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 «StudTasks»

Исполнитель

студентка 2 курса группы 4 Миневич Кристина Викторовна

(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 2](#_Toc104344762)

[1. Аналитический обзор литературы и постановка задачи 4](#_Toc104344763)

[1.1 Анализ прототипов 4](#_Toc104344764)

[1.2 Постановка задачи 8](#_Toc104344765)

[2. Анализ требований к программному средству и разработка функциональных требований 9](#_Toc104344766)

[2.1 Описание средств разработки 9](#_Toc104344767)

[2.2 Описание разрабатываемой функциональности приложения 10](#_Toc104344768)

[2.3 Спецификация функциональных требований 11](#_Toc104344769)

[3. Проектирование программного средства 12](#_Toc104344770)

[3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства 12](#_Toc104344771)

[3.2 Проектирование архитектуры приложения 14](#_Toc104344772)

[3.3 Схема работы приложения 16](#_Toc104344773)

[3.4 Проектирование логики сценариев использования 16](#_Toc104344774)

[3.5 Модель базы данных 17](#_Toc104344775)

[4. Создание (реализация) программного средства 20](#_Toc104344776)

[4.1 Выполняемые функции 20](#_Toc104344777)

[4.2 Реализация общей структуры проекта 20](#_Toc104344778)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 25](#_Toc104344779)

[6. Методика использования программного средства 32](#_Toc104344780)

[ЗАКЛЮЧЕНИЕ 41](#_Toc104344781)

[СПИСОК ЛИТЕРАТУРЫ 42](#_Toc104344782)

[Приложение А 43](#_Toc104344783)

[Приложение Б 45](#_Toc104344784)

[Графический материал 46](#_Toc104344785)

[Диаграмма вариантов использования 46](#_Toc104344786)

[Логическая схема базы данных 47](#_Toc104344787)

[Диаграммы классов 48](#_Toc104344788)

[Диаграмма последовательности 49](#_Toc104344789)

ВВЕДЕНИЕ

При современном темпе жизни всех нас объединяет одна проблема – нам катастрофически не хватает времени. Мы постоянно куда-то спешим, бежим, а времени все равно нет! Жизнь современных школьников и студентов не так проста: ежедневные пары, уроки, зачёты, горы домашнего задания, да ещё и какие-то дополнительные занятия сваливаются на голову как большой снежный ком.

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

Поэтому сейчас на первый план выходит умение эффективно распоряжаться своим временем. Уметь распределять его так, чтобы в первую очередь выполнялись действительно важные дела, которые двигают нас вперед, и только потом всё остальное.

Бумажные ежедневники и блокноты утратили актуальность, а им на смену пришли органайзеры, которые можно установить на свой смартфон или компьютер, поэтому темой моего курсового проекта является разработка программного средства «Планировщик задач StudTasks».

Данное программное обеспечение позволит студентам правильно распоряжаться своим временем и планировать задачи с указанием дедлайна и уровня важности. Программа автоматически при наличии невыполненных задач с указанным дедлайном уведомляет пользователя о них, а также формирует статистику по выполненным задачам и представляет данные в виде графика. В программе также присутствует возможность добавления и редактирования расписания занятий. Для сохранности личных данных используется система авторизации. Помочь найти задачу может строка поиска.

Итак, насыщенные студенческие будни требуют чёткого и детального планирования для достижения отличных результатов, и максимально эффективно с этой задачей поможет справиться органайзер StudTasks.

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 моего курсового проекта.

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 Анализ прототипов

Сейчас существует довольно много приложений, облегчающих жизнь студентам и не только. Рассмотрим наиболее популярные и удобные планеры, органайзеры и электронные ежедневники.

Программа *iStudiez* охотно придет на помощь тем, кто грызет гранит науки, и пока что не нуждается во взрослых органайзерах. Это приложение, которое совмещает в себе функции календаря и ежедневника. Как отмечают разработчики, их программа пригодится в первую очередь студентам. Интерфейс приложения показан на рисунке 1.1.1.



Рисунок 1.1.1 – Интерфейс приложения “iStudiez”

Приложение напоминает обычные ежедневники, которыми так любят пользоваться взрослые дяденьки. Главным элементом здесь является календарь со встроенным органайзером. Разработчики сделали правильное умозаключение: у школьников и студентов, в отличие от бизнесменов, нет каких-то важных встреч или оперативных совещаний, им всего лишь нужно расписание уроков и факультативных занятий. Такой функционал реализован очень просто: в специальном разделе нужно указать все изучаемые предметы (алгебра, информатика, английский и так далее), а потом внести их в расписание, ничего не печатая, просто выбирая нужный вариант из списка.

Также можно подробно настроить систему получения баллов, оценок и экзаменов. Тут стоит знать, что приложение использует западную систему. А все потому, что оно было создано иностранными разработчиками.

К достоинствам iStudiez можно отнести и наличие версий для смартфона и планшета. На тот случай, если вы забыли какой предмет дальше по расписанию или как зовут лектора, отчаиваться не стоит: информация будет доступна и на вашем iPhone. Также приложение поддерживает последние версии OS X и iOS.

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 учёбой.

*Trello* – программа для управления проектами небольших групп, разработанная Fog Creek Software. Веб-приложение напоминает вывеску. Связано это с тем, что с японского языка слово «trello» переводится как «рекламный щит». Трелло позволяет работать над проектами как индивидуально, так и в команде. Также инструмент даёт возможность подключить к работе над задачами и проектом сторонних исполнителей. Trello доступен на компьютерах в виде десктопной версии, в браузерной версии и версии для мобильных устройствах.

Проект в Трелло — отдельная доска, на которой в разных колонках размещаются карточки. Столбцы могут отображать статус задач, находящихся в них, как на канбан-доске или быть определённой сущностью. Задачи в Trello представляют из себя карточки. Карточки могут как содержать тело задачи (описание, требования и т. д.), так и содержать информацию о каком-нибудь объекте.

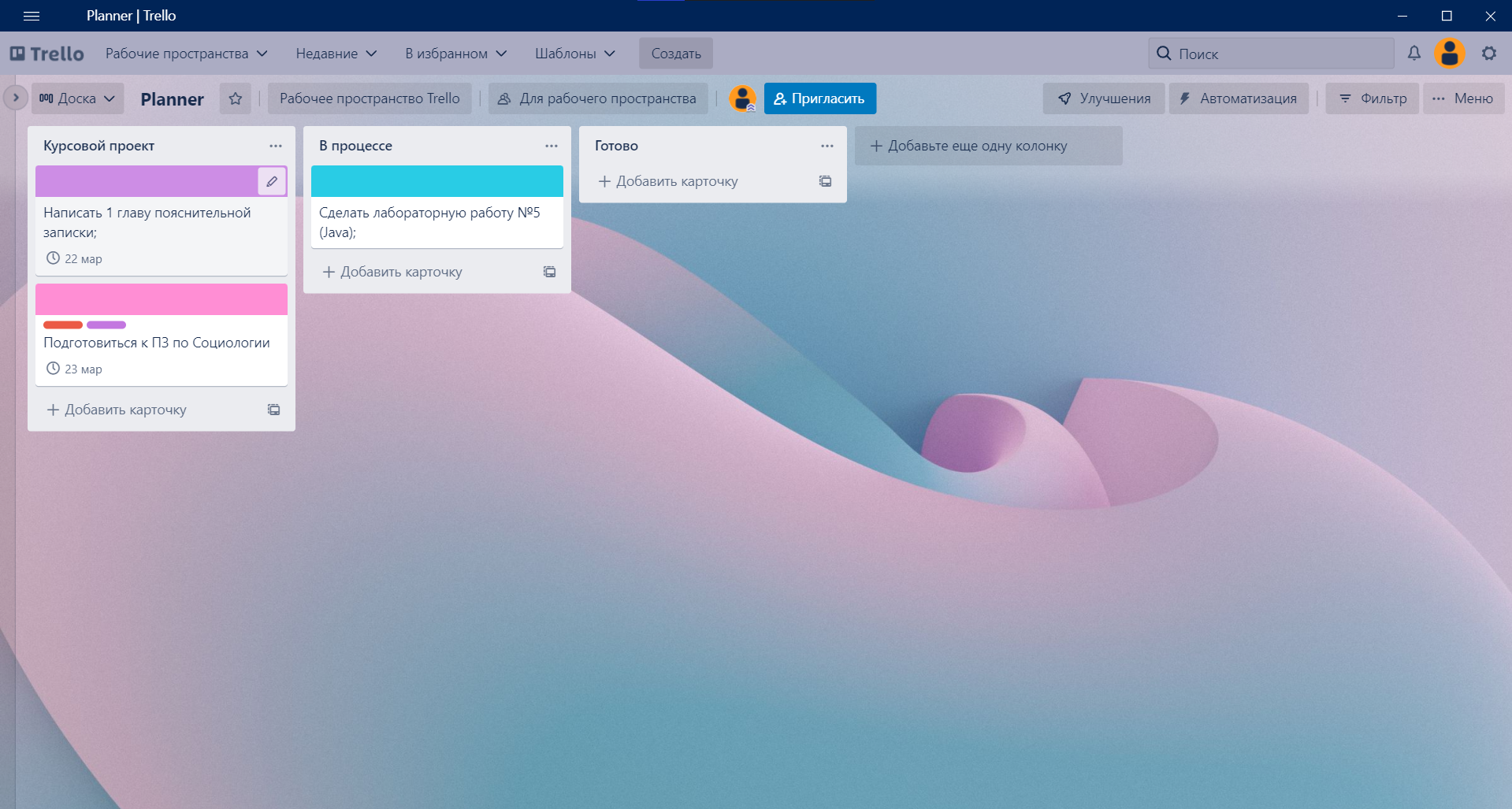
**

Рисунок 1.1.2 – Интерфейс приложения “Trello”

Карточки имеют множество возможностей. Они предназначены для обсуждений, голосований, загрузки файлов и данных. Есть возможность задавать дедлайны, назначать текстовые и цветовые метки. Важно заметить, что все участники команды видят изменения в реальном времени и могут наблюдать состояния отдельных исполнителей — онлайн или офлайн.

Плюсы:

* удобный и понятный интерфейс;
* многофункциональность в использовании (можно оставить доску заданий приватной, открывать отдельные карточки задач для других пользователей или сделать доску полностью открытой);
* использование с десктопа, данные синхронизируются между разными устройствами;
* есть возможность добавлять абсолютно любые документы в описание к задачам.

Минусы:

* нельзя фиксировать в программе время, затраченное на задачу;
* нет возможности выводить отчеты по задачам.

*ToDoist* – веб-сервис и набор программного обеспечения для управления задачами. Задачи могут также содержать заметки с файлами любого типа. Задачи можно помещать в проекты, сортировать по фильтрам, присваивать им метки, редактировать и экспортировать. Интерфейс приложения показан на рисунке 1.1.3.

