СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc136809779)

[1 Теоретические основы разработки обучающих приложений 4](#_Toc136809780)

[1.1 Обзор и анализ существующих обучающих приложений по основам программирования на языке С++ 4](#_Toc136809781)

[1.2 Достоинства и недостатки существующих обучающих приложений 5](#_Toc136809782)

[1.3 Постановка задачи на разработку 6](#_Toc136809783)

[2 Проектирование обучающего приложения по основам программирования на языке С++ 8](#_Toc136809784)

[2.1 Проектирование базы данных 8](#_Toc136809785)

[2.2 Описание этапов разработки приложения 11](#_Toc136809786)

[2.3 Описание основных элементов приложения 17](#_Toc136809787)

[2.4 Руководство администратора 25](#_Toc136809788)

# ВВЕДЕНИЕ

В современном мире, обучение языкам может быть вызывать определенные трудности, особенно для начинающих программистов и студентов специальностей 09.хх.хх, так как многие образовательные программы учебных заведений проходят либо без должного уровня подачи лекционных материалов, либо без практической части обучения.

Целью выпускной квалификационной работы является разработка обучающего приложения по основам программирования на языке С++, как средства для внедрения в учебный процесс заведения.

В соответствии с поставленной целью можно сформулировать следующие задачи разработки системы:

1. Определение состава задач;
2. Постановка задач системы;
3. Разработка технологии функционирования системы в условиях учебного заведения;
4. Определение структуры базы данных системы, построение информационной модели системы;
5. Построение интерфейса взаимодействия пользователя с системой.

Объектом исследования является учебное заведение, которое занимается обучением студентов языку программирования C++.

Предметом выпускной квалификационной работы является приложение, разрабатываемое на платформе Visual Studio с использованием системы управления базами данных MySQL.

# 1 Теоретические основы разработки обучающих приложений

## 1.1 Обзор и анализ существующих обучающих приложений по основам программирования на языке С++

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.

## 1.2 Достоинства и недостатки существующих обучающих приложений

Основными причинами популярности платформы «Sololearn» является её дружелюбный и понятный интерфейс и количество доступных для изучения языков и технологий.

Среди языков доступных для изучения: Python, C, C++, C#, Java, SQL, HTML, CSS, JavaScript, R, Kotlin, Golang, PHP, Swift 4, Ruby и другие.

Платформа «Sololearn» предоставляет бесплатный доступ к своим курсам с некоторыми лимитами в виде ограничений на доступ к модулю практических занятий и малому количеству возможных ошибок при выполнении тестов. При этом, платформа имеет платную подписку, которая убирает эти ограничения и к тому же, добавляет бонусные упражнения, расширяет видимые параметры статистики в личном кабинете и делает ваш вопрос в обсуждениях приоритетным среди других пользователей.

Другой популярный ресурс для изучения программирования – «Codecademy». Он предлагает широкий выбор курсов, которые охватывают различные языки программирования, включая Python, JavaScript, Ruby, HTML, CSS и многие другие.

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.

Платформа выделяется подходом к обучению. Если в «Sololearn» обучение почти всегда проходит в формате тестов, а изучение языков программирования по большей части представляет из себя изучение их синтаксиса, то на «Codecademy» внимание уделяется особенности конкретных языков и работе с ними в реальных условиях. К примеру, при обучении C++ внимание уделяется его низкоуровневости и в обучение входит процесс компиляции кода компилятором «MinGW», а при изучении SQL акцент делается на разницу в типах баз данных и системах управления ими. Также, на платформе, помимо курсов по языкам программирования доступны курсы по всем технологиям определённой сферы программирования, например, курс «Full-stack разработчик» или курс «Машинное обучение».

Большим минусом «Codecademy» является сильное ограничение в функциях при отсутствии платной подписки. Многие курсы являются эксклюзивными для её обладателей, а количество тем в бесплатных курсах составляет лишь половину от содержания платных.

Из данных качеств проанализированных обучающих приложений можно извлечь для разрабатываемого проекта подходящие и внедрить их.

## 1.3 Постановка задачи на разработку

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

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

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++.

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

В проекте библиотеки классов будут созданы классы-модели таблиц спроектированной базы данных со свойствами-столбцами, после чего создастся файл контекста, в котором прописывается строка подключения к базе данных со всеми необходимыми настройками.

После, заполняется база данных.

В проекте WPF будет использоваться 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.

