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 «Программное обеспечение информационных технологий»(программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема Программное средство «TaskWave»

Исполнитель

Студент(ка) 2 курса группы 5 Кравченко Алексей Дмитриевич

(Ф.И.О.)

Руководитель работы преп.-стажёр Север А.С.

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

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированные технологии программирования и стандарты проектирования"

|  |  |
| --- | --- |
| Специальность: 1-40 01 01 Программное обеспечение информационных технологий  Студент: Кравченко Алексей Дмитриевич | Группа: 5 |
| **Тема: Программное средство “TaskWave”** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2023 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора сервиса:
  + Поддерживать работу c базой данных;
  + Контролировать конфиденциальность личных данных;
  + Обновлять функционал приложения;
  + Контролировать регистрацию пользователей.
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Создавать одиночные проекты и задачи;
  + Управлять проектами;
  + Контролировать выполнение проектов и задач;
  + Настраивать личный кабинет;
  + Просматривать статистику выполнения проектов и задач.

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2023 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2023 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2023 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 02.04.2023 |  |
| 5 | Кодирование программного средства | 23.04.2023 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2023 |  |
| 7 | Оформление пояснительной записки | 07.05.2023 |  |
| 8 | Защита проекта | 20.05.2023 |  |

**5. Дата выдачи задания**  12.02.2023

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Север А.С.

(подпись)

Задание принял к исполнению 12.02.2023 *Кравченко А.Д.*

(дата и подпись студента)

**Содержание**

[ВВЕДЕНИЕ 6](#_Toc135313107)

[1 Аналитический обзор прототипов и литературных источников 7](#_Toc135313108)

[1.1 Обзор аналогичных решений 7](#_Toc135313109)

[1.2 Требования к проектируемому программному средству 9](#_Toc135313110)

[2 Анализ требований к программному средству и разработка требований 11](#_Toc135313111)

[2.1 Технические средства разработки 11](#_Toc135313112)

[2.1.1 Язык программирования C# и платформа .Net 11](#_Toc135313113)

[2.1.2 Entity Framework 12](#_Toc135313114)

[2.1.3 Windows Presentation Foundation (WPF) 13](#_Toc135313115)

[2.1.4 Microsoft SQL Server 13](#_Toc135313116)

[2.1.5 Figma 14](#_Toc135313117)

[2.2 Архитектура системы 15](#_Toc135313118)

[2.3 Диаграмма вариантов использования 17](#_Toc135313119)

[3 Проектирование программного средства 18](#_Toc135313120)

[3.1 Проектирования логической структуры базы данных 18](#_Toc135313121)

[3.2 Проектирования структуры окон 20](#_Toc135313122)

[3.3 Проектирование архитектуры приложения 23](#_Toc135313123)

[3.4 UML-диаграмма классов 24](#_Toc135313124)

[4 Реализация программного средства 27](#_Toc135313125)

[4.1 Основные классы и методы программного средства 27](#_Toc135313126)

[4.2 Классы для реализации паттерна MVVM 27](#_Toc135313127)

[4.3 Классы для реализации паттерна Command 29](#_Toc135313129)

[4.4 Классы для реализации взаимодействия с базой данных 31](#_Toc135313130)

[4.5 Методы для работы с файлами 32](#_Toc135313131)

[4.6 Методы для работы электронной почтой 33](#_Toc135313132)

[4.7 Методы для регистрации и авторизации пользователей 34](#_Toc135313133)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 35](#_Toc135313134)

[5.1 Тестировании регистрации и авторизации 35](#_Toc135313135)

[5.2 Тестирование заполнения формы для создания задачи 36](#_Toc135313136)

[6 Руководство по использованию 39](#_Toc135313137)

[ЗАКЛЮЧЕНИЕ 43](#_Toc135313138)

[Список использованных источников 44](#_Toc135313139)

[ПРИЛОЖЕНИЕ А 45](#_Toc135313140)

[ПРИЛОЖЕНИЕ Б 46](#_Toc135313141)

[ПРИЛОЖЕНИЕ В 49](#_Toc135313142)

[ПРИЛОЖЕНИЕ Г 62](#_Toc135313143)

[ПРИЛОЖЕНИЕ Д 64](#_Toc135313144)

[ПРИЛОЖЕНИЕ Е 67](#_Toc135313145)

[ПРИЛОЖЕНИЕ Ж 70](#_Toc135313146)

[ПРИЛОЖЕНИЕ З 82](#_Toc135313147)

# ВВЕДЕНИЕ

Тема курсового проекта "Программа для организации задач и управления проектами" имеет огромную актуальность и практическую значимость в современном мире, где эффективное управление задачами и проектами является неотъемлемой частью успешного функционирования организаций и достижения их целей.

Анализ достижений в области управления задачами и проектами показывает, что существующие методы и инструменты имеют свои преимущества, однако с ростом сложности и масштаба проектов возникают новые вызовы, требующие более эффективных и интегрированных решений. Быстрая смена требований, распределенные команды, необходимость в управлении ресурсами и сроками, а также взаимодействие между участниками проекта – всё это создает потребность в разработке специализированной программы для организации задач и управления проектами.