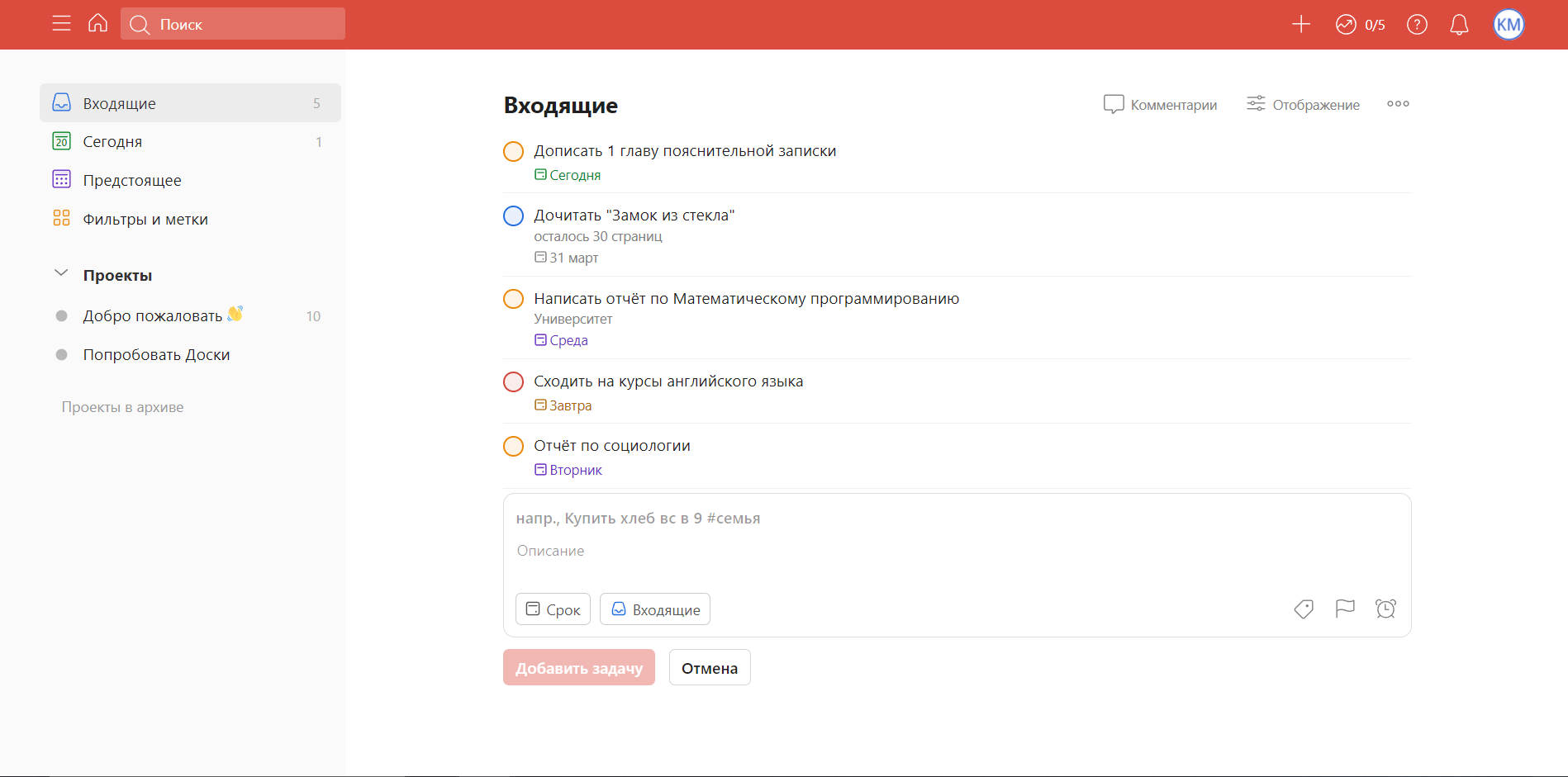


Рисунок 1.1.3 – Интерфейс приложения “ToDoist”

Для каждой из задач виден её проект и приоритет. Отсюда же можно быстро просмотреть комментарии и вложенные файлы. Слева от списка задач можно переключиться на входящие дела или на дела только за сегодня. Также можно быстро перейти в нужный проект или посмотреть задачи по меткам и пользовательским фильтрам.

Сервис отлично работает с горячими клавишами и поэтому управлять задачами можно без отрыва от клавиатуры. Любую задачу можно по-быстрому добавить во входящие или сразу же назначить ей проект, приоритет, теги и напоминание. Также к любой задаче можно добавить неограниченное количество комментариев и файловых вложений.

Фильтрация задач при помощи ключевых слов и меток не только помогает отобразить нужные сейчас дела, но и быстро находить задачи в сложных проектах.

ToDoist поддерживает десятки различных интеграций со сторонними сервисами. Благодаря которым, такие любители автоматизации, как я, смогут ещё более продуктивно планировать свой день и выполнять поставленные задачи в срок.

К недостаткам можно отнести то, что синхронизация и многие важные функции доступны только премиум-пользователям, продвинутые «фишки» не очевидны и про них нужно знать, нет гибкой настройки отображения задач.

*LeaderTask* – это простой и удобный органайзер и планировщик задач с функцией таск-менеджера. Приложение призвано помочь эффективно решать задачи как обычных пользователей, так и корпоративного сегмента, где нужно контролировать выполнение поручений сотрудников на постоянной основе.

Дизайн приложения выглядит лаконичным настолько, будто его и вовсе нет. В интерфейсе, показанном на рисунке 1.1.4, просто нет таких вещей, с которыми не удается совладать интуитивным способом. Это важно, потому что внешний вид интерфейса LeaderTask призван не удивлять, а максимально облегчить первоначальную адаптацию и быстрее включиться в работу.

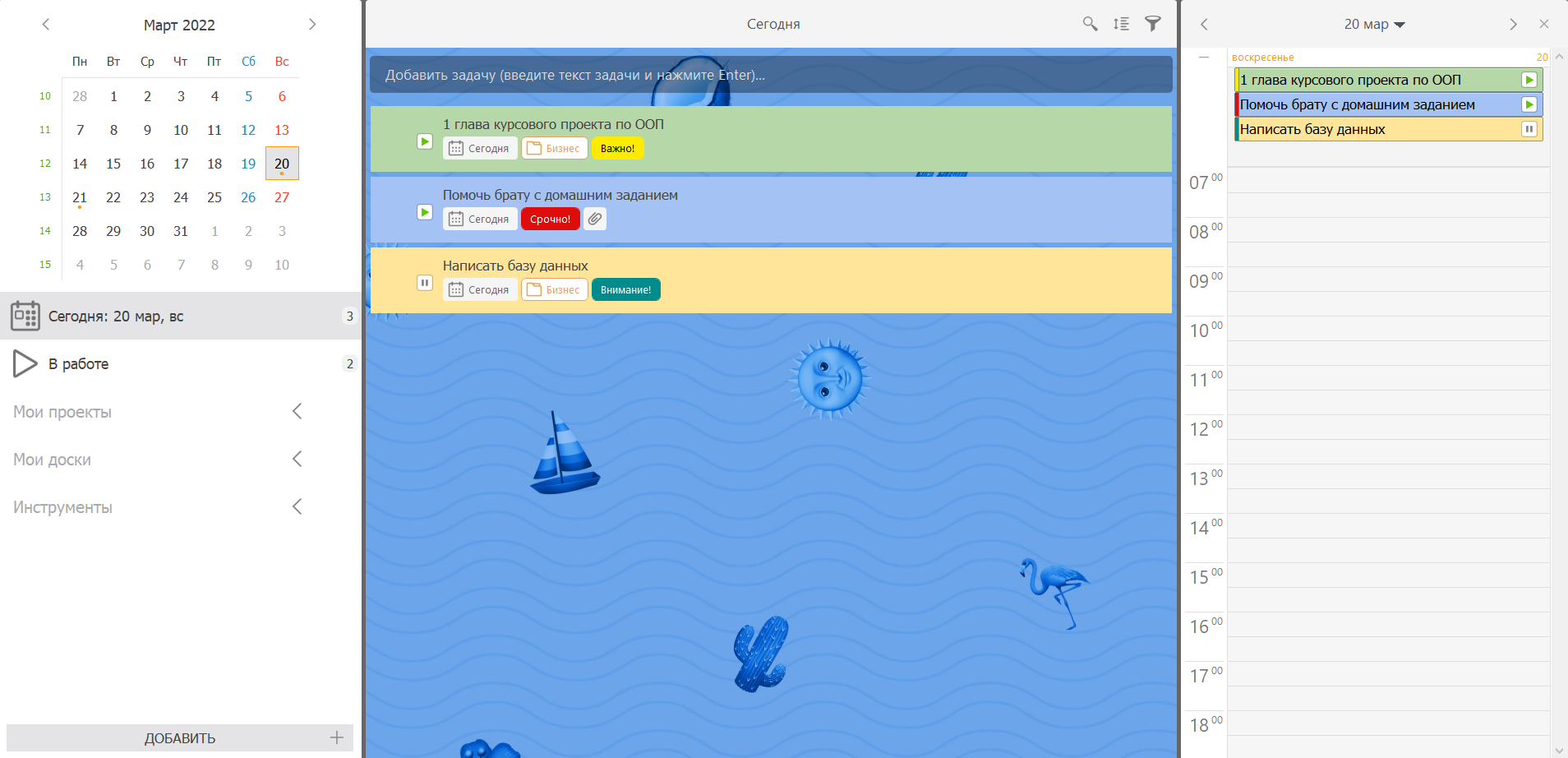


Рисунок 1.1.4 – Интерфейс приложения “LeaderTask”

Задачи в LeaderTask, это все ваши дела, которые вам необходимо выполнить. Вы можете присвоить цвета и метки задачам, а также установить дату выполнения.

Когда вы создаете задачу, вы вписываете её название, а также добавляете в неё разное содержимое: заметки, чек-листы, файлы.

Задачам можно установить статусы, использовать ссылки для доступа к задачам, а также отправлять ссылки внутри задач (на сторонние ресурсы, ссылки на локальные файлы, ссылки на приложенные файлы).

LeaderTask — это яркий пример совмещения двух, казалось бы, разных направлений — личного органайзера и коллективного таск-менеджера для организации работы компании. Причём, что удивительно, LeaderTask одинаково силен в обоих направлениях. Также важно отметить, что LeaderTask способен без потерь в скорости переваривать большие объемы тасков, ведь его база данных оптимизирована под высокие нагрузки.

Таким образом, данное приложение – это отличный таск-менеджер с органайзером для малого бизнеса и предпринимателей. В нем есть всё, что только может понадобиться даже самому взыскательному юзеру, а эргономичный интерфейс сервиса практически полностью раскрывает его огромный потенциал

1.2 Постановка задачи

Проанализировав прототипы приложений-планировщиков, я выделила основную задачу и функциональные требования проекта.

Основной задачей моего курсового проекта является разработка десктопного приложения, позволяющего пользователю добавлять задачи, составлять расписание занятий, хранить информацию в централизованной базе данных, а также возможность составления статистики выполнения задач. Интерфейс должен быть простым и удобным для использования.

Функциональные требования описаны в главе 2.

2. Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

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).

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.

База данных была разработана в Microsoft SQL Server.

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 посредством специального API. СУБД 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 бд, 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.

2.2 Описание разрабатываемой функциональности приложения

Программное средство предоставляет пользователю следующие функциональные возможности:

* идентификация пользователей;
* создание запланированных задач;
* просмотр уже выполненных задач;
* составление расписания занятий;
* хранение и отображение прогресса по предметам;
* хранение личной информации в базе данных;
* обеспечение обратной связи с администратором;

Дополнительные функции для администратора:

* просмотр всех пользователей;
* редактирование личной информации пользователей;
* возможность удалять пользователей из базы данных.

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

2.3 Спецификация функциональных требований

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно о базе данных описано в главе 3.

В программном средстве «StudTasks» при запуске необходимо реализовать регистрацию и авторизацию пользователей для дальнейшего использования приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для того, чтобы зарегистрироваться, необходимо ввести имя, фамилию пользователя, номер студенческого билета, телефон, email, логин и пароль. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных, а пользователь получает доступ к приложению.

В верхней части окна находятся настройки приложения: смена языка интерфейса и смена темы приложения.

Основной функцией ПС является составление расписания и добавление задач. Пользователь может просмотреть имеющиеся задачи, а также добавить новые, введя текст в окно для ввода и нажав кнопку «Save» в соответствующей вкладке окна. При добавлении задачи пользователь может установить предмет, для которого устанавливается задача, добавить дедлайн, выбрав дату в появившемся календаре, установить уровень важности, выбрав цифру в выпадающем списке, выполнить задачу или удалить ее.

Добавленные задачи можно фильтровать по определённому предмету и важности. Также возможен отдельный просмотр предстоящих задач и уже выполненных.

Необходимо реализовать возможность просмотра всех задач пользователя.

Кнопка для выхода из приложения должна возвращать пользователя к окну авторизации.

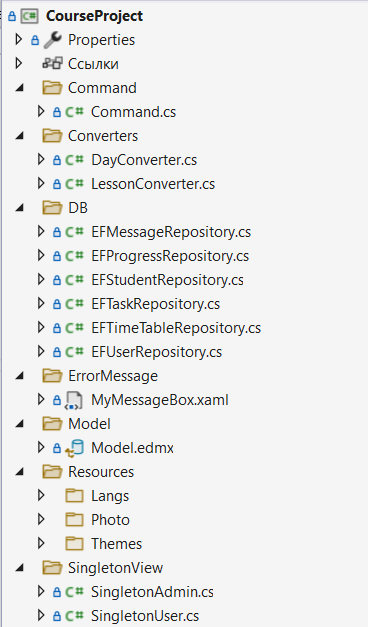
Таким образом, в ходе работы над этим разделом были сформулированы основные функциональные треования для проектирования программного средства.

3. Проектирование программного средства

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

3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства

Структура проекта представлена на рисунке 3.1.1.



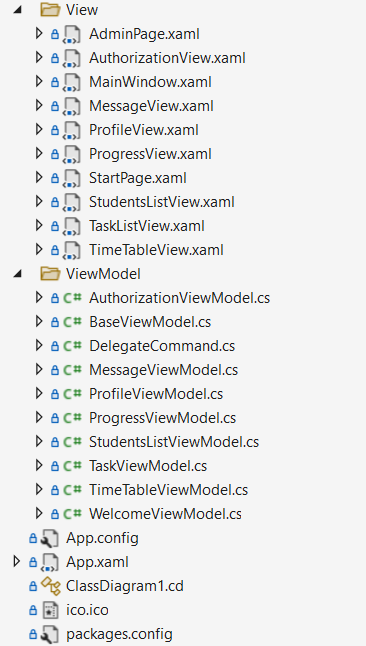


Рисунок 3.1.1 – Структура проекта

Описание приведенной структуры проекта представлено в таблице 3.1.1.

