МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство для студентов «Student365»

Исполнитель

студент 2 курса группы 4 Зворыкин Дмитрий Александрович

(Ф.И.О.)

Руководитель работы ассистент Мущук А.Н.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2022

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc104350953)

[1. Аналитический обзор литературы и постановка задачи 4](#_Toc104350954)

[1.1 Обзор прототипов 4](#_Toc104350955)

[1.2 Постановка задачи 9](#_Toc104350956)

[2. Анализ требований к программному средству и разработка функциональных требований 10](#_Toc104350957)

[2.1 Описание средств разработки 10](#_Toc104350958)

[2.2 Описание разрабатываемой функциональности программного средства 11](#_Toc104350959)

[3. Проектирование программного средства 13](#_Toc104350960)

[3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства 13](#_Toc104350961)

[3.2 Разработка общей схемы приложения 14](#_Toc104350962)

[3.3 Проектирование архитектуры приложения 16](#_Toc104350963)

[3.4 Проектирование базы данных 18](#_Toc104350964)

[4. Создание (реализация) программного средства 22](#_Toc104350965)

[4.1 Диаграмма классов 22](#_Toc104350966)

[4.2 Реализация архитектуры MVVM 22](#_Toc104350967)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 25](#_Toc104350968)

[6. Руководство по использованию 29](#_Toc104350969)

[ЗАКЛЮЧЕНИЕ 33](#_Toc104350970)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 34](#_Toc104350971)

[Приложение А 35](#_Toc104350972)

[Приложение Б 37](#_Toc104350973)

[Графический материал 39](#_Toc104350974)

[Диаграмма вариантов использования 39](#_Toc104350975)

[Логическая схема базы данных 40](#_Toc104350976)

[Диаграммы классов 41](#_Toc104350977)

ВВЕДЕНИЕ

Среди трендов последних лет очень четко выделяется направление самоорганизации. Приложения для повышения продуктивности занимают все более высокие позиции в рейтингах по количеству скачиваний в PlayMarket и AppStore. Тренд очевиден: современный человек сегодня – это человек, стремящийся контролировать и анализировать свою жизнь.

Не удивительно, что организованность напрямую связана с производительностью. Учащиеся не могут уследить за всеми делами и постоянно что-то забывают, создавая себе проблемы.

Программа Student365 ориентирована на применение в университете. Это не просто программа для студента куда он записывает расписание и заметки. В приложении есть 3 роли: студент, преподаватель и администратор. Студент может: просматривать расписание, выставлять пропуски, оценки, делать заметки, отмечать количество сданных лабораторных работ, отслеживая таким образом прогресс. Учитель может: добавить студента в группу, просмотреть всех студентов группы, изменить расписание для группы/подгруппы, добавить предметы в группу, поставить максимальное количество лабораторных работ по предмету, которые автоматически появятся у студента. Функции админа: добавление студентов, преподавателей и других админов.

Итак, насыщенные студенческие будни требуют чёткого и детального планирования для достижения хороших результатов, и максимально эффективно с этой задачей поможет справиться Student365, разработка которого и является темой курсового проектирования

Главная задача данного курсового проектирования – это разработка программного средства, которое реализует все вышеперечисленные функции и решает поставленные задачи. Язык разработки проекта – C#. При выполнении курсового проекта будут использованы принципы и приемы ООП. Также будут использоваться технологии Windows Presentation Foundation (WPF).

Для успешной реализации курсового проекта необходимо:

* провести анализ соответствующей литературы;
* ознакомиться с прототипами программных средств выбранной мной темы;
* определить функциональные требования;
* продумать структуру базы данных;
* продумать структуру проекта;
* реализовать программное средство;
* протестировать программное средство;
* написать руководство пользователя.

Содержание данной пояснительной записки отражает все этапы выполнения.

1. Аналитический обзор литературы и постановка задачи

Для того чтобы окончательно определиться с постановкой задачи моего курсового проекта, необходимо проанализировать прототипы программных средств выбранной темы.

1.1 Обзор прототипов

Основной прототип моей курсовой работы — это приложение расписания для БГУИР в Playmarket и Appstore. Так же, как и в моём приложении, студент не должен ничего делать и его регистрируют в системе после чего он может просматривать расписание своей или любой группы подгруппы и даже расписание экзаменов.

Интерфейс приложения показан на рисунке 1.1.1.

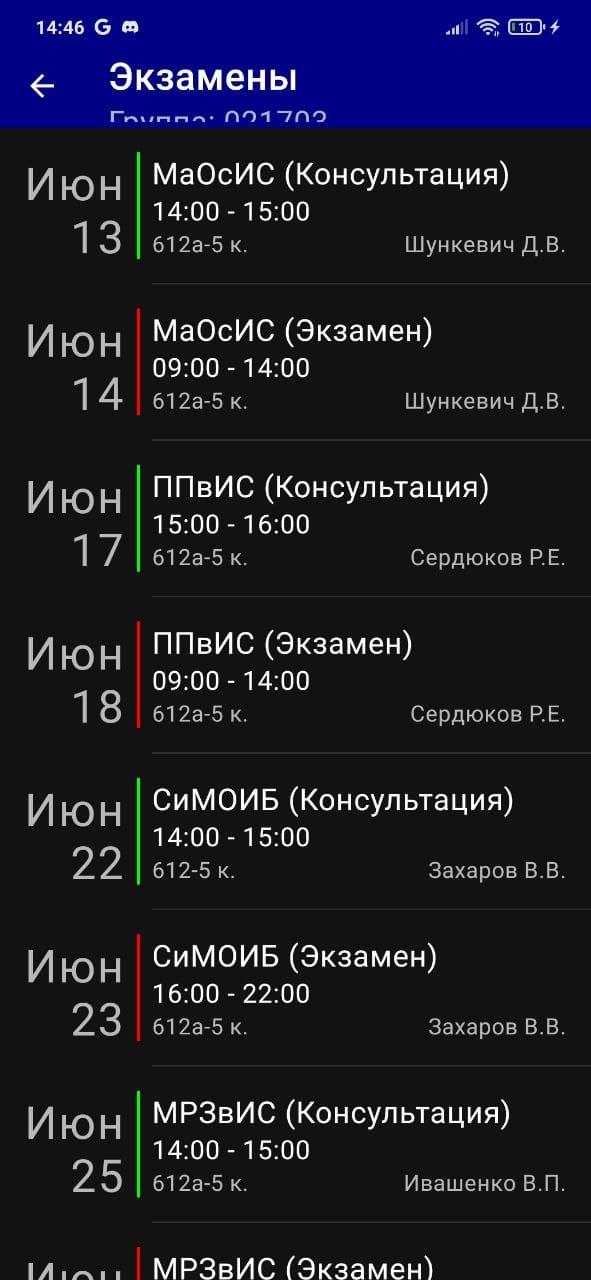
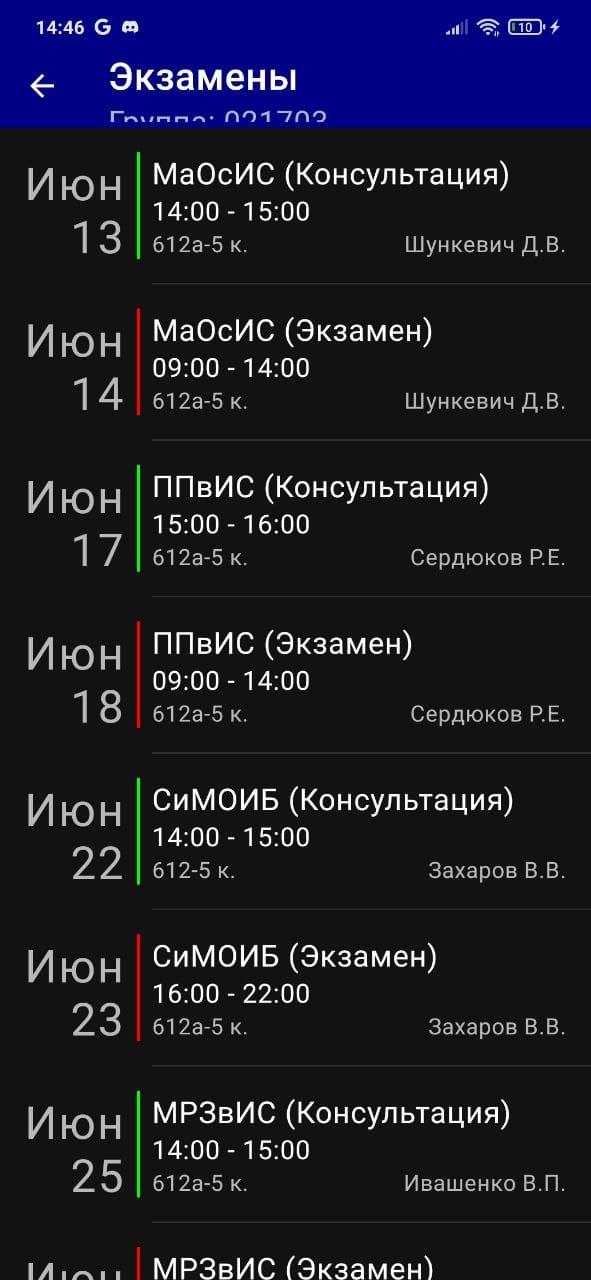


Рисунок 1.1.1 – Интерфейс приложения БГУИР

Исходя их названия приложения доступно только студентам БГУИР. К достоинствам можно отнести что это приложения для телефона, не требуется никакого ручного заполнения расписания и автоматическое обновление при изменении расписания, возможность сменить тему.

Из минусов: отсутствие десктопного приложения, нет дополнительного функционала такие как заметки или оценки и т.д.

Мне захотелось сделать что-то похожее, но с дополнительным функционалом и на ПК

Из других десктопных аналогов можно выделить класс приложений как органайзеры.

Например iStudiez–приложение, которое совмещает в себе функции календаря и ежедневника. Как отмечают разработчики, их программа пригодится в первую очередь студентам интерфейс представлен на рисунке 1.1.2



Рисунок 1.1.2 – Интерфейс приложения iStudiez

В приложении присутствует система оценок но она использует заподную систему поэтому в нашей стране это функция не особо полезная.

