# 1. Организационные вопросы оформления на предприятии, инструктаж по охране труда и технике безопасности, распределение по рабочим местам.

Проведён инструктаж по технике безопасности на рабочем месте.

Проведено информирование работников об условиях труда на рабочих местах, об уровнях профессиональных рисков, о предоставляемых гарантиях и компенсациях за работу во вредных и опасных условиях труда.

С целью организации процедуры информирования работников об условиях труда на их рабочих местах, уровнях профессиональных рисков, а также о предоставляемых им гарантиях, полагающихся компенсациях, в УЭТК СГУ установлены следующие формы информационного взаимодействия:

1. Включение соответствующих положений в трудовой договор работника;
2. Ознакомление работника с результатами специальной оценки условий труда на его рабочем месте;
3. Проведение консультаций и семинаров по охране труда, совещаний, встреч заинтересованных сторон, переговоров;
4. Использования информационных — ресурсов в — информационно-телекоммуникационной сети «Интернет»;
5. Проведение выставок, конкурсов по охране труда;
6. Изготовление и распространение информационных бюллетеней, плакатов, иной печатной продукции, видео- и аудиоматериалов;
7. Размещение соответствующей информации в общедоступных местах.

Проведено обеспечение оптимальных режимов труда и отдыха работников.

Работодатель обеспечивает режим труда и отдыха работников в соответствии с трудовым законодательством и иными нормативными правовыми актами, содержащими нормы трудового права.

Нормальная продолжительность рабочего времени работников УЭТК СГУ не может превышать 40 часов в неделю.

Продолжительность рабочего времени для работников устанавливается исходя из сокращенной продолжительности рабочего времени не более 36 часов в неделю.

Сокращенная продолжительность рабочего времени (не более 36 часов в неделю) также устанавливается для работников учреждения, условия труда которых по результатам специальной оценки условий труда отнесены к вредным условиям труда 3 или 4 степени.

К мероприятиям по обеспечению оптимальных режимов труда и отдыха работников УЭТК СГУ относятся:

1. обеспечение рационального использования рабочего времени;
2. обеспечение внутрисменных перерывов для отдыха работников, включая перерывы для создания благоприятных микроклиматических условий;
3. поддержание высокого уровня работоспособности и профилактика утомляемости работников.

Распределение по рабочим местам происходит в соответствии с нормативными положениями организации.

# 2 Ознакомление со структурой и характером деятельности предприятия

Университетский экономико-технологический колледж является структурным подразделением Сочинского государственного университета.

Педагогический штат колледжа насчитывает 110 профессионалов, в том числе кандидатов наук, преподавателей высшей и первой квалификационных категорий.

В настоящее время в колледже веется подготовка по 15 специальностям среднего профессионального образования. Для качественного обеспечения процесса обучения УЭТК обладает развитой материально-технической базой.

В колледже функционирует более 50 лабораторий и учебных кабинетов, оснащенных современным оборудованием: лаборатории информационно-коммуникационных систем, делопроизводства и оргтехники, служба приема и размещения гостей, товароведения продовольственных товаров и продукции общественного питания, учебная бухгалтерия, учебный банк, технологии разработки баз данных, системного программирования, учебный кулинарный цех, учебный кондитерский цех, другие лаборатории. Учебные кабинеты оснащены интерактивными досками, проекторами, комплектами наглядных пособий.

Вычислительный центр является одним из структурных подразделений Университетского экономико-технологического колледжа. Вычислительный центр взаимодействует с различными отделами колледжа по вопросам аппаратного и программного обеспечения компьютерной техники, используемой для непосредственной деятельности отделов.

# 3 Сбор материалов для составления технического задания по теме дипломной работы

## 3.1 Анализ структуры обучающих электронных приложений

В качестве обучающих электронных приложений для анализа взяты популярные приложения для самостоятельного изучения различных языков программирования.

Одним из самых популярных сервисов для изучения языков программирования в России, исходя из веб аналитики, предоставленной на ресурсе www.similarweb.com и оценок пользователей на площадке Google Play является «Sololearn». Этот сервис представляет собой локальную инфраструктуру, которая доступна посредством интернет-сайта и мобильного приложения и которая состоит из модулей:

1. Теоретического обучения языку программирования, состоящая из лекционных материалов и тестов к ним;
2. Практических занятий с языком программирования посредством онлайн компиляторов, предоставляемых сервисом;
3. Онлайн блога для обсуждений и соискания помощи в решении вопросов по программированию;
4. Авторизации пользователей (возможна авторизация посредством аккаунта Google, Apple и Facebook).

Теоретические и практические занятия по одной и той же теме совмещены и предоставляются последовательно в соответствующих курсах различных уровней сложности с разграничением на начинающих, продвинутых и профессиональных программистов. Так же, в присутствующем каталоге есть выбор прохождения курсов по конкретному языку.

Другой популярный ресурс для изучения программирования – «Codecademy».