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

В модуль «Авторизация» будет интегрирована система хэширования паролей методом шифрования SHA-256 для повышения безопасности работы системы.

Для работы подсистемы «Практика» в проект встроится компилятор кода C/C++ «MinGW». В соответствующую вкладку добавится текстовый редактор кода с возможностью компиляции и вывода результата кода в терминал.

После определения и постановки задачи на разработку можно приступать к проектированию приложения.

# 2 Проектирование обучающего приложения по основам программирования на языке С++

## 2.1 Проектирование базы данных

Для работы с базой данных была выбрана система управления базой данных MySQL. 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.

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

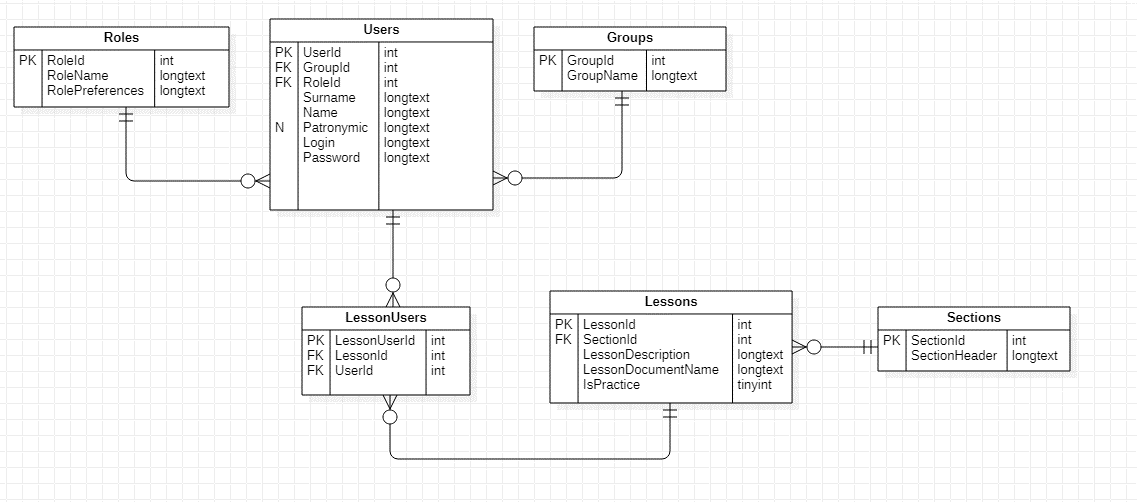


Рисунок 1 – ER-диаграмма базы данных

База данных спроектирована согласно третьей нормальной форме.

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.

2.1.1 Состав и описание таблиц базы данных

Среди сущностей базы данных имеются 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.

Для описания модели базы данных создан словарь данных (Таблица 1)

Таблица 1 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Словарь данных | | | | |
| Ключ | Поле | Тип данных | Null |
| 1 | 2 | 3 | 4 |
| Groups | | | | |
| PK | GroupId | int(11) | Нет |
|  | GroupName | longtext | Нет |
| Lessons | | | | |
| PK | LessonId | int(11) | Нет |
| FK | SectionId | int(11) | Нет |
|  | LessonDescription | longtext | Нет |
|  | LessonDocumentName | longtext | Нет |
|  | IsPractice | tinyint(1) | Нет |
| LessonUsers | | | | |
| PK | LessonUserId | int(11) | Нет |
| FK | LessonId | int(11) | Нет |
| FK | UserId | int(11) | Нет |
| Roles | | | | |
| PK | RoleId | int(11) | Нет |
|  | RoleName | longtext | Нет |
|  | RolePreferences | longtext | Нет |
| Sections | | | | |
| PK | SectionId | int(11) | Нет |
|  | SectionHeader | longtext | Нет |
| Users | | | | |
| PK | UserId | int(11) | Нет |
| FK | RoleId | int(11) | Нет |
| FK | GroupId | int(11) | Нет |
|  | Login | longtext | Нет |
|  | Password | longtext | Нет |
|  | Name | longtext | Нет |
|  | Surname | longtext | Нет |
|  | Patronymic | longtext | Да |

2.1.2 Определение ключей базы данных

В каждой таблице базы данных присутствуют столбцы первичных ключей - идентификаторов. В некоторых таблицах так же присутствуют внешние ключи. Их название образовано по формуле

*, (1)*

где k – название ключа, t – название таблицы с используемым идентификатором.