Таблица 3.1.1 – Описание структуры проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Properties | Свойства проекта, содержит информацию о сборке используемых ресурсах и настройках. |
| DB | Репозитории, которые являются промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. |
|  | |
| Продолжение таблицы 3.1.1 | |
| Model | Подпространство имён, содержащее описание сущностей для работы с БД. |
| Resources | Графические ресурсы приложения, а также словари ресурсов, в которых хранятся стили приложения, т.е. темы, и файлы для локализации приложения. |
| Converters | Конвертеры представлений и типов данных для отображения их в ListBox. |
| View | Представления, которые определяют визуальный интерфейс, через который пользователь взаимодействует с приложением. Т.е. это код xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов. |
| ViewModel | Содержит логику по получению данных из модели, которые затем передаются в представление. |
| App.xaml | Основная задача данного файла состоит в определении ресурсов, общих для приложения. |
| app.manifest | Файл манифеста сборки, определяющий её конфигурацию. |
| App.config | Файл с параметрами проекта. |

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

3.2 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 3.2.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

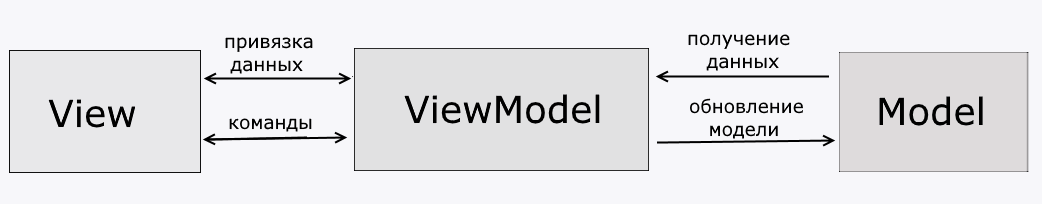


Рисунок 3.2.1 – Структура шаблона MVVM

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.

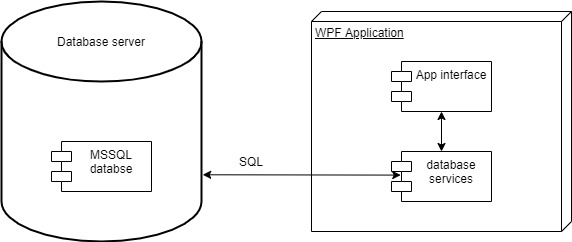


Рисунок 3.2.2 – Диаграмма размещения

3.3 Схема работы приложения

На рисунке 3.3 представлена схема навигации по страницам приложения.

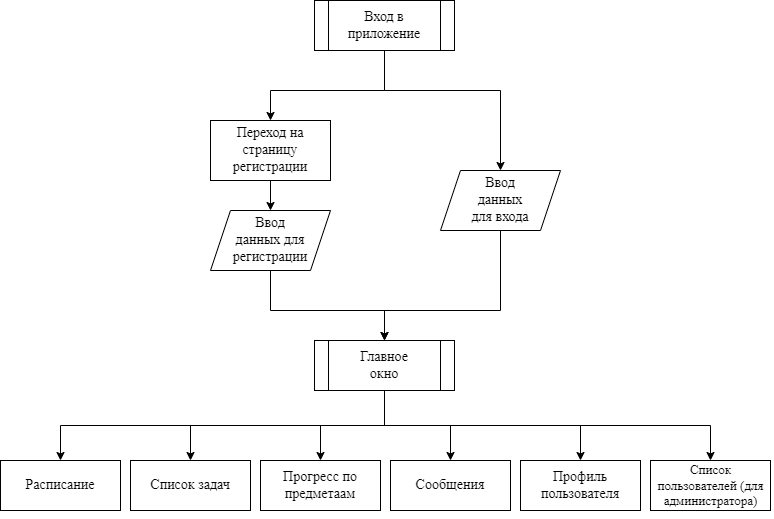


Рисунок 3.3 – Схема навигации по страницам приложения

При запуске приложения появляется окно авторизации, которое требует ввода логина и пароля для входа в приложение.

Если у вас нет учётной записи, то её можно создать, нажав «Sign Up», после чего вы будете направлены в форму регистрации, где необходимо заполнить соответствующие поля. В случае неверного ввода данных выведется соответствующее сообщение.

После успешной регистрации, а в дальнейшем и авторизации, запускается главное окно приложения, где происходит работа с отдельными страницами.

**3.4 Проектирование логики сценариев использования**

При проектировании приложения были рассмотрены различные сценарии использования. В графической части представлена диаграмма последовательности добавления задачи.

Из диаграммы видно, что, нажав на кнопку «New Task», у пользователя появится возможность добавить задачу. Для выбора предмета приложение отправит запрос в базу данных на получение информации обо всех заполненных в расписании предметах. Далее, когда база данных вернет информацию о них, она будет показана пользователю в ComboBox. Заполнив все оставшиеся поля, появится возможность сохранить задачу, нажав на кнопку «Save». После этого, приложение отправит запрос на добавление этого таска в соответствующую таблицу. Когда задание будет сохранено, оно отобразиться в левой части приложения.

3.5 Модель базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных;

Для реализации поставленной в курсовом проектировании задачи, была создана база данных StudTasks. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server 2019.

Созданная база данных содержит в себе 6 таблиц: User, Student, TimeTable, Task, Progress, Message. Они представлены ниже вместе со своими структурами.

На рисунке 3.5.1 представлена структура таблицы User.



Рисунок 3.5.1 – Структура таблицы User

В данной таблице хранится логин и пароль пользователя.

Всю информацию о студенте можно посмотреть в таблице Student, содержащая ФИО и контактные данные студента. Структура данной таблицы представлена на рисунке 3.5.2.

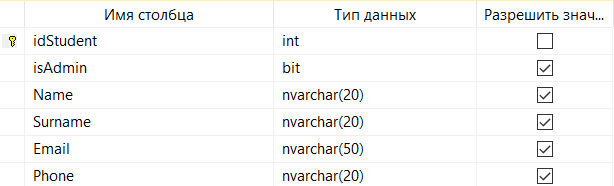


Рисунок 3.5.2 – Структура таблицы Student

Для хранения расписаний для каждой из групп была создана база данных TimeTable, представленная на рисунке 3.5.3.

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

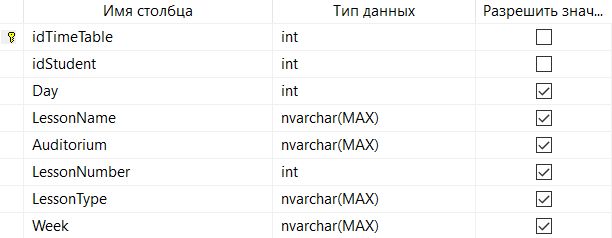


Рисунок 3.5.3 – Структура таблицы TimeTable

Прогресс по предметам хранится в базе данных Progress, структура которой представлена на рисунке 3.5.4.

В таблице содержится количество необходимых и выполненных задач по определенному предмету, название предмета только из расписания, прогресс по предмету.



Рисунок 3.5.4 – Структура таблицы Progress

В базе данных Task содержится информация о всех выполненных и невыполненных пользователем задачах. Структура базы данных представлена на рисунке 3.5.5.

Здесь хранится название заметки, ее содержимое, название предмета, к которому она относится, дата, когда необходимо выполнить, булевое значение, показывающее выполнение или не выполнение определенных задач, а также ссылку на идентификатор студента.

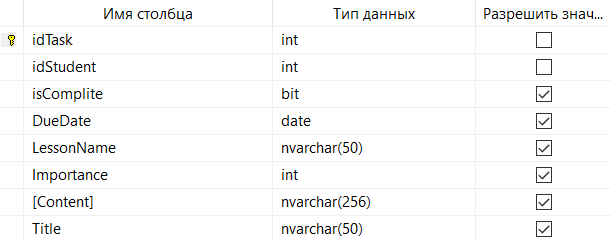


Рисунок 3.5.5 – Структура таблицы Task

Содержимое и дата создания отправленных администратором сообщений хранится в базе данных Message. Структура этой базы данных представлена на рисунке 3.5.6.

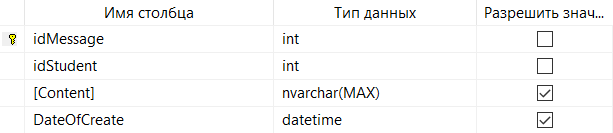


Рисунок 3.5.6 – Структура таблицы Message

Взаимосвязь всех таблиц базы данных представлена в графической части.

Для таблиц Task, Progress, TimeTable, Message внешним ключом является столбец idStudent, который соответствует первичному ключу таблицы Student.

Таблицы Task, Progress, TimeTable, Message соединены связями «один ко многим», т. е. несколько строк из дочерней таблицы зависят от одной строки в родительской таблице.

Таблицы Student и User соединены связью «один к одному», т. е. объекту одной таблицы можно сопоставить только один объект другой таблицы.

4. Создание (реализация) программного средства

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

4.1 Выполняемые функции

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 «StudTasks» представлена в графической части.

Рассмотрим реализацию команды для работы пользователя с расписанием, представленную в приложении Б.

Составление расписания происходит следующим образом: студент выбирает нужный день недели и время, далее в верхней панели заполняет нужную информацию и сохраняет её, нажав кнопку «Save». После этого данные сохраняются в базе данных, а студент в любое время может вернуться и изменить информацию.

4.2 Реализация общей структуры проекта

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

В Model находится структура модели базы данных приложения. При проектировании базы данных данного курсового проекта для взаимодействия с базой данных используется Entity Framework.

Entity Framework 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.

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.

Нередко все сущности (модели) наследуют 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() как показано ниже:

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

Листинг 4.2.1 – OnPropertyChanged

Для того, чтобы начать отслеживание свойства, необходимо при его установке вызывать метод OnPropertyChanged(“Имя свойства”) как показано ниже:

public int CountProgress

{

get { return countProgress; }

set

{

countProgress = value;

OnPropertyChanged("CountProgress");

}

}

Листинг 4.2.2 – Применение OnPropertyChanged

Для удобной работы с данными, используется паттерн «Репозиторий». На рисунке 4.2.1 показана его структура.

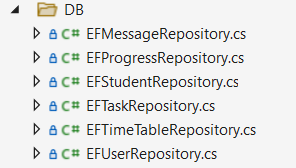


Рисунок 4.2.1 – Паттерн “Репозиторий”

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. Класс репозитория Task представлен в приложении А.

По такому же принципу и строятся и остальные репозитории.

На рисунке 4.2.2 показана структура пакета Themes, в котором находятся файлы стилей – LightTheme.xaml и DarkTheme.xaml. Эти файлы написаны на языке разметки xaml.

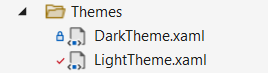


Рисунок 4.2.2 – Структура пакета Themes

В проекте используется паттерн MVVM, а это значит, что должны присутствовать ViewModels, которые связывают View и Model. Во ViewModel также должна информация обрабатываться и выводится/записываться в нужном виде. Ниже представлена структура пакета ViewModel (рисунок 4.2.3).

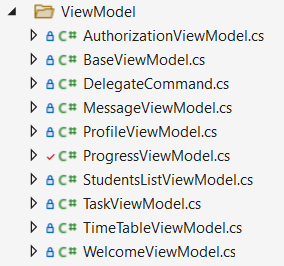


Рисунок 4.2.3 – Структура пакета ViewModel

На следующем рисунке 4.2.4 показана структура пакета View, в котором хранятся файлы представлений на языке разметки xaml, а также соответствующие им файлы на языке C#.

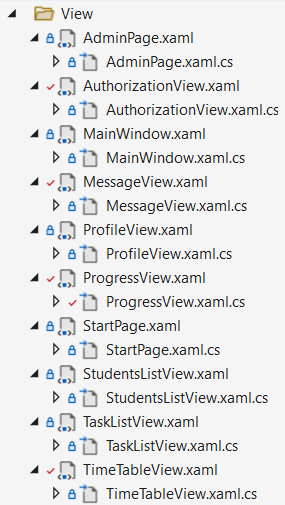


Рисунок 4.2.4 – Структура пакета View

В этом пакете находятся все «окна» и «страницы», которые мы видим, используя приложение.