Данный ресурс состоит из интернет-сайта и мобильного приложения. Из модулей можно выделить модуль авторизации, форум и теоретический модуль, совмещённый с итоговым заданием (практикой) по окончанию каждой темы.

Проанализировав структуру подобных приложений, можно приступать к выбору средств разработки.

## 3.2 Анализ программных средств, необходимых для разработки приложения по основам программирования на языке С++

Для разработки десктопного приложения необходимо выбрать язык и фреймворк под нужные задачи.

Electron JS — это JavaScript фреймворк, позволяющий написать полноценную программу используя Web технологии — HTML & CSS, JS. Electron JS представляет собой движок Chromium в котором исполняется весь код. У Electron'a есть один достаточно серьезный недостаток — большое потребление памяти, как физической, так и оперативной. Сегодня многие популярные приложения написаны на Electron'e — «Slack», «Skype», «Discord», «VSCode», «Atom», «Postman», «Insomnia» и другие. А с учетом непрекращающегося роста мощности компьютеров, все реже возникает проблема с переполнением памяти. Высокое потребление памяти не будет играть большой роли, если продукт будет хорош в своей сфере, будет грамотно написан код и распределены процессы.

Qt — очень мощный набор инструментов для создания кроссплатформенных приложений на языке С++ и Python (а также Ruby, PHP, C# и другие, но они не поддерживаются сообществом). Qt достаточно старый фреймворк, но он продолжает активно развиваться и на нем написаны такие программы как: «2ГИС» для Android, «Kaspersky Internet Security», «Virtual Box», «VLC Media Player», «Opera» и другие. Окружение рабочего стола KDE тоже написано с использованием Qt. Qt имеет среду разработки — «Qt Creator», которая включает в себя «Qt Designer» с помощью которого можно создавать графический интерфейс. Визуальное создание интерфейса позволяет легко и просто создавать интерфейс, перетаскивая различные виджеты (выпадающие списки, кнопки, переключатели) на форму. Qt имеет обширную документацию, активную поддержку сообщества и множество других преимуществ.

Windows Presentation Foundation (WPF) - бесплатная графическая подсистема с открытым исходным кодом (аналогичная WinForms), первоначально разработанная Microsoft для визуализации пользовательских интерфейсов в приложениях на базе Windows. WPF доступен для скачивания в пакетах среды разработки Microsoft Visual Studio. Он прост в написании понятного и быстрого кода. Язык разметки XAML, в сочетании с конструктором. позволяет гибко настраивать пользовательский интерфейс разрабатываемого приложения. WPF до сих пор поддерживается компанией Microsoft и имеет огромное количество расширений и пакетов, доступных для скачивания из пакетного менеджера Visual Studio «NuGet».

Для реализации модуля авторизации пользователей и модуля обучения понадобится реляционная база данных. Среди подобных выделяются SQLite, PostgreSQL и MySQL.

SQLite — это библиотека, встраиваемая в приложение, которое её использует. Будучи файловой базой данных, она предоставляет набор инструментов для более простой (в сравнении с серверными базами данных) обработки любых видов данных. Когда приложение использует SQLite, их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные (например, баз данных SQLite), а не какого-то интерфейса, что повышает скорость и производительность операций.

PostgreSQL — это самая продвинутая реляционная система управления базами данных, ориентирующаяся в первую очередь на полное соответствие стандартам и расширяемость. PostgreSQL, или Postgres, пытается полностью соответствовать SQL-стандартам ANSI/ISO. PostgreSQL отличается от других РСУБД тем, что обладает объектно-ориентированным функционалом, в том числе полной поддержкой концепта ACID (Atomicity, Consistency, Isolation, Durability). Будучи основанным на мощной технологии Postgres отлично справляется с одновременной обработкой нескольких заданий. Поддержка конкурентности реализована с использованием MVCC (Multiversion Concurrency Control), что также обеспечивает совместимость с ACID.

MySQL — это самая популярная из всех крупных серверных БД. Разобраться в ней очень просто и в сети о ней можно найти большое количество информации. Хотя MySQL и не пытается полностью реализовать SQL-стандарты, она предлагает широкий функционал. Приложения общаются с базой данных через процесс-демон.

Исходя из проведённого анализа, принято решение о разработке приложения на фреймворке C# WPF с подключением к серверной системе управления базами данных MySQL.

## 3.3 Описание этапов разработки подсистем

Для воспроизведения необходимых подсистем приложения, сперва необходимо разработать модель базы данных.

База данных должна поддерживать работу модулей «Авторизация» и «Обучение».

Разработка модели базы данных (Рисунок 1) осуществляется в приложении для создания диаграмм «StarUML».