К достоинствам iStudiez можно отнести и наличие версии для смартфона и приложении доступно на всех платформах от Мac до Windows

К недостаткам можно отнести то, что приложение является платным. Но при желании пользователи могут вначале попробовать iStudiez Lite, которая хоть и обладает некоторыми ограничениями, но также отлично подходит для планирования и структурирования информации, связанной с учебой.

**TimeTune**

TimeTune позволяет настроить не только расписание занятий, но и время на сборы, дорогу, выполнение домашнего задания и подготовку к завтрашнему дню. В приложении можно спланировать весь день до минуты и автоматизировать заполнение графиков. TimeTune незаменим для студентов, которые много занимаются дополнительно, совмещают учёбу с работой или не хотят забывать о хобби. Сервис условно-бесплатный, с рекламой и платным контентом, доступен на Android. интерфейс представлен на рисунке 1.1.3.

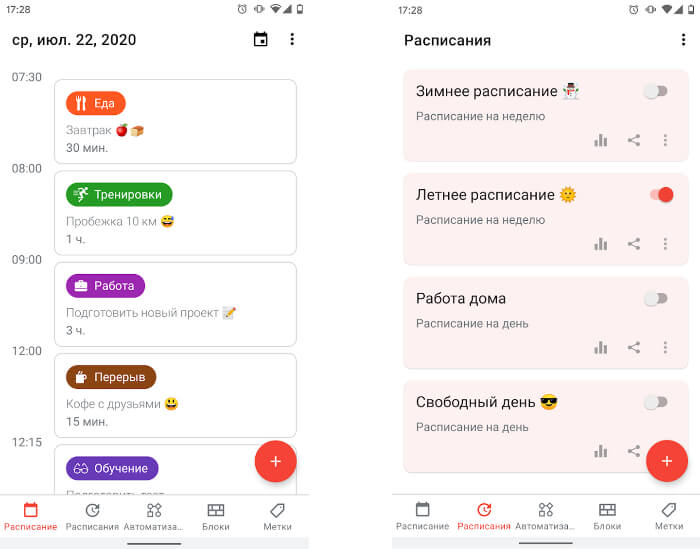


Рисунок 1.1.3 – Интерфейс TimeTune

**Weekly Planner**

Удобный и любимый пользователями календарь с постановкой задач, который присылает уведомления о делах. Кроме этого, здесь можно ставить цели, чтобы отмечать время их достижения. Этим функциям в Weekly уделено даже больше внимания, чем составлению расписания на неделю. Это может понадобиться для подготовки к сессии, написании диплома и курсовой работы, но в обычное время, возможно, приложение не окажется таким полезным. Оно доступно на всех платформах и абсолютно бесплатно интерфейс представлен на рисунке 1.1.4

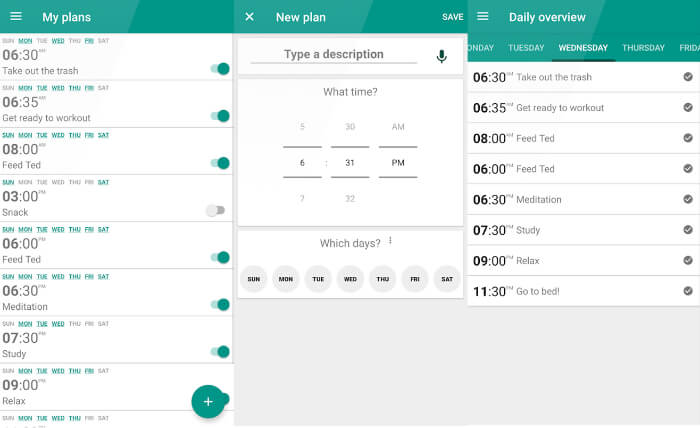


Рисунок 1.1.4 – Интерфейс Weekly Planner

**LightSchool**

Приложение было создано для школьников, но приспособить его для студенческой жизни очень легко. Здесь можно установить расписание и длительность занятий, отмечать место их проведения и записывать домашние задания. Есть даже теоретические материалы и простые встроенные функции для облегчения учебной жизни, но они будут актуальны разве что для первокурсников. Приложение бесплатно, без рекламы, доступно для Android. интерфейс представлен на рисунке 1.1.5

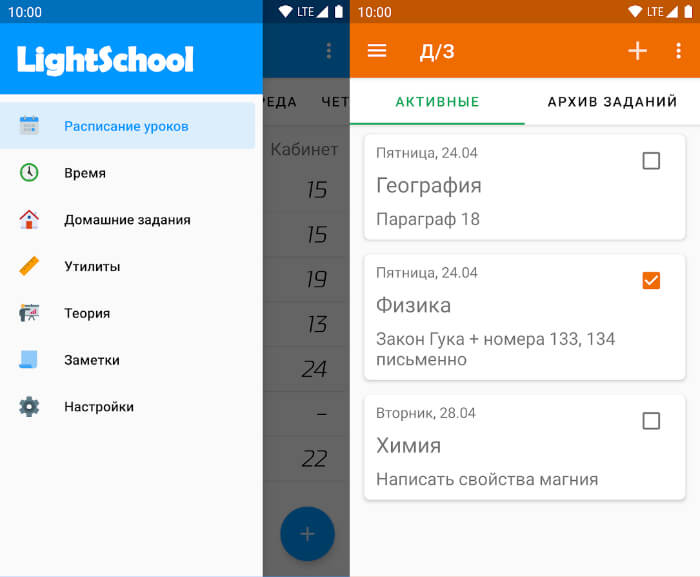


Рисунок 1.1.5 – Интерфейс LightSchool

Ну и в конце мой фаворит с простым названием “Расписание для студентов”

Можно составлять расписание на несколько недель, установить длительность занятия, настроить выходные, добавлять домашнее задание. Приложение для Android (версия для iOS в разработке), бесплатное и с рекламой. Из плюсов можно отметить отзывчивую команду, которая реагирует на отзывы и быстро добавляет обновления, которые просят пользователи.

Я каждый день использую это приложение. К концу семестра актуальность немного теряется т.к. уже запоминаешь расписание. Но в начале семестра очень полезное. Поддерживает push уведомления перед парой со временем начала и аудиторией. Единственный минус отсутствие приложения для десктопа. интерфейс представлен на рисунке 1.1.6.

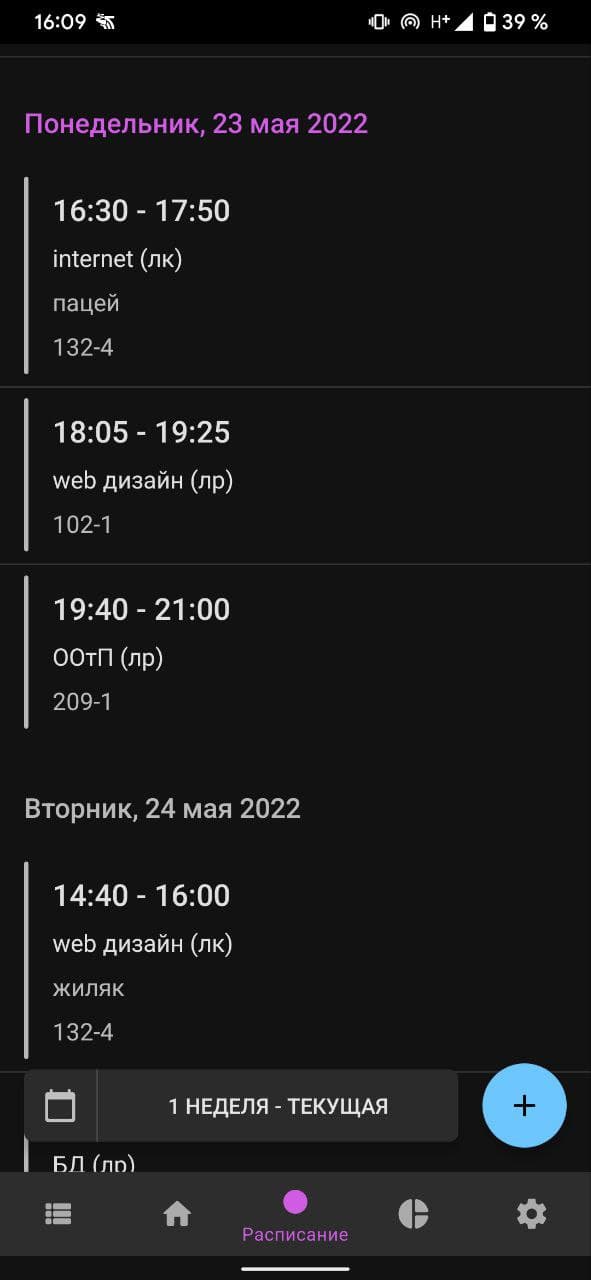


Рисунок 1.1.6 – Интерфейс Расписание для студентов

1.2 Постановка задачи

Проанализировав прототип, были выделены основная задача и функциональные требования проекта.

Целью курсового проекта стало написание десктопной игры, в которой возможно выполнять авторизацию и регистрацию пользователей, создавать вопросы и наборы вопросов, создавать игровые сессии. Пользователю в лице администратора можно будет изменять любые данные о пользователях, вопросах и наборах вопросов. Пользователю в лице клиента доступна возможность создавать свои вопросы и наборы вопросов, вступать в интеллектуальные схватки с другими пользователями.

Функциональные требования описаны в главе 2.

2. Анализ требований к программному средству и разработка функциональных требований

Для начала этапа разработки необходимо сформулировать функциональные требования к программному средству.

2.1 Описание средств разработки

Языком разработки курсового проекта является C#. При выполнении курсового проекта будут использованы принципы и приемы ООП. Также будут использоваться технологии Windows Presentation Foundation (WPF). Программный продукт реализован на объектно-ориентированном языке C# с использованием технологий WPF.

Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от WindowsForms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

База данных была разработана на SQL Server в приложении DataGrip.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

Производительность. SQL Server работает очень быстро.