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

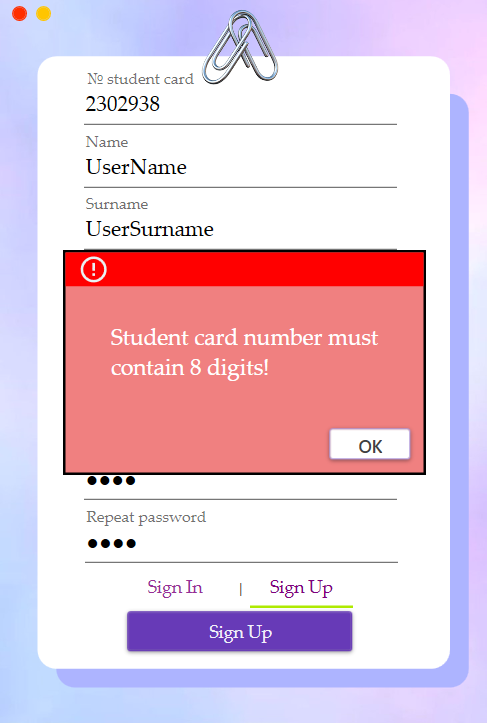
Таблица 4.2 – Описание структуры проекта

|  |  |
| --- | --- |
| Название | Краткое описание |
| AuthorizationView.xaml | Окно авторизации. Требует ввода логина и пароля. Если нет учетной записи, то здесь же можно зарегистрироваться. |
| MainWindow.xaml | Главное окно приложения. |
| MessageView.xaml | Окно объявлений. Здесь администраторы могут оставить сообщения, а пользователи их просмотреть. |
| ProfileView.xaml | Окно для просмотра информации о вошедшем в приложение пользователе. |
| ProgressView.xaml | Окно прогресса. Здесь пользователь может указать количество выполненных/всех лабораторных работ по какому-либо предмету отслеживая при этом прогресс. |
| StudentsListView.xaml | Окно только для администраторов. Здесь можно просмотреть список всех зарегистрированных пользователей, при необходимости удалить или редактировать с соответствующими ограничениями. |
| TaskListView.xaml | Окно задач. Здесь пользователь может добавить задачу по определенному предмету, отсортировать их, поставить отметку о выполнении и просмотреть список всех выполненных задач. При необходимости удалить. |
| TimeTableView.xaml | Окно с расписанием. Пользователь может заполнять расписание и его корректность будет зависеть только от заполнявшего. Имеется возможность заполнить разное расписание на две недели. |

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.

На рисунке 5.1 представлен результат обработки исключения при регистрации пользователя. Поле “Номер студенческого билета” уникально, поэтому в базе данных не могут храниться пользователи с одинаковым номером студенческого билета, а также значение этого поля должно состоять из 8 цифр.



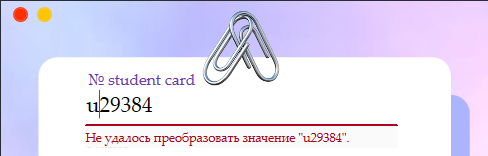


Рисунок 5.1 – Некорректный номер студенческого билета

Также не может быть несколько пользователей с одинаковым полем “Логин” (рисунок 5.2).

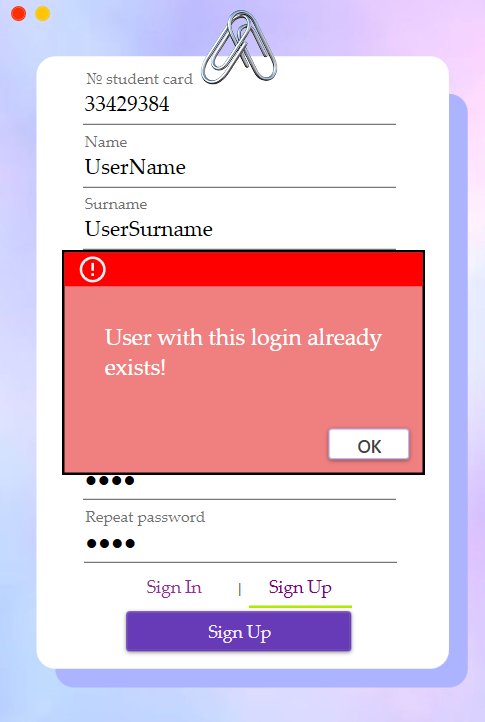


Рисунок 5.2 – Некорректный логин

В программе присутствует валидация на ввод номера телефона, а также на электронный адрес пользователя. Это предусмотрено с помощью регулярных выражений (рисунок 5.3).

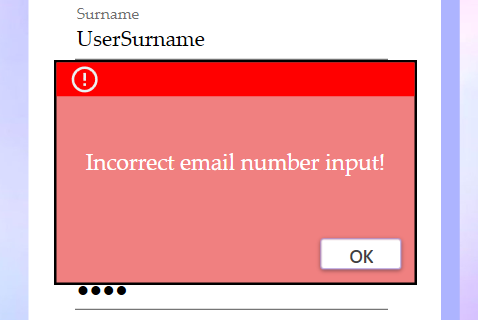


Рисунок 5.3 – Некорректный номер телефона и email

Также при регистрации необходимо продублировать пароль. Они должны совпадать (рисунок 5.4).

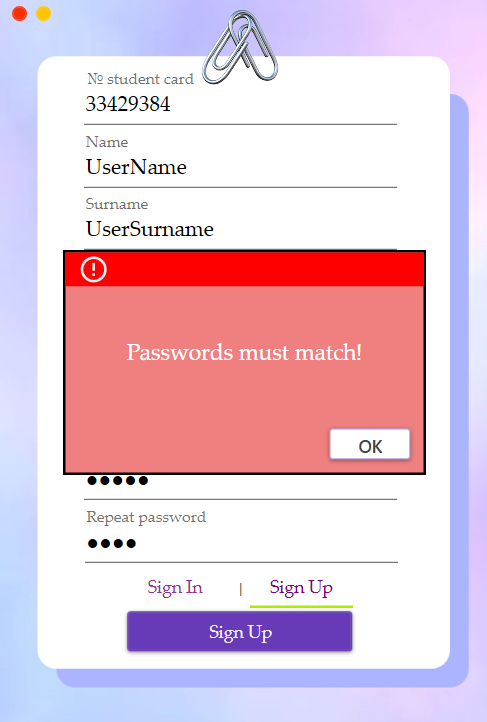


Рисунок 5.4 – Несовпадение паролей

На рисунке 5.5 представлен результат обработки исключения при входе в приложения не указывая логин и пароль.



Рисунок 5.5 – Данные не введены

Также возникают ошибки если данные неверно введены (рисунок 5.6).

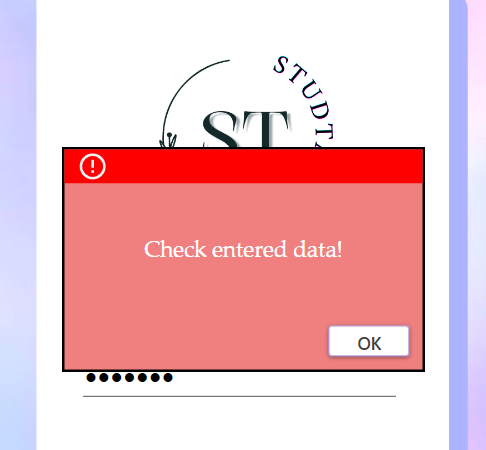


Рисунок 5.6 – Неверно введённые данные

На странице создания заметок и заданий присутствуют различные способы валидации и защиты от ошибок пользователя. Так, например, пользователь не может задавать дату раньше текущей. В СomboBox заносятся лишь те учебные предметы, которые пользователь внёс в расписание (рисунок 5.7).

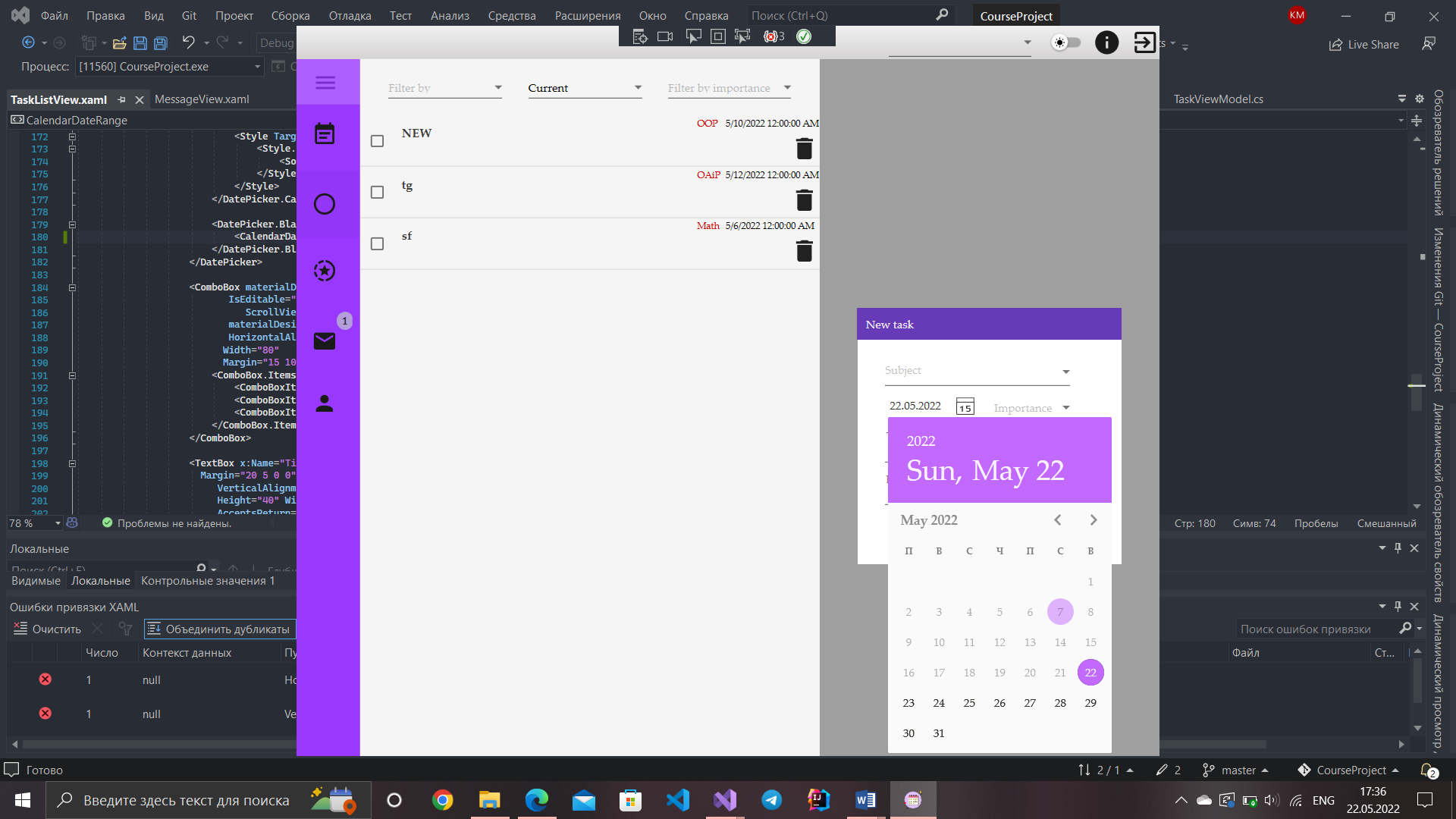


Рисунок 5.7 – Ограничение на выбор даты

При попытке создать задачу с незаполненными полями также возникает ошибка (рисунок 5.8).

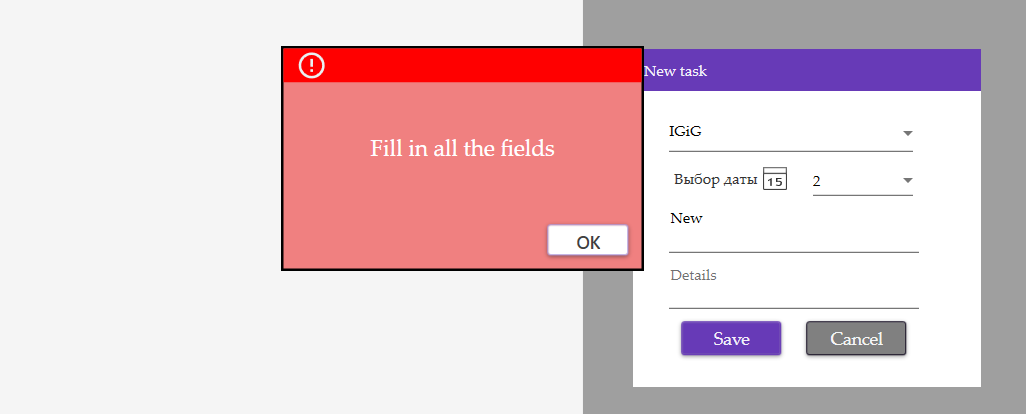


Рисунок 5.8 – Не все поля заполнены

На рисунке 5.9 представлена обработка ошибки при попытке студента удалить не свое сообщение.



