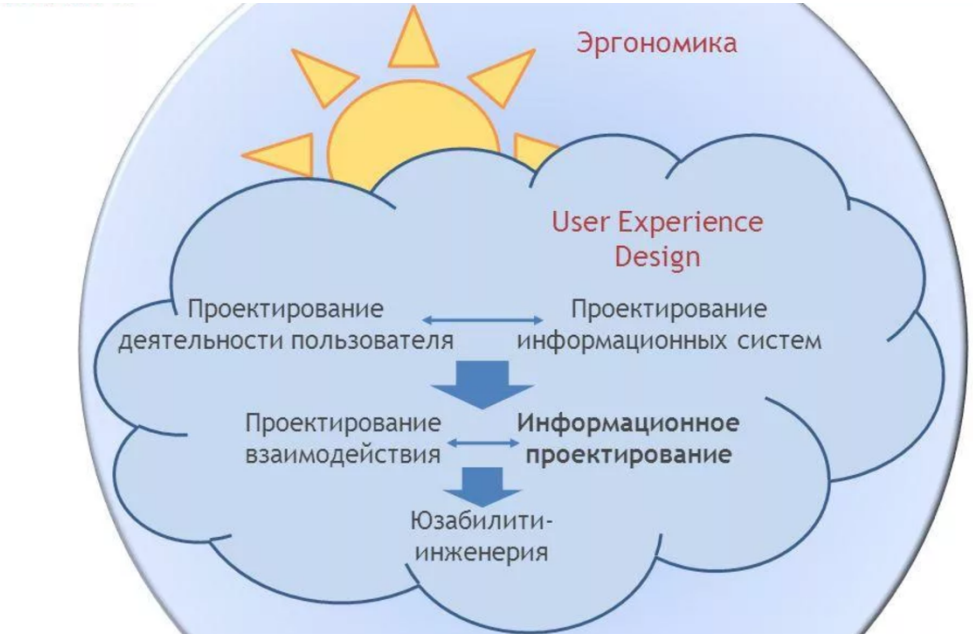
Изучением организационных, инженерных и технических аспектов создания ПО, включая методы разработки, занимается дисциплина, называемая программной инженерией.

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).

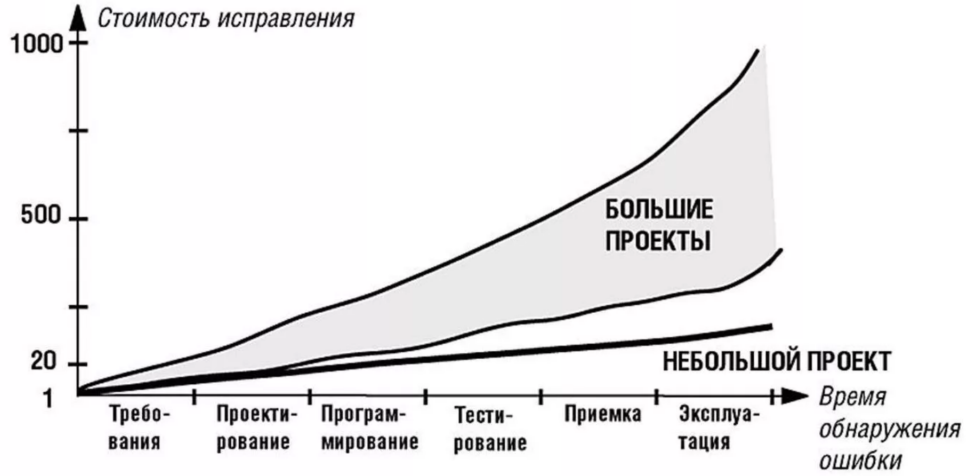
Для этого нужно учесть множество аспектов. Доходы от продаж значительно снизятся, если многие из пользователей не смогут воспользоваться программой только потому, что в их компьютерах процессоры слишком медленные или мало оперативной памяти, или потому что данные к системе часто поступают в искаженном 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.

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

Проверь себя!

1. **Какие технологии разработки ПО сейчас наиболее актуальны?**
2. **Какие требования предъявляются к современному ПО?**
3. **Поясните устно разработанную вами в задании выше схему основных принципов разработки сложных систем.**

МОДУЛЬ 3. ОСОБЕННОСТИ ОБУЧЕНИЯ В СООТВЕТСТВИИ СО СТАНДАРТАМИ ВОРЛДСКИЛЛС И СПЕЦИФИКАЦИЕЙ СТАНДАРТОВ ВОРЛДСКИЛЛС ПО КОМПЕТЕНЦИИ «ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ БИЗНЕСА»

Актуализация образовательных программ с учетом стандартов компетенции

Основная особенность ФГОС СПО в том, что они интегрируют систему требований к результатам освоения образовательных программ, берут за основу стандарты WorldSkills. Например, все ФГОС СПО (ТОП-50) предполагают проведение демонстрационного экзамена (ДЭ) по стандартам WorldSkills. результаты которого учитываются работодателями, имеющими соглашение с Союзом «Молодые профессионалы (Ворлдскиллс Россия)» (далее - Союз) о признании результатов ДЭ. Также ФГОС имеет серьёзные отличия в структуре. Так, сам стандарт теперь содержит лишь рамочные условия реализации образовательной программы, её детализация отображена в Примерной программе, выполнение которой обязательно, поскольку стандарт ссылается на неё.

Образовательный стандарт (ФГОС) только регулирует содержание и структуру основных образовательных программ, т.е. он предписывает, как должна быть с точки зрения государства устроена образовательная программа, описывает ее структуру.

Профессиональный стандарт – это стандарт, определяющий в конкретной области профессиональной деятельности требования к уровню компетенции и квалификации, к содержанию, качеству и условиям труда.

В образовательных стандартах указываются компетенции, которые должны быть сформированы у выпускника, для вариативной и обязательной части, а также какими образовательными средствами должна быть сформулирована та или иная компетенция.

В профессиональных стандартах сформулированы требования работодателей к взаимодействию системы образования и рынка труда. Их объединяет то, что профессиональные стандарты описывают требования к качеству и содержанию труда в определенной области профессиональной деятельности.

При освоении профессиональных стандартов необходимо задуматься о новых методах и технологиях обучения в СПО специалистов среднего звена. Модернизация профессионального образования в СПО позволит повысить качество образования.

Модернизация образования должна выполняться, как минимум, в трех направлениях: содержание образования, технологии образования и мотивация.

Содержание образования	Технологии обучения	Мотивация
• Разработка образовательных программ на основе ФГОС, ПС и WSSS • Гибкость и адаптивность ППССЗ	• Внедрение стандартов WS в учебный процесс (УП, ЭК); • Реализация практико-ориентированной модели обучения (ЛПЗ – 50%); • Реализация принципов дуального образования; • Сближение образовательной деятельности и производства (ПП по ПМ).	• Построение индивидуальных образовательных маршрутов по стандартам WS; • Чемпионатное образование; • Коучинг в образовании; • Профориентация

Приведем некоторые примеры внедрения стандартов Ворлдскиллс Россия в образовательный процесс:

- **внедрение требований профессиональных стандартов и стандартов WSR с содержание дисциплин и модулей;**
- **изменение содержания дисциплин с исключением устаревшей информации и сокращением часов по соответствующим дисциплинам;**
- **изменение часов вариативной части в соответствии с WSSS;**
- **проведение экзаменов и квалификационных экзаменов с применением элементов и стандарта WSR;**
- **оценивание курсовых проектов с применением элементов и стандарта WSR;**
- **проведение демонстрационных экзаменов;**
- **проведение конкурсов профессионального мастерства и олимпиад с применением стандартов WSR.**

Задание: составьте проект внедрения стандартов Ворлдскиллс в образовательный процесс вашего колледжа. Результат представьте в виде схемы.

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.

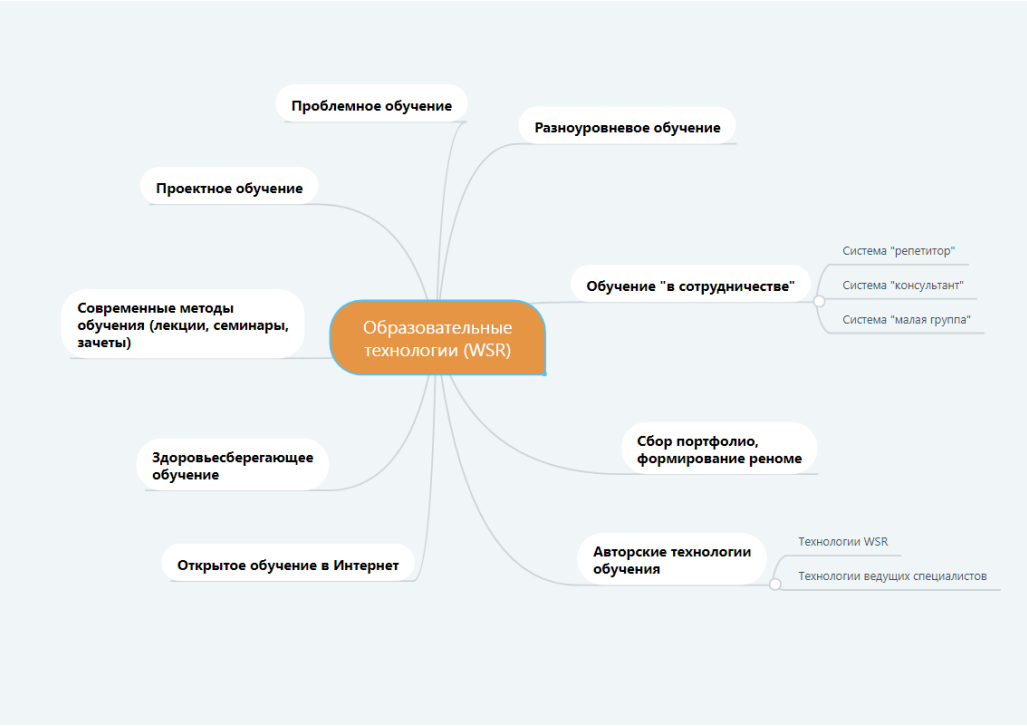
Во–вторых, по прежнему актуальность имеют образовательные технологии предметного обучения. При технологизации предметного обучения важно учитывать ценностно-деятельностный аспект всех этапов изучения предмета и его профессиональную специализацию.

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.

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, WrapPaneli 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**
11	**Понятие окна в WPF** **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

Фрагмент РП

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.

Адаптивное обучение

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

- **выбор мест прохождения практики с учетом требований их доступности – 65,9% образовательных организаций, обучающих инвалидов;**
- **проведение текущего контроля успеваемости, промежуточной и государственной итоговой аттестации обучающихся с учетом ограничений их здоровья – 59,4%;**
- **использование методов обучения, исходя из их доступности для обучающихся инвалидов и обучающихся с ОВЗ – 56,6%;**
- **использование практико-ориентированного (дуального) обучения – 26,2%;**
- **разработка индивидуальных учебных планов и индивидуальных графиков обучающихся инвалидов и обучающихся с ОВЗ– 23,4%;**
- **обеспечение обучающихся инвалидов и лиц с ОВЗ печатными и электронными образовательными ресурсами в формах, адаптированных к ограничениям их здоровья – 18,9%;**
- **включение в вариативную часть образовательных программ среднего профессионального образования адаптационных дисциплин – 5,6%**

Методическая основа разработки адаптированной ОП

- **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).

- «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).
- 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.

Задание: актуализация ФГОС 09.02.07 по одному профессиональному модулю (на выбор).
Шаблоны для акутализации и видео по актуализации доступны в системе дистанционно обучения.

Контроль по модулю программы:

1. **Рабочая программа ПМ, составленная по требованиям образовательной организации слушателя**
2. **Модель выпускника**
3. **Модуль внедрения стандартов Ворлдскиллс в образовательный процесс колледжа**

МОДУЛЬ 4. КУЛЬТУРА БЕЗОПАСНОГО ТРУДА. ОСНОВЫ БЕЗОПАСНОГО ТРУДА И ЭФФЕКТИВНАЯ ОРГАНИЗАЦИЯ РАБОЧЕГО МЕСТА В СООТВЕТСТВИИ СО СТАНДАРТАМИ ВОРЛДСКИЛЛС И СПЕЦИФИКАЦИЕЙ СТАНДАРТОВ ВОРЛДСКИЛЛС ПО КОМПЕТЕНЦИИ

Вредные и (или) опасные производственные факторы

Физические:
- **подвижные части производственного оборудования;**
- **острые кромки, заусенцы и шероховатость на поверхно- стях заготовок, инструментов и оборудования;**
- **повышенная или пониженная температура поверхностей оборудования, материалов; воздуха рабочей зоны;**
- **повышенный уровень шума на рабочем месте, вибрации;**
- **повышенная или пониженная влажность воздуха, подвиж- ность воздуха, ионизация воздуха;**
- **повышенная напряженность электрического поля, магнит- ного поля;**
- **недостаточная освещенность рабочей зоны;**
- **повышенная пульсация светового потока;**
- **повышенное значение напряжения в электрической цепи, замыкание которойможет произойти через тело человека**

Химические:
- **токсические;**
- **раздражающие;**
- **сенсибилизирующие**

Биологические:
- **патогенные микроорганизмы (бактерии, вирусы)**

Психофизиологические:

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.

Жидкости на рабочем столе

В процессе выполнения задания одному из обучаемых захотелось попить воды. Она пролила воду на рабочее место и повредила его. Работая возможно обучаемым уронила жидкость на периферийные устройства ПК.

В результате – произошло замыкание оборудования.

В процессе выполнения задания одному из обучаемых захотелось попить воды. Она пролила воду и поставила емкость с водой на соседний пустой стол.

В результате риск повреждения ПК минимален.

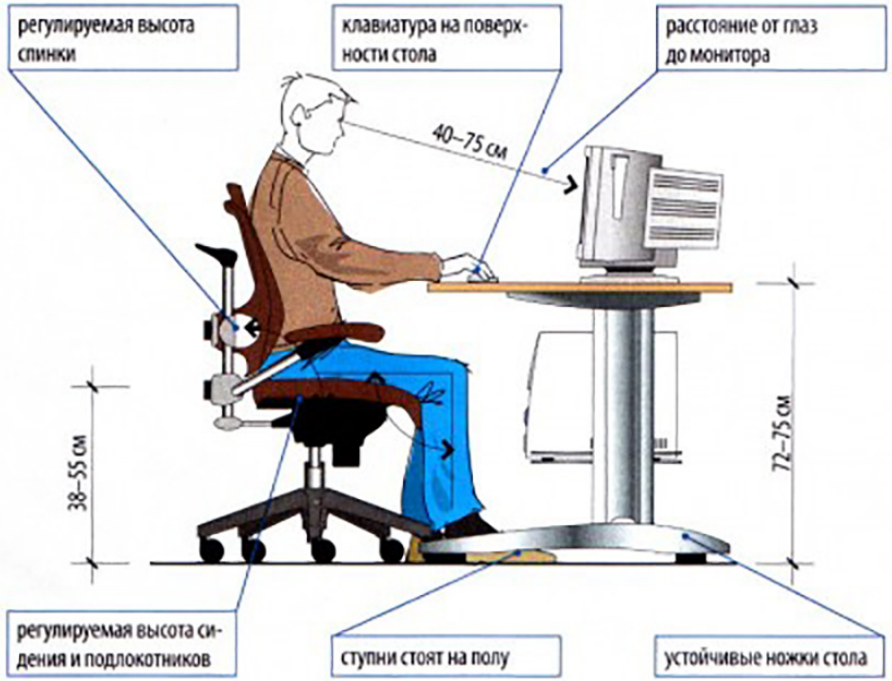
Работа с канцелярскими ножами

В процессе выполнения задания один из обучаемых выдвинул лезвие канцелярского ножа более чем на одно деление и положил его на стол после использования, не спрятав лезвие обратно.

В результате – обучаемый получил травму руки.

В процессе выполнения задания обучаемый не выдвинул лезвие более чем на одно деление и прежде чем положить на стол после использования, спрятал лезвие обратно.

В процессе работы с указанным инструментом у обучаемого риск порезать руку минимален



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.

ОТРАБОТКА СИТУАЦИОННЫХ ЗАДАЧ ПО ОТ И ТБ

Вопрос: первый день чемпионта, участник при выполнении задания задевает висячие от компью- тера провода, компьютер гаснет, участник ничего не сохранил, поднимает руку и объясняет ситуацию ГЭ. Кто виновен? Действия ГЭ.
Ваш ответ:

Правильный ответ:

Вопрос: при застройке площадки НЧ 2019 в комнату экспертов и комнату ГЭ были установлены пороги. Есть ли нарушение ТБ, каковы вероятные действия ГЭ, в какой день?

Ваш ответ:

Правильный ответ:

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

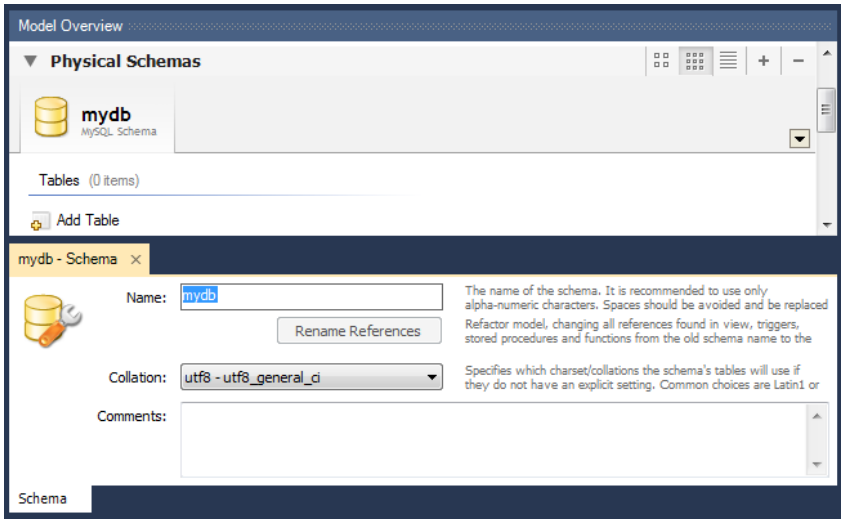
средств насчитывает около 300 различных CASE-систем, наиболее мощные из которых так или иначе используются практически всеми ведущими западными фирмами.

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.