Надежность и безопасность. SQL Server предоставляет шифрование данных.

Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language).

Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов – Transact-SQL – реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;

model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;

code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран первый подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек. Приложение должно иметь возможность запускаться без использования интегрированных средств разработки.

2.2 Описание разрабатываемой функциональности программного средства

В программном средстве присутствуют 3 роли: студент, преподаватель, администратор.

У роли студента следующие функциональные возможности:

• авторизация в приложении при условии регистрации админом студента;

• просмотр расписания, установленное преподавателем 1 или 2 недели (устанавливается автоматически либо пользователем) без возможности изменения расписания;

• просмотр списка студентов своей (или любой) группы;

• оставлять пометки и записки;

• отмечать сданные лабы из предметов выставленными преподавателями для каждой группы отдельно;

• добавлять пропуски занятий и автоматический их подсчёт по предметам;

• выставлять оценки по предметам;

У роли учителя следующие функциональные возможности:

• изменять расписание для группы/подгруппы;

• просматривать список студентов для любой группы;

• добавлять студентов (с разрешения админа) в группу;

• ставить предметы группы и количество лабораторных работ необходимых для сдачи;

У роли Администратора следующие функциональные возможности:

• регистрация студентов, преподавателей, админов;

• удаление из базы данных студентов, преподавателей, админов;

• изменение в базе данных студентов, преподавателей, админов.

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

В графической части записки представлена UML-диаграмма возможностей, которая отражает функциональность программного средства с точки зрения получения значимого результата для пользователя.

3. Проектирование программного средства

Проектирование программного средства – процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства

В данном приложении используется архитектурный паттерн MVVM (Model-View-ViewModel). Он позволяет отделить логику приложения от визуальной части. MVVM состоит из трех частей:

Модель описывает используемые в приложении данные.

Представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением.

Модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление.

Структура проекта представлена на рисунке 2.2.1.

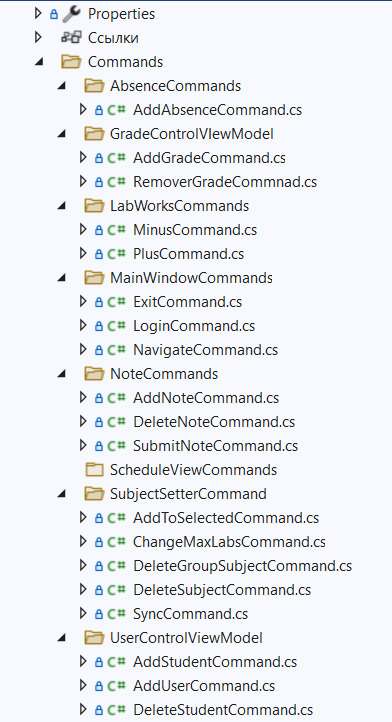
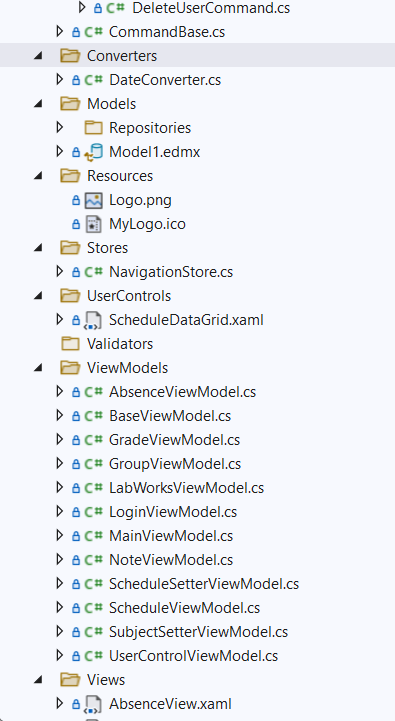
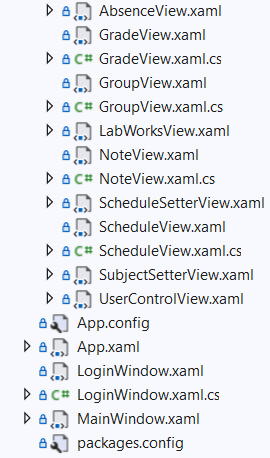


Рисунок 2.2.1 – Структура проекта

Таблица 2.1 – Структура проекта «Student365»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Properties | Свойства проекта, содержит информацию о сборке используемых ресурсах и настройках. |
| Models | Подпространство имён, содержащее описание сущностей для работы с БД. |
| Repositories | Репозитории, которые являются промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.Также содержит паттерн Unitofwork |
| Resources | Графические ресурсы приложения. |
| Converters | Конвертеры представлений и типов данных для отображения их |
| View | Представления, которые определяют визуальный интерфейс, через который пользователь взаимодействует с приложением |
| ViewModel | Содержит логику по получению данных из модели, которые затем передаются в представление. |
| App.config | Файл с параметрами проекта. |
| Сommands | Cодержит логику работы команд определенных во viewmodel |
| Localization | Содержит ресурсы локализации |
| Store | Паттерн Store. Содержит NavaigationsStore |

Таким образом, сформированная таблица помогает понять общую стркутура проекта проектируемого программного средства.

3.2 Разработка общей схемы приложения

Общая схема окон приложения для администратора представлена на рисунке 3.2.1, на рисунке 3.2.2 – студента, на рисунке 3.2.3 – преподавателя.

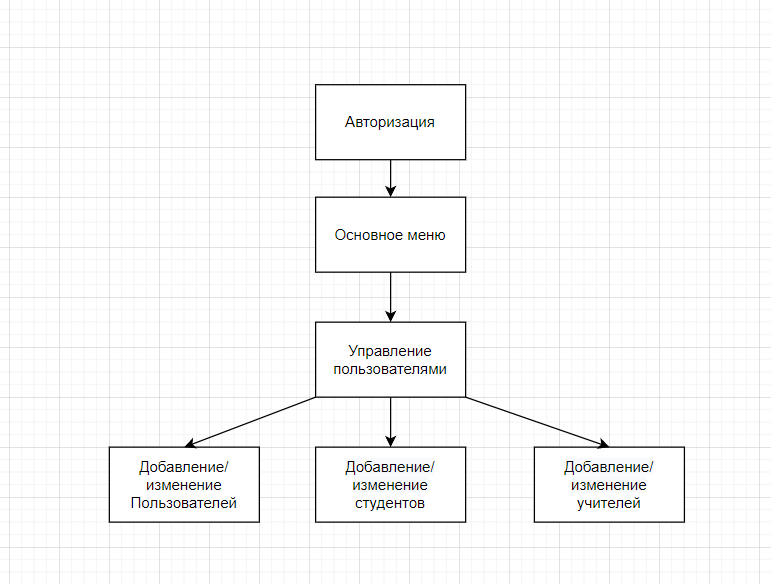


Рисунок 3.2.1 – Схема окон администратора



Рисунок 3.2.2 – Схема окон студента

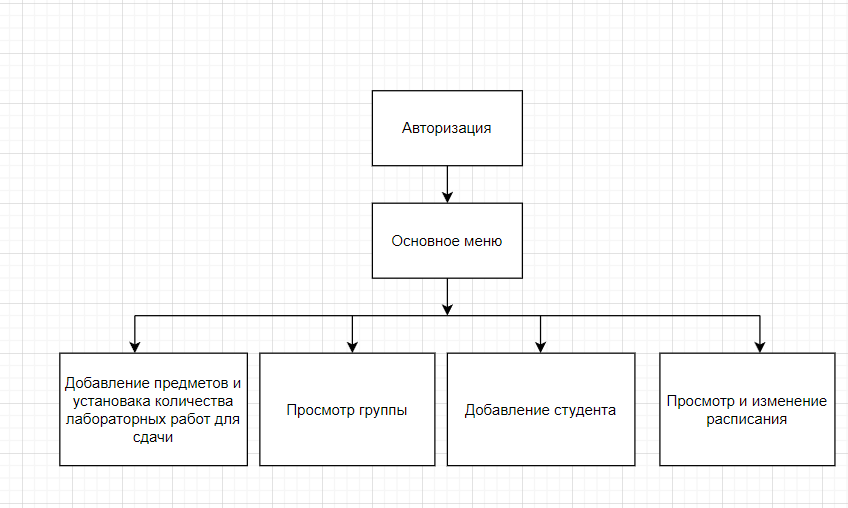


Рисунок 3.2.3 – Схема окон преподавателя

При запуске приложения открывается окно авторизации.

После успешной авторизации запускается главное окно приложения, где происходит работа с отдельными страницами.

3.3 Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.3.1 – Структура шаблона MVVM

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Нередко все сущности (модель) наследует интерфейс INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать интерфейс INotifyPropertyChanged, нужно объявить свойство PropertyChanged и метод OnPropertyChanged().

Для того, чтобы начать отслеживание свойства, необходимо вызывать метод OnPropertyChanged(“Имя свойства”).

Model или Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server. На диаграмме размещение на рисунке 3.3.2 показано графическое представление инфраструктуры приложения.

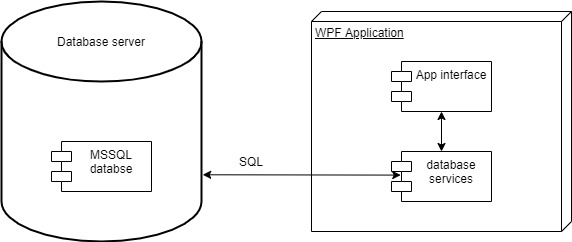


Рисунок 3.3.2 – Диаграмма размещения

3.4 Проектирование базы данных

Проектирование баз данных – процесс создания схемы базы данных и определения необходимы ограничений целостности.

При проектировании базы данных курсового проекта использовался Entity Framework. Он представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из базы данных, но и получать объекты, связанные различными ассоциативными связями.

Логическая модель базы данных представлена в графической части. Для реализации поставленной в курсовом проектировании задачи была создана база данных Student365. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server 2018.