Рисунок 5.9 – Попытка удалить чужое сообщение студентом

Также при добавлении в прогресс предмета, который уже отслеживается, появляется соответствующее сообщение об ошибке (рисунок 5.10)

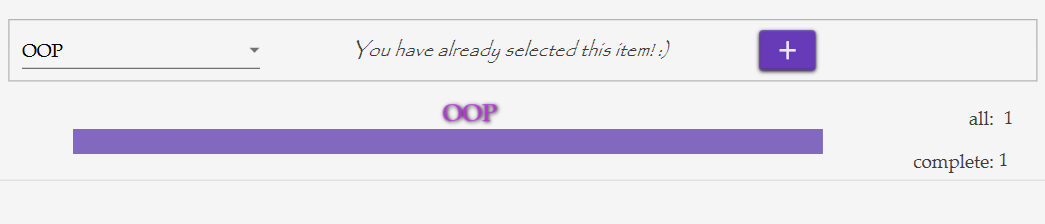


Рисунок 5.10 – Попытка добавить в прогресс существующий предмет

Если не заполнено расписание, то прогресс отслеживать нельзя (рисунок 5.11).

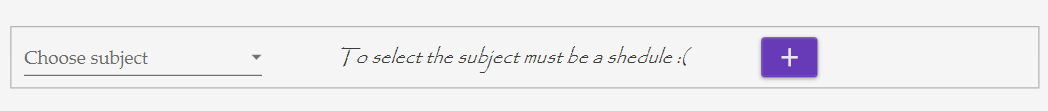


Рисунок 5.11 – Попытка добавить в прогресс несуществующий в расписании предмет

Таким образом, большинство возникающих ошибок в данном приложении были откорректированы. Однако, например, корректность расписания будет зависеть лишь от заполнявшего.

6. Методика использования программного средства

При запуске приложения открывается окно авторизации, представленное на рисунке 6.1.

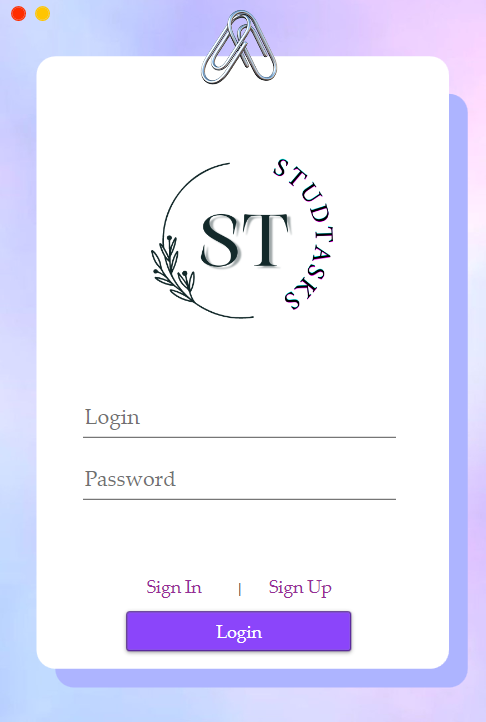


Рисунок 6.1 – Окно авторизации

Окно авторизации требует ввода логина и пароля для начала работы с приложением. Однако, если у вас нет учетной записи, то её можно создать, нажав «Sign Up», после чего вы будете направлены в форму регистрации, где необходимо заполнить соответствующие поля (рисунок 6.2). В случае неверного ввода данных, выведется соответствующее сообщение.

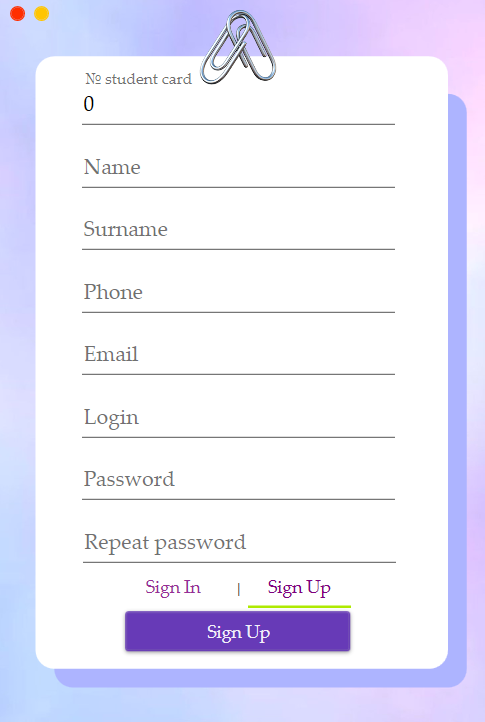


Рисунок 6.2 – Окно регистрации

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

После успешной регистрации необходимо вернуться в окно авторизации и ввести ваши данные, указанные при регистрации.

После ввода логина и пароля на форме авторизации будет осуществляться проверка правильности введенных данных и поиск пользователя с такими данными в базе данных.

В случае неверно введенных данных выведется сообщение об ошибке. В ином случае окно авторизации закроется и отроется главное окно, где уже можно непосредственно взаимодействовать с отдельными страницами. Главное окно администратора представлено на рисунке 6.3, пользователя – на рисунке 6.4.

При входе в приложение можно увидеть краткое руководство по использованию приложения. Стрелками указано на каких вкладках выполняются те или иные задачи.

Также приложение предусматривает выбор темы оформления: тёмная или светлая.

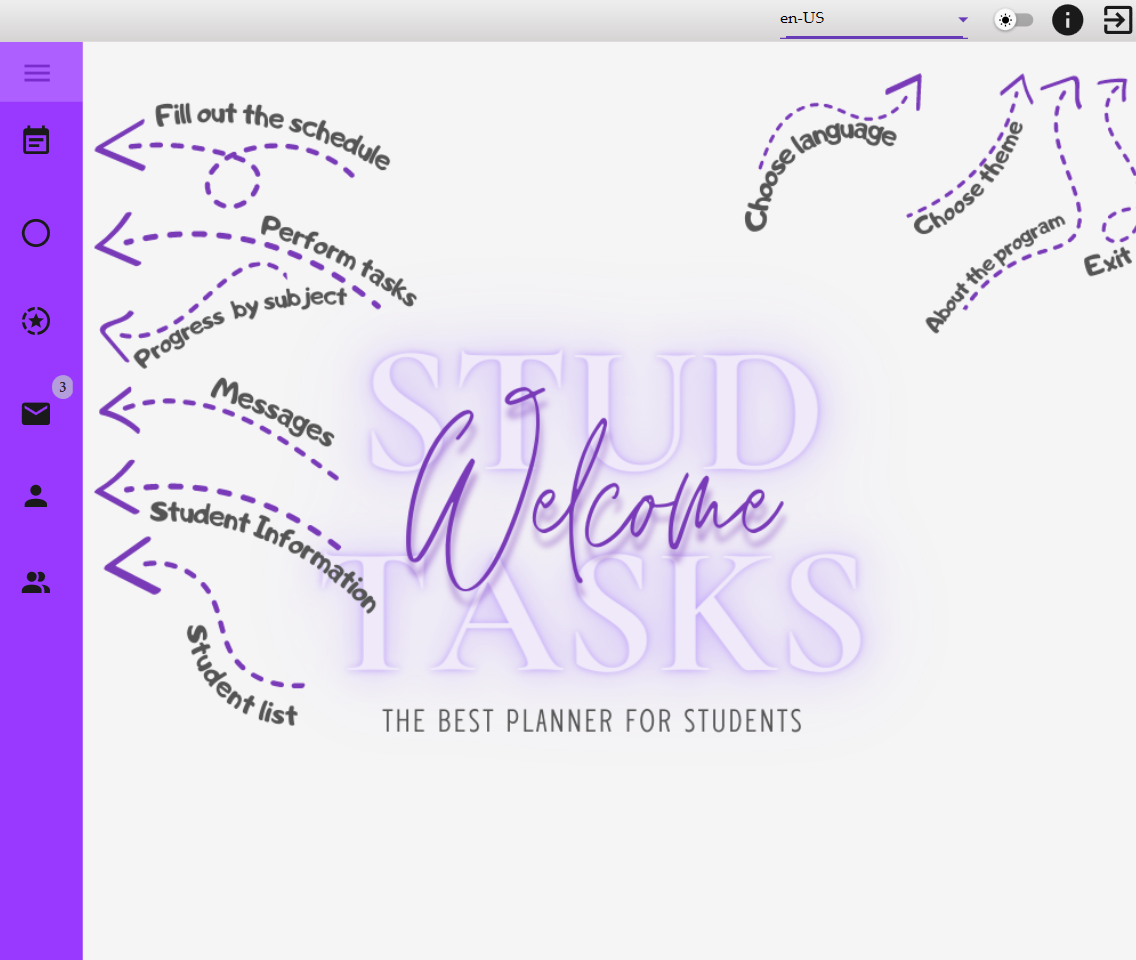


Рисунок 6.3 – Главное окно администратора

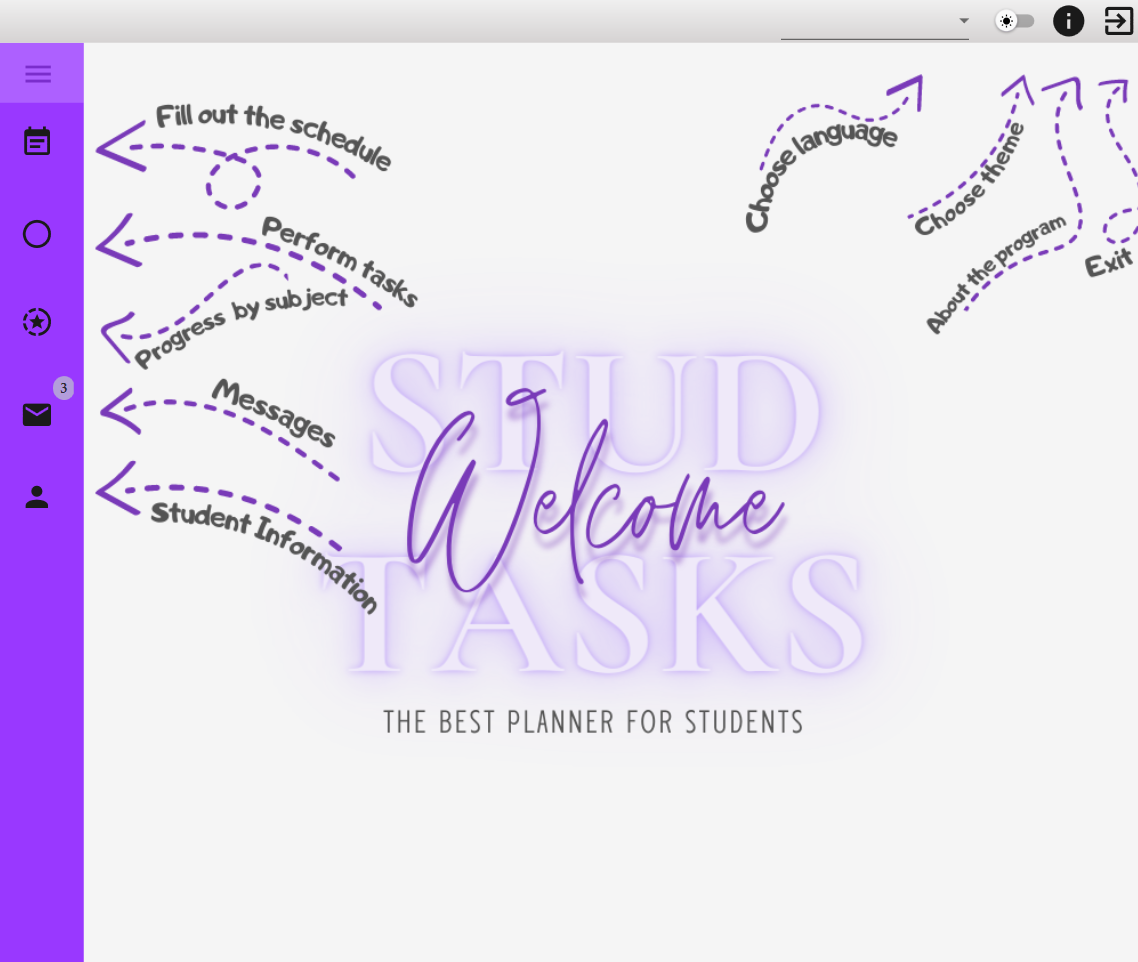


Рисунок 6.4 – Главное окно пользователя

Рассмотрим подробнее каждую из страниц.