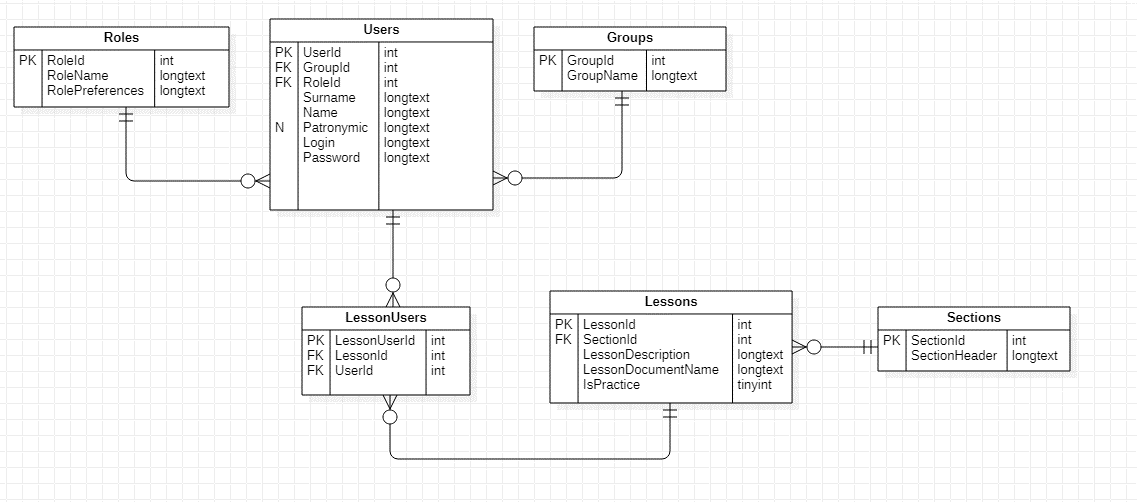


Рисунок 1 – ER-диаграмма базы данных

Среди сущностей базы данных, на диаграмме представлены таблицы:

1. Roles – для хранения ролей пользователей;

2. Groups – для хранения групп пользователей (как учебных групп студентов, так и рабочих групп модерации и администраторов);

3. Users – для хранения пользователей, информации об их роли, группе, личных данных и данных для авторизации;

4. Sections – для хранения названий разделов лекционного материала;

5. Lessons – для хранения пути до лекционных материалов внутри приложения и дополнительной информации о них;

6. LessonUsers – для записи истории просмотренных пользователем лекций.

Исходя из нужных функций, в базе данных будет храниться информация для работы с авторизацией пользователей и данные о предоставляемом лекционном материале по изучению языка C++. Ради увеличения быстродействия, сами материалы будут храниться непосредственно в локальной директории приложения, а на сервере будет храниться лишь относительный путь до файла.

После проектирования базы данных, в Visual Studio создаются два проекта: приложение WPF .NET Core 6.0 и библиотека классов C#. Проекты создаются для сборки приложения и модели базы данных соответственно. Происходит установка дополнительных пакетов для проектов.

В проекте библиотеки классов создаются классы-модели таблиц спроектированной базы данных со свойствами-столбцами, после чего создаётся файл контекста, в котором прописывается строка подключения к базе данных со всеми необходимыми настройками. Всё это происходит посредством ORM-системы, которая связывает базы данных с концепциями объектно-ориентированных языков программирования.

После, заполняется база данных. В таблице «Roles» создаются роли: студент, модератор, администратор. В таблице «Groups» создаётся несколько групп. Таблицы «Sections» и «Lessons» заполняются информацией о лекционном материале. Таблица «LessonUsers» не нуждается в заполнении.

В проекте WPF используется архитектура MVVM (Model, View, ViewModel). Она подразумевает под собой то, что проект WPF разбивается на тройную структуру со структурными единицами в соответствии с расшифровкой аббревиатуры.

К категории «Model» относятся все файлы, которые никак не взаимодействуют с пользовательским интерфейсом и являются, например, объектами передачи данных базе данных или конвертерами данных (хэширование паролей).

«View» же, наоборот, всё, что имеет отношения только к работе пользовательского интерфейса приложения (все файлы разметки и пользовательских элементов интерфейса).

Всё что находится между этими элементами обрабатывается файлами «ViewModel» – связующими звеньями и посредническими классами в работе приложения.

Главное правило данной архитектуры гласит, что «View» не должен знать о существовании «Model» и наоборот.

Смысл использования MVVM состоит в том, чтобы упростить разработку и масштабирование проекта WPF. Используя этот метод, можно легко отследить неполадки в работе приложения, провести диагностику, а также, упростить разработчику навигацию по проекту при разработке.

Была создана диаграмма прецедентов, в которой представлены все доступные для каждой роли функции (Рисунок 2).