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

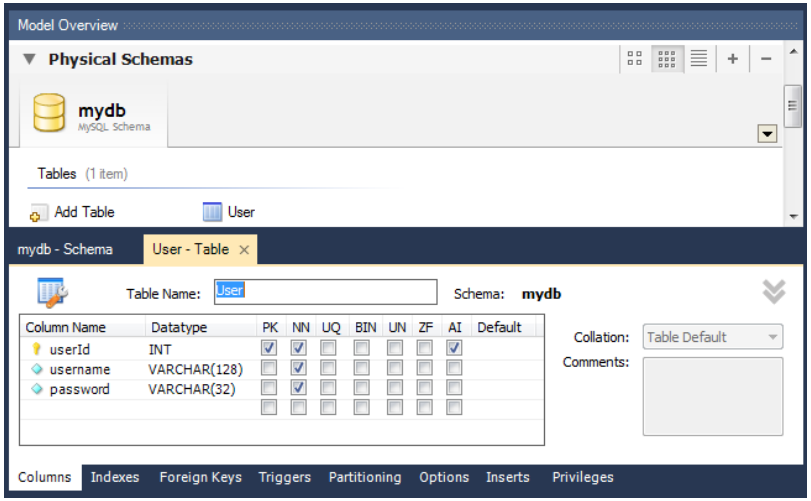
Для добавления модели нажимаем "Models" или выбираем "File → New Model" (Ctrl + N):



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

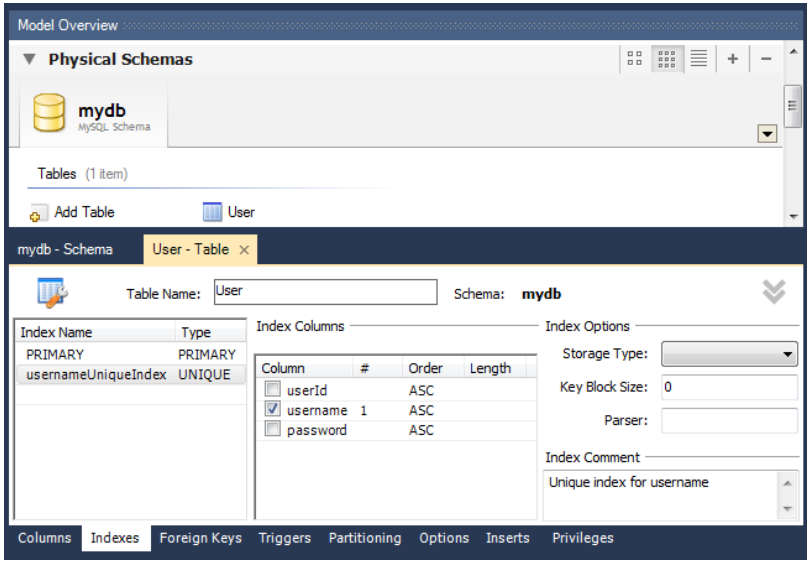
Список баз данных проекта и список таблиц в пределах базы данных будет располагаться во вкладке "Physical Schemas".Чтобы создать таблицу, дважды кликаем на "+Add Table":



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:

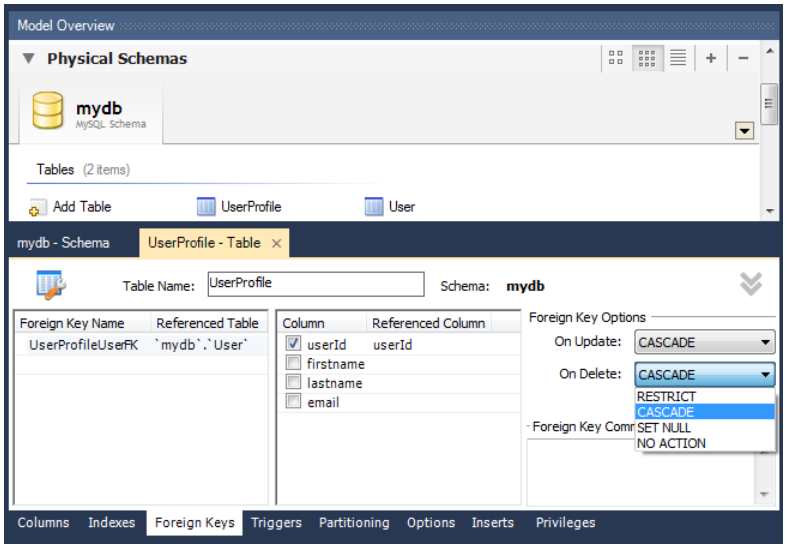


к

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 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:

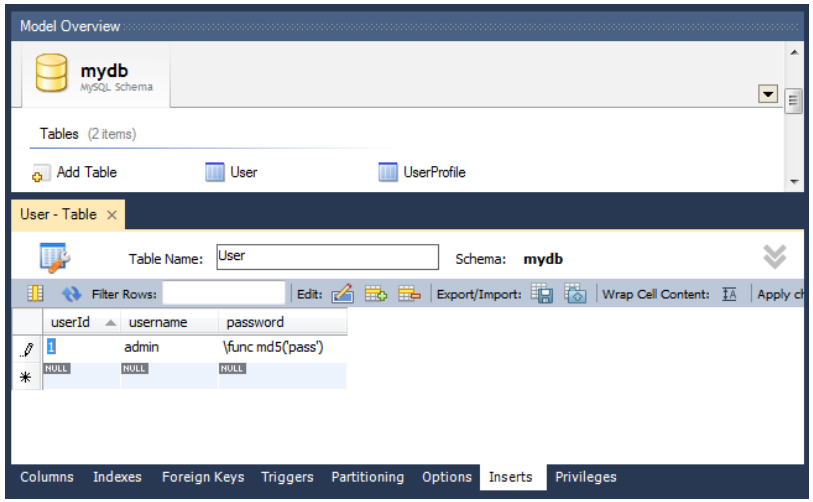
- **RESTRICT - выдавать ошибку при изменении / удалении родительской записи**
- **CASCADE - обновлять внешний ключ при изменении родительской записи, удалять дочернюю запись при удалении родителя**
- **SET NULL - устанавливать значение внешнего ключа NULL при изменении / удалении родителя (неприемлемо для полей, у которых установлен флаг NOT NULL!)**
- **NO ACTION - не делать ничего, однако по факту эффект аналогичен RESTRICT**

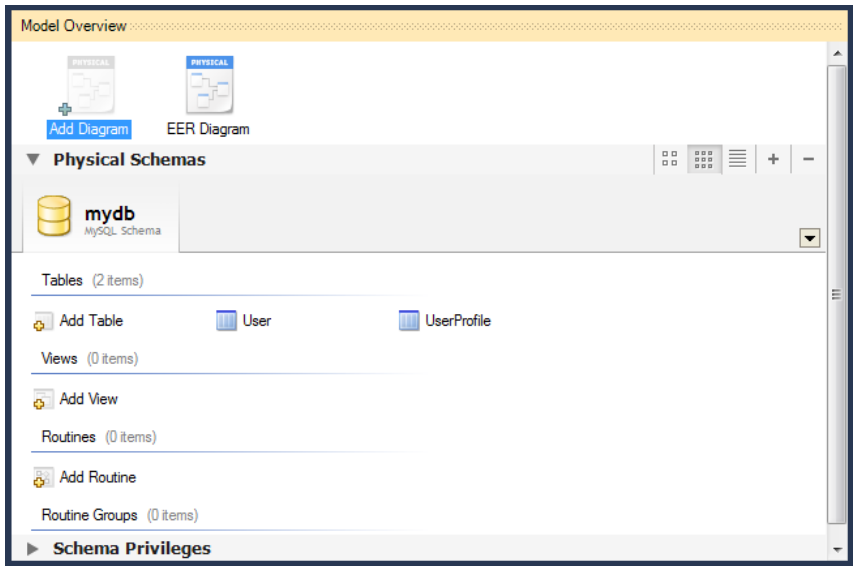
В приведённом примере добавили к дочерней таблице UserProfile внешний ключ для связи с родительской таблицей User. При редактировании поля userId и удалении позиций из таблицы User аналогичные изменения будут автоматически происходить и со связанными записями из таблицы UserProfile.

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":

в





Как видно из примера, в случае, если перед записью 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")

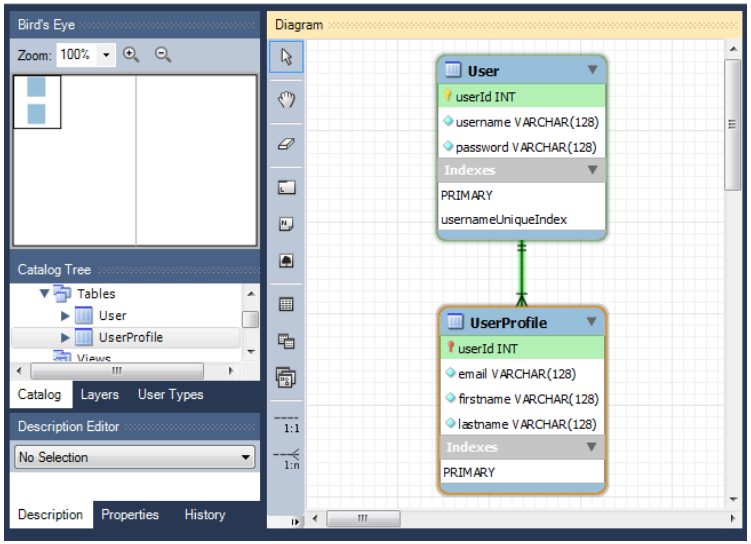
Для представления схемы данных, сущностей и их связей в графическом виде в MySQL Workbench существует редактор EER-диаграмм. Для создания диаграммы в верхней части экрана управления базой данных дважды кликаем на иконку "+Add Diagram":

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)

Для

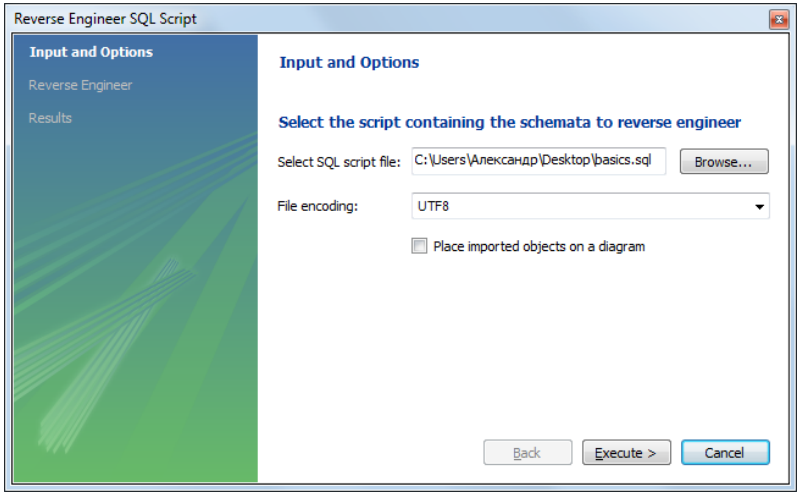


Если уже есть схема данных, её можно без труда импортировать в MySQL Workbench для дальнейшей работы. импорта модели из SQL файла выбираем "File → Import → Reverse Engineer MySQL Create Script...", после чего выбираем нужный SQL файл и жмём "Execute >"

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)**

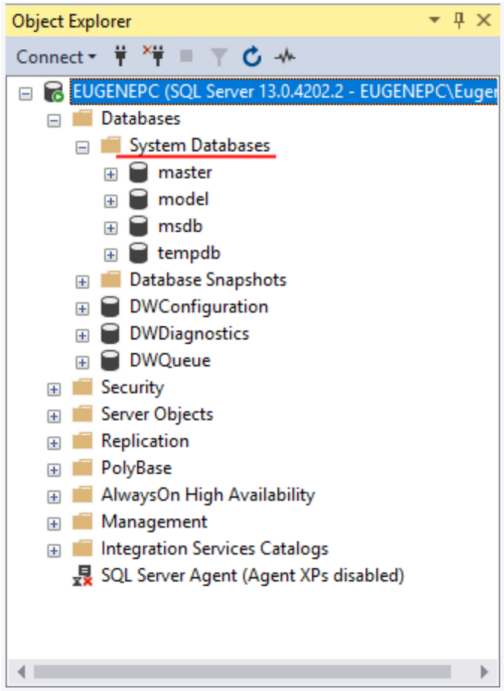
- **Функции: также код SQL, который выполняет определенную задачу**

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.

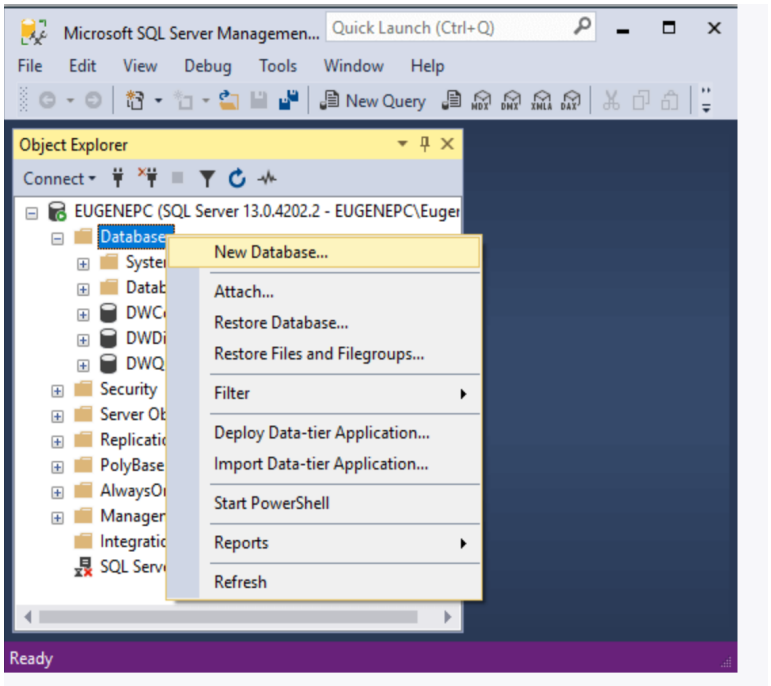
• tempdb: эта база данных используется как хранилище для временных объектов. Она заново пересоздается при каждом запуске сервера.

Все эти базы можно увидеть через SQL Server Management Studio в узле Databases -> System Databases:

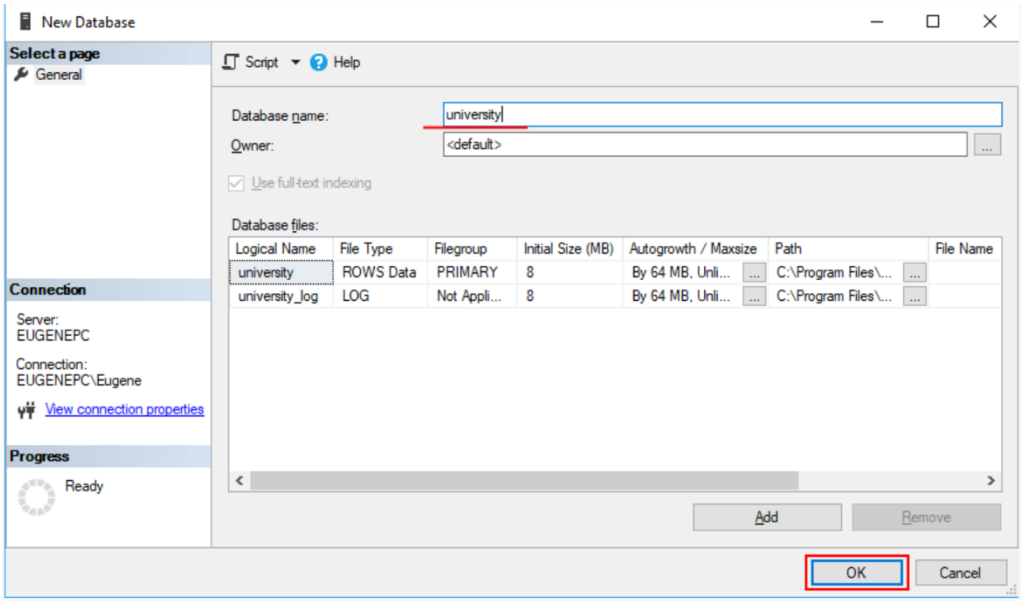
Эти базы данных не следует изменять.

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)**

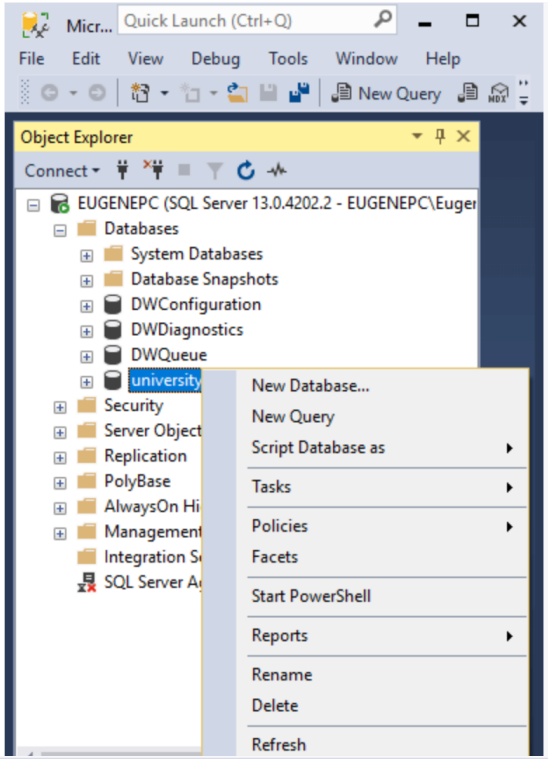
- **Filegroup: обозначет группу файлов. Группа файлов может хранить множество файлов и может использоваться для разбиения базы данных на части для размещения в разных местах.**
- **Initial Size (MB): устанавливает начальный размер файлов при создании (фактический размер может отличаться от этого значения).**
- **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.**

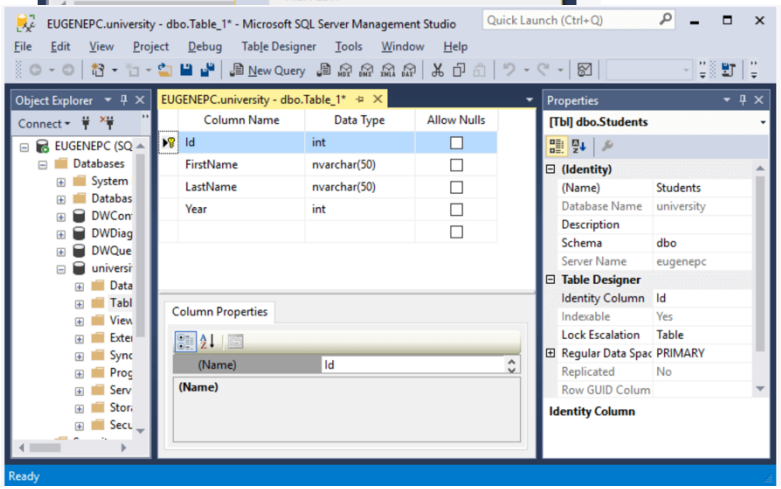
После ввода названия базы данных нажмем на кнопку OK, и БД будет создана.

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.

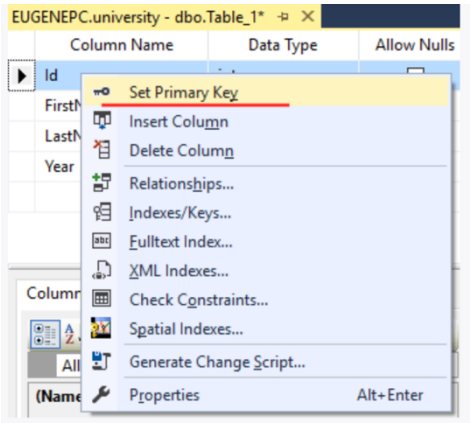
Allow Nulls: может ли отсутствовать значение у столбца, то есть может ли он быть пустым

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).

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.

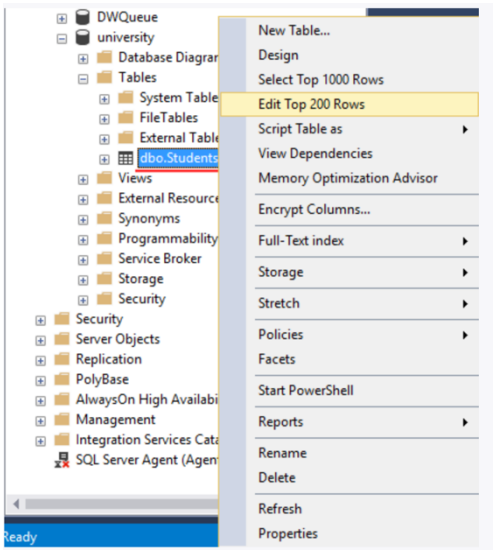


И в конце нам надо отметить, что столбец Id будет выполнять роль первичного ключа (primary key). Первичный ключ уникально идентифицирует каждую строку. В роли первичного ключа может выступать один столбец, а может и несколько.

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.

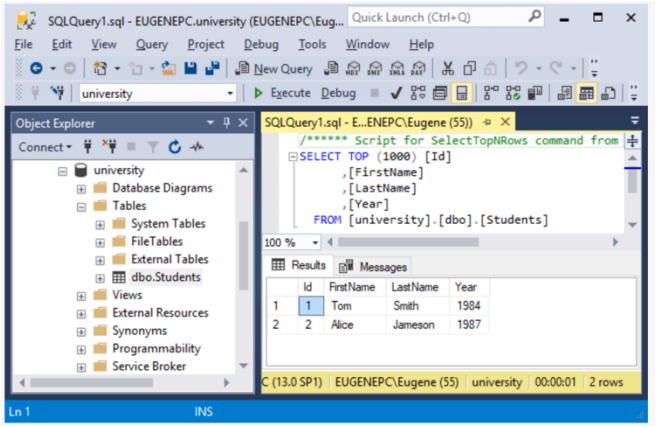
Для добавления начальных данных можно выбрать опцию Edit Top 200 Rows. Она открывает в виде таблицы 200 первых строк и позволяет их изменить. Но так как у нас таблица только создана, то естественно в ней будет никаких

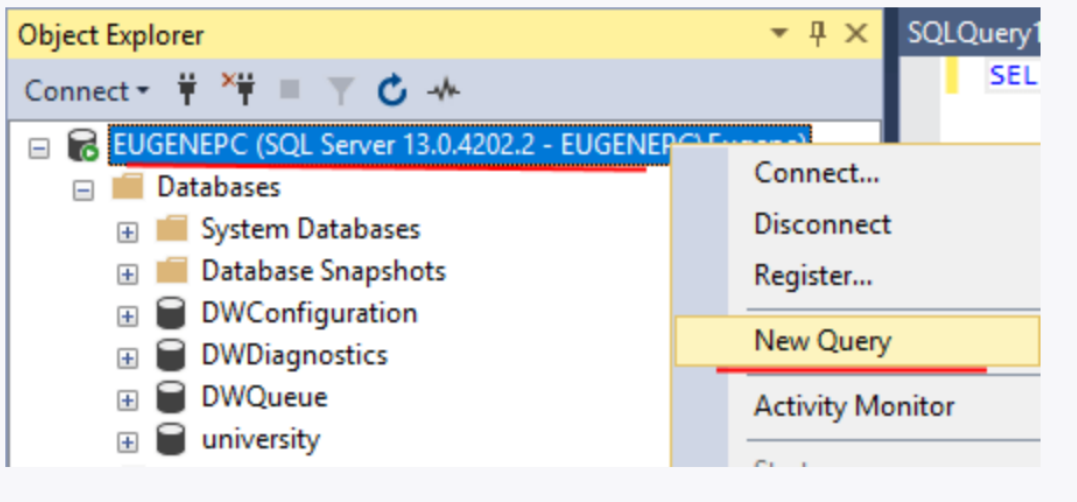
данных. Введем пару строк - пару студентов, указав необходимые данные для столбцов:

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:



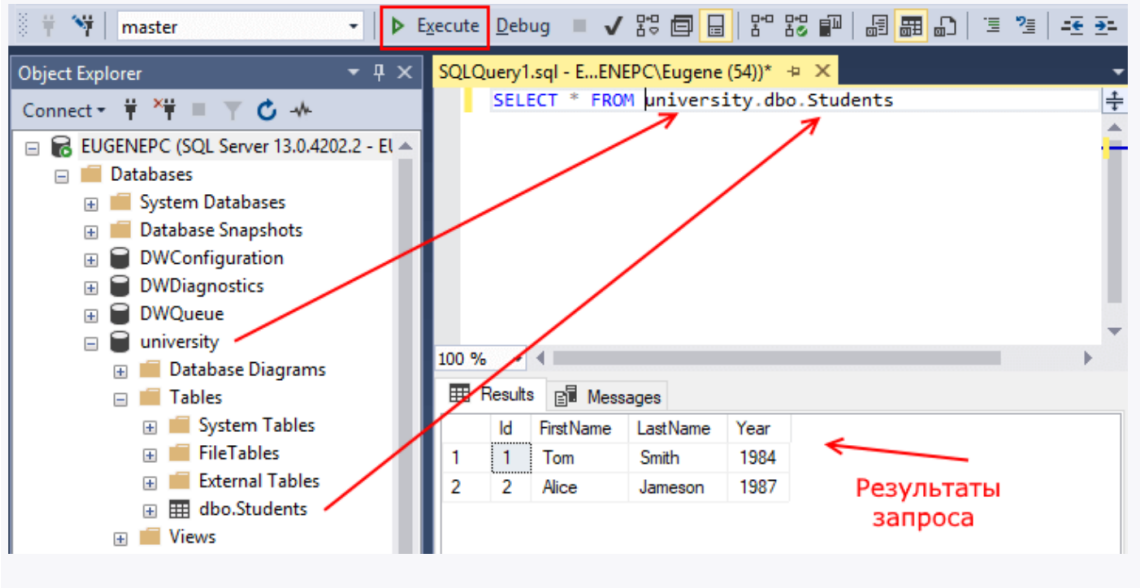


После этого в центральной части программы откроется окно для ввода команд языка SQL.