Созданная база данных содержит в себе 10 таблиц: Absence, Grade, GroupSubjects, LabWorks, Notes, Schedule, Students, Subjects, Teachers, Users.

На рисунке 3.4.1 представлена структура таблицы Users.

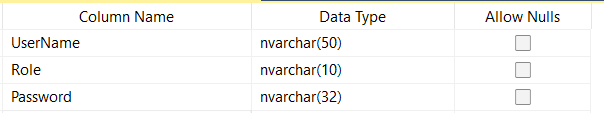


Рисунок 3.4.1 – Структура таблицы Users

В данной таблице хранится логин, пароль, роль пользователя.

Всю информацию о студенте можно посмотреть в таблице Students, структура которой представлена на рисунке 3.4.2.

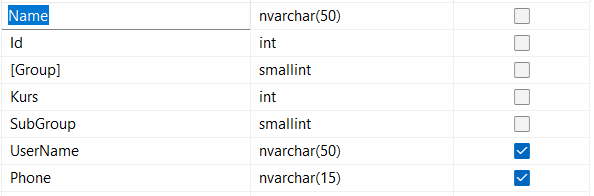


Рисунок 3.4.2 – Структура таблицы Students

Всю информацию о преподавателе можно посмотреть в таблице Teachers, структура которой представлена на рисунке 3.4.3

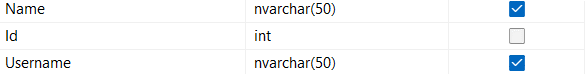


Рисунок 3.4.3 – Структура таблицы Teachers

Для хранения заметок есть таблица Notes представленная на рисунке 3.4.4

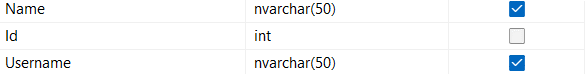


Рисунок 3.4.4 – Структура таблицы Notes

Для отмечания отсутствия есть таблица Absence представленная на рисунке 3.4.5

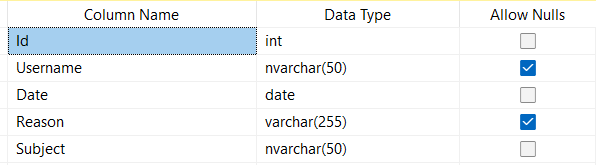


Рисунок 3.4.5 – Структура таблицы Notes

На рисунке 3.4.6 представлена структура таблицы Grade.

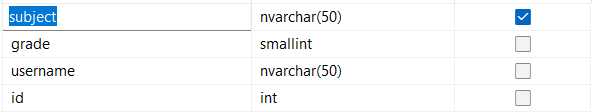


Рисунок 3.4.6 – Структура таблицы Grade

В данной таблице хранится информация об оценках.

На рисунке 3.4.7 представлена структура таблицы LabWorks.

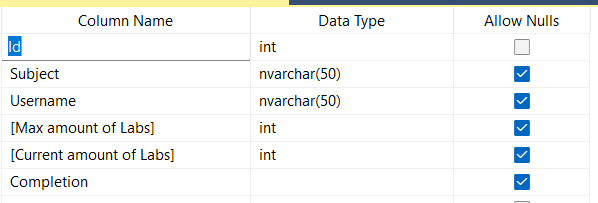


Рисунок 3.4.7– Структура таблицы LabWorks

В данной таблице хранится информация о лабораторных работах. Столбец Completion считается автоматически как Max amount of Labs/Current amount of Labs.

На рисунке 3.4.8 представлена структура таблицы Subjects. В ней находится названия всех предметов.

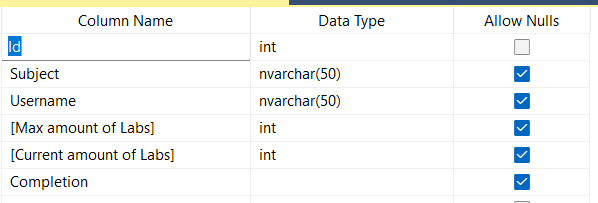


Рисунок 3.4.8 – Структура таблицы Subjects

На рисунке 3.4.9 представлена структура таблицы GroupSubjects.

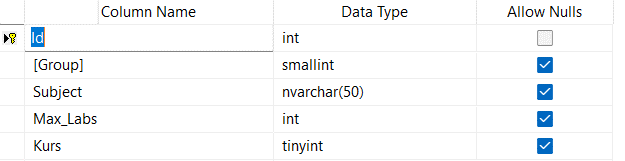


Рисунок 3.4.9 – Структура таблицы Subjects

В ней находится информация о предметах для определенной группы. Нужна чтобы выставлять количество лабораторных на сдачу по предмету всей группе сразу.

На рисунке 3.4.10 представлена структура таблицы Schedule.

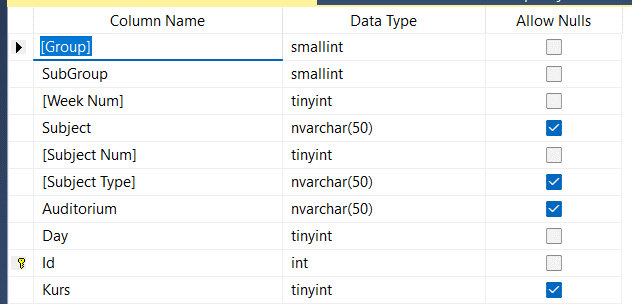


Рисунок 3.4.10 – Структура таблицы Schedule

Здесь хранится номер дня недели расписания, название предмета, номер аудитории, порядковый номер пары, тип пары (лекция, лабораторное занятие, практическое), номер недели, курс, группа.

4. Создание (реализация) программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

4.1 Диаграмма классов

Структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На данной диаграмме классов изображены классы программы, методы, поля и свойства классов. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов). Диаграмма классов проекта   
 представлена в графической части.

4.2 Реализация архитектуры MVVM

При создании приложения использовался паттерн проектирования MVVM. Он заключается в разделении представления от бизнес-логики. Это достигается за счёт ввода новой логической конструкции ViewModel. Он связывает представление и бизнес-логику приложения.

Нередко все сущности (модель) наследует интерфейс INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать интерфейс INotifyPropertyChanged, нужно объявить свойство PropertyChanged и метод OnPropertyChanged().

Для того, чтобы начать отслеживание свойства, необходимо вызывать метод OnPropertyChanged(“Имя свойства”).

Для удобной работы с данными, используется паттерн Repository. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

Чтобы не создавать отдельный класс репозитория для каждой модели базы данных, можно его обобщить, затем в классе UnitOfWork указать все обобщенные репозитории. На рисунке 4.2.1 показана структура для работы с базой данных.

Класс репозитория для работы с записями студентов представлен в приложении А.

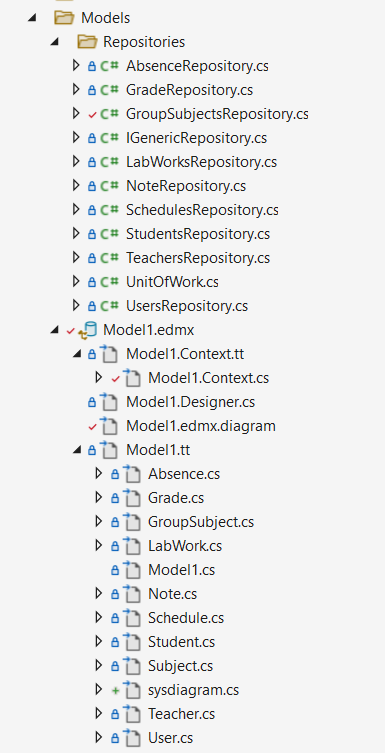


Рисунок 4.2.1 – Структура проекта для работы с базой данных

В приложении есть пакет View, который содержит все представления. Каждая отдельная страница имеет свой пакет, с файлами, используемыми представлением.

Структура пакета View представлена на рисунке 4.2.2.

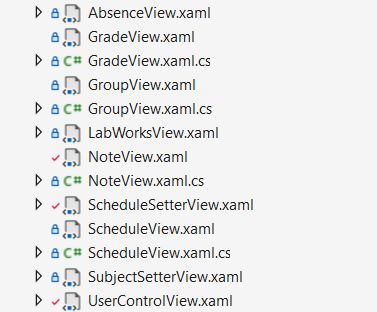


Рисунок 4.2.2 – Структура проекта для View

Пакет ViewModel содержит классы, которые настраивают логику отображения моделей на слое представления. В ViewModel обычно содержится модель, к которой создаются привязки.

Структура пакетов для Model и ViewModel представлена на рисунке 4.2.3

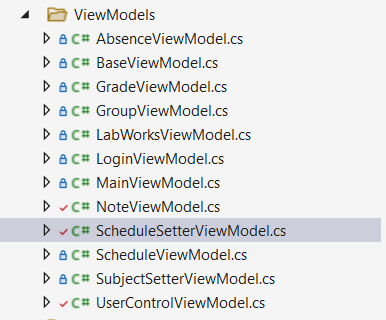


Рисунок 4.2.3 – Структура проекта для viewmodel

Класс ViewModel GradeViewModel приведён в приложение Б.

Назначение каждого файла в проекте описано в таблице ниже (таблица 4.2).

Таблица 4.2 – Описание структуры проекта

|  |  |
| --- | --- |
| Название | Краткое описание |
| LoginWindow.xaml | Окно авторизации. Требует ввода логина и пароля. |
| MainWindow.xaml | Главное окно приложения. |
| UserControlView.xaml | Окно управления админами, студентами, преподавателями. |
| SubjectSetterView.xaml | Окно для установки предметов группы и максимального количества лаб |
| ScheduleView.xaml.xaml | Окно просмотра расписания |
| ScheduleSetterView.xaml | Окно изменеия расписаня по группам |
| NoteView.xaml | Окно заметок |
| LabWorksView.xaml | Окно для просмотра прогресса выполненных лабораторных работ |
| GroupView.xaml | Окно для просмотра всех студентов группы |
| GradeView.xaml | Окно для просмотра и добавления оценок |
| AbsenceView.xaml | Окно для просмотра и добавления пропусков |

5. Тестирование, проверка работоспособности и анализ полученных результатов

Для обеспечения корректности работы программы обрабатываются различные ошибки, возникающие в процессе работы. Данное программное средство использует подключение к базе данных, следовательно, неправильно введенные данные или же их отсутствие может повлечь за собой неработоспособность приложения.