После определения ключей база данных становится функциональной готовой к использованию.

## 2.2 Описание этапов разработки приложения

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 и получают названия «SledgePlus.WPF» и «SledgePlus.Data» соответственно. Происходит установка дополнительных пакетов для проектов:

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.

2.2.1 Разработка проекта «SledgePlus.Data»

В проекте библиотеки классов создаются классы-модели для каждой из таблиц спроектированной базы данных со свойствами-столбцами (Рисунок 2), 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.

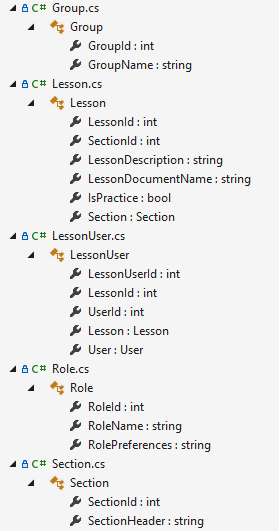


Рисунок 2 – Библиотека классов «SledgePlus.Data»

Используя ORM-инструмент Entity Framework, база данных собирается на сервере из заданной в файле контекста информации (Рисунок 3).



Рисунок 3 – Контекст базы данных

Сборка происходит через команды Entity Framework’а (Рисунок 4).

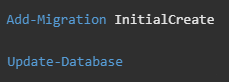


Рисунок 4 – Команды для сборки базы данных

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 5) или через десктопное приложение для работы с запросами SQL и данными MySQL Workbench.

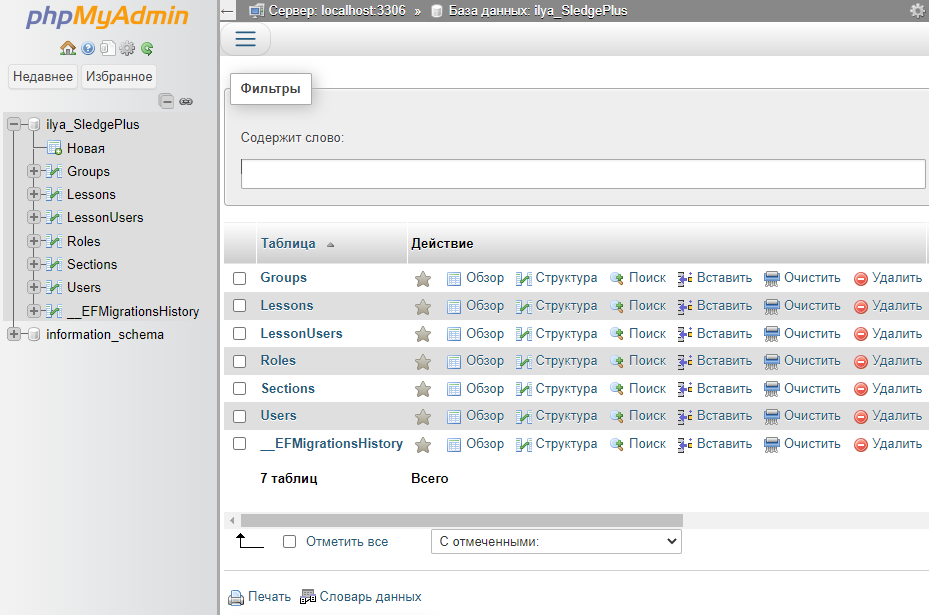


Рисунок 5 – Интерфейс веб-приложения phpMyAdmin

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

Дополнительно в базу данных импортируется информация об учебных материалах.

2.2.2 Разработка проекта «SledgePlus.WPF»

В проекте WPF используется архитектура MVVM (Model, View, ViewModel).

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 6).