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.

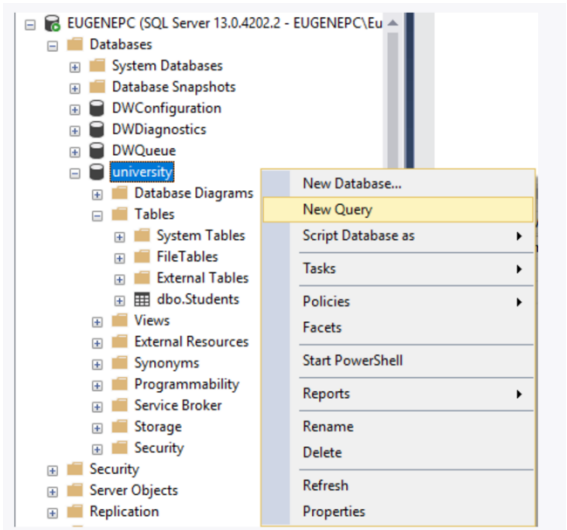


После ввода запроса нажмем на панели инструментов на кнопку Execute, либо можно нажать на клавишу F5.

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:

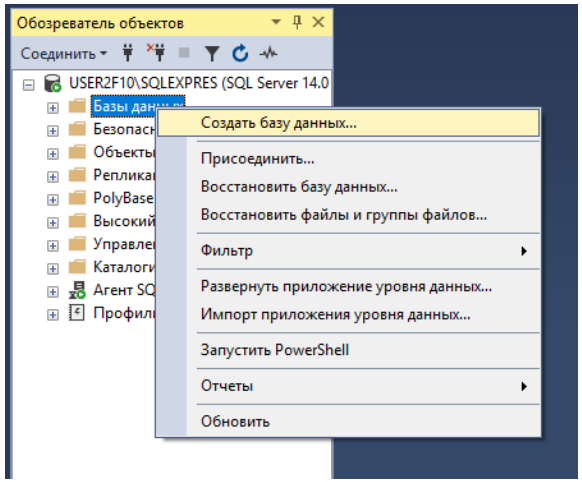
```
1        SELECT * FROM Students
```

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.) .**

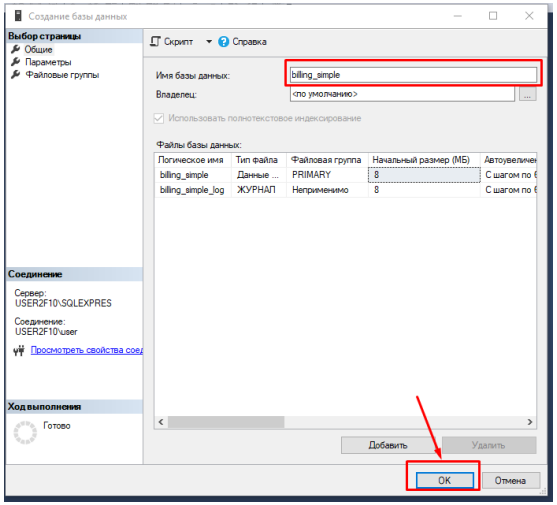
Задание: Работа с запросами, импорт из файлов формата .xlsx



Создать базу данных: Для это откроем MS SQL Management Studio, авторизуемся в обозревателе объектов создадим базу данных с именем billing_simple.

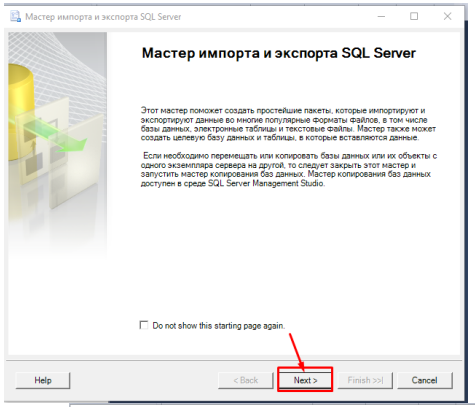
Выделить блок «Базы данных», нажать ПКМ, выбрать «Создать базу данных». В открывшемся окне введем имя базы данных billing_simple

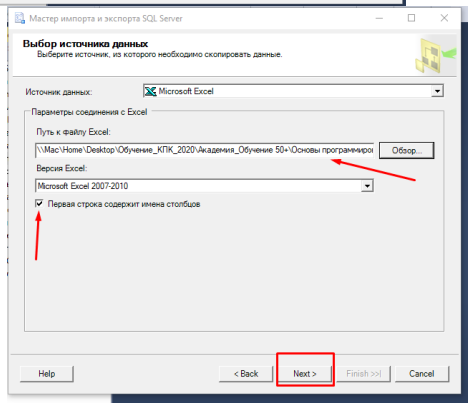
2. Импортируем данные (из формата .xlsx) из документа с именем billing Для этого выделим в обозревателе объектов выделим имя базы данных billing_simple, нажмем ПКМ, в разделе задачи выберем пункт «Импорт данных»:

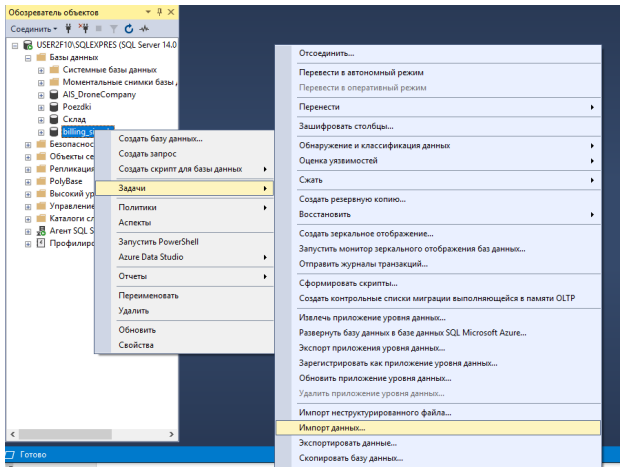


Далее:

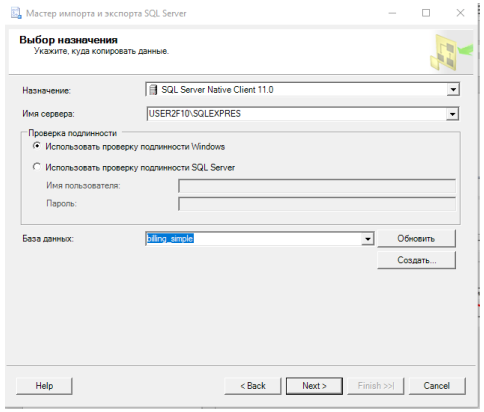
В открывшемся диалоговом окне укажите источник для загрузки, место расположения файла:



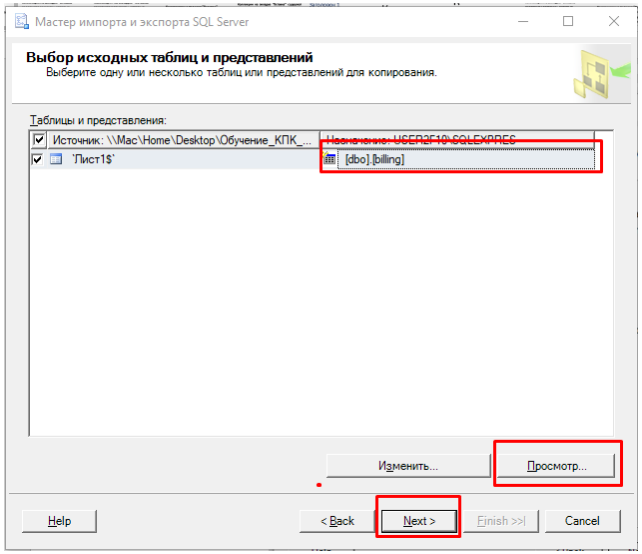


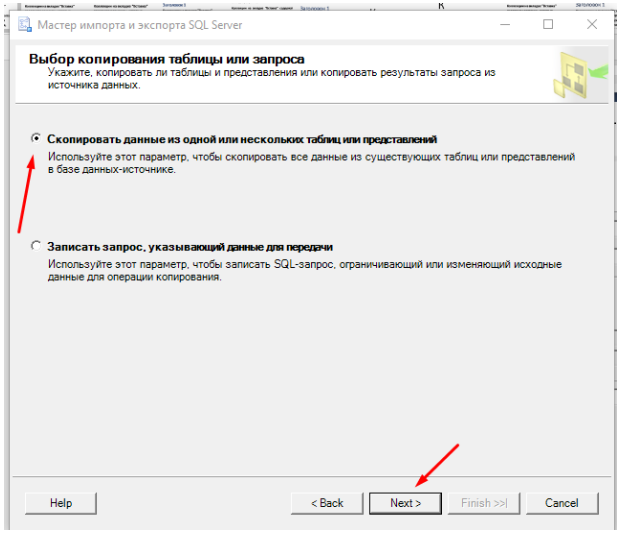


В новом окне выберите назначение SQL Server Native Client 11.0 и нажмите «Далее»



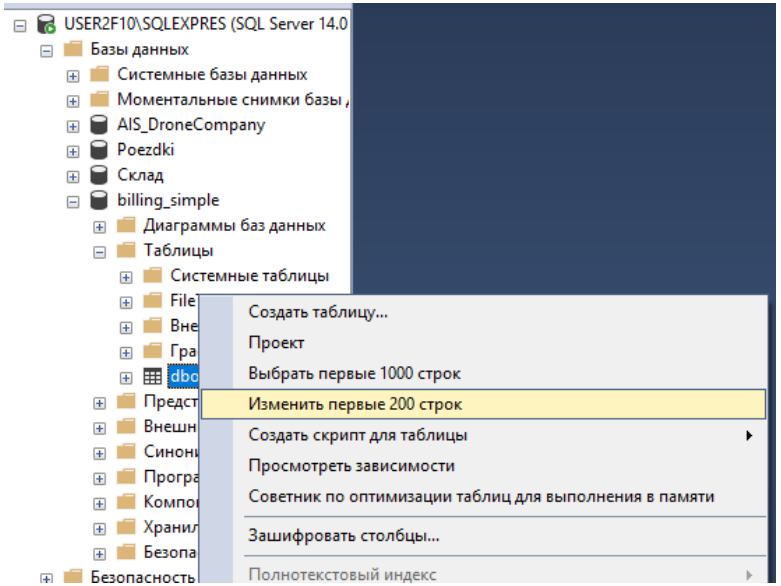
В новом окне измените имя поля в поле «Просмотр», нажать таблицы на billing, можно просмотреть «Далее».

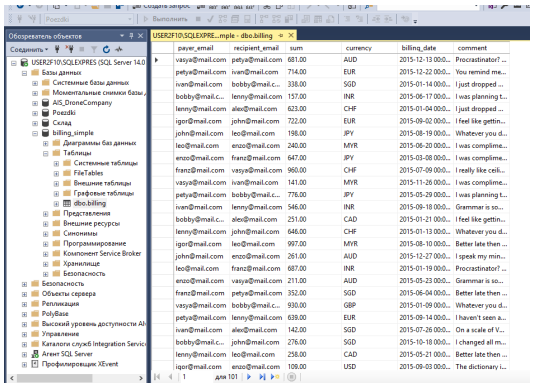




Далее нажать Финиш и закрыть окно импорта.

В обозревателе объектов проверьте содержимое таблицы. Для этого необходимо выделить имя таблицы billing, нажать ПКМ, выбрать «Изменить первые 200 строк».





Должно получиться так:

Задание: В таблице billing создайте запросы:

1) Добавьте в таблицу одну запись о платеже со следующими значениями: email плательщика: 'pasha@mail.com'
email получателя: 'katya@mail.com'
сумма: 300.00
валюта: 'EUR'
дата операции: 14.02.2018
комментарий: 'Valentines day present)'

2) Измените адрес плательщика на 'igor@mail.com' для всех записей таблицы, где адрес плательщика 'alex@mail.com'.

3. Импортировать таблицы из файлов «Client», «sale», «category»

МЕТОДИКА ОЦЕНКИ КОНКУРСНОГО ЗАДАНИЯ

Совместно с преподавателем проведите оценку выполненных заданий другого слушателя по приведенным ниже аспектам:

ERD	O	Определены основные сущности	1
	O	Определены типы данных	1
	O	Определены ключевые поля	1
	O	Определены связи	1
	O	Реализована 3 НФ	1
Импорт	O	Обработка данных для импорта реализована	1
	O	Данные импортированы в полном объеме	1
Экспорт БД	O	Сформирован скрипт с данными	1
	O	Сформирован *.bak	1
Восстановление	O	Развертывание БД из скрипта	1
	O	Расзвертывание БД из *.bak	1

ПРИМЕРЫ ЗАДАНИЙ ДЛЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА:

"Проектирование базы данных"

Задание для выполнения

1.Определите количество и структуру таблиц БД:

✓ выделите в исходной информации данные, относящиеся к отдельным информационным объектам (например, информацию об успеваемости студентов можно разделить на данные о студентах и на данные о полученных ими оценках);

✓ каждому из полученных наборов данных поставьте в соответствие отдельную таблицу (например, данные о студентах и о полученных ими оценках будут находиться в разных таблицах). В результате, в первом приближении, будет определён набор полей для каждой из таблиц;

✓ **проверьте полученные таблицы. Таблица должна содержать данные об однотипных объектах, информация в одной таблице не должна дублироваться, не должно быть повторений между таблицами;**

✓ **при необходимости переносите поля из одних таблиц в другие.**

2.Выполните проверку полей в каждой полученной таблице. Дайте корректные имена полям таблиц. Определите тип данных в каждом поле таблиц:

✓ **данные следует разбивать на наименьшие логические единицы (например, в таблицу следует включить три поля – Фамилия, Имя и Отчество, а не одно общее поле ФИО);**

✓ **имена полей, по возможности, должны быть краткими. Использовать пробелы в именах полей не рекомендуется.**

3.Определите первичные ключевые поля в каждой таблице:

✓ **в качестве первичного ключевого поля следует использовать поле, содержащее уникальные значения (например, поле с номером зачётной книжки, в таблице, содержащей данные о студентах);**

✓ **при необходимости в таблицу можно добавить новое поле с уникальными значениями (код, номер события, индивидуальный номер и т.д.);**

✓ **в ряде случаев можно использовать составной ключ, содержащий значения двух, и более полей, при этом сочетание значений должно быть уникальным.**

4.Установите связи между таблицами. Определите типы связей:

✓ **таблицы БД связываются по полям, содержащим одинаковые значения. Правильно установленные связи соответствуют реальным связи между информационными объектами;**

✓ **связь должна обеспечивать целостность данных (например, связывать таблицу с данными о студентах с таблицей с данными об оценках необходимо по полю с номером зачетной книжки, а не по полю, содержащему имя студента);**

✓ **при необходимости в таблицу можно добавить новое поле или заменить существующее, если это требуется для установления связи (например, в таблицу с данными об оценках может быть добавлено поле с номером зачётной книжки, или данным полем заменено поле, содержащее имя студента).**

Вариант 1

Назначение базы данных: Учёт пассажирских перевозок.

Информация, которая должна содержаться в базе данных: Номер маршрута; название маршрута; начальная остановка; конечная остановка; продолжительность маршрута; фамилия, имя, отчество водителя; адрес водителя; контактный телефон водителя; государственный номер автобуса; марка автобуса; пассажировместимость; дата поездки; выполненный маршрут; время начала поездки; время окончания поездки; водитель, выполнивший маршрут; автобус, выполнивший маршрут.

Вариант 2

Назначение базы данных: Учёт товаров, поступающих на склады фирмы.

Информация, которая должна содержаться в базе данных: Наименование товара поступившего на склад; количество поступившего товара; дата поступления; номер склада; наименование поставщика товара; адрес поставщика; телефон поставщика; банковские реквизиты поставщика; наименование товара; артикул товара; вид товара; стоимость единицы товара; наименование производителя товара; адрес производителя; телефон производителя; банковские реквизиты производителя.

Вариант 3

Назначение базы данных: Отслеживание деятельности туристической фирмы.

Информация, которая должна содержаться в базе данных: Фамилия, имя, отчество клиента; адрес клиента; контактный телефон клиента; вид тура; страна и населенный пункт тура; характеристика тура; наименование туроператора; адрес туроператора; контактные телефоны туроператора; реквизиты туроператора; дата заключения договора; дата начала путешествия; дата окончания путешествия; клиент, заключивший договор; приобретённый тур; туроператор, продавший тур.

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 студент или специалистов: 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.

Выполните задание по построению UML диаграмм, выданное Вам в системе дистанционного обучения

Платформы для разработки программных решений

Компьютерная платформа — 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 студент или специалист 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

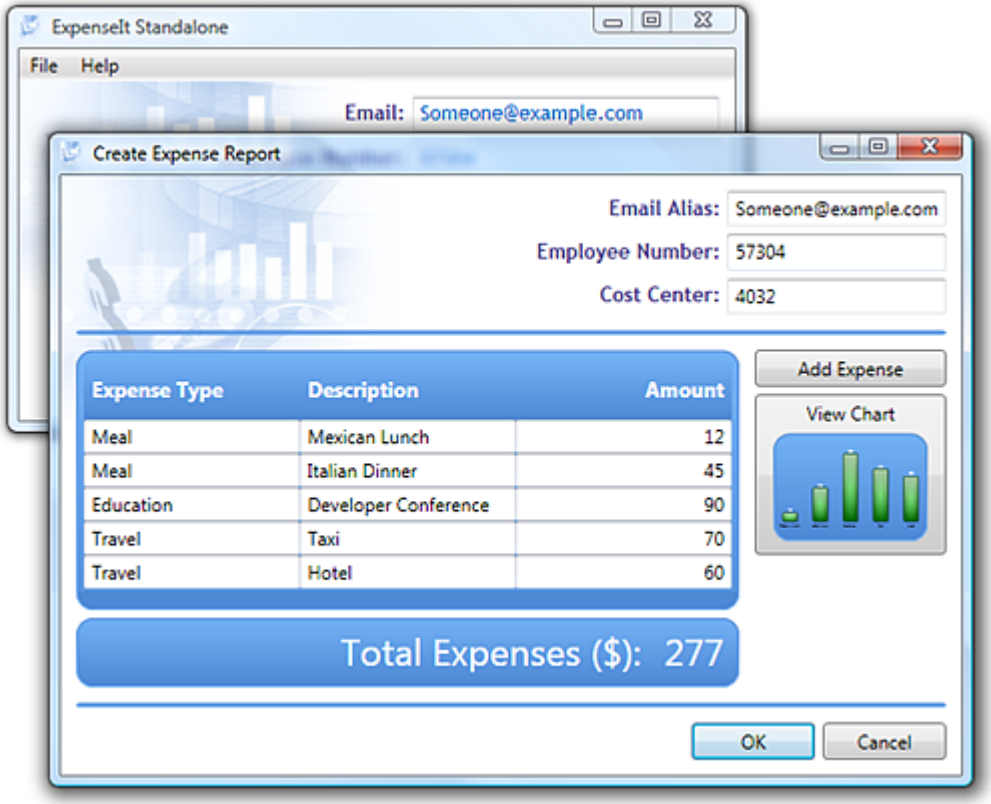
WPF такое поведение обычно реализуется в коде, который связан с разметкой. Этот тип кода называется кодом программной части.