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим некоторые такие сценарии и посмотрим на их обработку.

В моём курсовом проекте валидация данных в основном сделана на основе интерфейса IDataErrorInfo и благодаря ему определения CanExecute моих команд.

В момент регистрации, возможна такая ситуация, в которой пользователь ничего не ввёл или ввёл неверные данные. Обработка данного сценария приведена на рисунке 5.1.

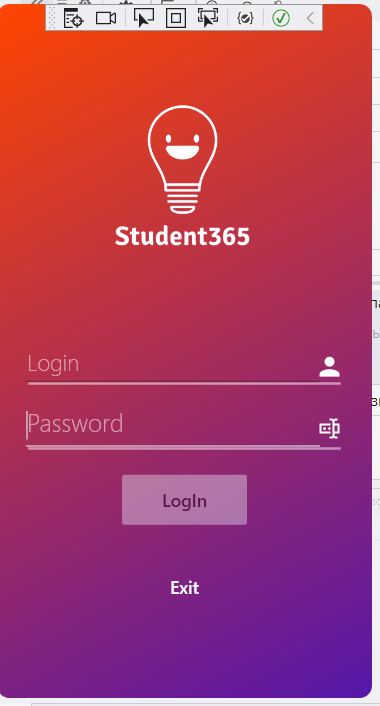


Рисунок 5.1 – Любое поле оставлено пустым

Кнопка Login не активна до момента ввода обоих полей.

Сценарий, при котором введён логин, и пароль но данной комбинации нет в базе данных, приведен на рисунке 5.2.

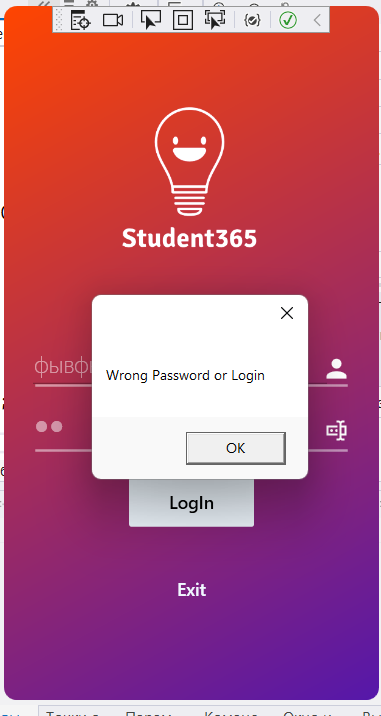


Рисунок 5.2 – Несуществующий пользователь

Сценарий, при котором Админ пытается добавить пользователя с уже существующим логином представлен на рисунке 5.3.

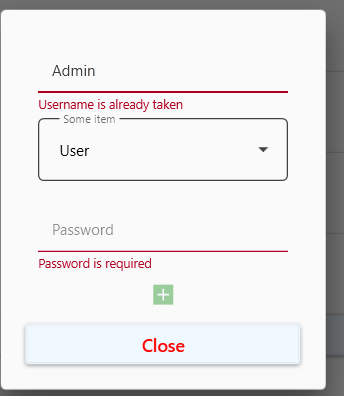


Рисунок 5.3 – Занятый логин и отсутствующий пароль

Сценарий, при котором Админ пытается добавить студента с неправильно заполненными полями, приведён на рисунке 5.4 .

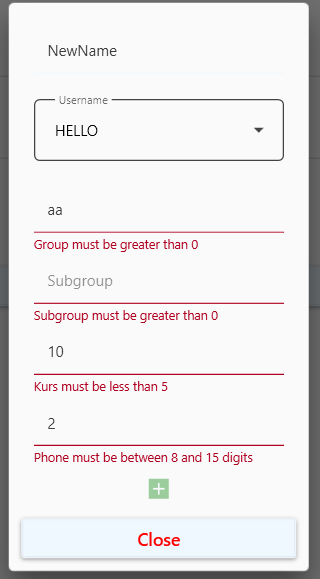


Рисунок 5.4 – Неверные данные при регистрации студента

Сценарий, при котором Студент пытается добавить оценки по предмету с неправильно заполненными полями, приведён на рисунке 5.5

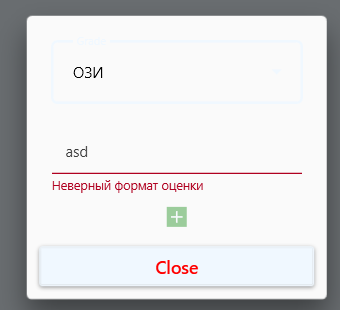


Рисунок 5.5– Неверные данные при добавлении оценки

6. Руководство по использованию

При запуске приложения у вас появится окно авторизации, где вы выбрать либо зарегистрировать нового пользователя, либо совершить вход в существующий аккаунт . Для смены регистрации и входа существует отдельная кнопка.

Окно авторизации представлено на рисунке 6.1.

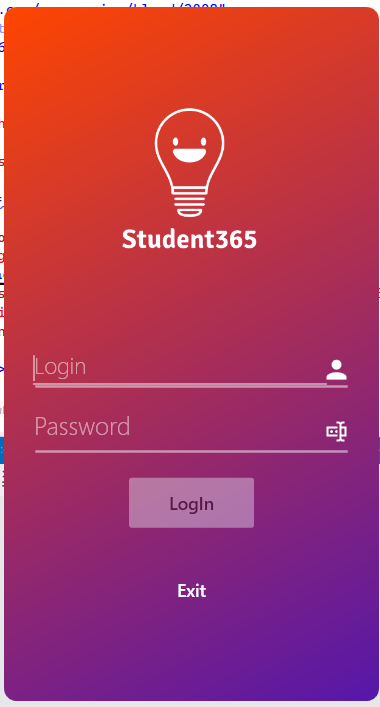


Рисунок 6.1 – Окно авторизации

После успешного входа под вашей учётной записью у вас появится главное меню, которое изображено на рисунке 6.2.

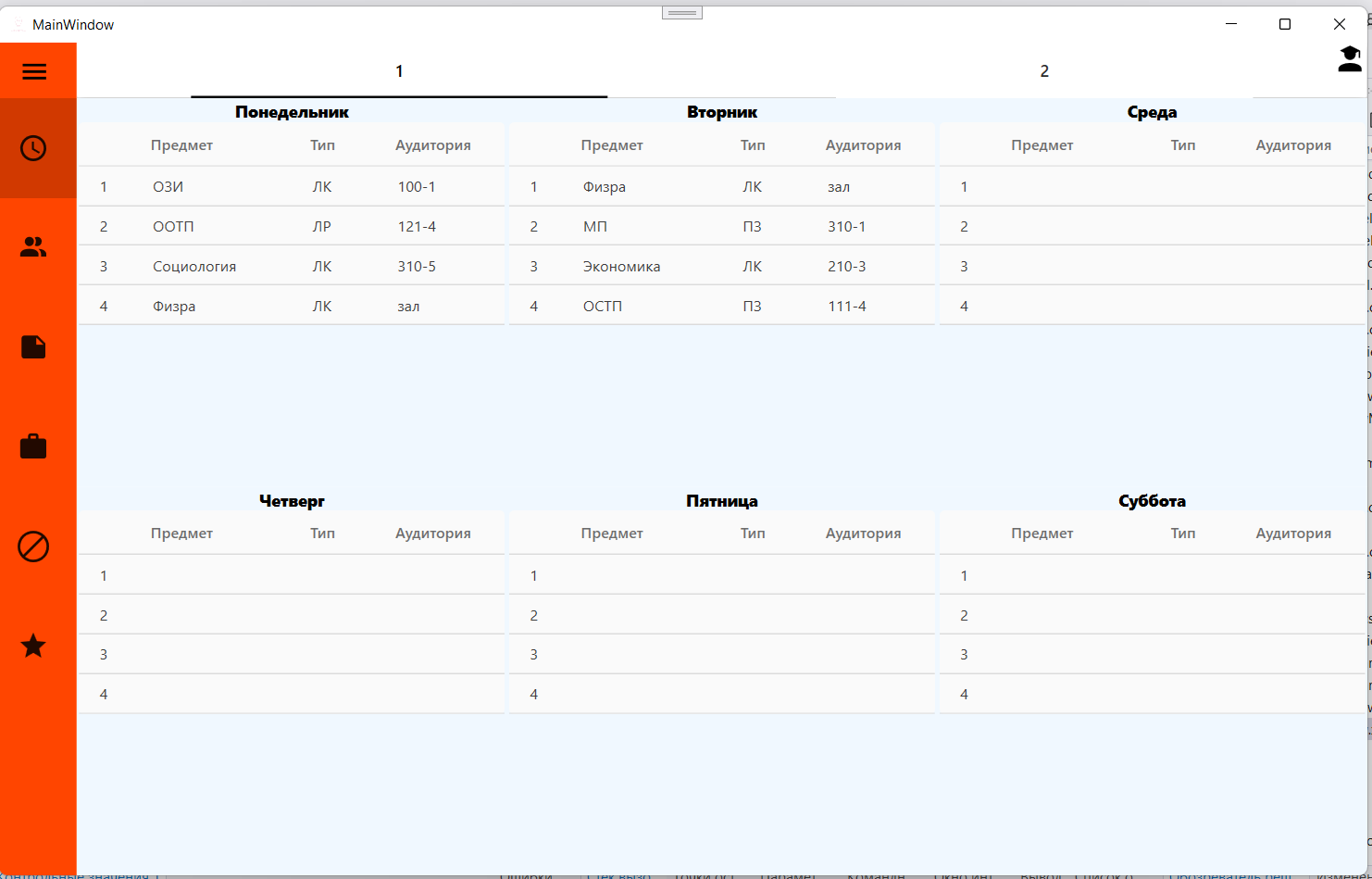


Рисунок 6.2 – Главное меню приложения

В нем изображены вкладки, ведущие на все страницы приложения. Справа сверху имеется иконка, показывающая, является ли текущий пользователем клиентом администратором или студентом. Данное вспомогательное меню представлено на каждой странице приложения.