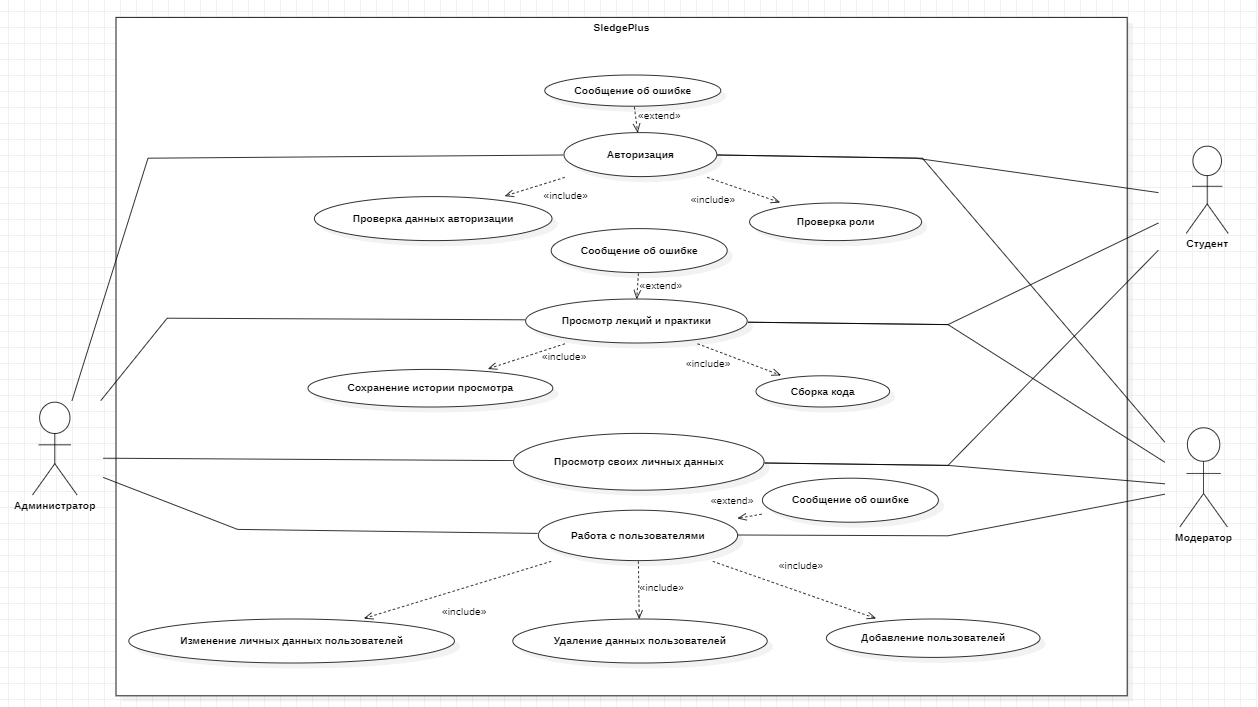


Рисунок 6 – Use-case диаграмма

В проект WPF добавляется подсистема авторизации пользователя. В этот модуль интегрирована система хэширования паролей (Рисунок 7) методом шифрования SHA-256 для повышения безопасности работы системы.

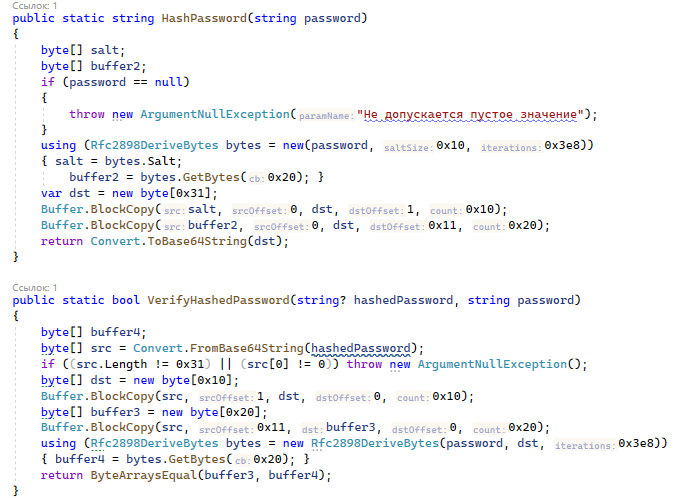


Рисунок 7 – Методы хэширования паролей в системе

Визуализация модуля 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 «LBase» по пути установленного приложения.

Добавляется подсистема практики. В проект встраивается компилятор кода C/C++ «MinGW».

Встраивание в приложение сторонних модулей, нее связанных на прямую с C# реализуется посредством встраивания в проект события пост-сборки (Рисунок 8).

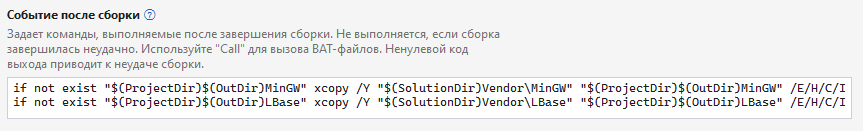


Рисунок 8 – Событие после сборки

Помимо структурных элементов архитектуры, в приложении присутствуют элементы, отвечающие за бизнес-логику продукта, промежуточное хранение данных, реализацию паттернов проектирования, а также шаблоны и специально созданные файлы исключений. Для всего это, в проекте создана соответствующая каталожная логика хранения.