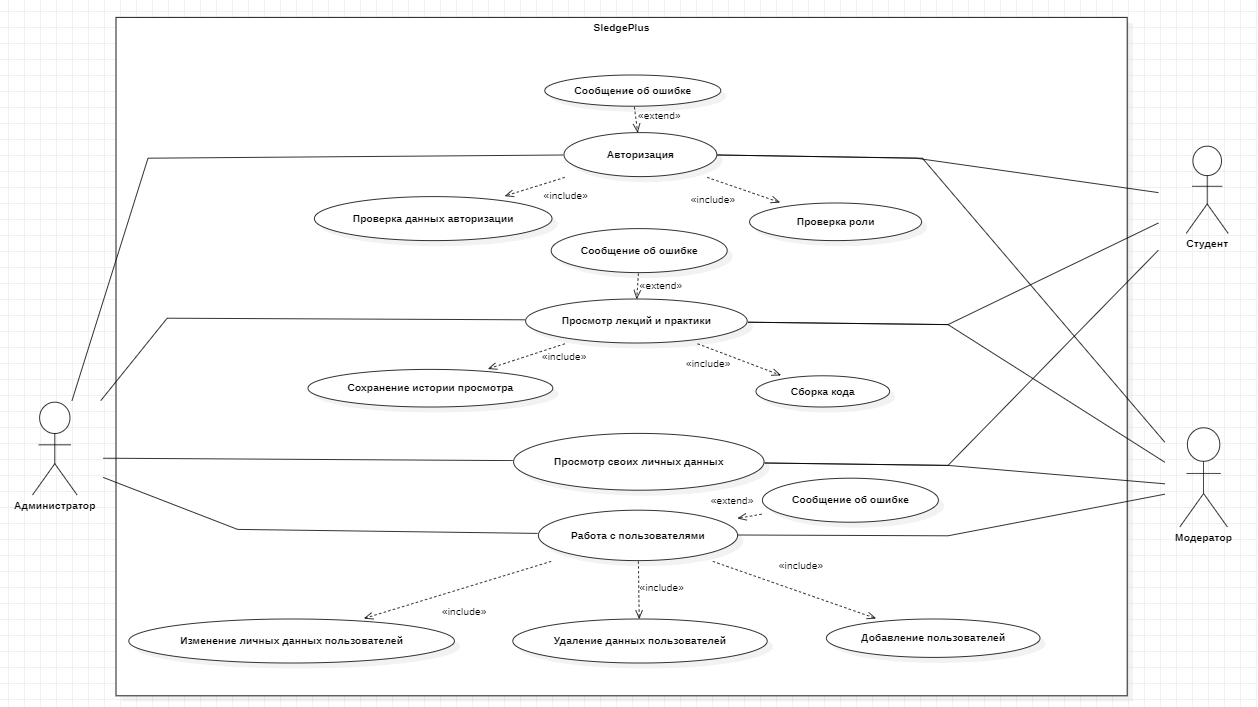


Рисунок 2 – Use-case диаграмма

Приложение состоит из единственного окна. Навигация по приложению будет происходить через кнопки в верхней части окна, которые будут менять текущий пользовательский элемент как страницу.

Схема взаимодействия интерфейса приложения представлена на Wireframe диаграмме (Рисунок 3).