Рассмотрим подробнее каждую из страниц.

На рисунке 6.3 представлено окно обзора группы вверху есть поле куда можно ввести номер группы для просмотра любой группы. ПО умолчанию он ставиться на группу текущего студента

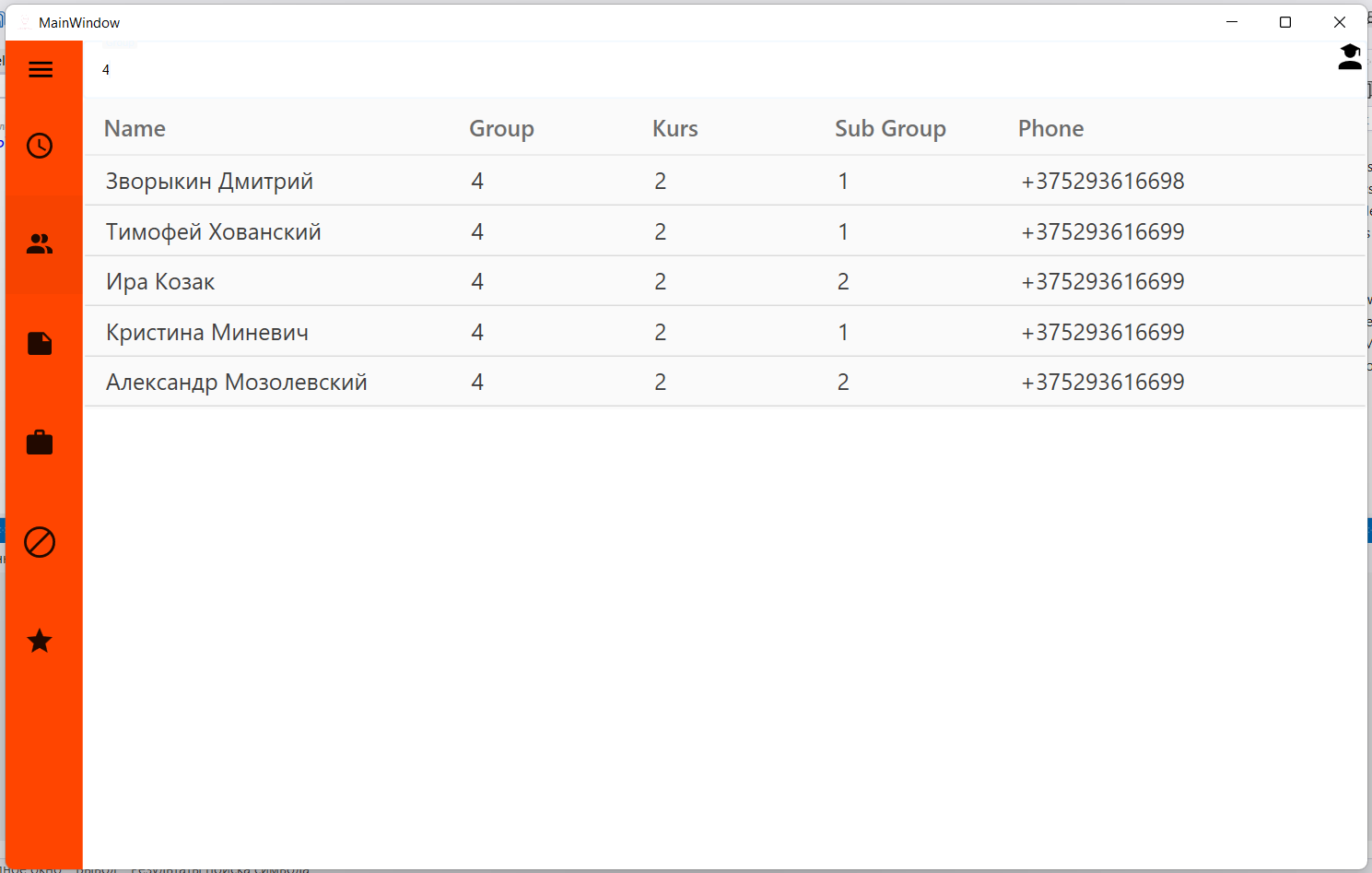


Рисунок 6.3 – Окно обзора группы

На рисунке 6.4 представлено окно заметок. Чтобы добавить заметку, нужно написать название заметки в поле, нажать кнопку добавления заметки. После чего можно писать сам текст заметки, для сохранения необходимо нажать кнопку подтверждения. Для удаления необходимо выбрать заметки и нажать красную кнопку удаления

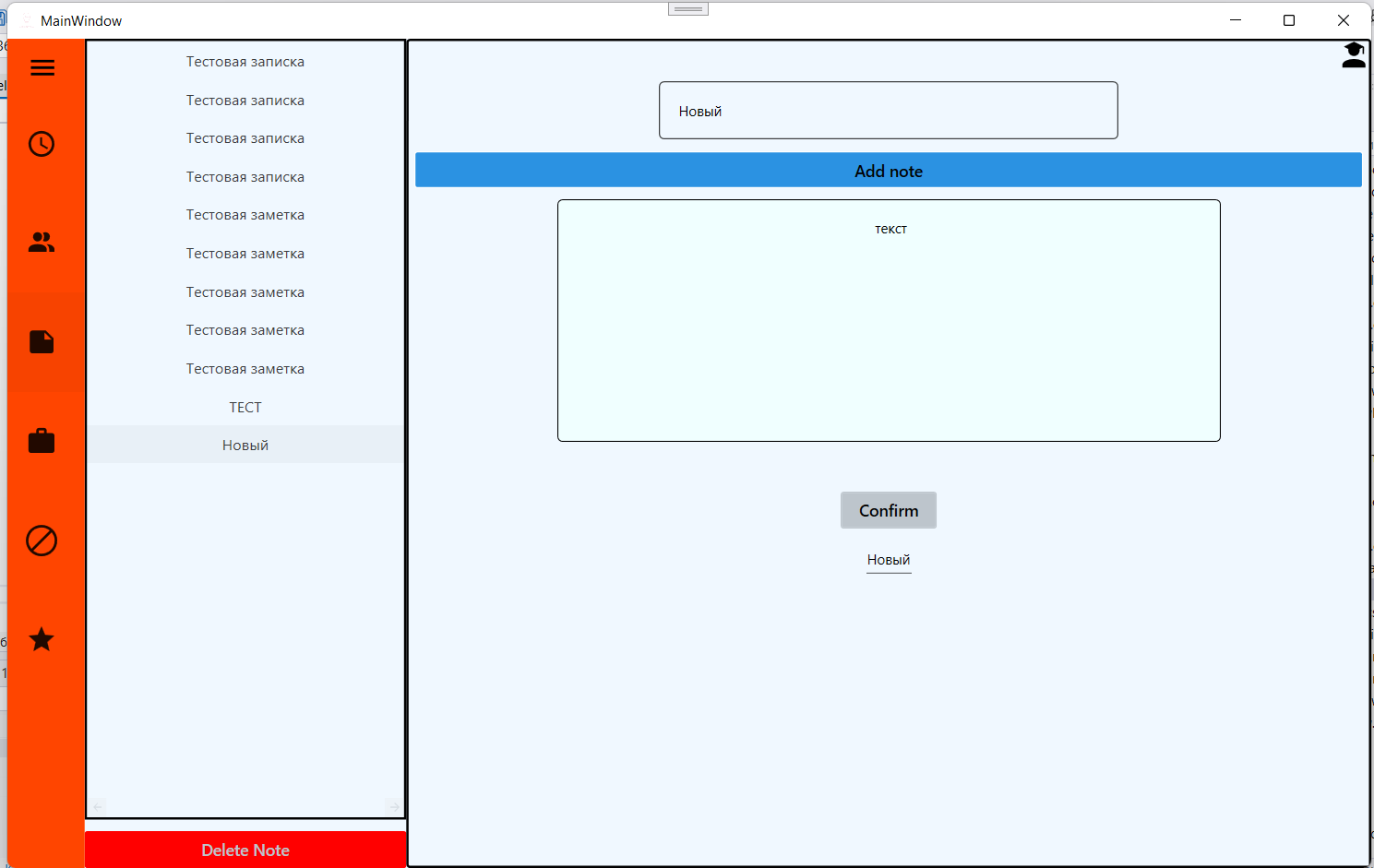


Рисунок 6.4 – Окно заметок

На рисунке 6.5 представлено окно лабораторных работ. Оно предназначено для отслеживания прогресса сдачи лабораторных работ. Лабораторные работы и количество их на сдачу устанавливается учителе для всей группы одновременно.

Для того чтобы отметить сданную лабу необходимо нажать ка зелёный пдюс на строке желаемого предмета. При необходимости можно пометить лабораторную как несданную нажав на красный минус.