Страница Tasks, на которой пользователь может добавить задачи по предметам, представлена на рисунке 6.5.

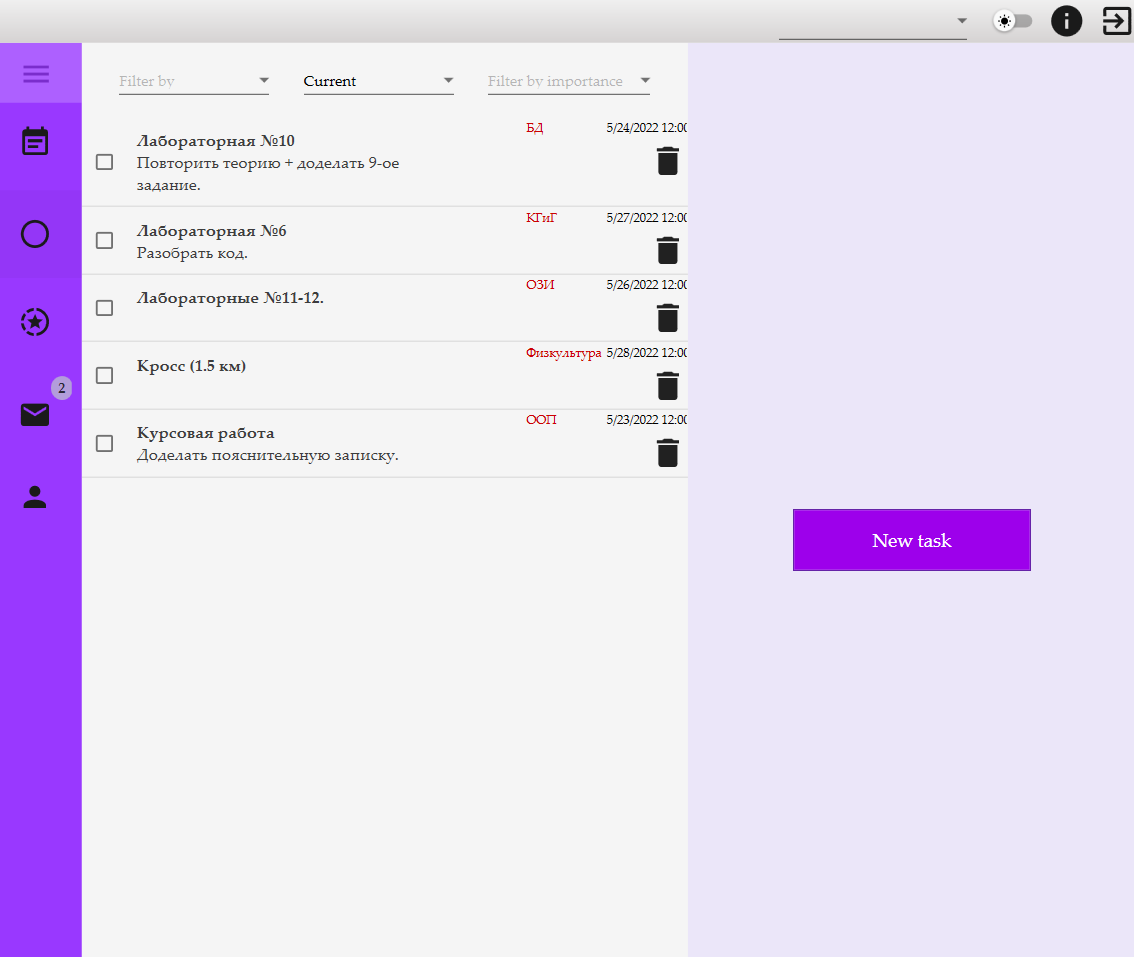


Рисунок 6.5 – Страница Tasks

Чтобы добавить новую задачу необходимо нажать на кнопку «New Task» в правой части страницы. После этого появится форма, на которой необходимо заполнить поля для новой задачи (рисунок 6.6).

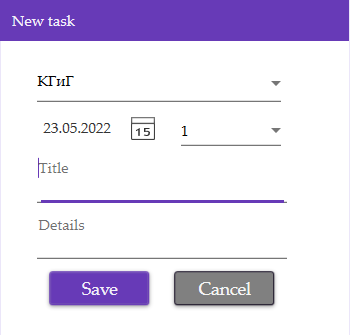


Рисунок 6.6 – Добавление новой задачи

После того, как все поля будут заполнены, необходимо нажать кнопку «Save» и задача появится в списке слева.

Если вы выполнили какую-либо задачу, необходимо поставить галочку, после чего задача перенесется в список выполненных задач. Чтобы их посмотреть, необходимо выбрать вкладку «Past».

Чтобы отсортировать задачи по определенному предмету, необходимо его выбрать из списка предметов, заполненных в расписании. На рисунке 6.7 представлен список всех предметов из заполненного ранее расписания и сортировка по предмету ООП. Здесь сразу же выводятся и выполненные по этому предмету задачи.

Чтобы просмотреть все выполненных и невыполненные задачи необходимо выбрать пункт «All».

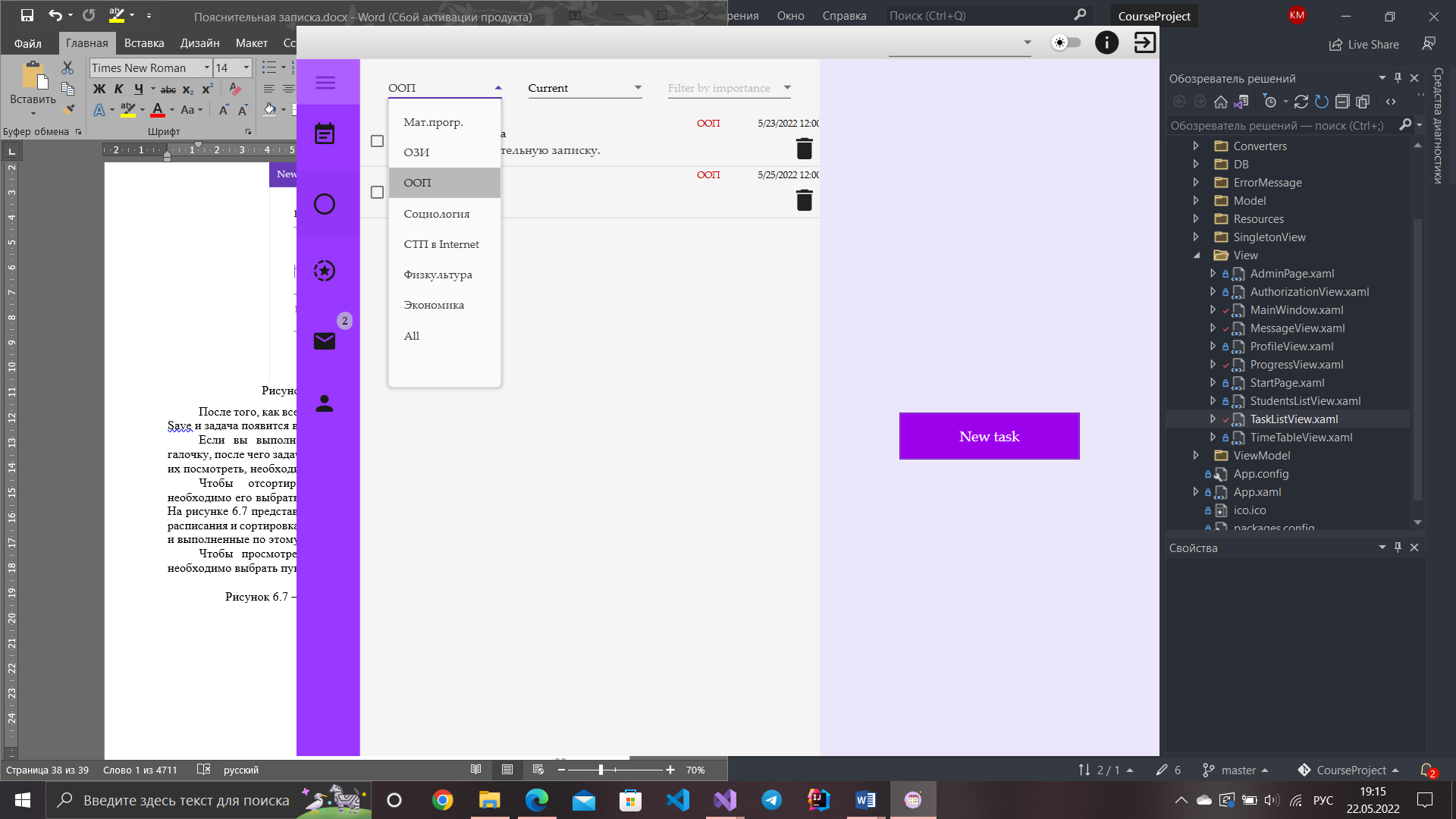


Рисунок 6.7 – Выбор предмета сортировки и сортировка по OOП

Следующая страница Progress отображена на рисунке 6.8.

Для отслеживания прогресса по предмету необходимо выбрать его из выпадающего списка, в котором находятся предметы из расписания. При добавлении задания по определённому предмету на странице Task, прогресс по данному предмету появляется автоматически. При удалении последнего задания по предмету, прогресс по данному предмету тоже удаляется.

Если расписание не заполнено, то прогресс отслеживать нельзя, выводится соответствующее сообщение.

При попытке добавить в прогресс уже отслеживаемый предмет также выводится соответствующее сообщение об ошибке.

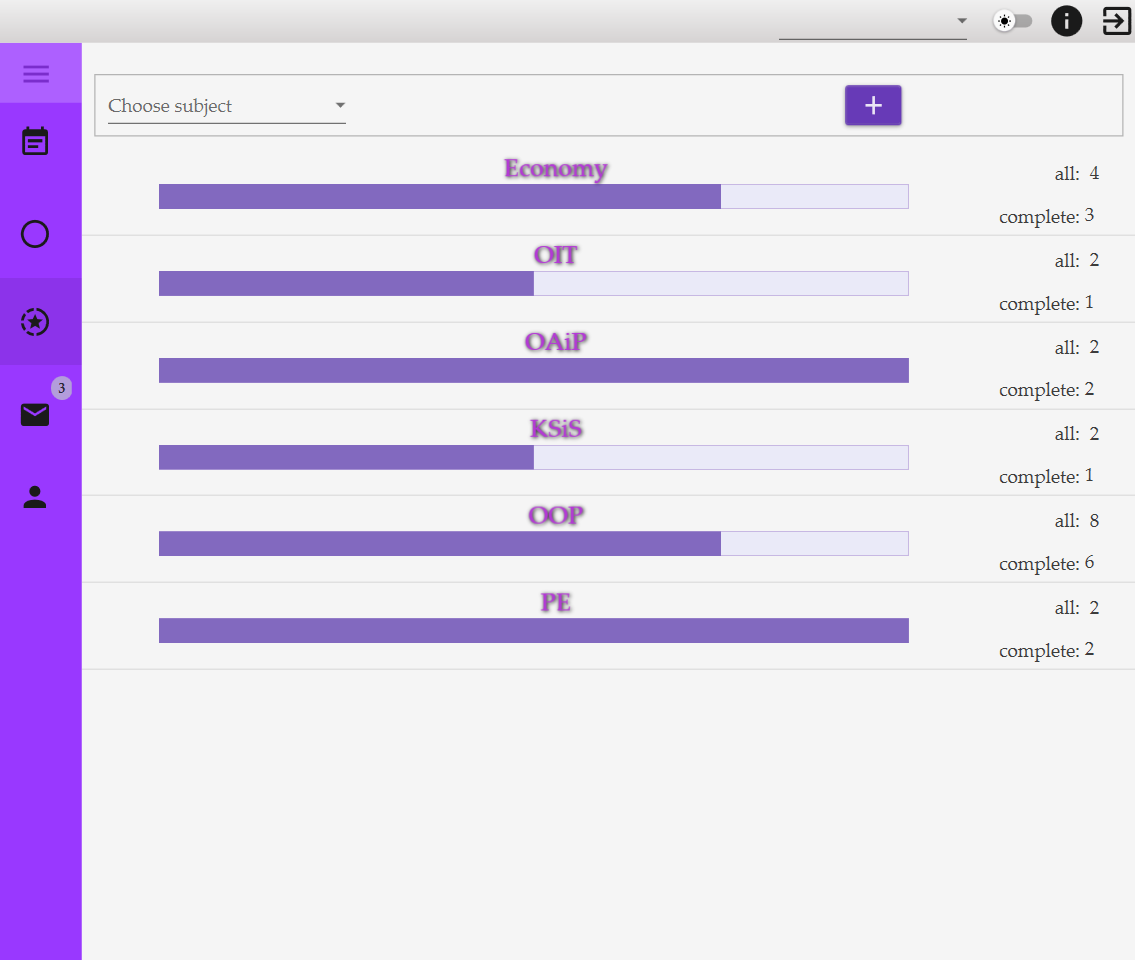


Рисунок 6.8 – Страница Progress

Прогресс отображается в процентном соотношении количества выполненных задач и необходимых. Полоса заполняется на это количество процентов.