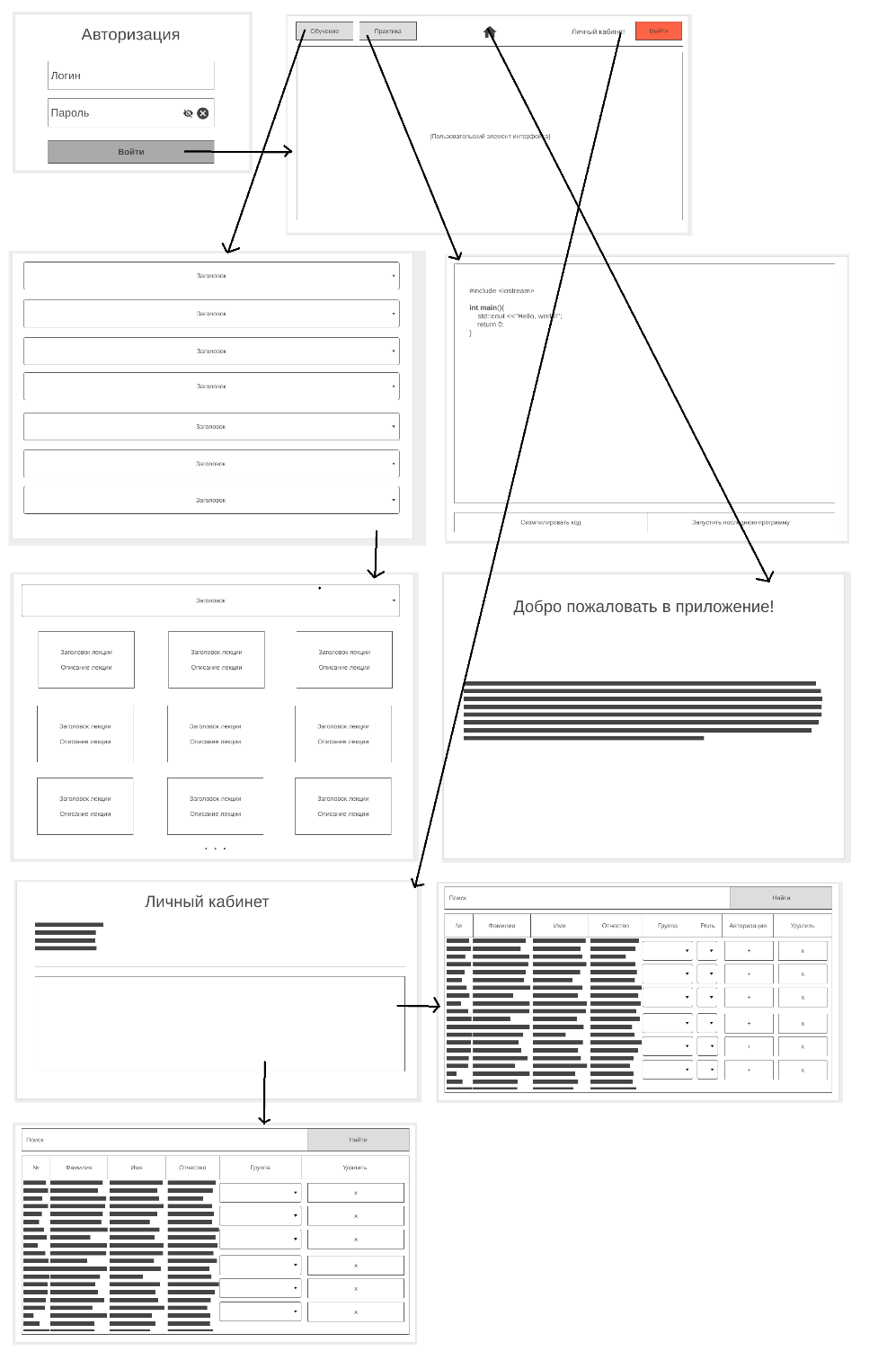


Рисунок 3 – Схема логического взаимодействия между структурными элементами приложения

Работа подсистем приложения описывается диаграммами перехода состояний (Рисунок 4).