Все элементы интерфейса взаимодействуют между собой благодаря файлам ViewModel, в которых содержатся их свойства. Логика их взаимодействия в свою очередь определяется сервисами. Созданы сервисы хранения, команд, конвертеров, работы с данными, работы с текстом.

## 2.3 Описание основных элементов приложения

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 9).

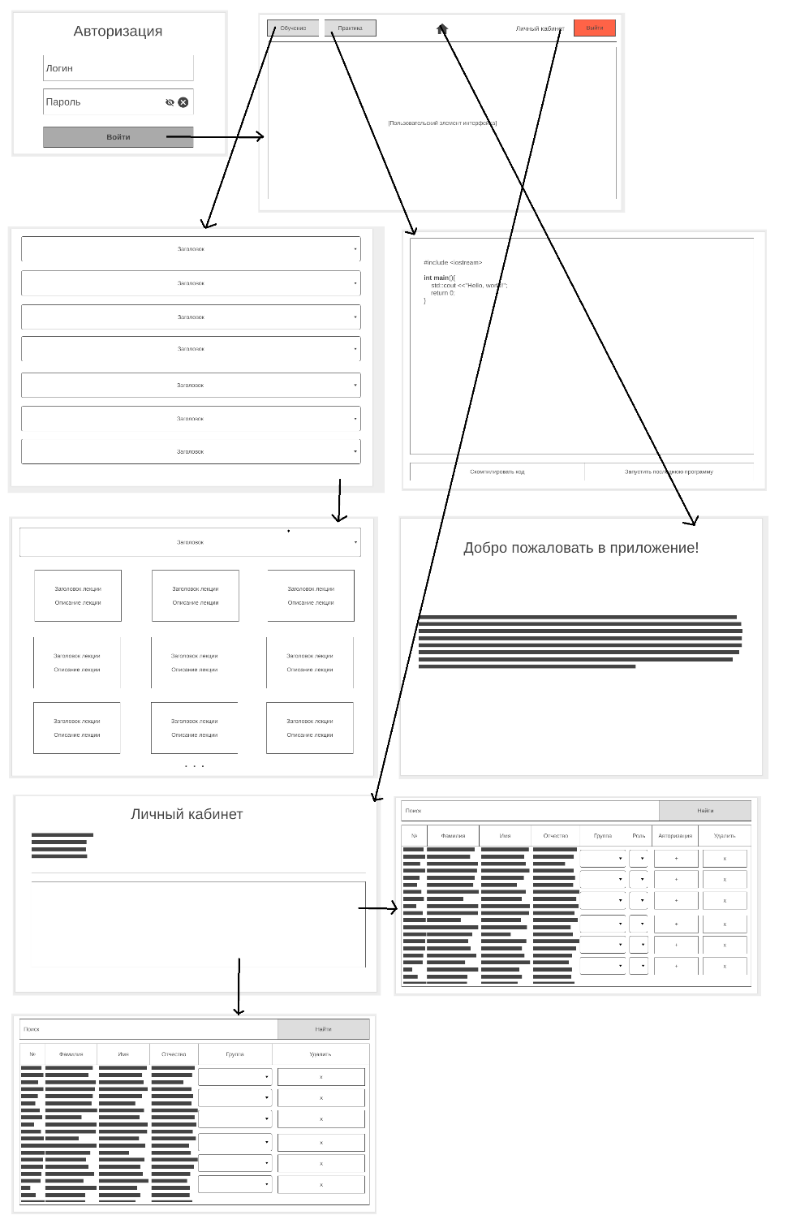


Рисунок 9 – Схема логического взаимодействия между структурными элементами приложения

Работа подсистем приложения описывается диаграммами перехода состояний (Рисунок 10).