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

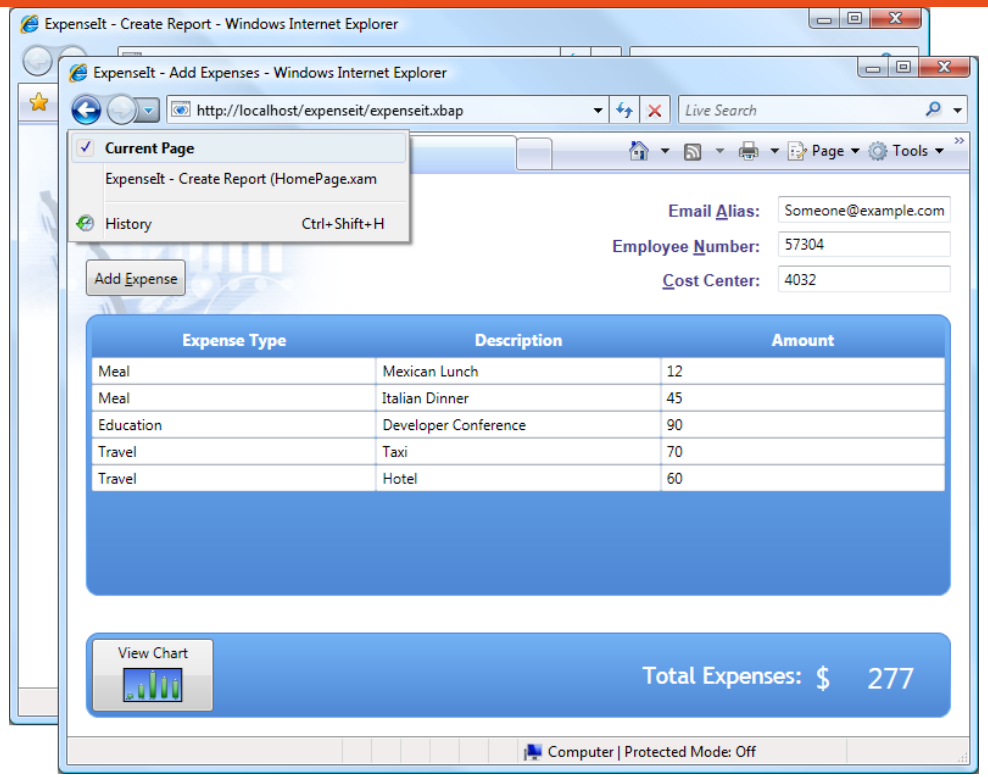
Для автономных приложений можно создавать доступные из меню и панелей инструментов окна и диалоговые окна с помощью класса Window. На следующем рисунке показано автономное приложение с главным и диалоговым окном.



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, FlowDocumentReader, FlowDocumentScrollViewer и StickyNoteControl.**
- **Ввод: TextBox, RichTextBox и PasswordBox.**
- **Структура: Border, BulletDecorator, Canvas, DockPanel, Expander, 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 TabControl.**
- **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.**

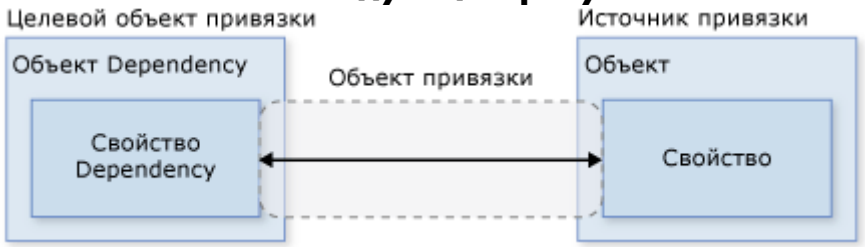
Привязка данных

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.

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).

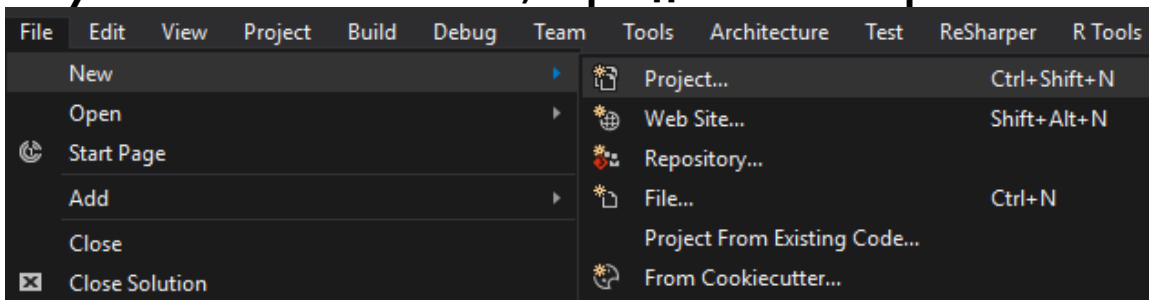
1.6 СОЗДАНИЕ "1. ГЛАВНЫЙ ЭКРАН СИСТЕМЫ" Создайте главное меню системы, как указано в "1. Главный экран системы"в презентации. Каждое окно / страница приложения, который имеет "? дней? часов и? минут до начала гонки в "в нижней части экрана должно автоматически обновляться в режиме реального времени. 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.

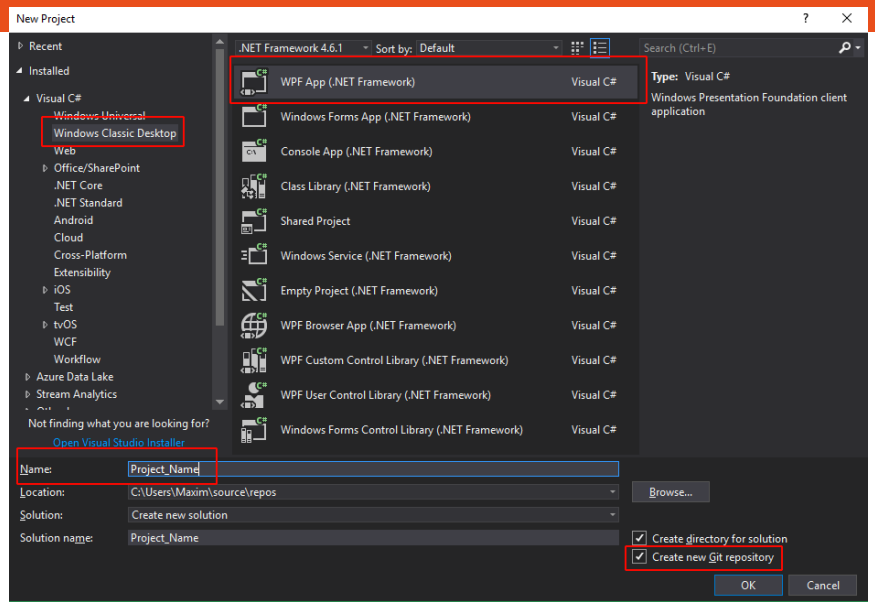
1.8 СОЗДАНИЕ ОКНА "10. СПИСОК БЛАГОТВОРИТЕЛЬНЫХ ОРГАНИЗАЦИЙ" Создать страницу, как описано в "10. Список благотворительных организаций»в презентации. Эта страница отображает все благотворительные организации, перечисленные в базе данных вместе с их логотипами (при условии, что они есть в общих ресурсах), чтобы показать бегунам благотворительные организации, которые их могут поддержать.

СОЗДАНИЕ ПРОЕКТА

Запустите Vi sual Studio 2017, перейдите File – Project.



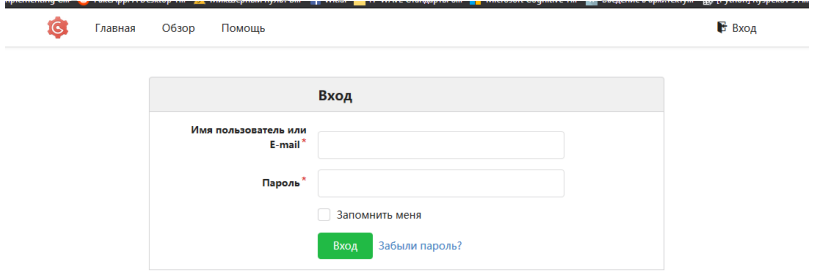
Выберите Visual C# - Windows Classic Desktop – WPF App, укажите имя проекта, а также поставьте галочку «Create new git repositoty» чтобы включить проект в систему контроля версий.



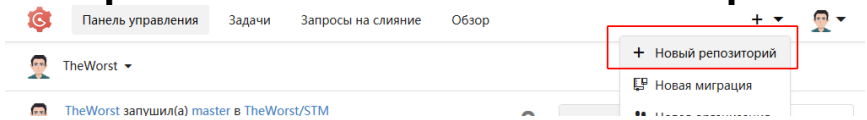
После нажатия кнопки «OK» создастся проект с пустой формой.

СОЗДАНИЕ РЕПОЗИТОРИЯ

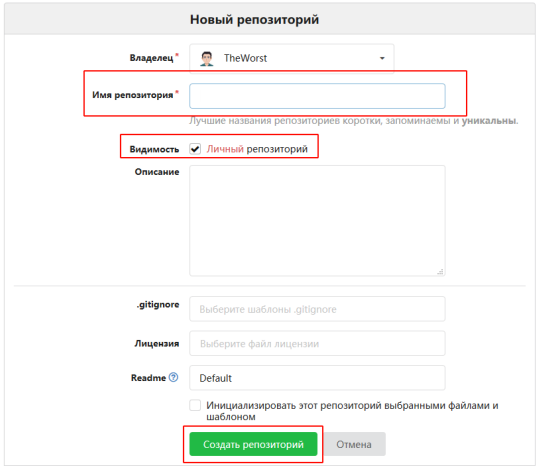
Откройте в браузере GOGS. Авторизуйтесь под своей учетной записью.



На верхней панели нажмите «+» и выберите «Новый репозиторий».



После чего укажите имя репозитория, поставьте галочку что репозиторий будет являться личным. И

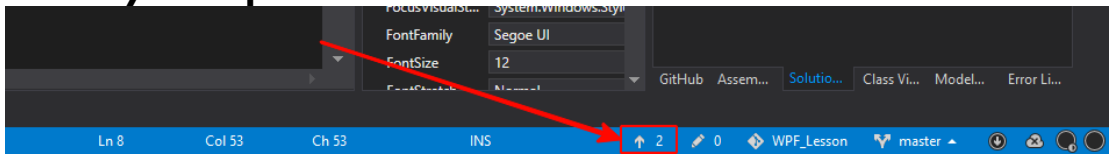


создайте его путем нажатия на «Создать репозиторий».

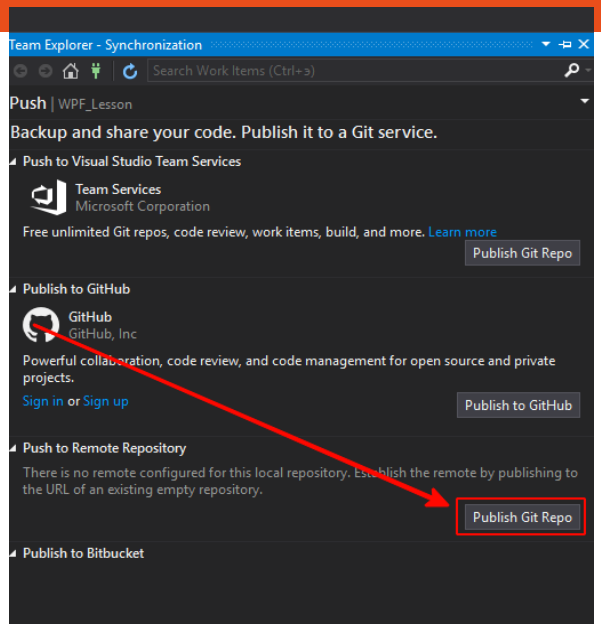
После создания репозитория появится следующее окно.

Создадим в проекте «Resource Dictionary».

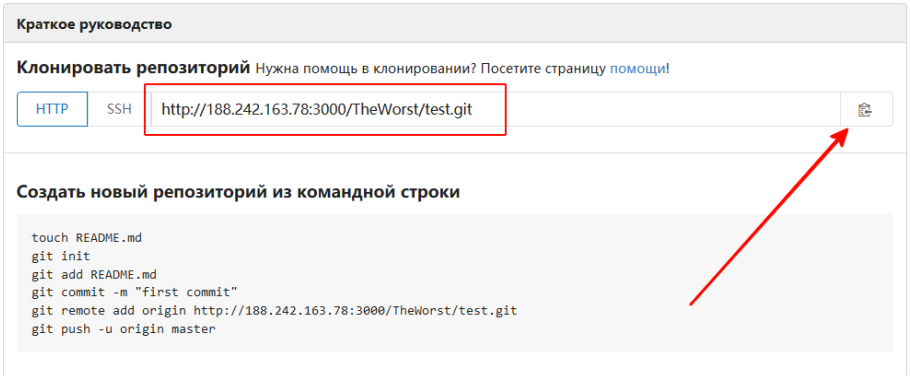
Для того чтобы подключить проект, откройте проект в Visual Studio, затем внизу нажмите на стрелку направленную вверх.



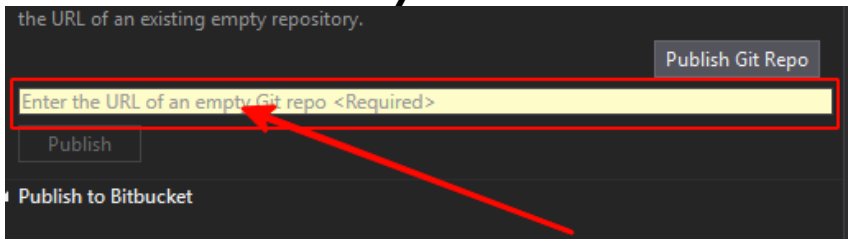
После нажатия у вас откроется окно «Team Explorer», в котором нажмите «Publish Git Repo»



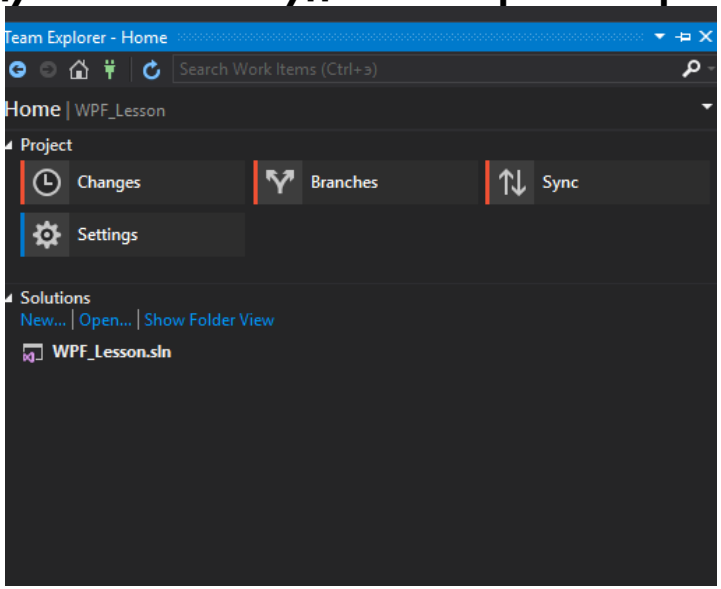
После чего у вас откроется окно в которое необходимо вставить ссылку на репозиторий. Для получения ссылки, откройте GOGS, авторизуйтесь, зайдите в созданный репозиторий (урок Создание репозитория) и там скопируйте на него ссылку.



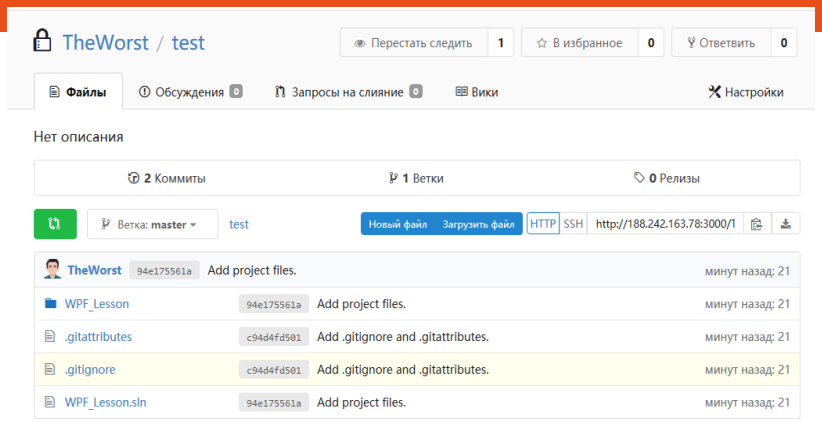
После того как скопируете ссылку вставьте ее в поле подключения удаленного репозитория в «Team Explorer» и нажмите кнопку «Publish»



После нажатия кнопки, проект будет сохранен на удаленном репозитории, а также создастся связь между локальным и удаленным репозиторием.



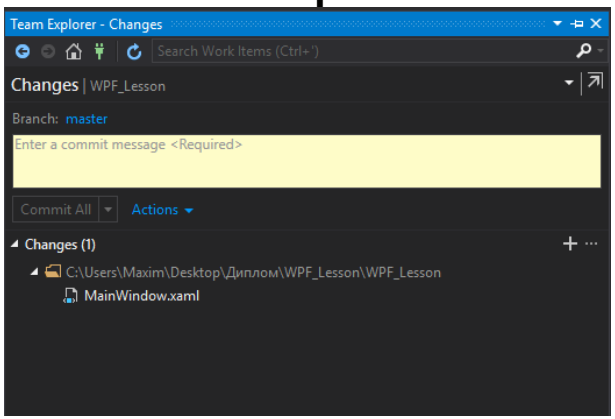
Проверить результат сохранения проекта можно на GOGS в репозитории.



СОХРАНЕНИЕ ИЗМЕНЕНИЙ В РЕПОЗИТОРИИ

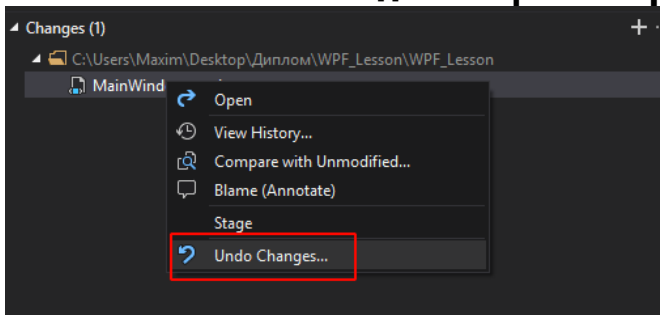
По мере того как будут вноситься правки в проект, счетчик изменений будет расти (указывается внизу Visual Studio)

Чтобы сохранить проект в репозитории, необходимо нажать на счетчик (карандаш). После этого появится окно Team Explorer.

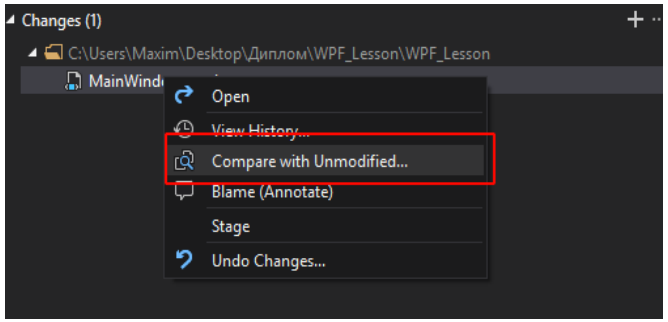


В данном окне показывается ветка (branch) в которую будет сохранятся изменения и файлы (Changes) в которых произошли изменения.

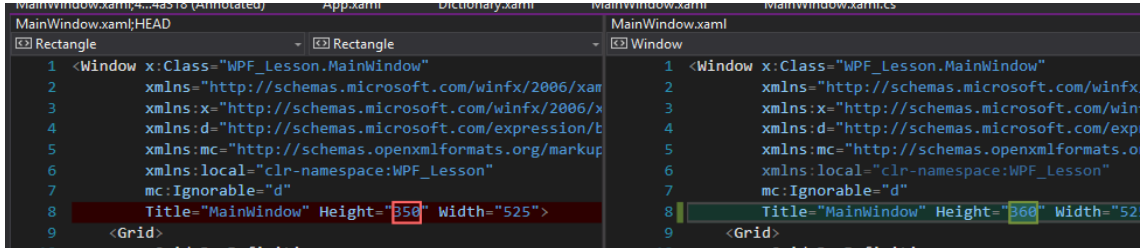
Если изменения были произведены ошибочно, то их можно откатить на предыдущею версию, путем нажатия на необходимом файле правой кнопкой мыши и выбрав пункт «Undo Changes»



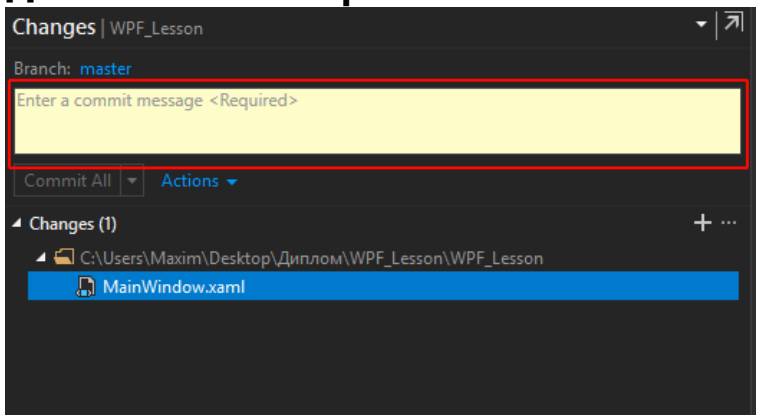
Если необходимо посмотреть изменения в файле (сравнить с предыдущей версией), то необходимо нажать правую кнопку мыши на нужном файле и выбрать пункт меню «Compare with Unmodified»



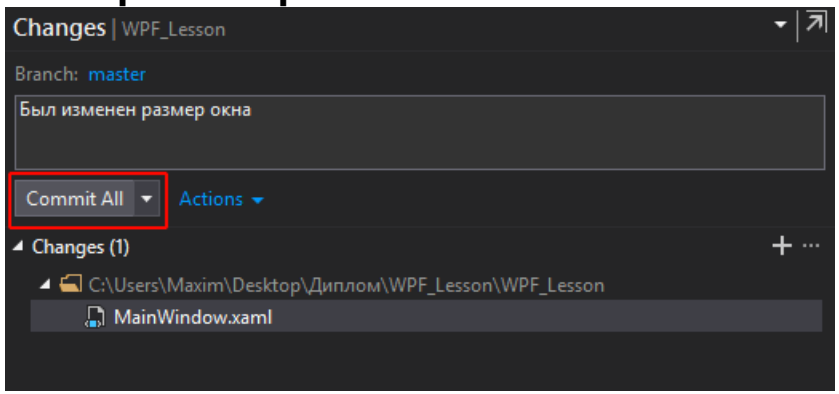
После чего откроется окно с изменениями, в одном окне предыдущая версия, в другом актуальная.