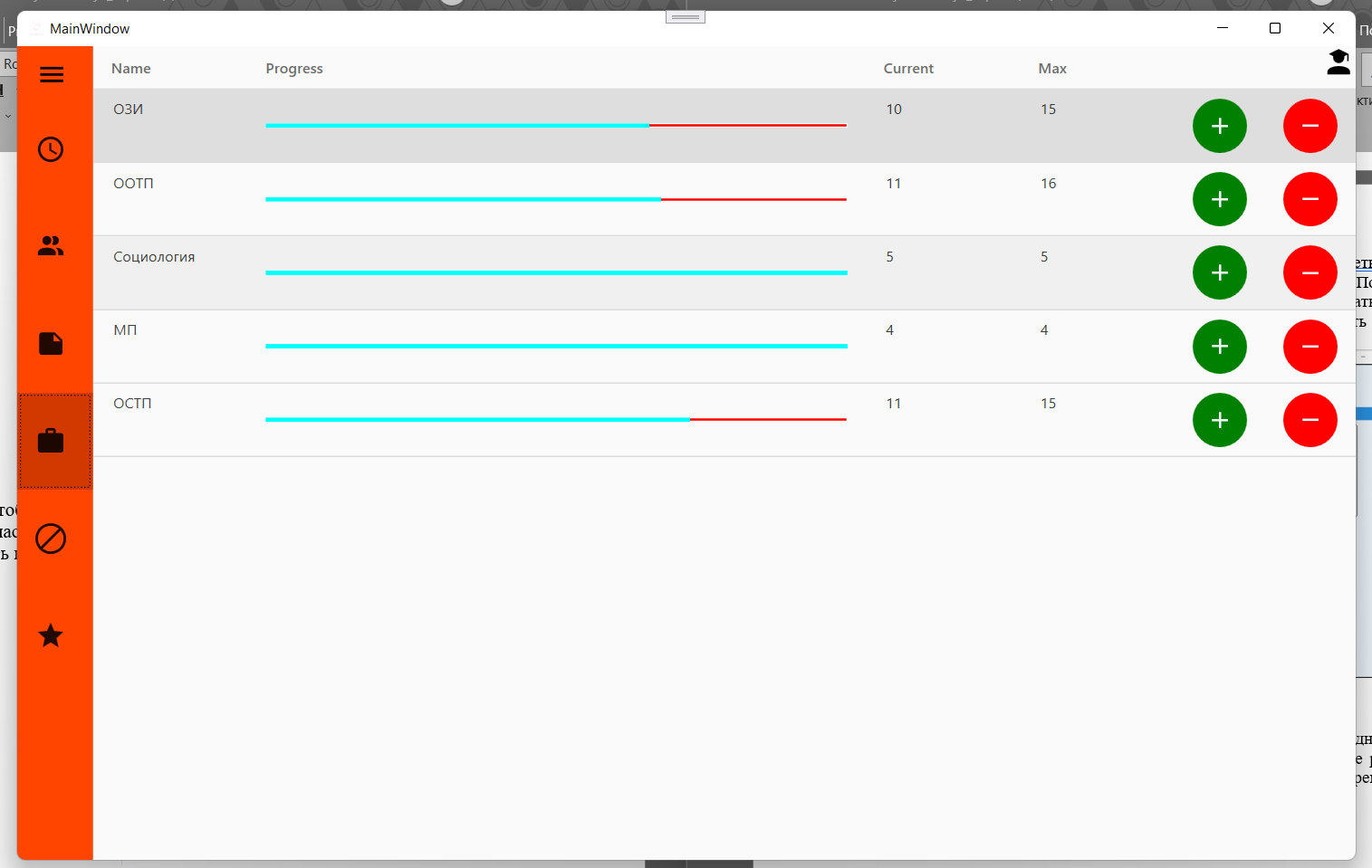


Рисунок 6.5–Окно лабораторных работ

На рисунке 6.6 представлено окно просмотра и отмачивании пропусков. Чтобы добавить пропуск, необходимо нажать на кнопку сверху и заполнить форму, которая включает: предмет, дату пропуска и причину отсутствия. После этого, она добавляется в список всех пропусков в таблицу и к общему подсчёту пропусков.

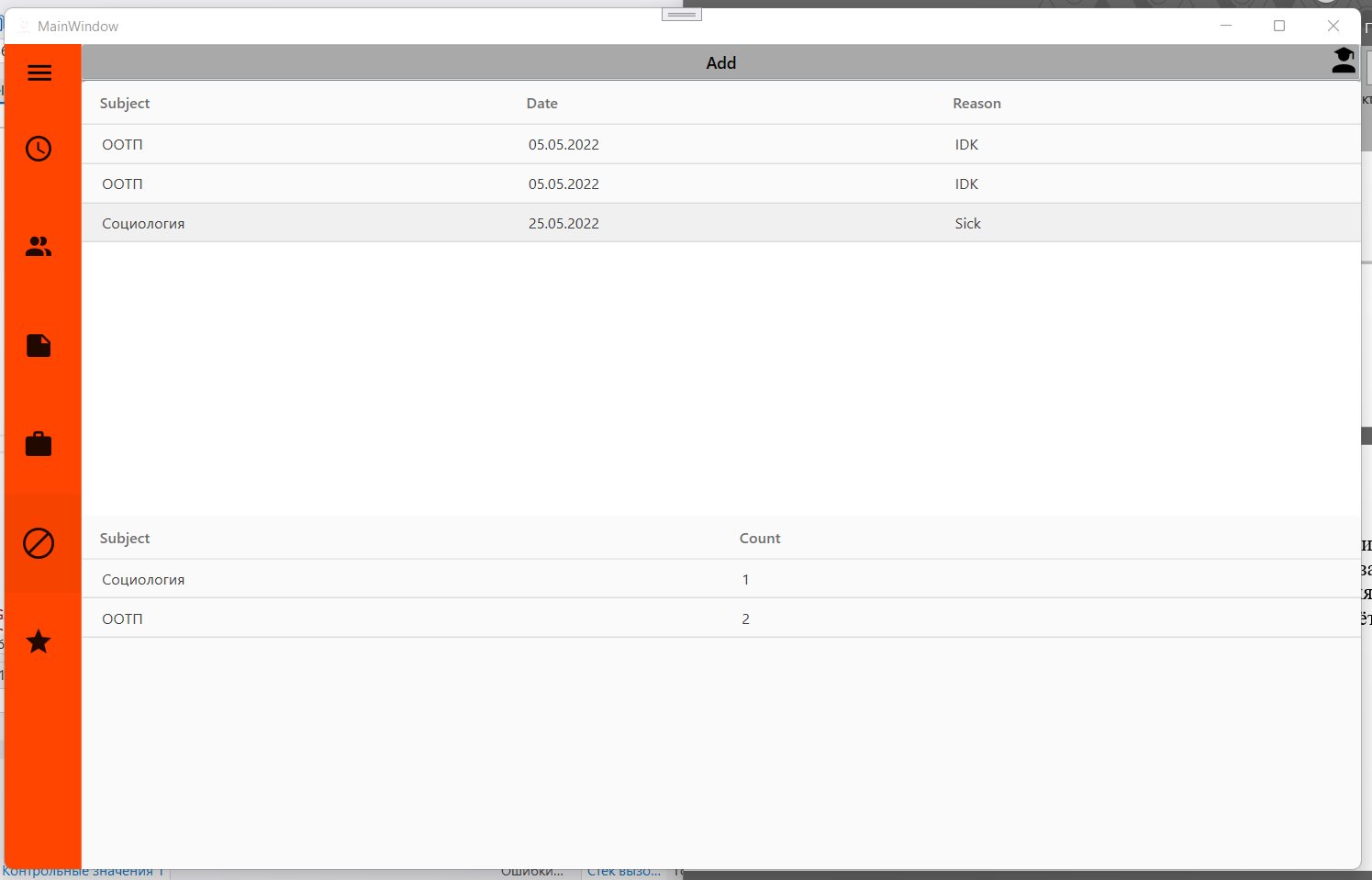


Рисунок 6.6 – Окно пропусков

На рисунке 6.7 представлено окно просмотра и добавления оценок. Для того, чтобы добавить оценку, необходимо нажать на кнопку сверху и заполнить форму: предмет и оценка. После чего она добавиться в таблицу ниже. При необходимости можно удалить оценку нажатием на неё в таблице.

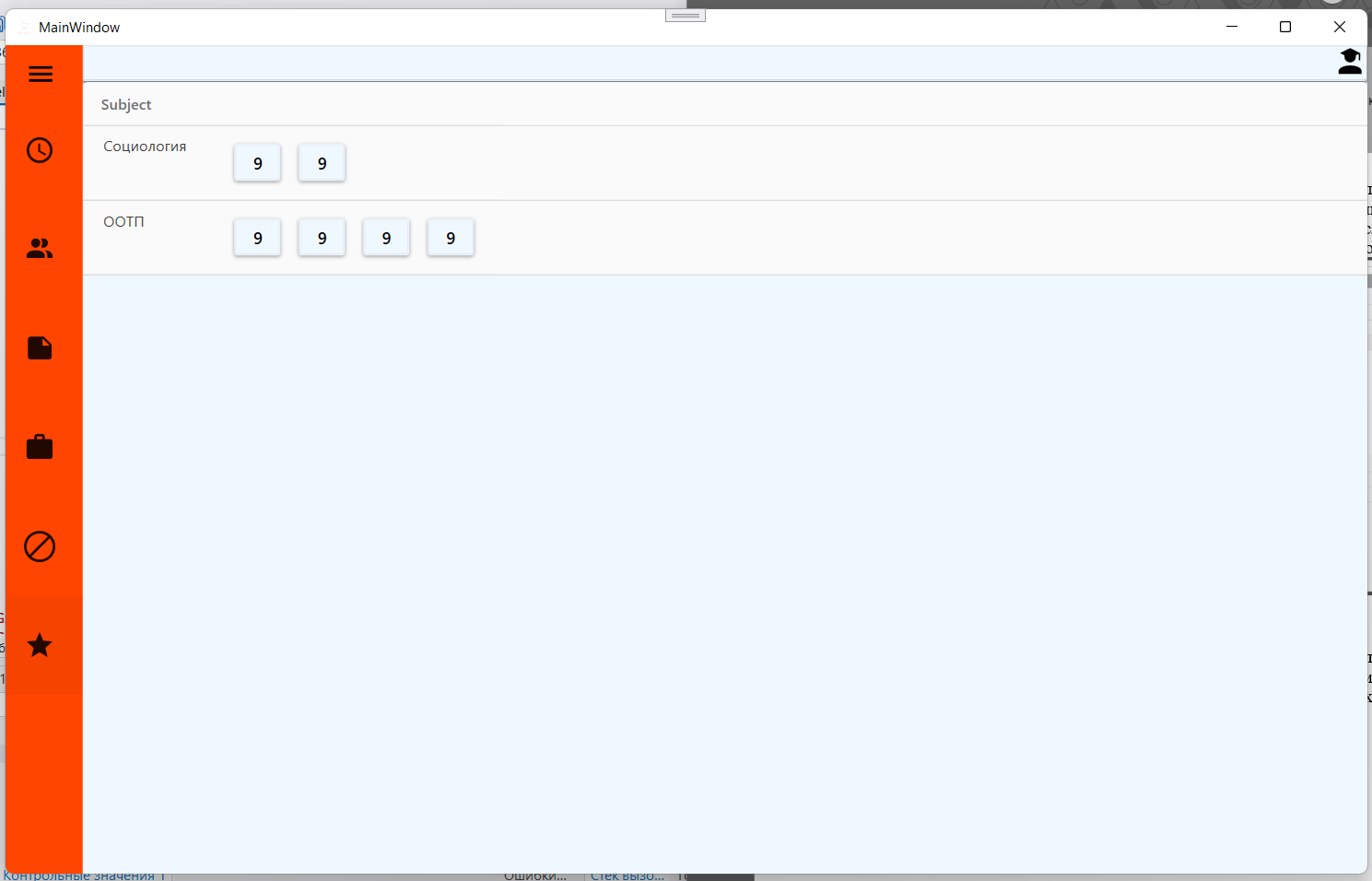


Рисунок 6.7 – Окно оценок

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было разработано программное средство “Student365”, которое помогает студентам в учёбе.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