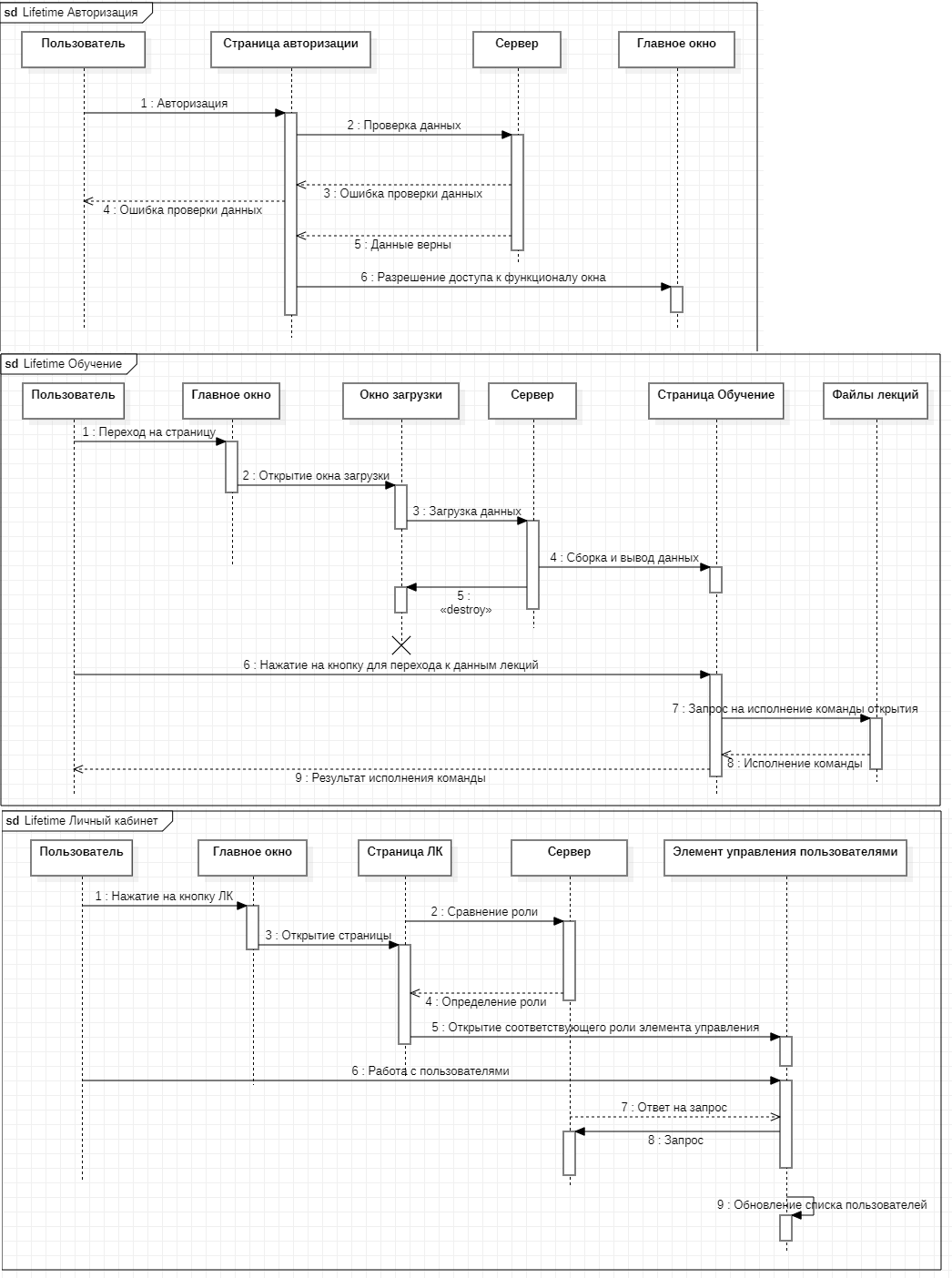


Рисунок 4 – Диаграммы переходов состояний

В проект WPF добавляется подсистема авторизации пользователя. В этот модуль будет интегрирована система хэширования паролей методом шифрования SHA-256 для повышения безопасности работы системы. Визуализация модуля будет представлена в виде начальной страницы авторизации пользователя, а после её успешного прохождения, в виде страницы личного кабинета, где у различных ролей будут находиться соответствующие им элементы управления пользователями, где модератор сможет добавлять, изменять и удалять личные данные пользователей, а администратор сможет производить любые манипуляции с пользователями. После успешного прохождения авторизации, приложение пользователя направит на приветственную страницу с текстом описания модулей и навигации по ним.

Создаётся подсистема обучения. В соответствующую вкладку добавится пользовательский элемент, «ViewModel» которого берет информацию о структуре загруженного в базу данных курса обучения с сервера. Файлы с лекционным материалом будут браться из директории по пути установленного приложения.

Добавляется подсистема практики. В проект встраивается компилятор кода C/C++. В соответствующую вкладку добавится текстовый редактор кода с возможностью компиляции и вывода результата кода в терминал.

После проектирования плана разработки приложения можно переходить к его реализации.

4 Разработка программного обеспечения на основе технического задания по теме дипломного проекта

На основе данной модели создаётся библиотека классов C#.

Устанавливаются NuGet-пакеты:

1. Entity Framework Core – пакет для работы с ORM проектированием базы данных;

2. Microsoft.Hosting и Microsoft.DependencyInjection – пакеты для работы с внедрением зависимостей и упрощению взаимодействия с доступом к экземплярам классов;

3. MaterialDesign – набор готовых пользовательских элементов интерфейса для облегчения работы с WPF и улучшения внешнего вида приложения;

4. AvalonEdit – пакет с пользовательским элементом интерфейса для работы с текстовым редактором для модуля практики.

Используя ORM-инструмент Entity Framework, база данных собирается на сервере из заданной в файле контекста информации (Рисунок 5).



Рисунок 5 – Контекст базы данных

Сборка происходит через команды Entity Framework’а (Рисунок 6).

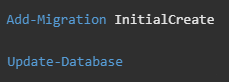


Рисунок 6 – Команды для сборки базы данных

Команды вводятся в консоль диспетчера пакетов последовательно. Команда Add-Migration [\_] принимает в качестве аргумента произвольное название миграции. Миграция – это автоматически генерирующийся код, который фиксирует внесённые в модель базы данных изменения, которые преобразуются в SQL-запрос и последовательно передаются на сервер после ввода команды Update-Database. Такой подход обеспечивает не только контроль поведения базы данных, но и мгновенный доступ к модели базы данных в связанных проектах без использования SQL-запросов.

База данных заполняется через средства веб-приложения «phpMyAdmin» (Рисунок 7) или через десктопное приложение для работы с запросами SQL и данными MySQL Workbench.