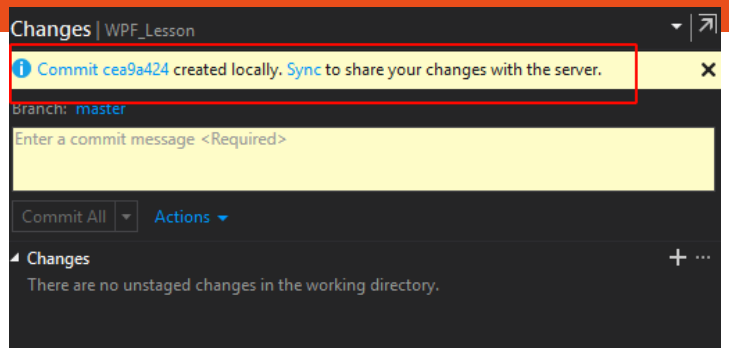
Для того чтобы сохранить изменения необходимо написать текст об изменениях в данной версии.



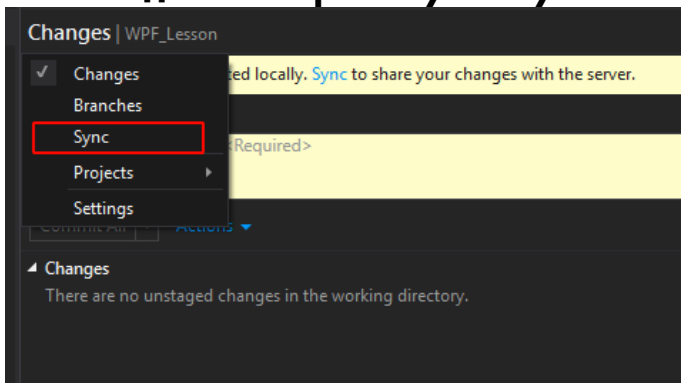
После ввода изменений, необходимо нажать кнопку «Commit All» для сохранения изменений в локальный репозиторий.



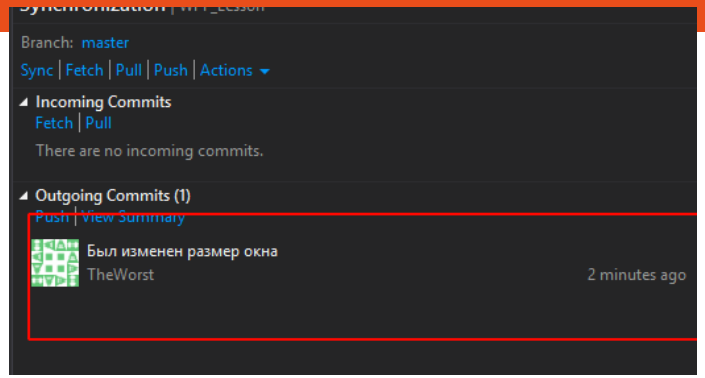
После чего будет выведено сообщение о успешном сохранении проекта. Так же будет предложено синхронизировать локальный репозиторий с удаленным.



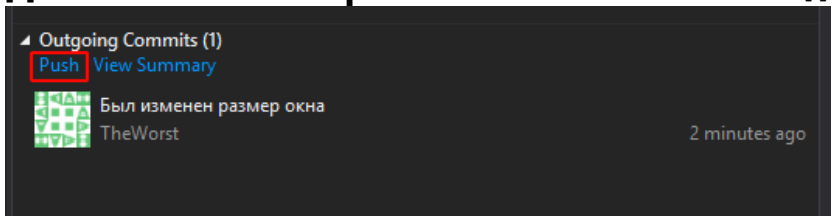
Теперь необходимо отправить изменения локального репозитория на удаленный репозиторий. Для этого необходимо выбрать пункт Sync из выпадающего меню.



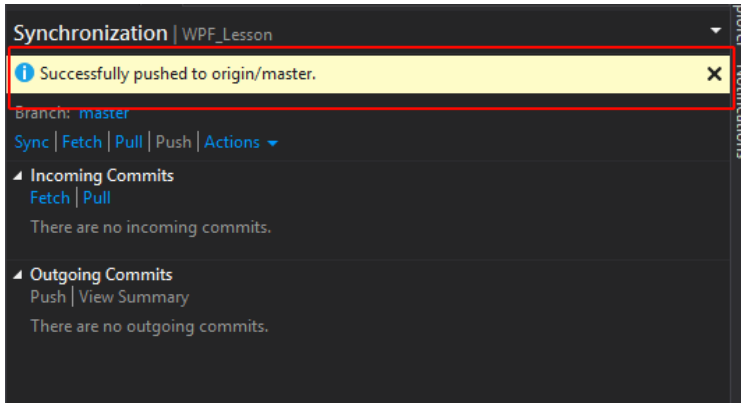
В данном пункте меню, внизу будет отображаться ваш коммит (изменения).



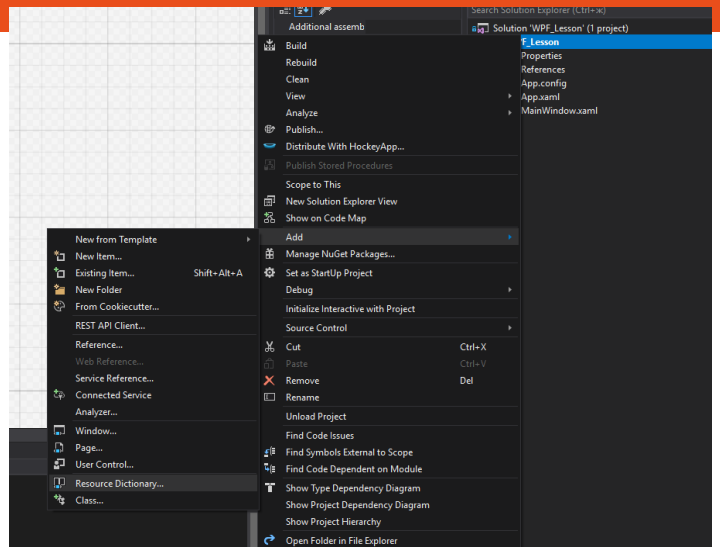
Для того чтобы отправить изменения необходимо нажать на кнопку «Push»



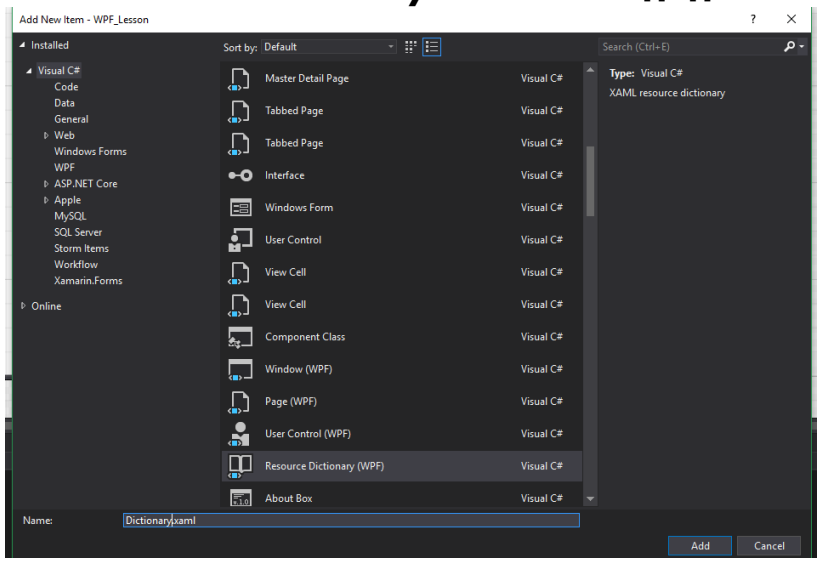
После нажатия кнопку локальный репозиторий будет сохранен на удаленном. И после успешного завершения будет выведено соответствующее сообщение.



СОЗДАНИЕ СЛОВАРЯ СТИЛЕЙ

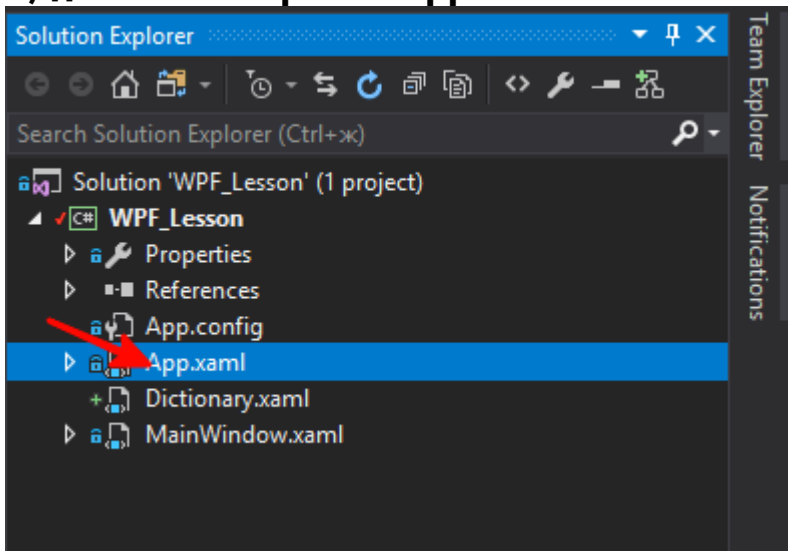


Укажем имя «Dictionary.xaml» и создадим его.



Для лабораторных работ

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>
```

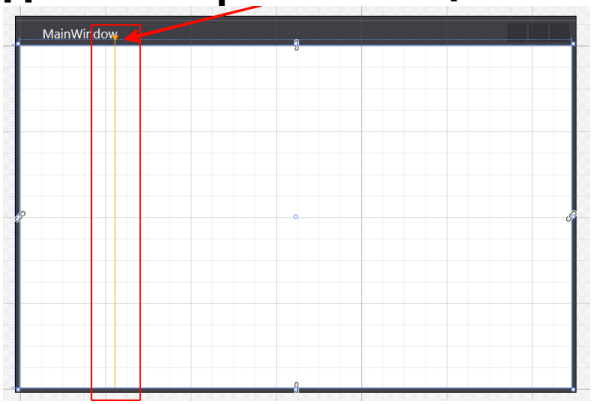
После данных операций можно начинать писать стили и константы. Напишем первый стиль, который будет менять задний фон и внутренний отступ, и он применяться ко всем компонентам типа «Label»

```
<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>
```

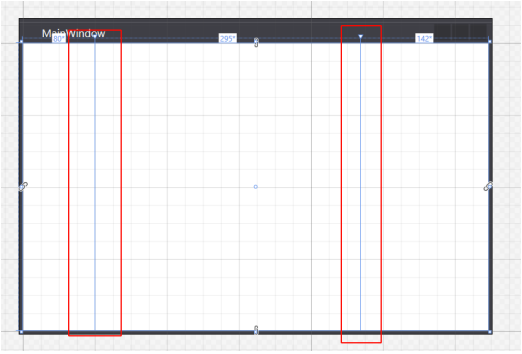
РАЗМЕЩЕНИЕ КОНТЕНТА ПО ЦЕНТРУ ФОРМЫ

Откройте форму или страницу. Выберите компонент Grid, после чего по краям (сверху и слева) появится возможность добавлять строки и столбцы.

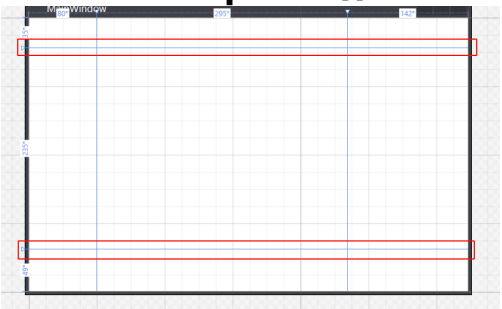
Добавим первый столбец.



Добавим еще один столбец.



Таким же образом добавим две строки.



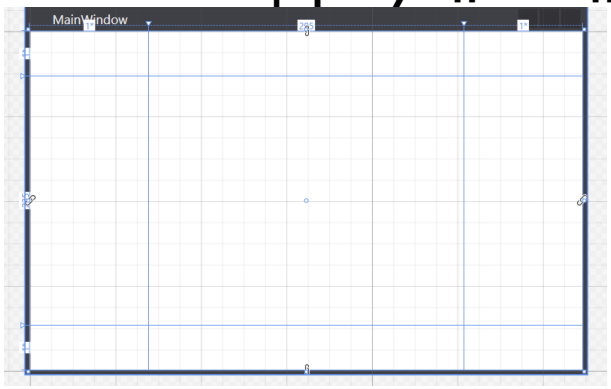
После всех добавление мышкой, мы можем увидеть следующий код XAML.

```
        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>
```

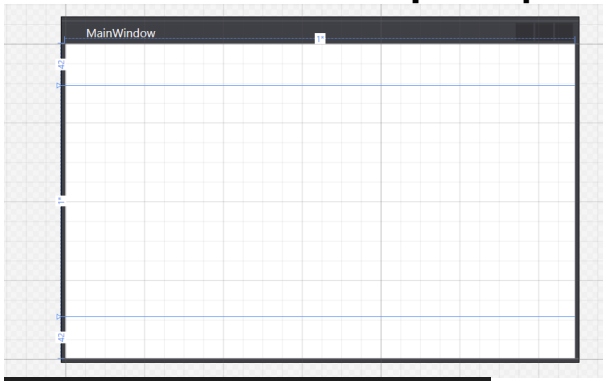
После чего на форме увидим следующее.



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).

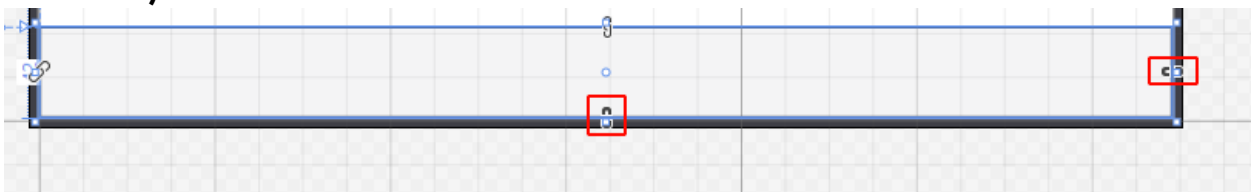
СОЗДАНИЕ БАЗОВОЙ ФОРМЫ

Создадим несколько строк с фиксированными размерами.




```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="42"/>
        <RowDefinition/>
        <RowDefinition Height="42"/>
    </Grid.RowDefinitions>
</Grid>
```

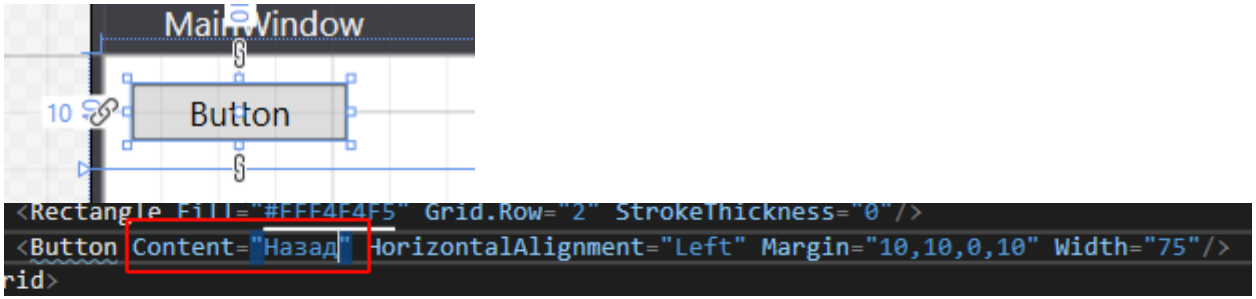
Добавим компонент «Rectangle» на форму и разместим его в 3 строке. Компонент необходимо растянуть на всю ширину и высоту строки, а затем закрепить высоту и ширину (чтобы он мог растягиваться)



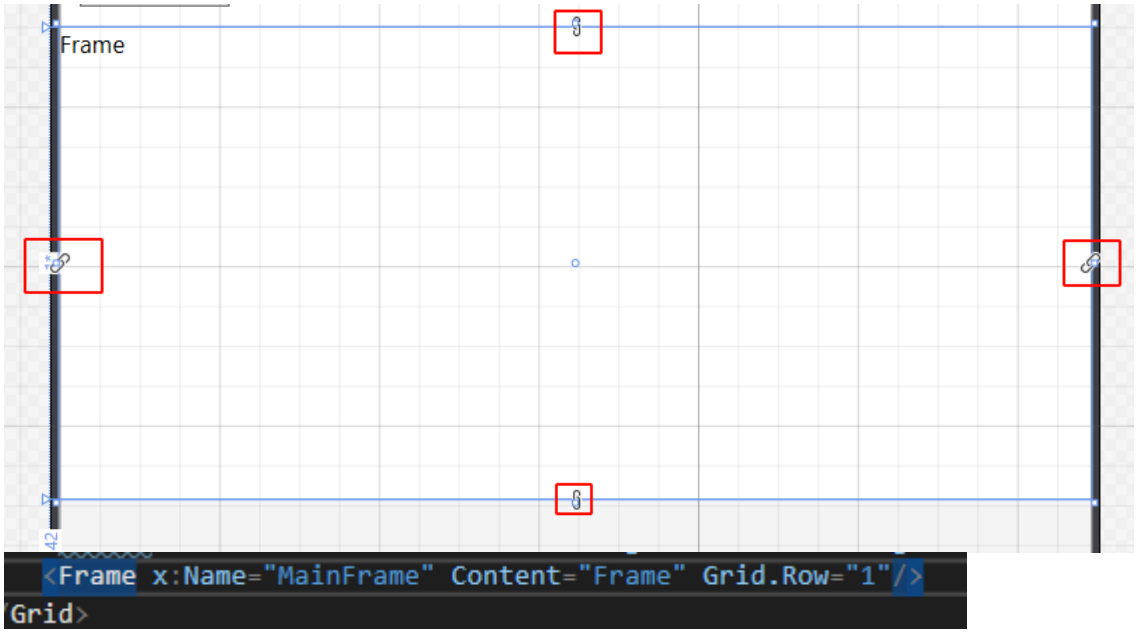
Укажем у компонента «Rectangle» свойство «StrokeThickness» равное 0, это задаст размер границы прямоугольника.