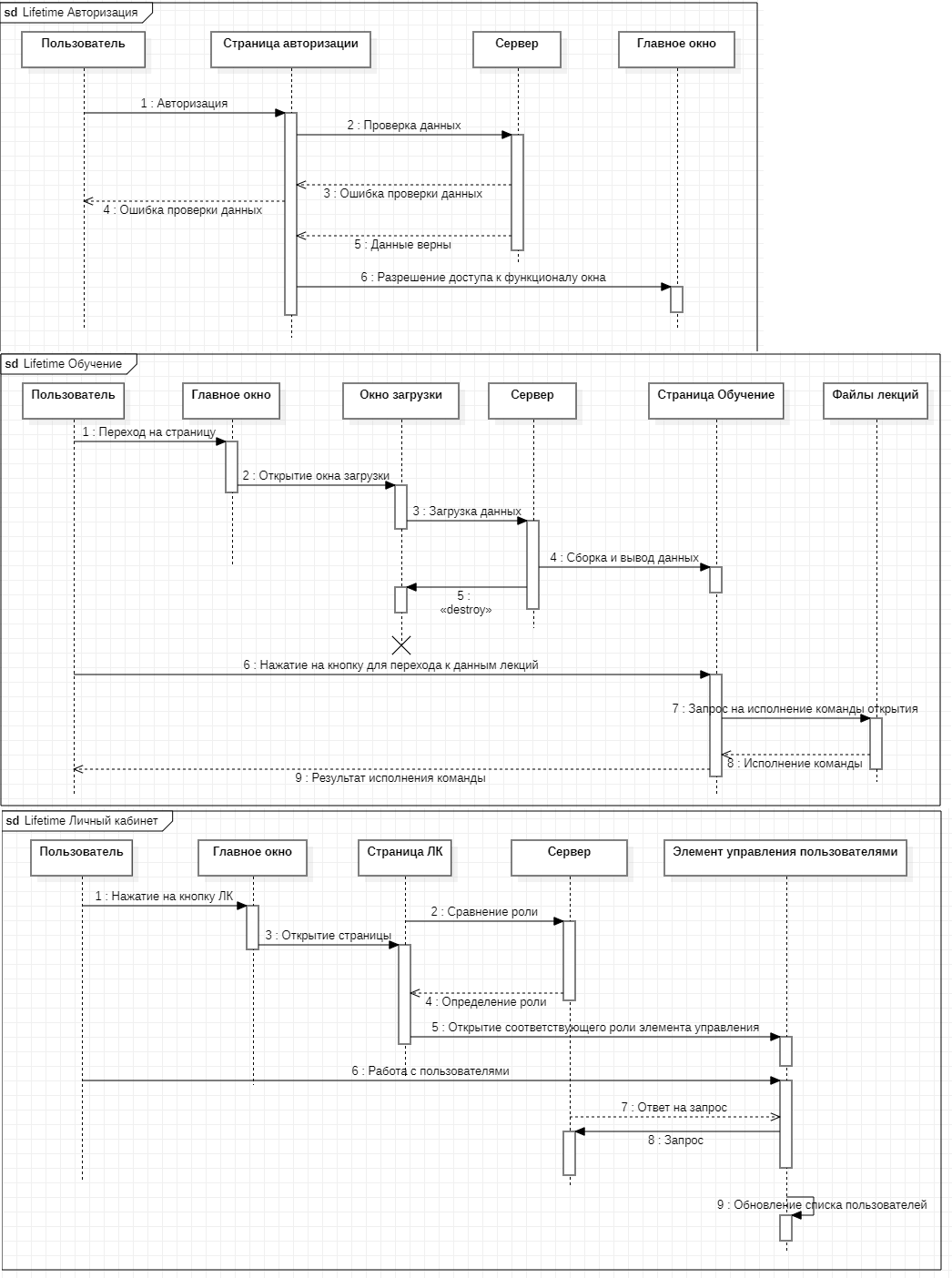


Рисунок 10 – Диаграммы переходов состояний

Главное окно приложения (Рисунок 11) состоит из поля навигации в верхней части и контейнера элементов интерфейса в нижней