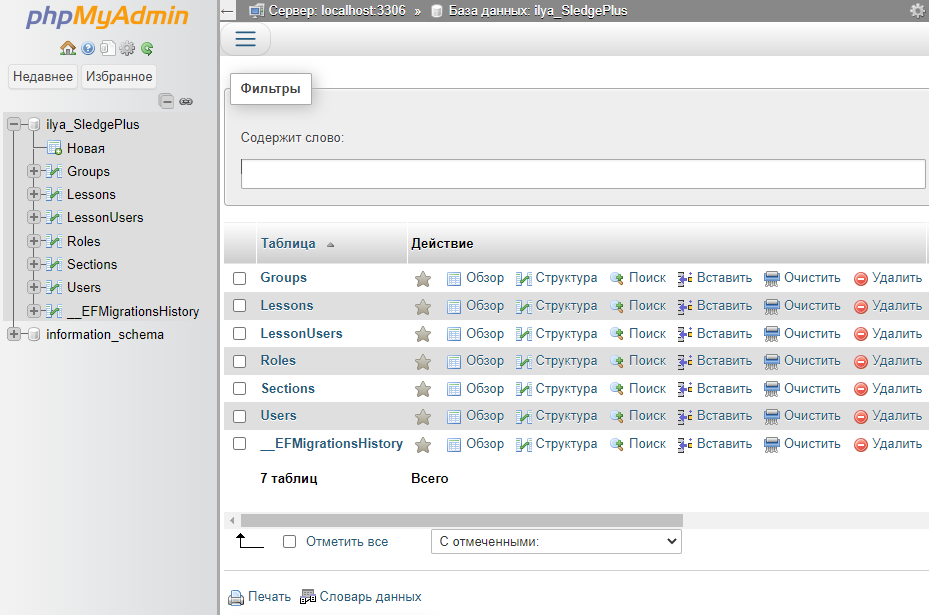


Рисунок 7 – Интерфейс веб-приложения phpMyAdmin

По умолчанию создаются роли: «Администратор», «Модератор» и «Студент»; группа «-»; пользователь «admin» с паролем «Admin123».

Дополнительно в базу данных импортируется информация об учебных материалах.

На этом этапе, создание и настройка базы данных завершена.

Создаётся пустой шаблонный проект C# WPF, который сразу разбивается на каталоги в соответствии с архитектурой MVVM.

В каталоге «View» размещается главное окно приложения (Рисунок 8). Оно состоит из поля навигации в верхней части и контейнера элементов интерфейса в нижней

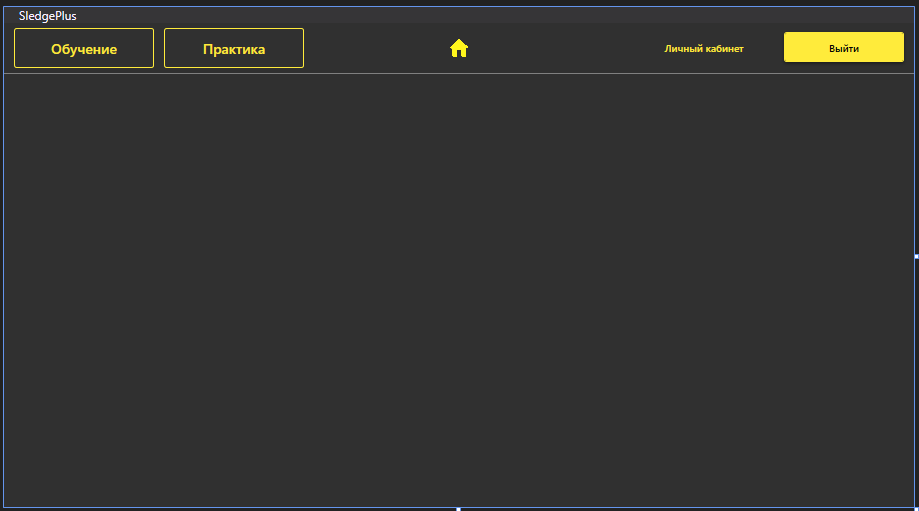


Рисунок 8 – Главное окно приложения

Здесь же создаются остальные элементы интерфейса, помещаемые в контейнер главного окна.

Элемент управления «Авторизация» (Рисунок 9) состоит из названия, текстового поля для вывода ошибки в случае её появления, двух интерактивных полей для логина и пароля и кнопки «Войти».

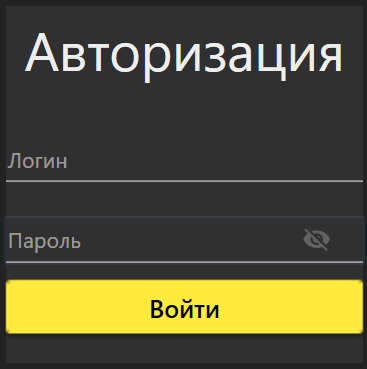


Рисунок 9 – Элемент управления «Авторизация»

К каждой кнопе пользовательских элементов прикреплена команда, вызывающаяся после выполнения нажатия на неё. Файлы с кодом для команд в архитектуре MVVM относятся к типу «Сервисы», являются подмодулями файлов-ViewModel’ей и хранятся в отдельном каталоге. В случае с кнопкой «Войти», команда выполняет запрос к серверу, ища пользователя с введённым логином, и в случае нахождения, дехэшируя пароль пользователя из базы данных с данным логином, проверяет запись на совпадение с введёнными значениями. В случае ненахождения пользователя или несовпадения пароля, пользователь получает текст с ошибкой.