```
</Grid.RowDefinitions>
<Rectangle Fill="#FFF4F4F5" Grid.Row="2" StrokeThickness="0"/>
```

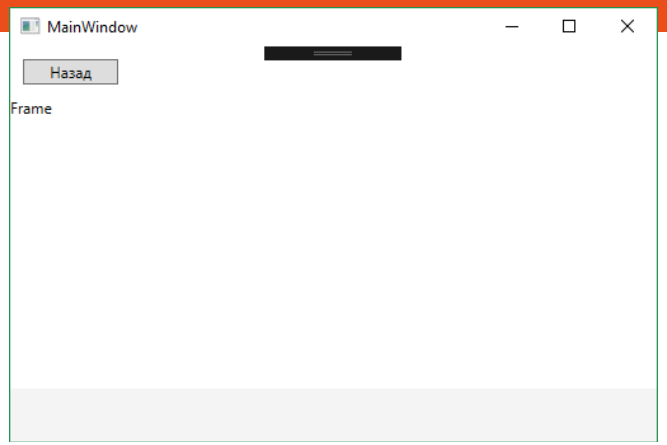
Добавим на форму кнопку и разместим в левом углу первой строки. И укажем имя кнопки «Назад», которая в последствии будет выполнять соответствующею функцию.



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»



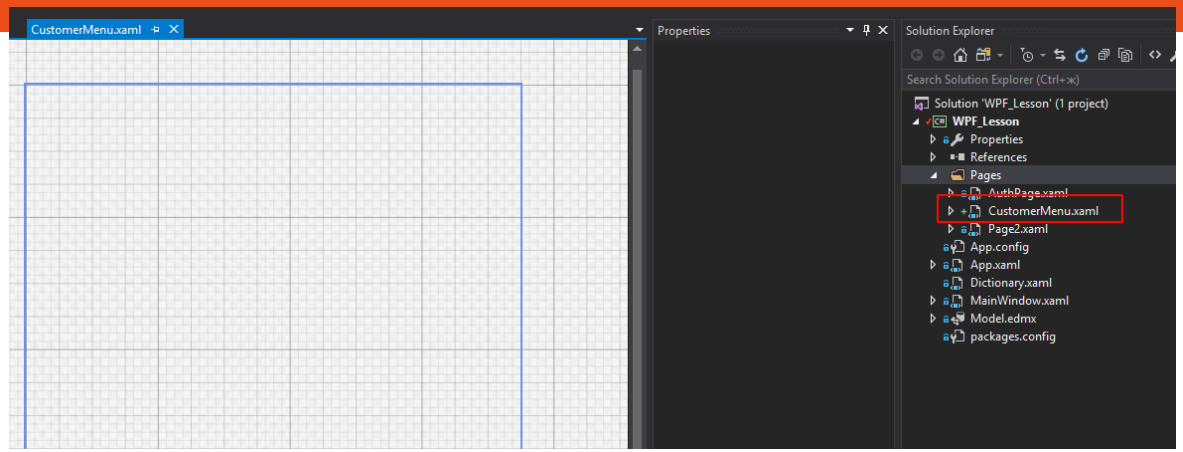
Запустим проект и увидим примерно следующее.



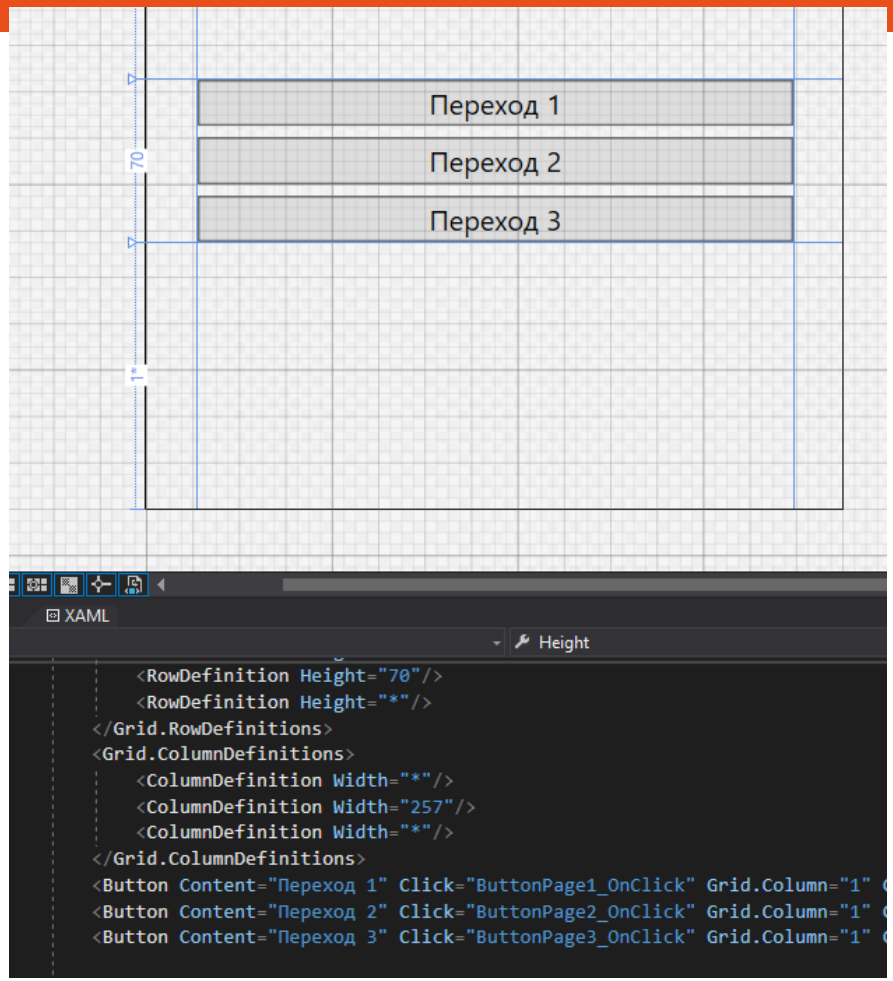
Для выполнения практического задания можно ознакомиться с обучающими видео

СОЗДАНИЕ МЕНЮ ПОЛЬЗОВАТЕЛЯ

Создадим новую страницу, назовем ее CustomerMenu.



Добавим несколько кнопок, разместим их по центру и добавим им обработчик события клик.

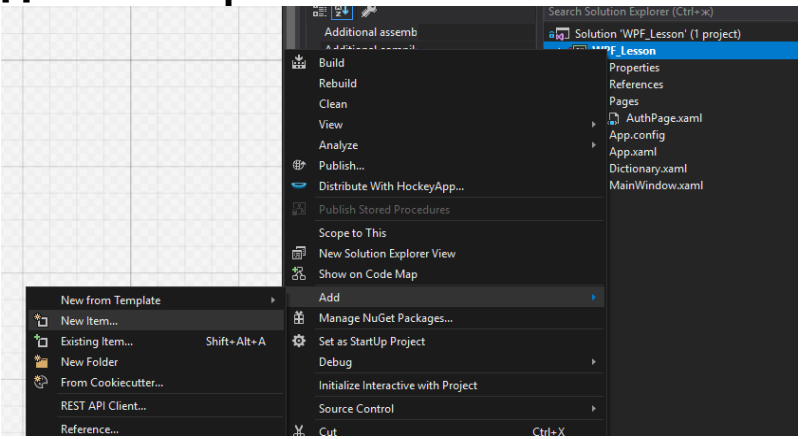


И в коде каждому обработчику укажем свой переход

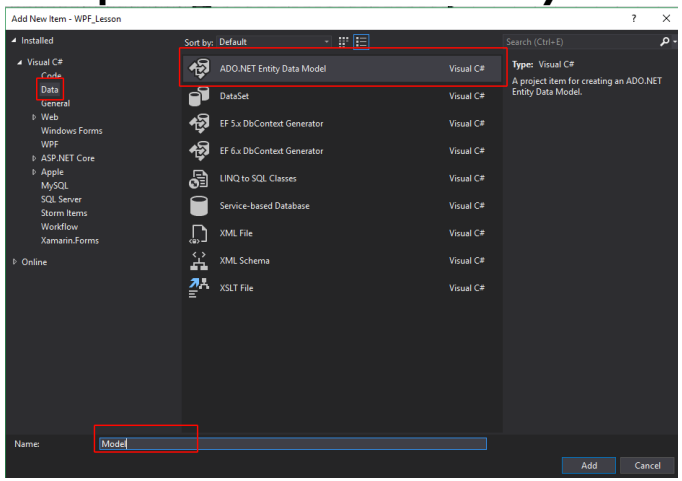
```
– references | 0 changes | 0 authors, 0 changes
private void ButtonPage1_OnClick(object sender, RoutedEventArgs e)
{
    NavigationService?.Navigate(new Page2());
}

– 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());
}
```

СОЗДАНИЕ ПОДКЛЮЧЕНИЯ К БАЗЕ ДАННЫХ

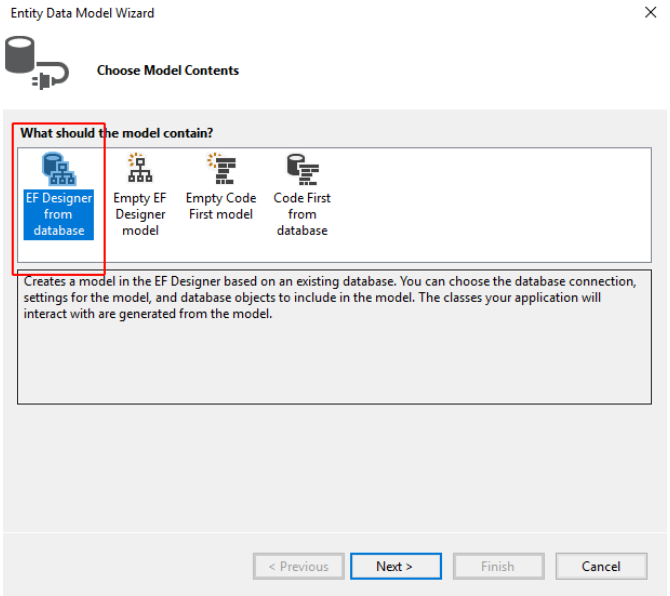
Добавим в проект новый объект



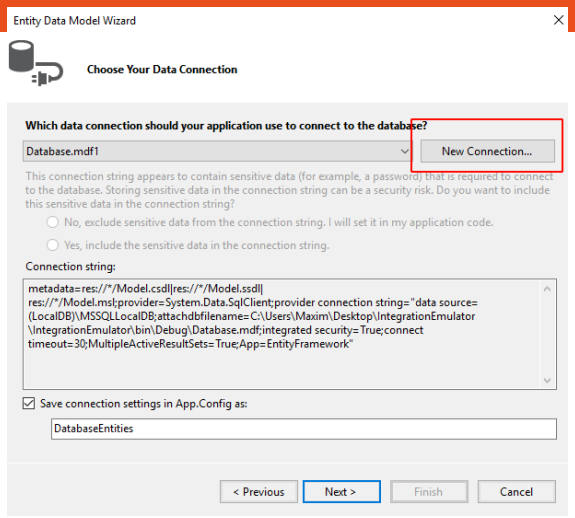
Выберем Data – ADO.NET Entity Data Model, назовем Model и добавим.



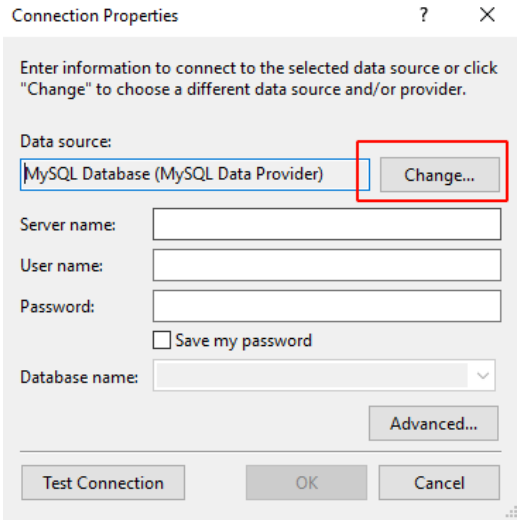
Появится диалоговое окно и выберем пункт «EF Designer from database»



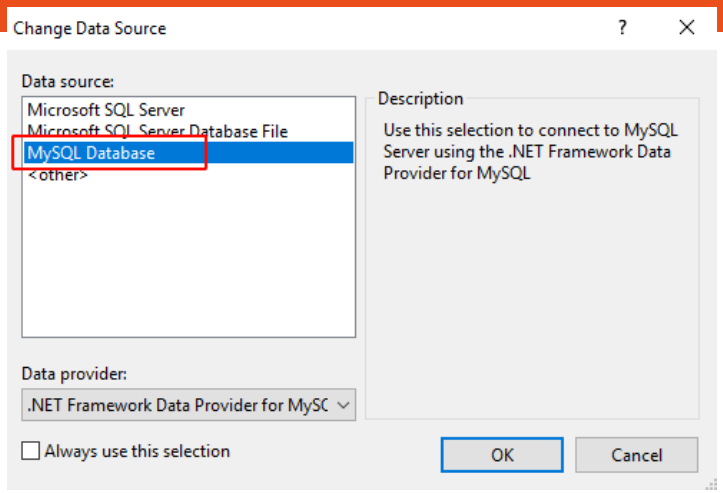
После появится следующее окно. Нажмем «New Connection»



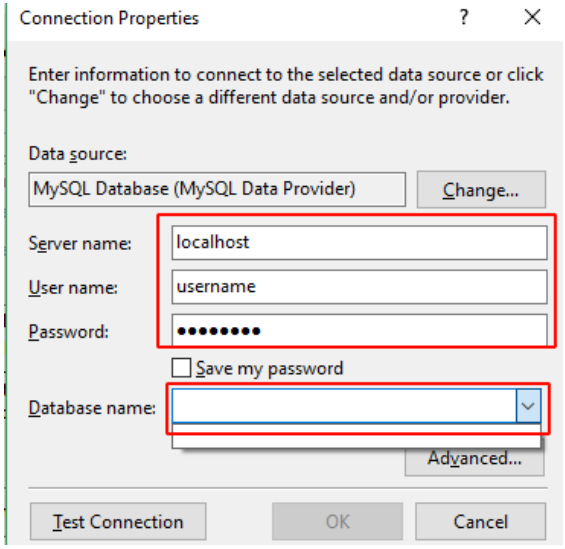
В данном окне выберем «Change».



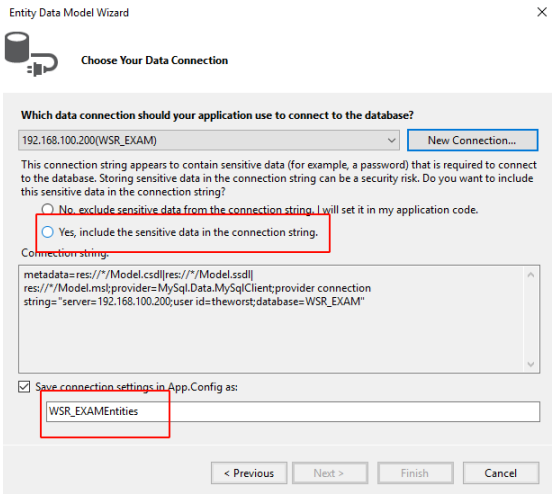
В следующем окне выберем «MySQL Database» и нажмем «OK».



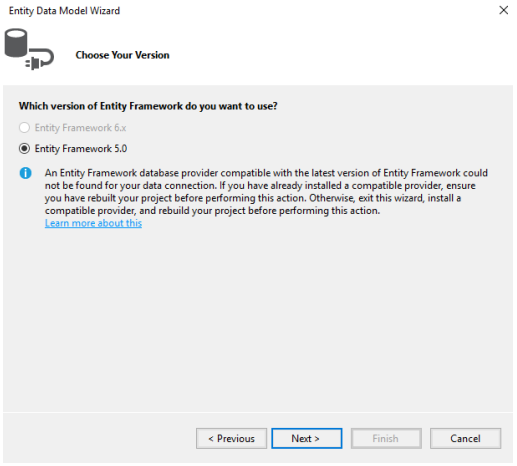
После выбора введем данные сервера: IP адрес, логин и пароль. После выберем нужную базу данных.



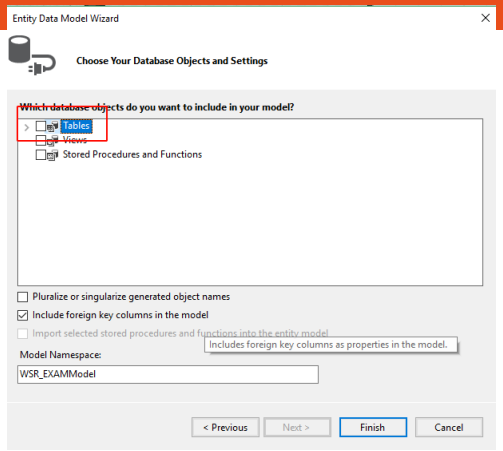
На следующем окне, выберем созданное подключение и поставим «Yes, include the sensitive data in the connection string», а также укажем имя нашего объекта базы данных «Entitites».



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();
}
```

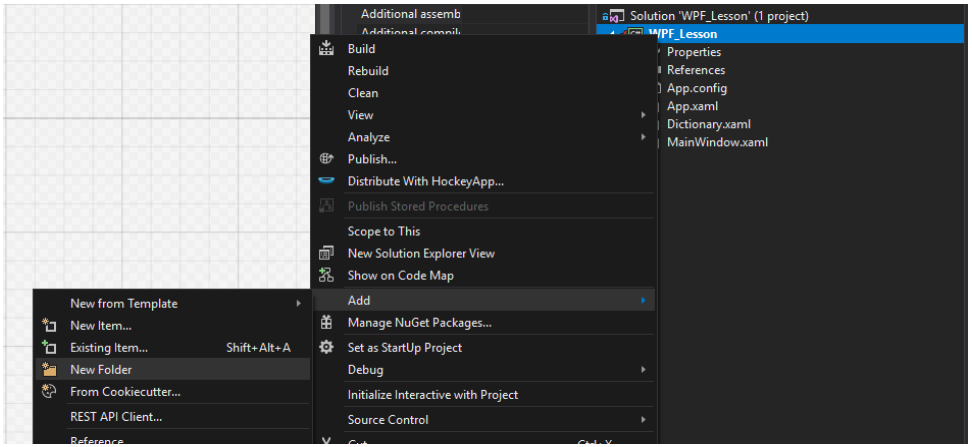
Получим пользователя по определенным критериям

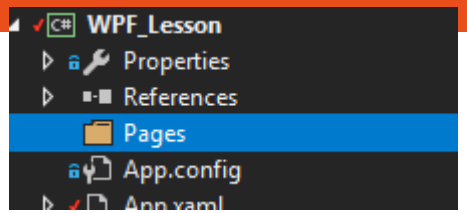
```
using (var db = new Entities())
{
    var user = db.User.AsNoTracking().FirstOrDefault(u => u.Login == "max" && u.Password == "test");
}
```

Для выполнения практического задания можно ознакомиться с обучающими видео

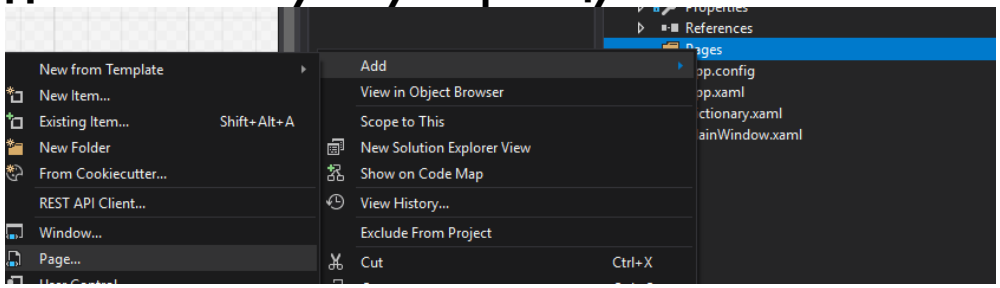
Создание формы авторизации

Добавим в проект новую папку и назовем ее «Pages», в этой папке будут находится страницы (авторизация, регистрация и другие)

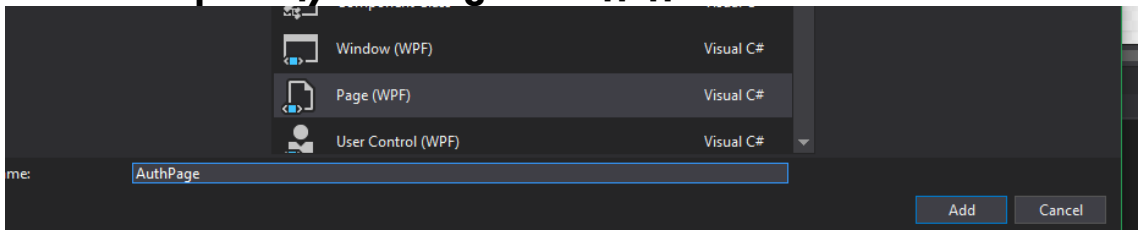


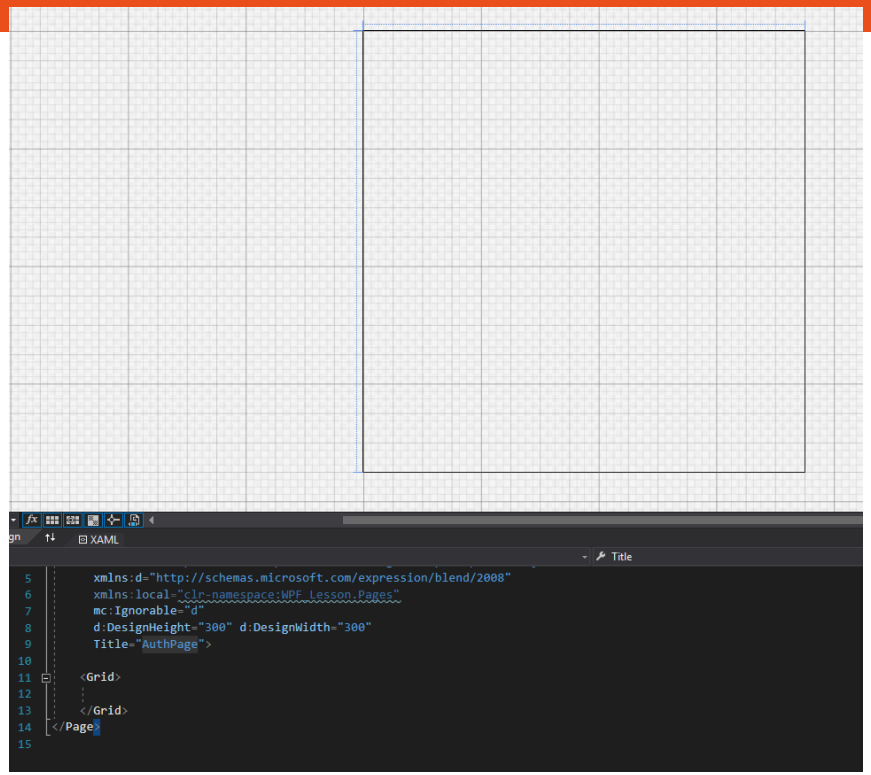


Добавим в папку новую страницу.