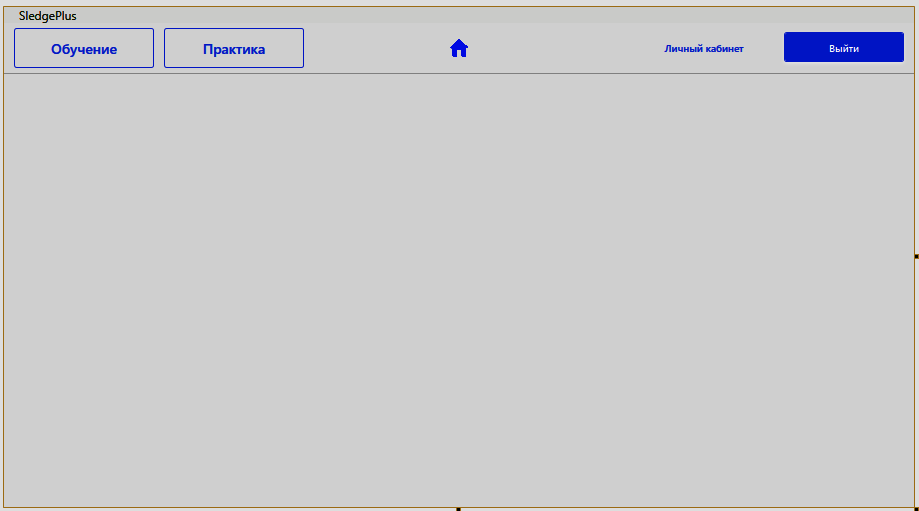


Рисунок 11 – Главное окно приложения

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

Элемент управления «Авторизация» (Рисунок 12) состоит из названия, текстового поля для вывода ошибки в случае её появления, двух интерактивных полей для логина и пароля и кнопки «Войти».

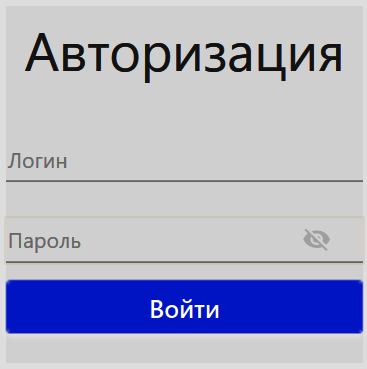


Рисунок 12 – Элемент управления «Авторизация»

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 13). 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 13 – 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 14). 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.

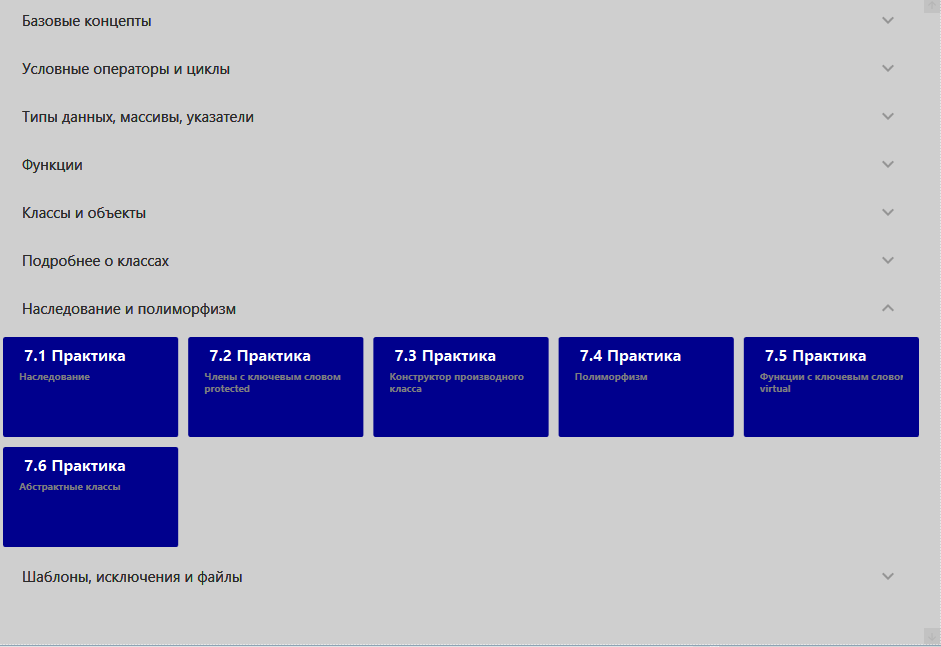


Рисунок 14 – Интерфейс модуля «Обучение»

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);  }  }  } |

Интерфейс личного кабинета (Рисунок 15) состоит из панели с личными данными текущего пользователя в текстовом виде и контейнера с элементом, зависящим от роли пользователя.