Заполнить расписание можно на странице TimeTable, которая представлена на рисунке 6.10.

Изменение расписания показано ниже:



Рисунок 6.9 – Изменить расписание

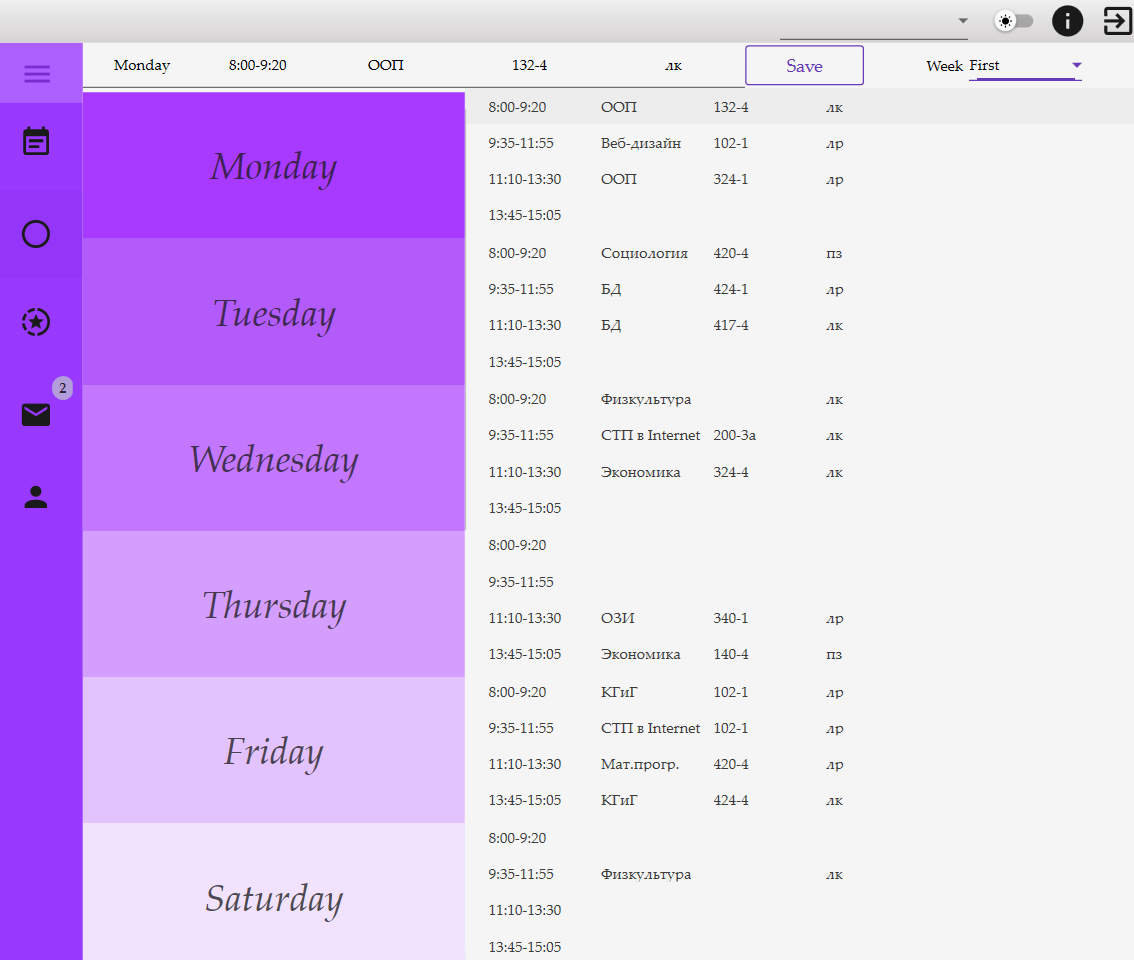


Рисунок 6.10 – Страница TimeTable

Для редактирования расписания необходимо выбрать строку с нужной парой и ввести необходимые данные в поля сверху. Данные сохраняются при нажатии кнопки «Save».

У студента имеется возможность заполнять разные расписания для двух недель. Для переключения между первой и второй неделей используется ComboBox.

Следующая страница Message необходима для того, чтобы обратную связь между администратором и пользователями. Все сообщения отображаются и для других пользователей, и для администратора.

Внешний вид страницы изображен на рисунке 6.11. Также на вкладке отображается количество сообщений в чате.

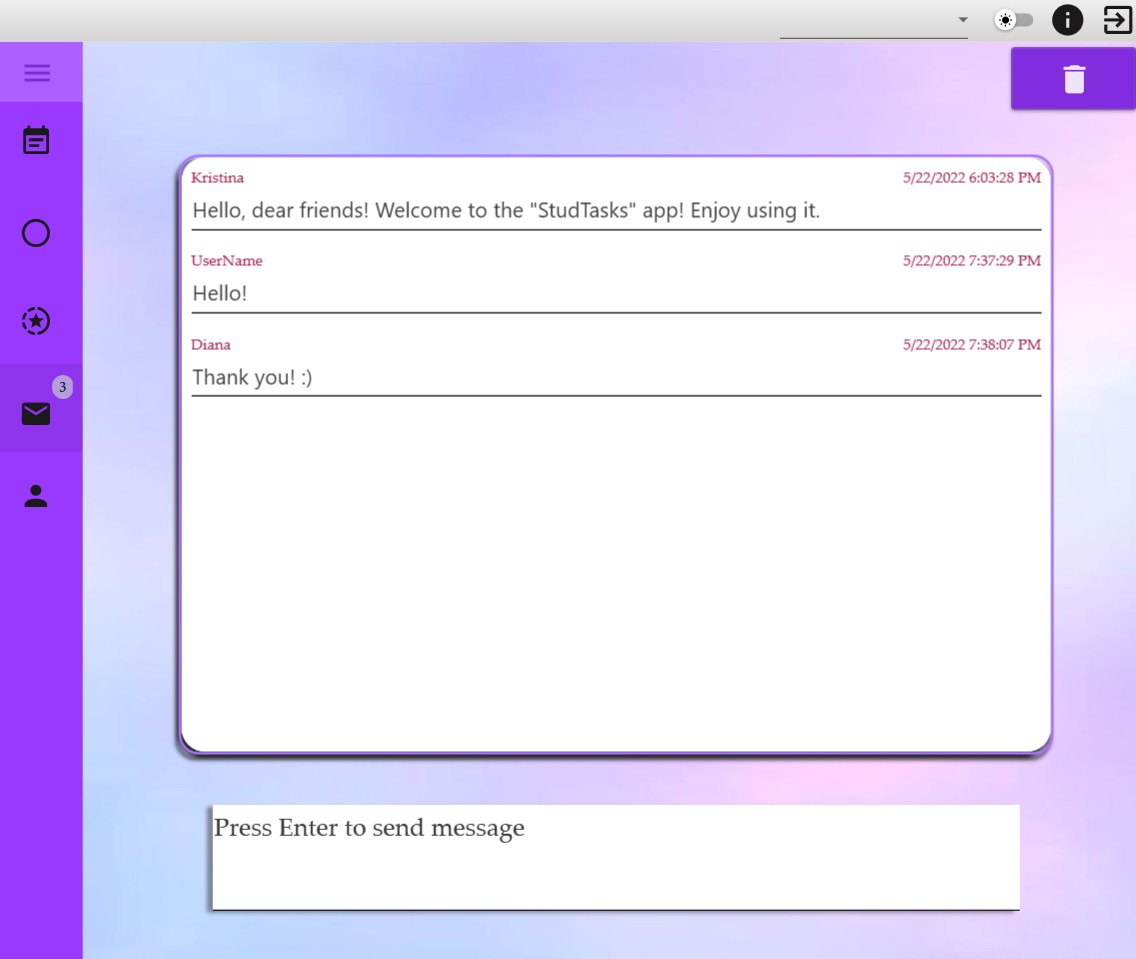


Рисунок 6.11 – Страница Message

Чтобы отправить сообщение, необходимо ввести его в текстовую области внизу окна и нажать клавишу Enter.

Для того, чтобы удалить сообщение вам необходимо выбрать его и нажать на кнопку удалить в правом верхнем углу страницы. Если вы не администратор и захотите удалить сообщение, которое отправили не вы, то выведется соответствующее сообщение об ошибке.

Список группы доступен только администраторам (рисунок 6.12).

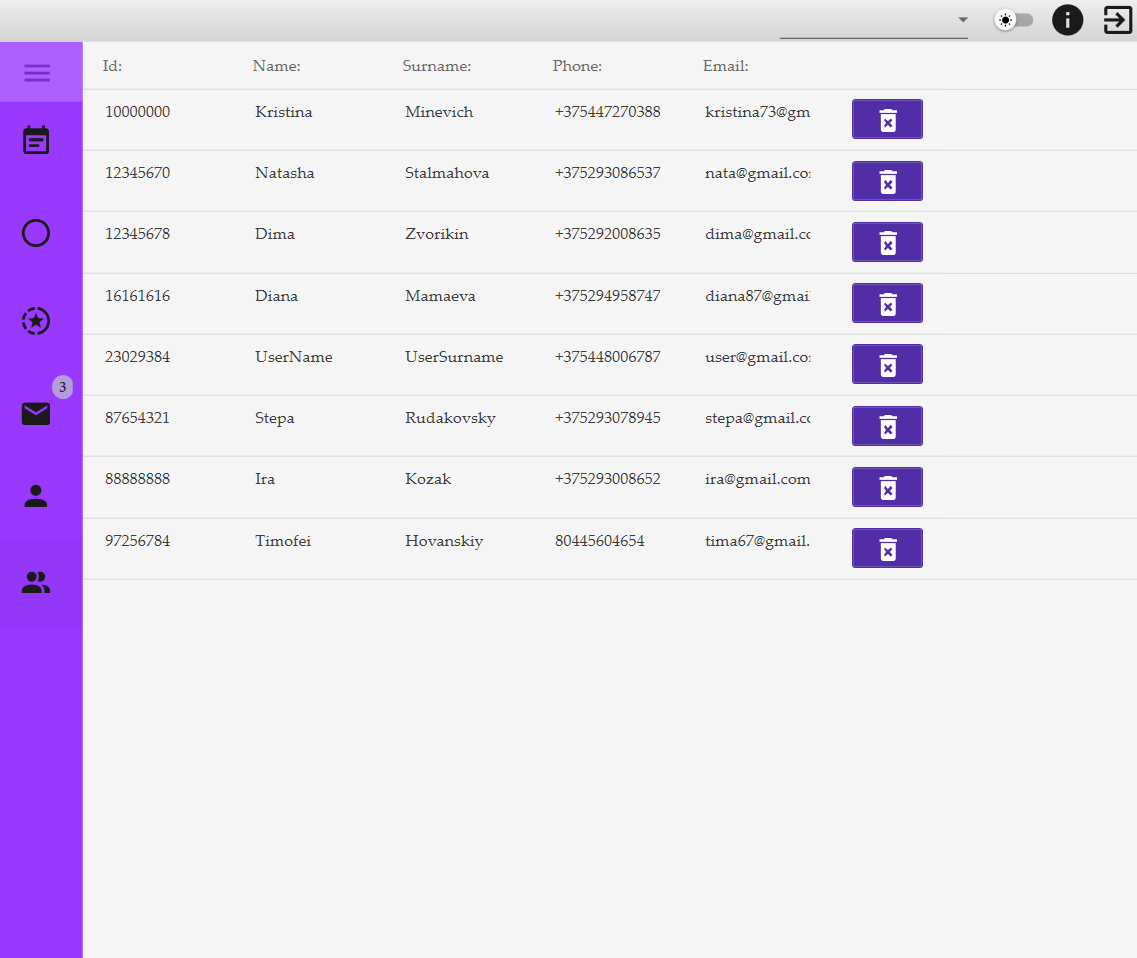


Рисунок 6.12 – Страница Students List

На этой странице администратор видит всех зарегистрированных пользователей, может удалить студента, нажав на кнопку в колонке с удаляемым пользователем. Вместе с пользователем удаляются его расписание, все задачи и отслеживаемый им прогресс. Дополнительно администратор может изменить информацию о студенте.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было разработано программное средство “StudTasks”, которое помогает студентам организовывать и планировать свои дела.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

регистрация пользователей;

авторизация пользователей;

создание запланированных задач;

просмотр уже выполненных задач;

фильтрация задач по определённому предмету и важности;

составление расписания занятий;

хранение и отображение прогресса по предметам;

хранение личной информации в базе данных;

отправка и просмотр сообщений от администратора и других студентов.