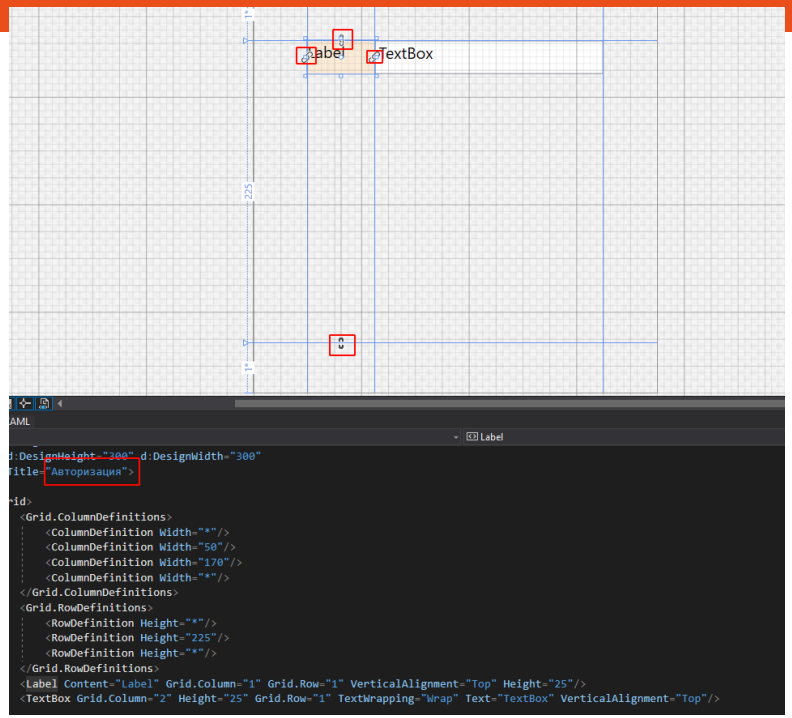


Назовем страницу AuthPage и создадим ее. После чего появится пустая страница.

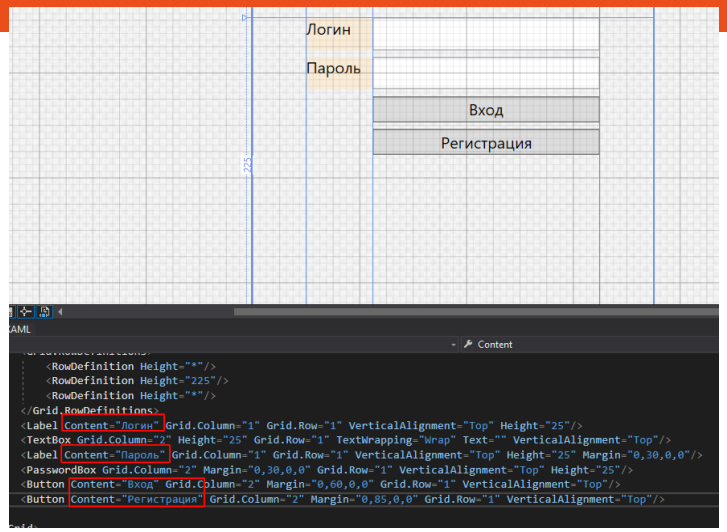




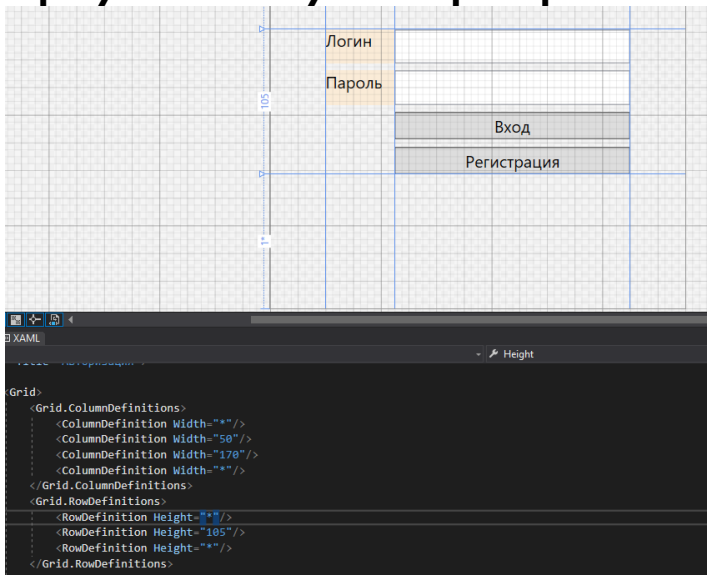
Добавим на форму компоненты «Label» и «TextBox», а затем от центруем их и добавим еще один столбец (для размещения лейблов). Также не забудем переименовать страницу, а затем закрепить лейбл и текстбокс, чтобы они растягивались по ширине.



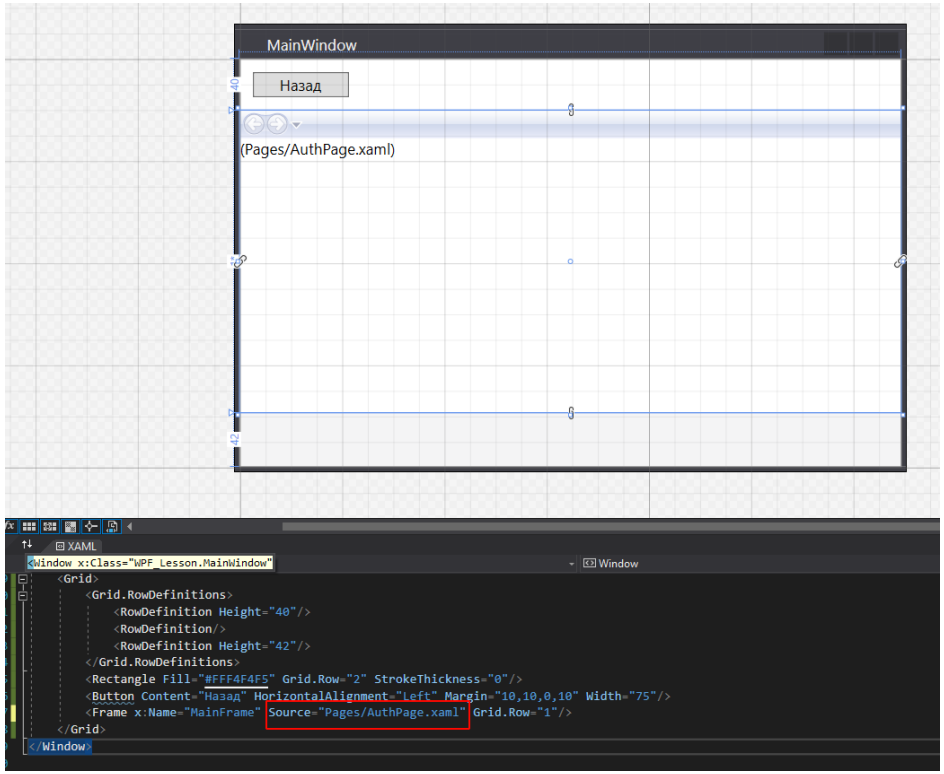
Добавим остальные компоненты по аналогии и переименуем их.



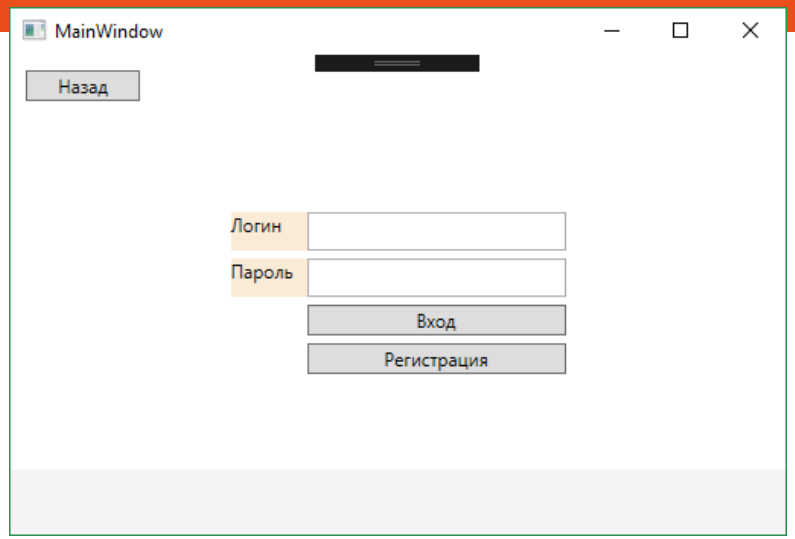
В результате получится примерно так.



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"/>
```

```
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 лабораторных работ

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>
```

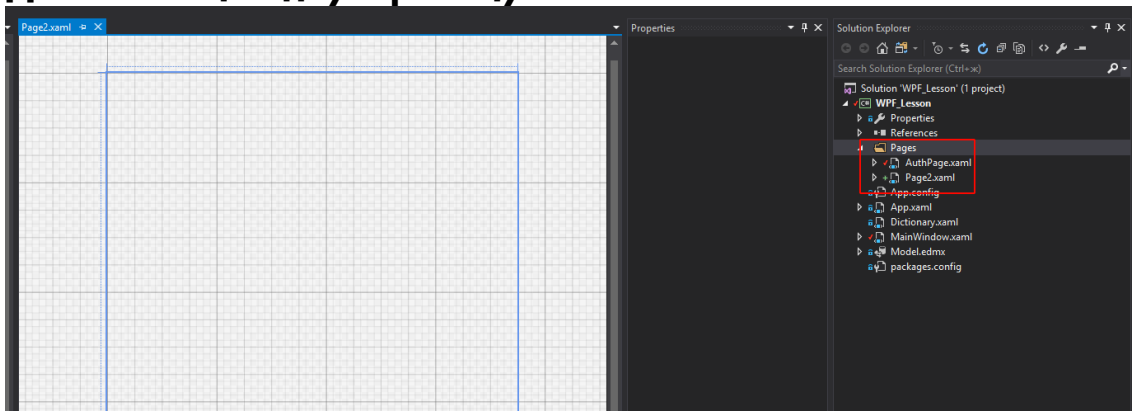
В обработчик события напишем следующий код. Сначала мы проверяем что получили ли мы страницу на вход, затем устанавливаем заголовок формы в соответствии с шаблоном, после в зависимости от страницы отображаем или скрываем кнопку «Назад».

```
0 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».**

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.

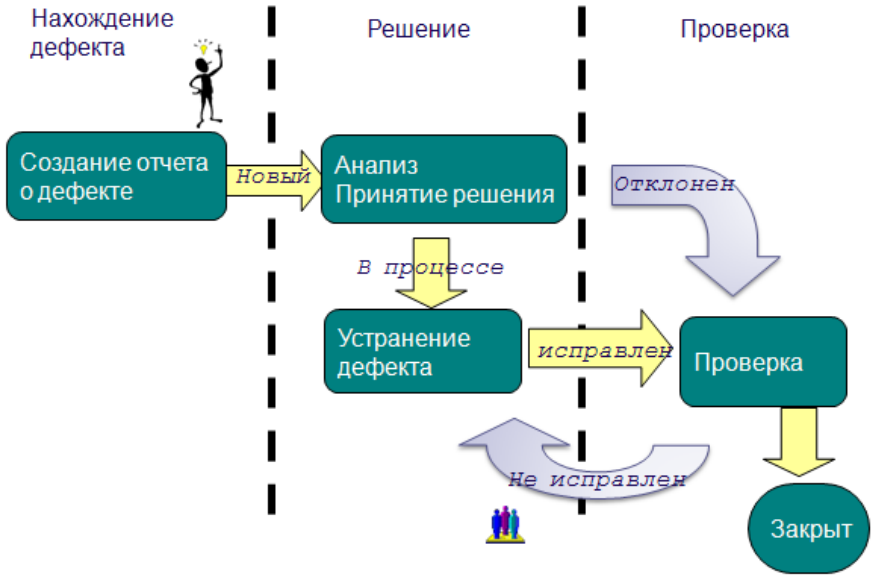


Рисунок 2 - Жизненный цикл дефекта

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.

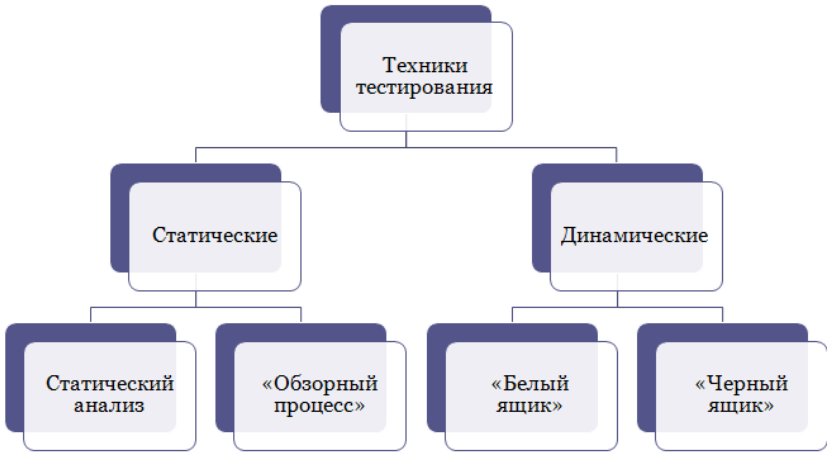


Рисунок 3 - Техники тестирования программного обеспечения

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 студента или специалистами.

Цель этого вида тестирования – изолировать отдельные части программы, протестировать их и показать, что в отдельности они работоспособны.

Преимущество этого вида тестирования в том, что студент или специалисты 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.

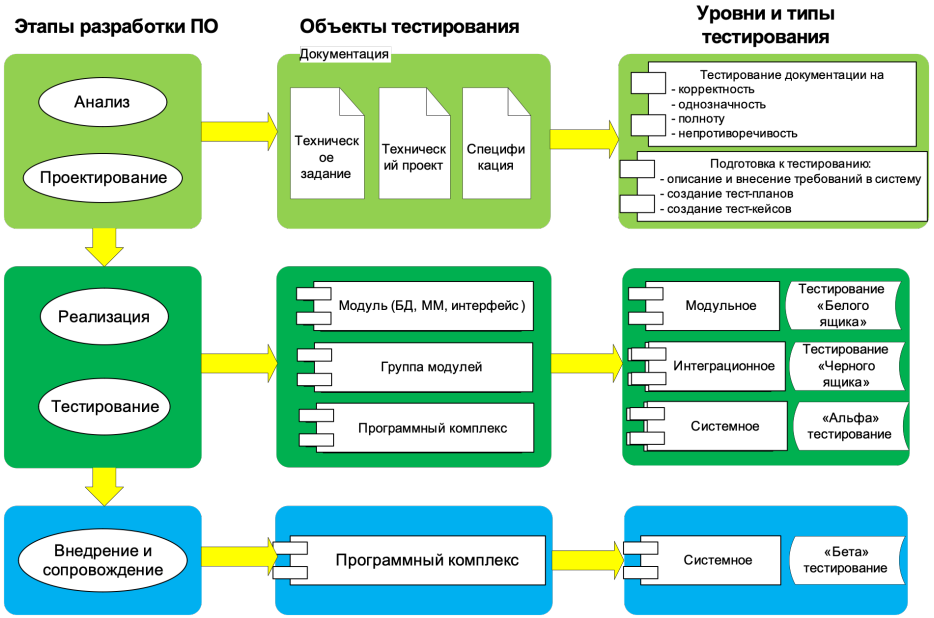


Рисунок 4 – Место тестирования в процессе разработки ПО

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:

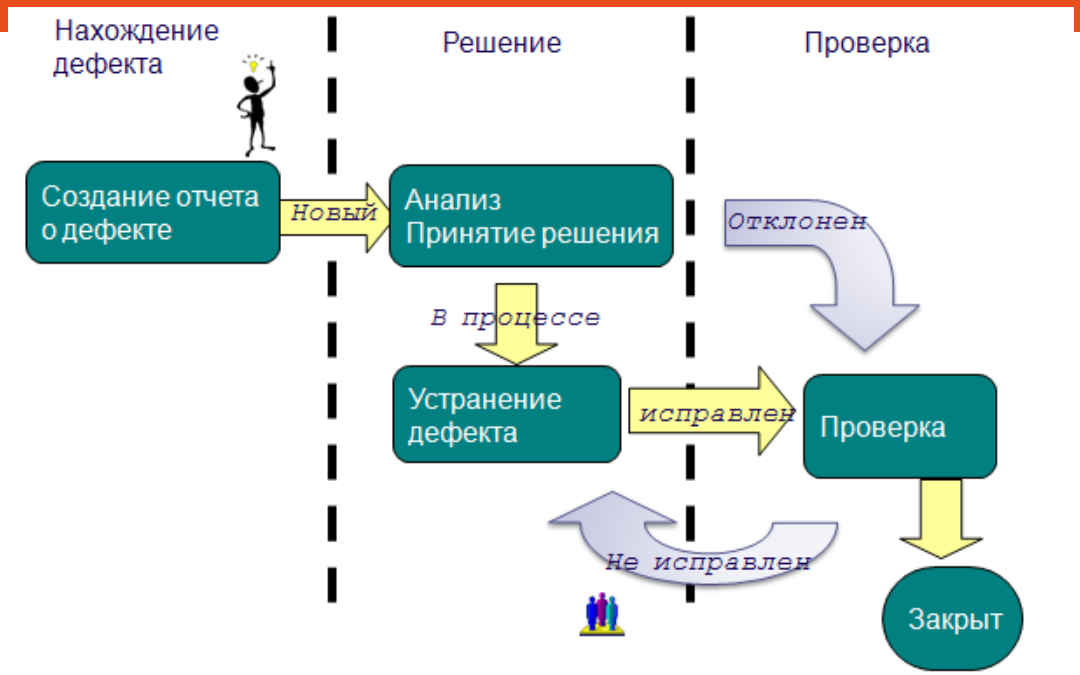


Рисунок 5 – Модель работы с дефектами

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 студента или специалиста, отвечающего за дефект, студент или специалист 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 студента или специалиста для исправления.

Всем дефектам со статусом «Исправлен» руководитель группы тестирования присваивает статус «Закрыт».

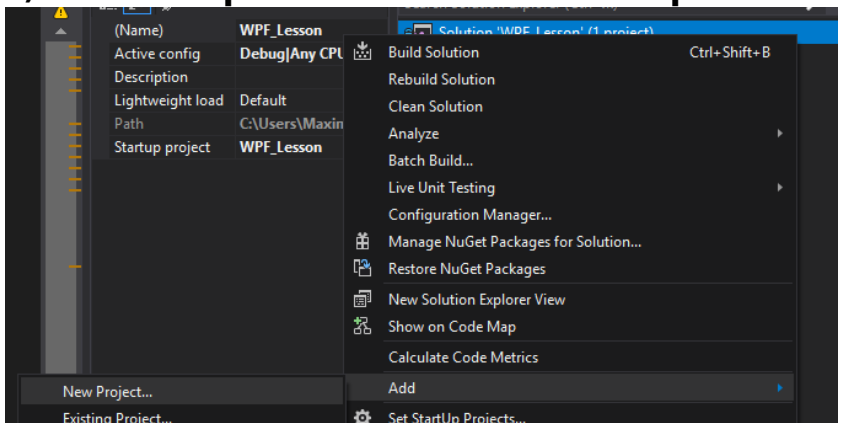
Кроме статусов атрибутами дефекта являются приоритет и важность. Приоритет дефекта показывает насколько быстро нужно исправить дефект, по приоритету определяется очередность выполнения задачи. Важность характеризует критичность дефекта по отношению к тестируемому приложению. Например, по признаку «важность» 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».

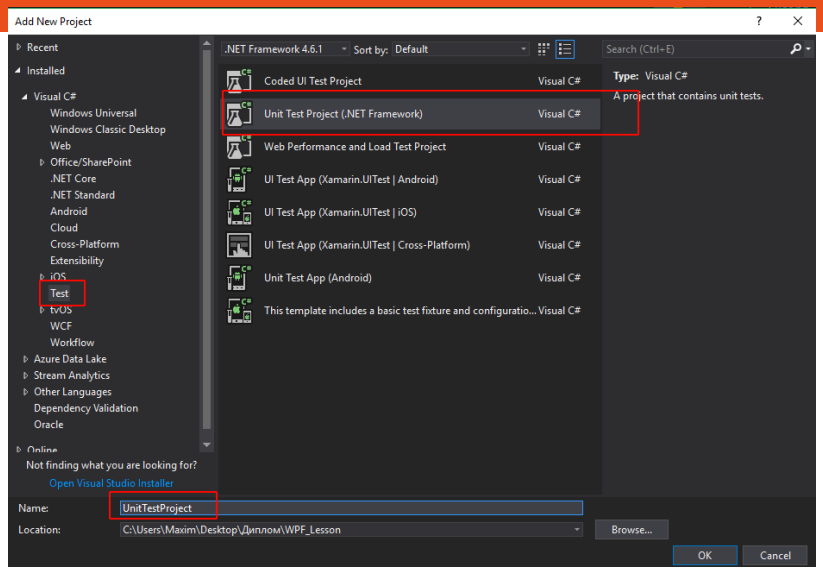
Такая модель работы с дефектами позволяет полностью контролировать состояние тестируемого проекта, распределяет роли между участниками процесса тестирования и делает процесс работы с дефектами открытым и отслеживаемым как для тестировщиков, так и для разработчиков.