регистрация пользователей;

авторизация пользователей;

создание и просмотр заметок;

создание и просмотр оценок;

создание и просмотр пропусков;

составление расписания занятий;

просмотра расписаний;

просмотр группы;

хранение и отображение прогресса по предметам;

хранение личной информации в базе данных;

Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке.

Проектирование осуществлялось по паттерну проектирования MVVM.

Для удобной работы с данными, был реализован паттерн Repository,UnitOfWork,Store.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1) Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2021. – 175 с.

2) Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с. [3]. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: http://msdn.microsoft.com/library/

4) METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

5) ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru

6) StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>

7) Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>.

Приложение А

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Student365.Models.Repositories

{

internal class StudentsRepository : IGenericRepository<Student>

{

private DbContext \_context = new Student365Entities();

private DbSet<Student> \_dbSet;

public StudentsRepository()

{

\_dbSet = \_context.Set<Student>();

}

public void Create(Student item)

{

\_dbSet.Add(item);

\_context.SaveChanges();

var groupsubj = UnitOfWork.GroupSubjectsRepository.GetAllSubjectsByGroupId(item.Group, item.Kurs);

foreach (var grrouSubject in groupsubj)

{

UnitOfWork.LabWorksRepository.AddLabInfo(grrouSubject);

}

}

public Student FindById(int id)

{

return \_dbSet.Find(id);

}

public IEnumerable<Student> Get()

{

return \_dbSet.ToList();

}

public IEnumerable<Student> Get(Func<Student, bool> predicate)

{

return \_dbSet.Where(predicate).ToList();

}

public void Remove(Student item)

{

\_dbSet.Remove(item);

\_context.SaveChanges();

}

public void Update(Student item)

{

\_context.Entry(item).State = EntityState.Modified;

\_context.SaveChanges();

Продолжение листинга

}

public ObservableCollection<Student> GetGroup()

{

var group = GetCurrentUserGroup();

var kurs = GetCurrentUserKurs();

return new ObservableCollection<Student>(\_dbSet.Where(x => x.Group == group && x.Kurs == kurs).ToList());

}

public int GetCurrentUserGroup()

{

return \_dbSet.Where(x => x.UserName == UnitOfWork.CurrentsUser.UserName).Select(x => x.Group)

.FirstOrDefault();

}

public ObservableCollection<Student> GetGroup(int group, short Kurs)

{

return new ObservableCollection<Student>(\_dbSet.Where(x => x.Group == group && x.Kurs == Kurs).ToList());

}

public int GetCurrentUserSubGroup()

{

return \_dbSet.Where(x => x.UserName == UnitOfWork.CurrentsUser.UserName).Select(x => x.SubGroup)

.FirstOrDefault();

}

public ObservableCollection<Student> GetAllStudents()

{

return new ObservableCollection<Student>(\_dbSet.ToList());

}

public List<Student> GetAllStudentsByGroupId(int groupid, short Kurs)

{

return \_dbSet.Where(x => x.Group == groupid && x.Kurs == Kurs).ToList();

}

public int GetCurrentUserKurs()

{

return \_dbSet.Where(x => x.UserName == UnitOfWork.CurrentsUser.UserName).Select(x => x.Kurs)

.FirstOrDefault();

}

}

Листинг – Класс StudentRepositoty

Приложение Б

public class GradeViewModel : BaseViewModel, IDataErrorInfo

{

public ObservableCollection<Custom> Subjects

{

get => \_subjects;

set

{

\_subjects = value;

OnPropertyChanged();

}

}

private ObservableCollection<Custom> \_subjects;

private ObservableCollection<string> \_subjects\_list;

public ObservableCollection<string> Subjects\_list

{

get => \_subjects\_list;

set

{

\_subjects\_list = value;

OnPropertyChanged();

}

}

private string \_newSubject;

public string NewSubject

{

get => \_newSubject;

set

{

\_newSubject = value;

OnPropertyChanged();

}

}

private string \_grade;

public string Grade

{

get => \_grade;

set

{

\_grade = value;

OnPropertyChanged();

}

}

private KeyValuePair<int, short> \_selected;

public KeyValuePair<int, short> Selected

{

get => \_selected;

set

{

\_selected = value;

OnPropertyChanged();

}

}

public ICommand AddCommand { get; }

public ICommand Remove { get; }

Продолжение листинга

public GradeViewModel()

{

GetSubjects();

Subjects\_list = new ObservableCollection<string>(

UnitOfWork.GroupSubjectsRepository.GetAllSubjectsByGroupId(UnitOfWork.StudentsRepository

.GetCurrentUserGroup(), UnitOfWork.StudentsRepository.GetCurrentUserKurs()).Select(x => x.Subject));

AddCommand = new AddGradeCommand(this);

Remove = new RemoverGradeCommnad(this);

}

string IDataErrorInfo.Error

{

get { return null; }

}

public string this[string columnName]

{

get

{

string error = null;

switch (columnName)

{

case nameof(Grade):

var grade = int.TryParse(Grade, out int n);

if (!grade)

return "Неверный формат оценки";

// Validate property and return a string if there is an error

if (n > 10 || n < 1)

{

return "Оценка должна быть в диапазоне от 0 до 10";

}

break;

case nameof(NewSubject):

if (string.IsNullOrWhiteSpace(NewSubject))

error = "Не может быть пустым";

break;

}

return error;

}

}

public void GetSubjects()

{

Subjects = new ObservableCollection<Custom>();

var grades = UnitOfWork.GradeRepository.GetAllGradesByCurrentUser();

var subjects = grades.Select(x => x.subject).Distinct();

foreach (var subject in subjects)

{

Subjects.Add(new Custom()

{

Subject = subject,

Grades = grades.Where(x => x.subject == subject)

.ToDictionary(x => x.id, x => x.grade1)

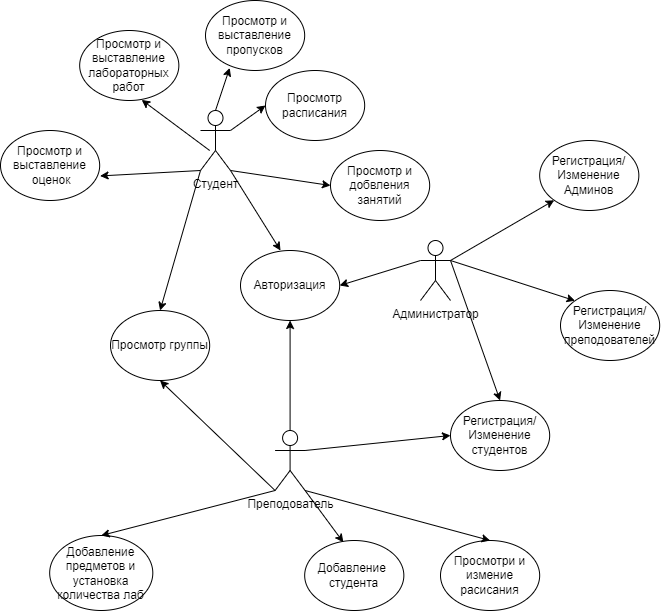
});

}

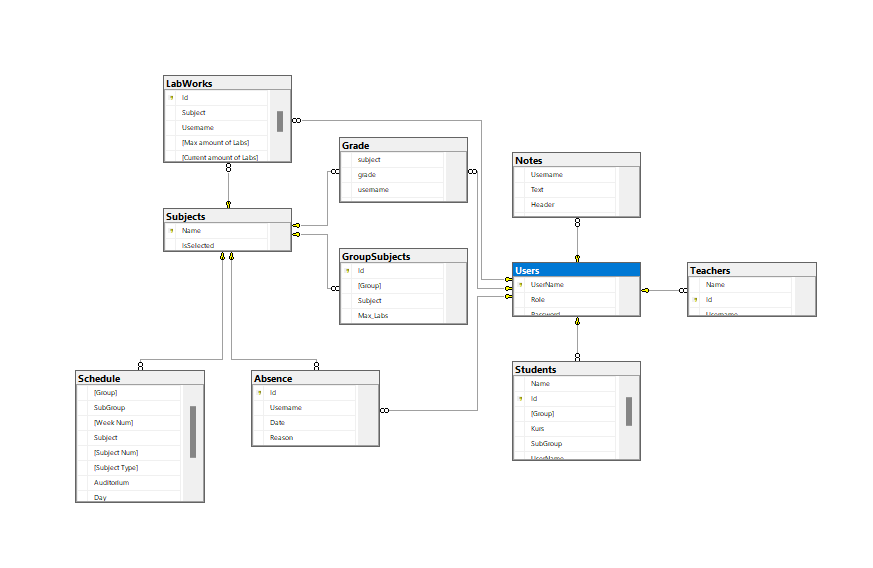
Листинг – Класс GradeViewModel

Графический материал

Диаграмма вариантов использования



Логическая схема базы данных



Диаграммы классов