Дополнительные функции для администратора:

просмотр всех пользователей;

возможность редактирования информации о пользователях;

возможность просматривать, оставлять и удалять сообщения в общем чате.

Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке. Кроме того, оно имеет удобный и понятный интерфейс.

Проектирование осуществлялось по паттерну проектирования MVVM.

Для удобной работы с данными, был реализован паттерн Repository.

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

СПИСОК ЛИТЕРАТУРЫ

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/>

3) Эндрю Троелсон «Язык программирования C# 6.0 и платформа .NET 4.6»

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

class EFTaskRepository

{

private StudTasksEntities context;

public EFTaskRepository()

{

context = new StudTasksEntities(); // Создаем контекст для работы с БД

}

public IEnumerable<Model.Task> getTasks() // Получение всех задач

{

context.Task.Distinct().OrderByDescending(p => p.DueDate).Load(); // Загружаем все задачи в контекст

return context.Task; // Возвращаем все задачи

}

public void addTask(Model.Task task) // добавление задачи

{

context.Task.Add(task);

context.SaveChanges();

}

public void SaveTask() // Сохранение изменений в БД

{

context.SaveChanges();

}

public void Update(Model.Task task)

{

context.Entry(task).State = EntityState.Modified; // Обновляем задачу

context.SaveChanges();

}

public void UpdateAll()

{

context.SaveChanges();

}

public void Remove(Model.Task task)

{

context.Task.Remove(task);

context.SaveChanges();

}

public Model.Task GetTaskById(Model.Task task) // Получение задачи по ее id

{

return context.Task.FirstOrDefault(p => p.idTask == task.idTask);

}

public IEnumerable<Model.Task> GetTasksById(Student student)

{

return context.Task.Where(p => p.idStudent == student.idStudent);

}

Продолжение листинга

public void ChangeComplite(Model.Task task) // Изменение статуса задачи

{

GetTaskById(task).isComplite = task.isComplite;

context.SaveChanges();

}

public void RemoveById(Model.Task task)

{

context.Task.Remove(GetTaskById(task));

context.SaveChanges();

}

public void RemoveByStudId(Student student) // Удаление задач по id студента

{

foreach (Model.Task task in getTasks())

{

if (task.idStudent == student.idStudent)

{

context.Task.Remove(task);

}

}

context.SaveChanges();

}

public void OrderTasks(Student student, string subject) // Сортировка задач по предмету

{

context.Task.Where(p => p.idStudent == student.idStudent).OrderBy(p => p.LessonName == subject).Load();

}

public IEnumerable<Model.Task> getEnum(Student student, string subject

{

return context.Task.Where(p => p.idStudent == student.idStudent && p.LessonName == subject);

}

public IEnumerable<Model.Task> getEnumByImportance(Student student, int importance)

{

return context.Task.Where(p => p.idStudent == student.idStudent && p.Importance == importance);

}

}

Листинг – Класс EFTaskRepository

Приложение Б

public ICommand UpdateLineCommand

{

get

{

return updateLine ?? (updateLine = new DelegateCommand(

obj =>

{

try

{

TimeTable updated;

using (StudTasksEntities db = new StudTasksEntities())

{

updated = db.TimeTable.FirstOrDefault(x =>

x.idTimeTable == selectedTimeTable.idTimeTable);

updated.Auditorium = SelectedTimeTable.Auditorium;

updated.LessonName = SelectedTimeTable.LessonName;

updated.LessonType = SelectedTimeTable.LessonType;

if (updated.LessonType.Length>2)

{

MyMessageBox.Show("The lesson type must consist of two letters", MessageBoxButton.OK);

return;

}

db.SaveChanges();

}

}

catch (Exception e)

{

MyMessageBox.Show(e.Message, MessageBoxButton.OK);

}

}

));

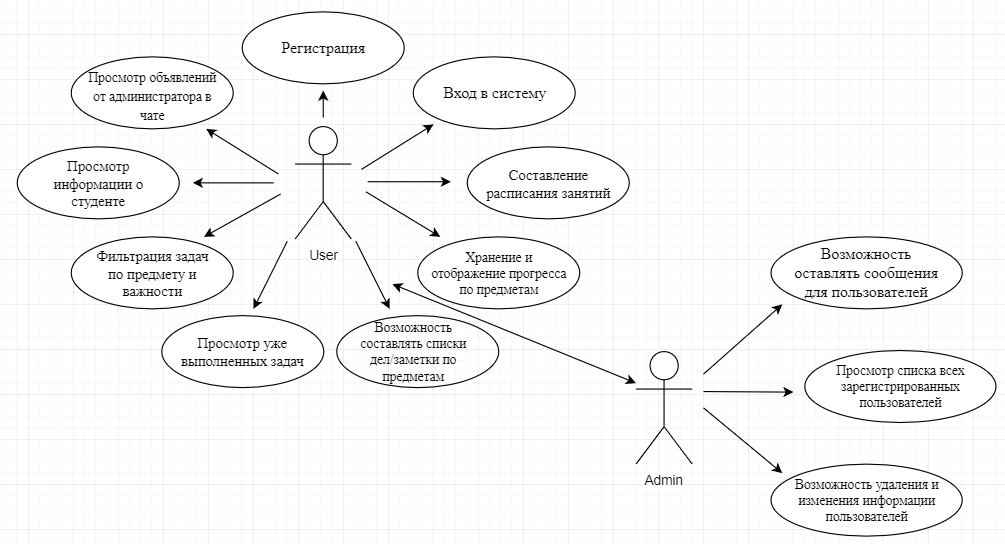
}

}

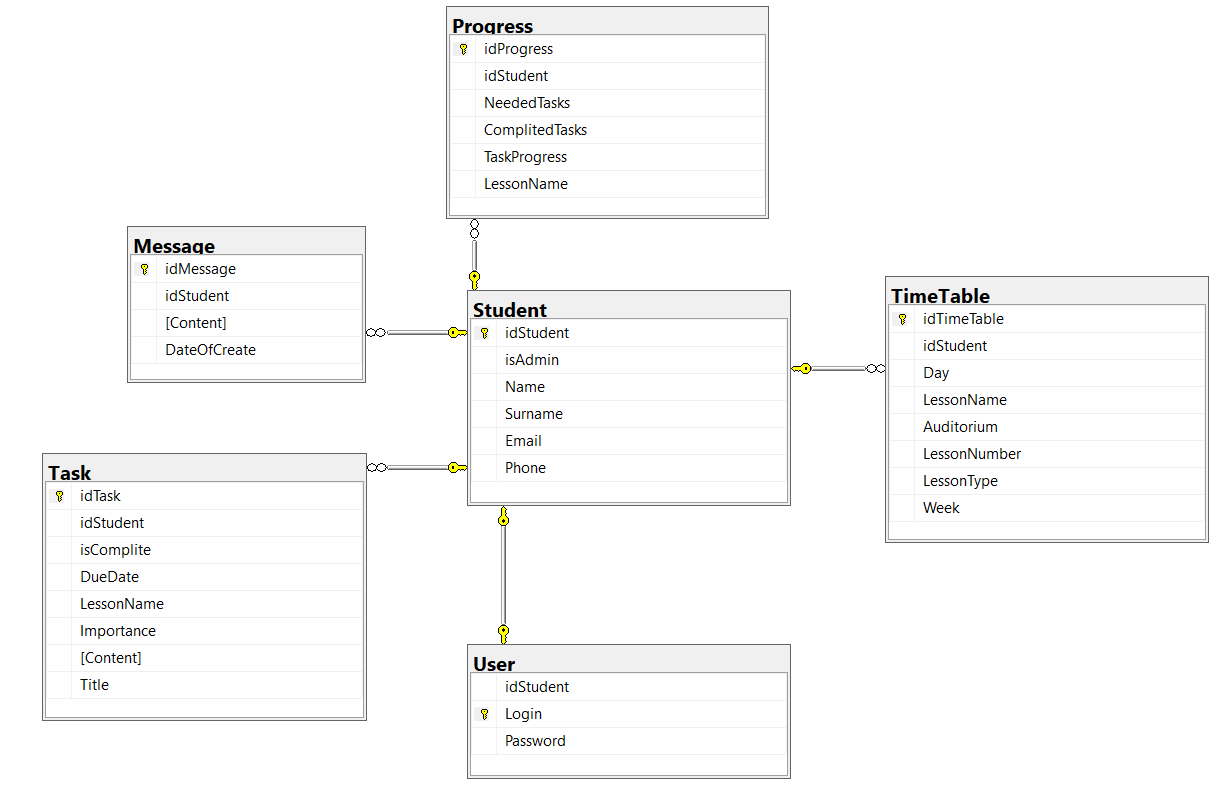
Листинг – Команда добавления расписания

Графический материал

Диаграмма вариантов использования



Логическая схема базы данных



Диаграммы классов

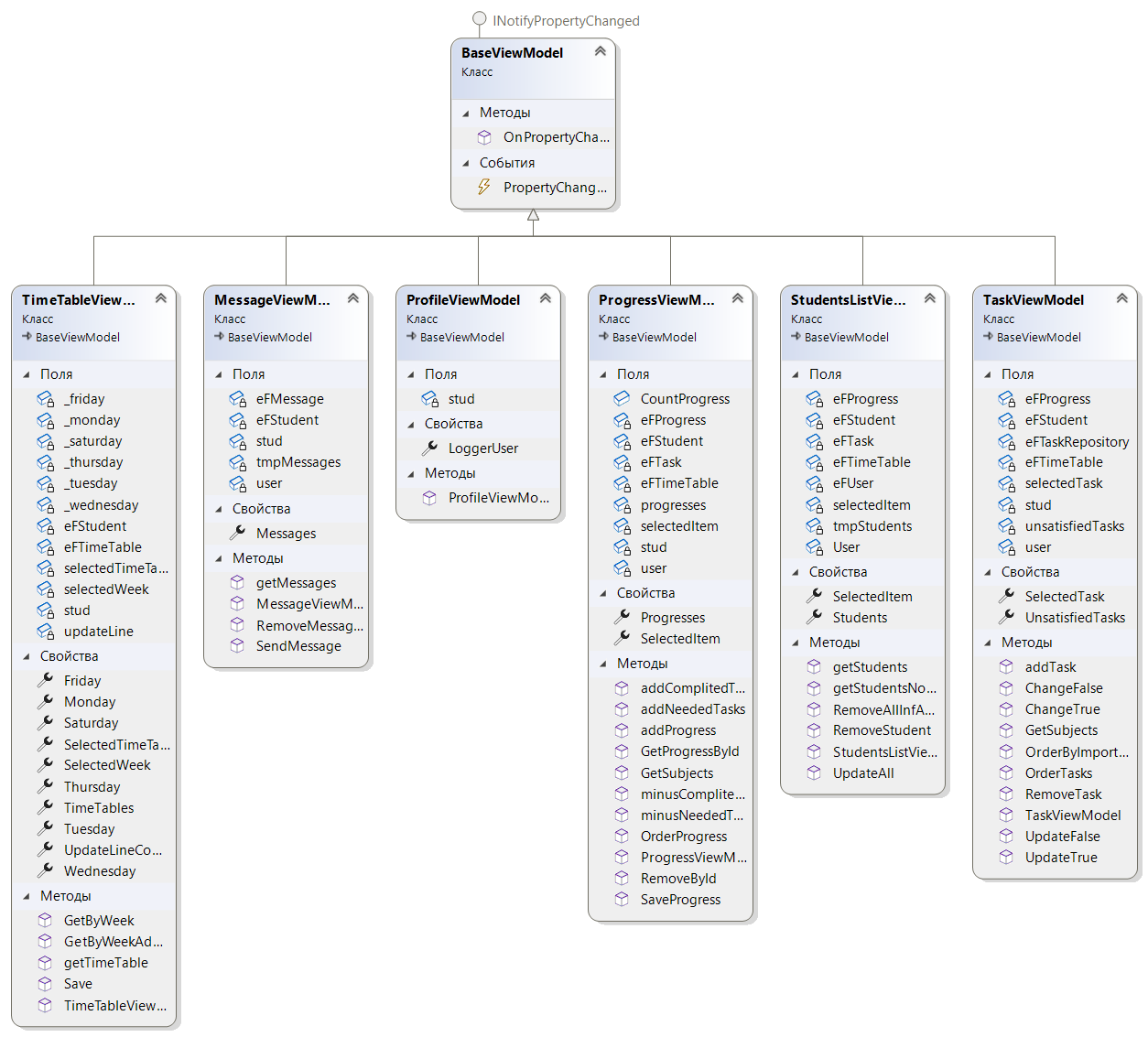


Диаграмма последовательности