Следующий элемент – совсем небольшая приветственная страница с общей информацией о приложении и о навигации по нему.

Далее, интерфейс для модуля практики (Рисунок 10). С помощью установленного ранее пакета «AvalonEdit» на страницу добавляется текстовый редактор для кода с подсветкой синтаксиса языка C++, в котором по умолчанию, при запуске приложения, находится текст с написанным кодом для вывода в консоль предложения «Hello, world!». Ниже располагаются кнопки для компилирования кода, написанного в редакторе и для запуска скомпилированной этим кодом программы на C++.



Рисунок 10 – Интерфейс модуля «Практика»

Компиляция кода происходит посредством встроенного в приложение легковесного компилятора «MinGW» для языков C, C++ и ASM.

Интерфейс модуля обучения представляет собой множество подгружаемых из базы данных элементов выпадающих списков-разделов с элементами кнопок-лекций внутри, собирающихся последовательно один за другим (Рисунок 11). Теоретические материалы выделяются жёлтым цветом, материалы с практическими заданиями – ярко-жёлтым. Дополнительно, если в базе данных есть запись о том, что пользователь уже открывал эту лекцию или практику, то элемент выделяется зелёным.

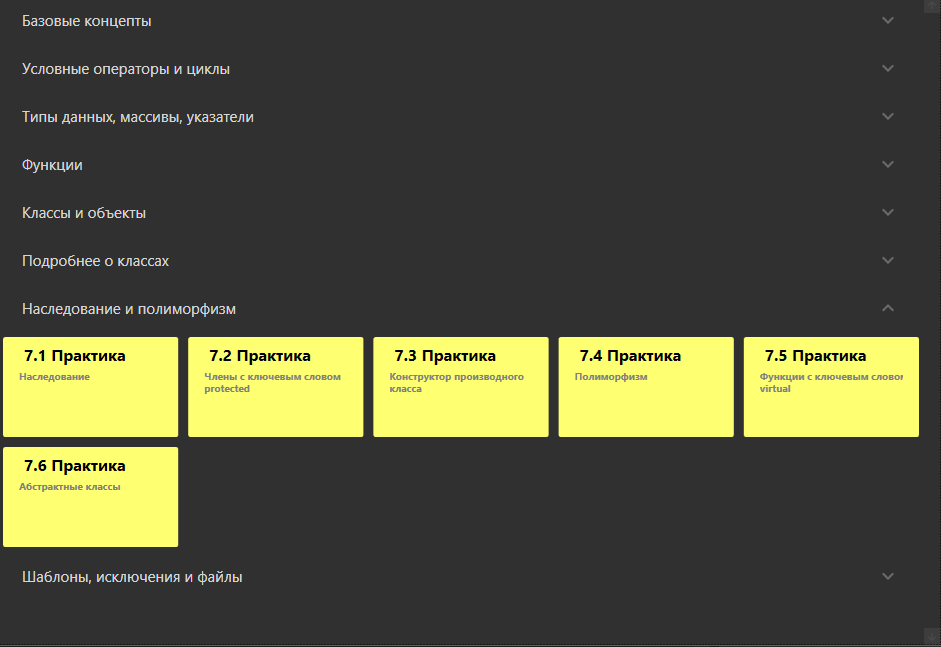


Рисунок 11 – Интерфейс модуля «Обучение»

Элементы лекций представляют из себя кнопки. В директории приложения лежат документы с расширением «.docx», которые открываются после нажатия элемент. Реализацию сборки страницы можно посмотреть в приведённом листинге 1.

Листинг 1 – Команда сборки страницы

|  |
| --- |
| public async Task Build()  {  if (Sections.Any()) Sections.Clear();  Label label = new();  var dbSections = await \_appDbContext.Sections.ToListAsync();  var dbLessons = await \_appDbContext.Lessons.ToListAsync();  foreach (var section in dbSections){  InnerSectionsItems = new ObservableCollection<LessonItemViewModel>();  label.FirstNumber += 1;  label.SecondNumber = 0;  foreach (var lesson in dbLessons.Where(x => x.SectionId == section.SectionId)){  var new Label = $"{label.FirstNumber}.{label.SecondNumber+=1} {GetLabelName(lesson)}";  var item1 = new LessonItemViewModel(\_host); |

Продолжение листинга 1

|  |
| --- |
| await Task.Run(async () => item1.Build(lesson.LessonId, newLabel, lesson.LessonDescription, new OpenLessonDocument(\_host, lesson.LessonDocumentName, lesson.IsPractice), await GetItemColor(lesson)));  lock (\_innerSectionsItemsLock)  {  InnerSectionsItems.Add(item1);  }  }  var item = new ExpanderLessonItemViewModel(\_host);  await Task.Run(() => item.Build(section.SectionHeader, InnerSectionsItems));  lock (\_sectionsLock)  {  Sections.Add(item);  }  }  } |