Рассмотрим пример создания unit-test.

СОЗДАНИЕ ПРОЕКТА ТЕСТИРОВАНИЯ

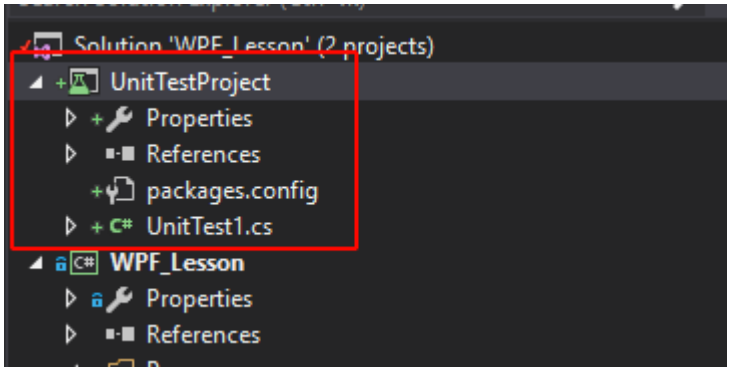
Откроем созданное решение, нажмем правой кнопкой мыши на решении и выберем «New Project»





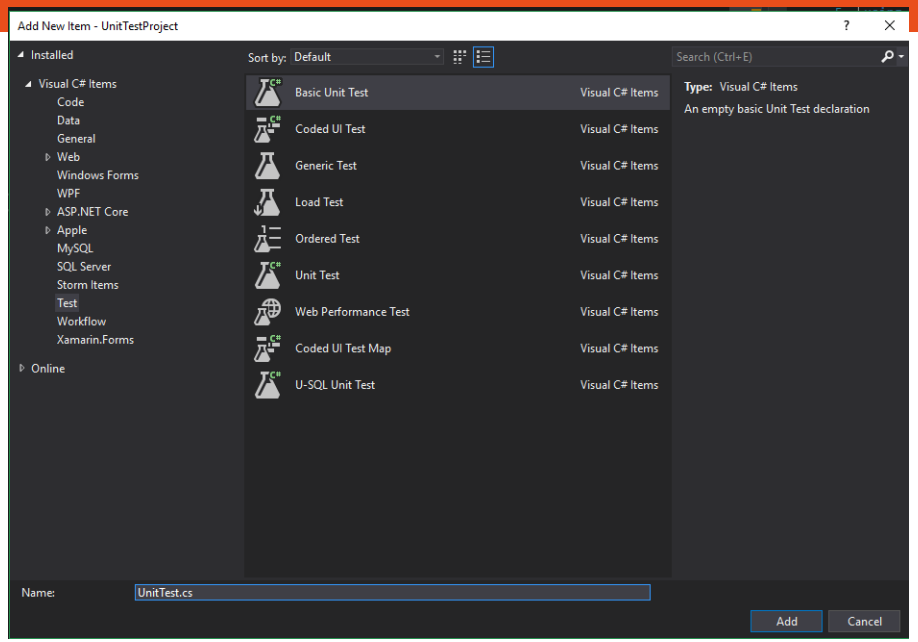
Далее выберем Test – Unit Test Project.

В решение добавится новый проект.



СОЗДАНИЕ ЮНИТ ТЕСТА

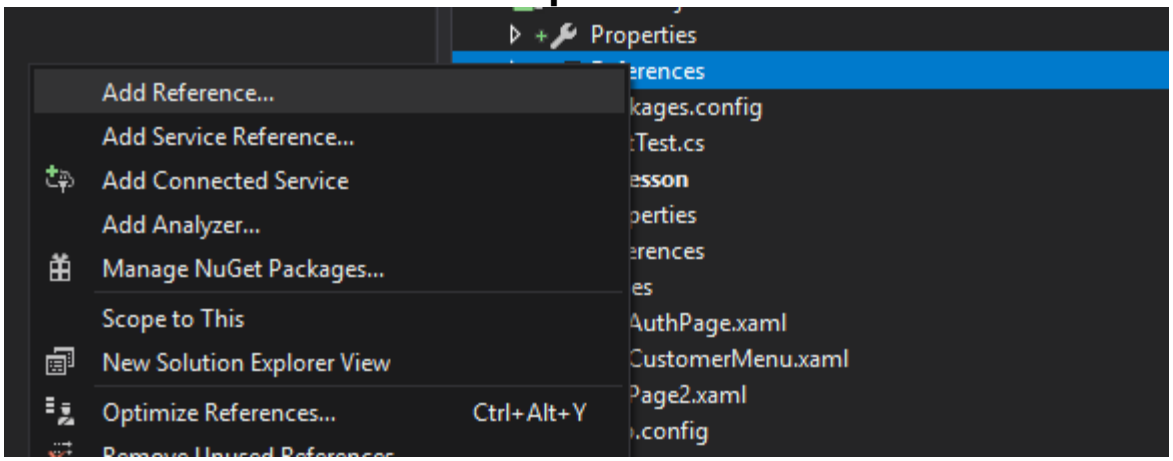
Добавим в проект базовый юнит-тест.



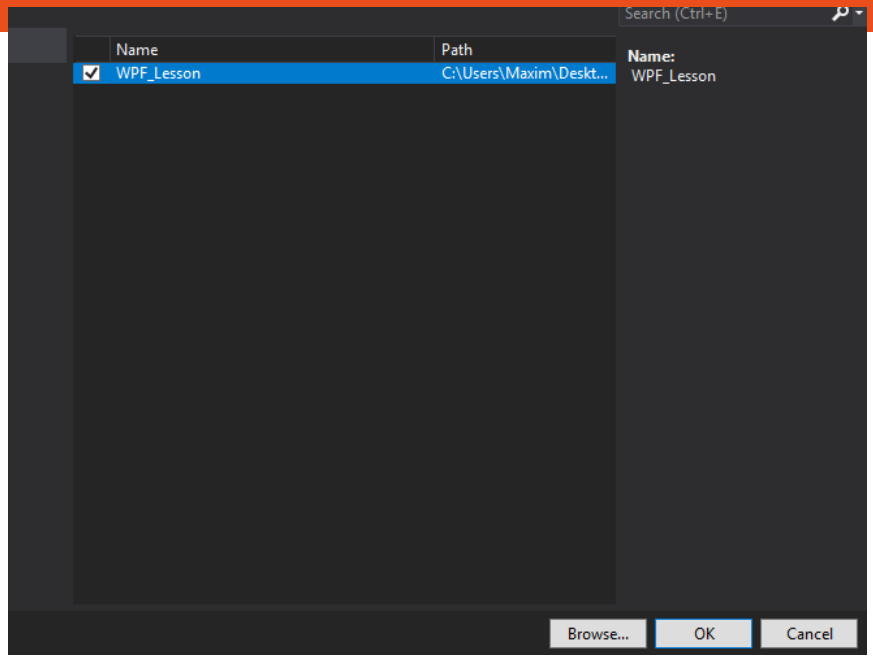
И напишем первый тест. Тест проверяет различные варианты ошибок.

```
[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:



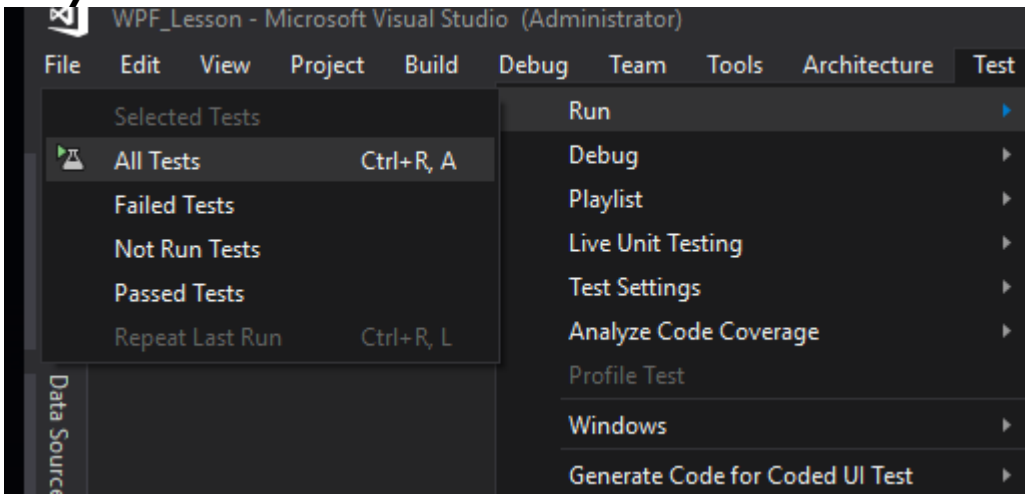
Для тестирования авторизации, нужно немного видоизменить форму авторизации:

```
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;
    }
}
```

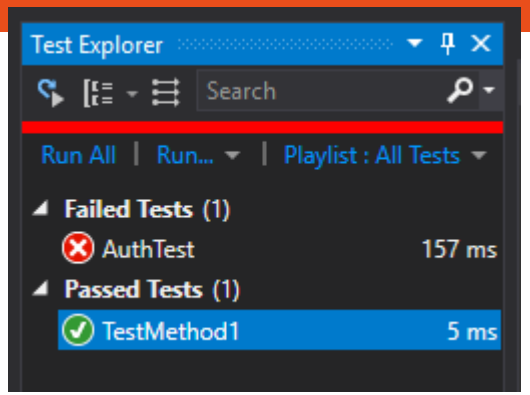
Теперь можно начинать тестировать авторизацию. Создадим новый метод для проверки авторизации и напишем следующий код:

```
[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("34645645745745745745745", "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-lkn;**
2) **Выполнить тестирование и оформить тестовую документацию в соответствие с заданием https://drive.google.com/drive/folders/1NrRGxRfGmf0HGT4fTCFvZpe1oCLd-lkn.**

ПРОЕКТИРОВАНИЕ СОДЕРЖАНИЯ УЧЕБНО-ПРОИЗВОДСТВЕННОГО ПРОЦЕССА С УЧЕТОМ СПЕЦИФИКАЦИИ СТАНДАРТА ВОРЛДСКИЛЛС

В системе современного образования необходимо разработать и внедрить в учебный процесс педагогические технологии, повышающие интенсивность, качество, уровень мотивации, привлекательность процесса познания. В системе практико-ориентированного обучения формируется следующий практический опыт: сравнения, оценки явлений, процессов, выявление причинно-следственных связей, постановка задач и необходимость дальнейшего пополнения предметных знаний. Реализация практико-ориентированного обучения предполагает рассмотрение практики как источника познания, как предмет познания с интегрированным подходом к анализу фактов, как средства познания. Поэтому организация учебного процесса в рамках практико-ориентированного подхода помогает создать уровень актуализации знаний, в котором их социальные и личные потребности реализуются вместе с познавательными потребностями.

Пример задания для проведения экзамена по профессиональному модулю:
Время выполнения задания – 8 часов.

Разработать программное обеспечение деятельности студенческого общежития. Общежитие имеет несколько корпусов. Каждый корпус характеризуется количеством комнат, а также количеством мест в каждой комнате (от 2 до 4). Информация о проживающих в каждом корпусе хранится в отдельных файлах (по одному файлу для корпуса): номер комнаты, ФИО проживающего, статус проживающего (студент или сотрудник), дата заселения, дата выселения (планируемая или фактическая), группа (для студентов) или должность (для сотрудников).
Комендант общежития должен использовать локальное приложение, позволяющее реализовать следующие функции:

1) **Изменить/добавить информацию о корпусе.**

2) Добавить информацию о вновь заселенных студентах или сотрудниках. Если студент или сотрудник оставлял заявку на заселение через сайт, информация о заявке удаляется.

3) Изменить информацию о выселенных студентах (заменить дату выселения).

4) Получить список свободных мест по каждому корпусу (сотруднику комната предоставляется целиком, т.е. место в комнате, занятой сотрудником, не является свободным).

5) Получить список свободных комнат по каждому корпусу (комната является свободной, если в ней никто не проживает).

6) Получить информацию о комнате (включая совместно проживающих) для указанного проживающего.

7) Получить список проживающих.

8) Просмотреть заявки на заселение, поступающие через сайт.

Любой желающий может просмотреть информацию об общежитии через сайт:

1) Получить информацию о корпусах общежития.

2) Получить количество свободных мест по каждому корпусу.

3) Через специальную форму оставить заявку на заселение.

Заявка на заселение содержит ФИО, статус претендента (студент или сотрудник), группу (для студента) или должность (для сотрудника), контактный телефон и адрес электронной почты. Заявки хранятся в отдельном файле с указанием даты и времени поступления заявки. Сохраняются только уникальные заявки. Заявка считается уникальной, если ранее не зарегистрировано заявок от человека с такими же ФИО и группой (должностью), ни в одной заявке не был указан тот же контактный телефон или адрес электронной почты.

Требования к программному обеспечению:

1) Формирование списка свободных мест осуществлять в отдельном потоке для каждого корпуса.

2) **Формирование списка свободных комнат реализовать в отдельном процессе, взаимодействие с которым организовать на базе одного из механизмов (анонимный или именованный канал, передача сообщений и т.п.).**

3) **Для обработки данных использовать собственные классы.**

4) **В локальном приложении предусмотреть строку состояния, отображающую наименование совершаемого в данный момент действия и текущую дату.**

5) **В локальном приложении предусмотреть всплывающие подсказки, объясняющие назначение компонент формы.**

6) **Программа должна иметь справочную систему в виде набора HTML-файлов, содержащую описание входных и выходных данных, а также инструкцию пользователя. Для вызова Справки предусмотреть отдельный пункт меню, для отображения – специальную форму.**

7) **Страницы сайта должны быть сверстаны с учетом просмотра их на различных устройствах с разными разрешениями, то есть иметь резиновую верстку для просмотра на компьютере при различных разрешениях до 1024px; иметь мобильную версию для просмотра на телефонах и планшетах вплоть до минимального разрешения – 320px.**

8) **Дата и время поступления заявки устанавливаются автоматически при попытке сохранения.**

9) **В форме заявки предусмотреть контроль корректности ввода телефонного номера и адреса электронной почты.**

План выполнения

1. **Составить спецификации входных и выходных данных.**

2. **Используя инструментальные средства разработать алгоритмы получения списка свободных мест , получения списка свободных комнат, подсчета количества свободных мест , поиска информации о комнате для указанного проживающего.**
3. **Используя инструментальные средства разработки программного обеспечения реализовать программный модуль для коменданта.**
4. **Выполнить отладку и тестирование программного модуля. Для этого создать проект тестирования; подключить библиотеки тестирования и разработанный программный проект; на каждый разработанный Вами метод разработать метод теста; реализовать все необходимые модульные тесты; осуществить тестирование и убедиться, что все тесты успешно исполняются. Результаты тестирования сохранить для демонстрации.**
5. **Разработать руководство пользователя и интегрировать его в программный модуль.**
6. **Разработать сайт, реализующий требуемые функции.**

	Системное + прикладное программирование	**70**
	Текст программы	
	Отсутствуют избыточные элементы (циклы, массивы и т.п., без которых можно обойтись)	**1**
	Отсутствуют глобальные переменные	**1**
	Отсутствуют лишние переменные, неиспользуемые блоки памяти	**1**
	Текст программы структурирован	**1**

	Программа выполнена в строгом соответствии с разработанными алгоритмами обработки данных	**1**
	Реализованы проверки на завершение функции или метода неудачей	**1**
	Реализована проверка диапазонов вводимых значений	**1**
	Реализованы значения по умолчанию	**1**
	Идентификаторы имеют значимые имена или комментарии	**1**
	Имеются комментарии к функциональным блокам и структуре программы	**1**
	Реализована функция изменения информации о корпусе	**1**
	Реализована функция добавления информации о заселенных студентах	**1**
	Реализована функция изменения информации о выселенных студентах	**1**
	Реализована функция получения списка свободных мест по каждому корпусу	**1**
	Реализована функция получения списка свободных комнат по каждому корпусу	**1**
	Реализована функция получения информации о комнате указанного проживающего	**1**
	Реализована функция получения списка проживающих	**1**
	Реализована функция просмотра заявок на заселение	
	Реализован собственный класс с несколькими полями	**2**
	В классе реализован метод, отображающий значения всех полей	**1**
	Функции обработки реализованы через методы класса	**2**

	Реализована многопоточность	**2**
	Реализована многопроцессная обработка данных	**2**
	Для ввода даты используется компонент выбора даты	**1**
	Используются системные функции для отображения текущей даты в строке состояния	**1**
	Реализован механизм взаимодействия между процессами (с использованием каналов, сообщений и т.п.)	**2**
	Работа с файлами реализована с использованием системных функций (API-функций)	**2**
	Реализована форма отображения html-файлов (справка)	**1**
	Интерфейс пользователя	
	Реализовано главное меню (все пункты меню работоспособны)	**2**
	Имеется строка состояния	**1**
	Реализована проверка корректности вводимых через поля ввода данных (отсутствует возможность ввода недопустимых значений или символов)	**2**
	Компоненты графического интерфейса имеют значимые имена	**1**
	Реализовано контекстное меню	**1**
	Имеются всплывающие подсказки	**1**
	Приложение имеет многооконный интерфейс (для работы с различными функциями программы предусмотрены отдельные формы)	**2**

	Реализовано программное заполнение элементов типа «список» и/или «таблица»	**2**
	Спецификация	
	Перечислены все входные данные	**1**
	Указаны типы входных данных	**1**
	Указаны диапазоны допустимых значений входных данных	**1**
	Перечислены все выходные данные	**1**
	Указаны типы выходных данных	**1**
	Указаны диапазоны допустимых значений выходных данных	**1**
	Документирование	
	Руководство пользователя содержит краткое описание возможностей и назначения программного модуля	**1**
	Руководство пользователя содержит описание условий применения программного модуля	**1**
	Руководство пользователя содержит описание выполняемых функций	**2**
	Руководство пользователя содержит описание операций технологического процесса обработки данных, необходимых для выполнения функций	**2**
	Руководство пользователя содержит описание аварийных ситуаций (диагностических сообщений)	**1**
	Алгоритм	

	Алгоритм структурирован	**1**
	Для описания алгоритма используются специализированные инструментальные средства	**1**
	Описание алгоритма соответствует действующим стандартам	**1**
	Алгоритмы соответствуют требованию детерминированности	**1**
	Алгоритмы соответствуют требованию массовости	**1**
	Разработан алгоритм получения списка свободных мест	**1**
	Разработан алгоритм получения списка свободных комнат	**1**
	Разработан алгоритм подсчета количества свободных мест	**1**
	Разработан алгоритм поиска информации о комнате для указанного проживающего	**1**
	Тестирование и отладка	
	На каждый разработанный метод объявлен и разработан метод теста	**2**
	Приведены результаты тестирования по каждому тесту	**2**
	Web-программирование	**30**
	Справочная система	
	Однородность текста и равномерность распределения пустых мест	**1**
	Все страницы сайта имеют одинаковый пользовательский интерфейс и выполнены в одном стиле	**1**
	Стили для справки хранятся в отдельном файле (нет inline стилей)	**1**

	Имеется вертикальное меню для работы с разделами справки	**1**
	Имеется не менее 1 страницы с описанием входных данных	**1**
	Имеется не менее 1 страницы с описанием выходных данных	**1**
	Имеется не менее 1 страницы с описанием инструкции пользователя	**1**
	Применяется блочная верстка	**1**
	Валидность HTML справки	**1**
	Сайт	
	Выполнена верстка страницы получения информации о свободных местах в общежитии	**1**
	Реализовано получение количества свободных мест	**2**
	Разработана форма заявки на заселение	**1**
	Реализована проверка корректности ввода телефонного номера с выдачей соответствующего сообщения	**1**
	Реализована проверка корректности ввода адреса электронной почты с выдачей соответствующего сообщения	**1**
	Реализована проверка уникальности заявки на заселение	**2**
	Реализовано сохранение заявок в файл	**1**
	Реализована автоматическая запись даты и времени поступления заявки	**1**
	Реализован переход между страницами сайта	**1**

	Реализован вывод на сайте информации о корпусах общежития	**1**
	Страницы сайта оформлены в едином стиле	**1**
	Однородность текста и равномерность распределения пустых мест на сайте	**1**
	Стили для сайта хранятся в отдельном файле (нет inline стилей)	**1**
	Идентичное отображение страниц сайта в браузерах Firefox, Chrome, IE	**1**
	Применяется блочная верстка	**1**
	Изменение разрешения не создает горизонтальных скроллбаров	**1**
	У всех интерактивных элементов на странице присутствуют три состояния: покой, наведение, нажатие	**1**
	Валидность HTML сайта	**1**
	Валидность CSS сайта	**1**

Шкала перевода баллов в оценку (вариативно):

Оценка "5" - от 33 баллов
Оценка "4" - 25-32 баллов
Оценка "3" - 20-24 баллов

Перечень используемого ПО:

MS Visual Studio (15 и выше)
MS SQL Server
OpenServer
MS Visio
Workbench (My SQL Connect)
MS Office
Entity Framework
Notepad++

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

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 проектирования	**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-lkn);**
2) **Подготовить задание для демонстрационного экзамена на основании предложенного шаблона в соответствие с представленной предметной областью.**