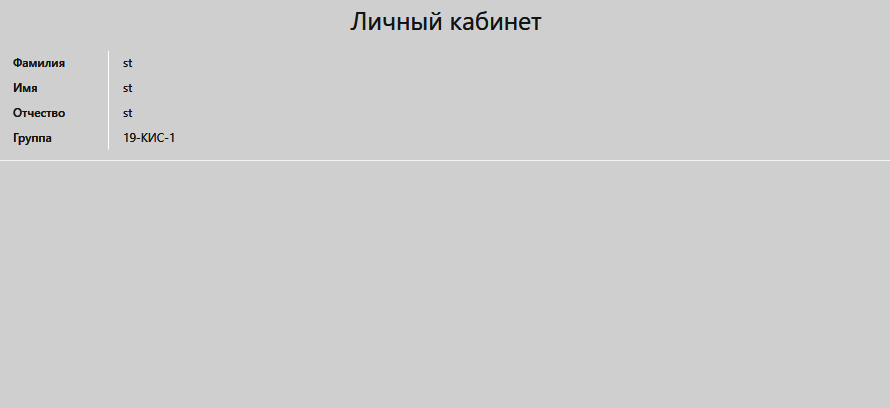


Рисунок 15 – Интерфейс личного кабинета

Контейнер внизу страницы включает в себя элемент DataGrid с расширенными свойствами и поисковую строку для него. Отличие модернизированного элемента DataGrid заключается в переопределении этого элемента в коде приложения и добавления в него нового свойства. Код нового элемента представлен в листинге 2.

Листинг 2 – Модернизированный DataGrid

|  |
| --- |
| public class UsersDataGrid : DataGrid  {  public UsersDataGrid()  {  SelectionChanged += DataGridCustom\_SelectionChanged;  }  private void DataGridCustom\_SelectionChanged(object? sender, SelectionChangedEventArgs e)  {  if (Items.SourceCollection == null) return;  ItemsList = new ObservableCollection<User>((ObservableCollection<User>)Items.SourceCollection);  }  public IEnumerable<User> ItemsList  {  get => (IEnumerable<User>)GetValue(ItemsListProperty);  set => SetValue(ItemsListProperty, value);  }  public static readonly DependencyProperty ItemsListProperty =  DependencyProperty.Register(  nameof(ItemsList),  typeof(IEnumerable<User>),  typeof(UsersDataGrid),  new PropertyMetadata(null));  } |

Этот элемент различается для пользователей «Модератор» и «Администратор» тем, что модератор не может изменять данные для авторизации и роли пользователей. Остальные данные могут изменяться обоими ролями (Рисунок 16).



Рисунок 16 – Элементы управления пользователей с ролями «Администратор» и «Модератор»

После нажатия на кнопку авторизации для какого-либо пользователя, вы попадаете на страницу с регистрацией пользователя (Рисунок 17).

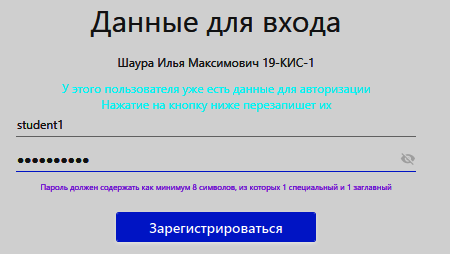


Рисунок 17 – Страница регистрации

После успешной регистрации, происходит переход обратно на страницу личного кабинета.

## 2.4 Руководство администратора

Система «SledgePlus» предоставляет возможность обучения студентов языку программирования C++. Кроме того, система предоставляет администратору возможность самостоятельно вносить изменения в лекционный материал и управлять аккаунтами студентов и модераторов.

Приложение позволяет:

1. Добавлять/изменять/удалять данные о пользователях;
2. Осуществлять пользователю компиляцию и запуска кода, написанного на языке C++;
3. Просмотр и редактирование учебных материалов, представленных в виде документов формата .docx.

Система «SledgePlus» может эксплуатироваться и выполнять заданные функции при соблюдении требований, предъявляемых к техническому, системному и прикладному обеспечению.

Порядок проверки работоспособности

Для проверки доступности системы «SledgePlus» с рабочего места пользователя необходимо выполнить следующие действия:

1. Запустить программу «SledgePlus»;
2. В форме аутентификации ввести пользовательский логин и пароль, нажать на кнопку «Войти»;
3. Убедиться, что вход в систему успешен;
4. В случае если приложение «SledgePlus» не запускается следует обратиться к администратору.

Описание операций. Выполняемые функции и задачи

Администратору доступен полный функционал приложения. Пользователь с этим уровнем доступа, при переходе во вкладку «Личный кабинет» может воспользоваться панелью управления пользователями (Рисунок 18).