Целью данного курсового проектирования является разработка и реализация программного решения, которое позволит эффективно организовывать и управлять задачами и проектами. Главной целью программы будет обеспечение прозрачности, координации и улучшения процессов управления, а также повышение эффективности и результативности проектных и командных деятельностей.

В основе проектирования программы для организации задач и управления проектами лежат несколько принципов, направленных на достижение поставленных целей. Во-первых, основным принципом является универсальность и гибкость программного решения, которое должно быть применимо в различных отраслях и типах проектов. Во-вторых, важным принципом является удобство использования и интуитивно понятный интерфейс, чтобы пользователи могли быстро освоить программу и эффективно работать с ней. В-третьих, важным принципом является возможность эффективного взаимодействия и коммуникации между участниками проекта. Программа должна обеспечивать удобные средства обмена информацией, комментирования и обратной связи, а также возможность работы в распределенных командах. В-четвертых, в основе программного решения должна быть система управления задачами и проектами, которая позволяет назначать ответственных исполнителей, определять сроки выполнения задач, отслеживать прогресс и контролировать выполнение.

Дополнительно, программа должна обеспечивать возможность генерации аналитики, чтобы пользователи могли оценивать эффективность работы.

В итоге, разработка и реализация программного решения, соответствующего указанным принципам и требованиям, позволит организациям эффективно управлять своими задачами и проектами, повышая результативность и достигая поставленных целей.

# Аналитический обзор прототипов и литературных источников

## Обзор аналогичных решений

Одним из первых этапов в создании программного продукта является анализ прототипов и литературных источников. На сегодняшний день можно встретить достаточно большое количество программных решений со схожей тематикой. По этой причине, мы рассмотрим ряд таких программных средств.

Первый аналог – **«Trello»**. На рисунке 1.1 изображён интерфейс приложения:

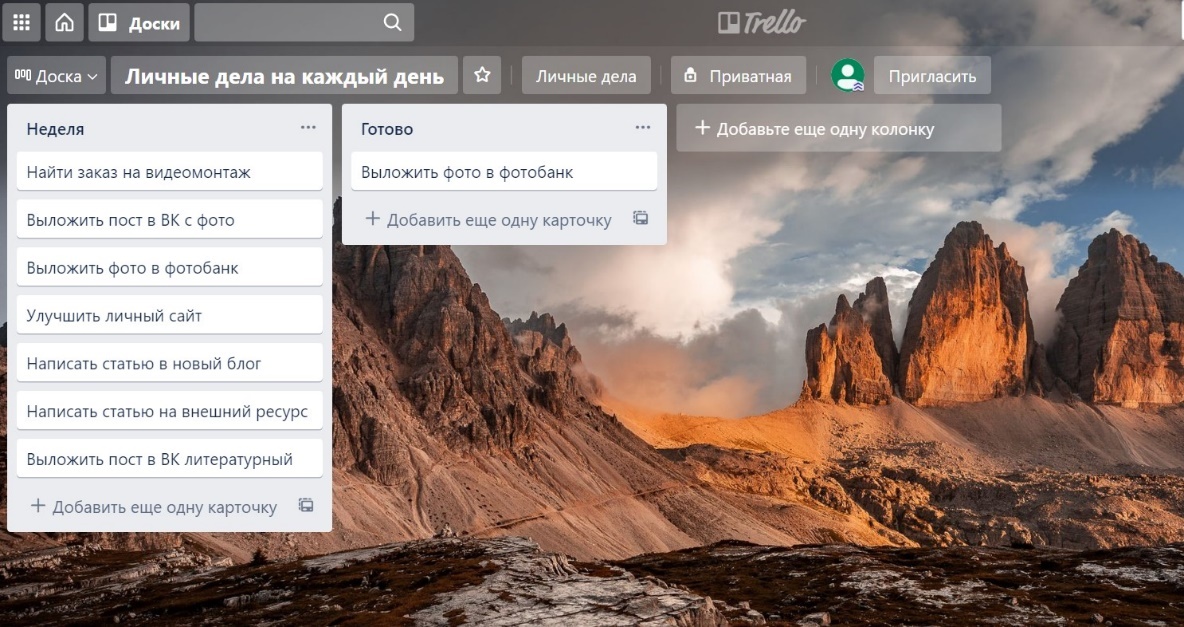


Рисунок 1.1 – Интерфейс приложения «Trello»

Trello - это программное средство для организации задач и управления проектами, которое предоставляет удобные и интуитивно понятные инструменты для визуализации и отслеживания рабочих процессов. Оно позволяет пользователям создавать доски, списки и карточки, которые могут представлять собой задачи, проекты, идеи или любую другую единицу работы.

Основной принцип Trello - это применение методологии Канбан, которая базируется на принципе визуального управления задачами. Каждая доска в Trello представляет определенный проект или рабочий поток, а списки на доске отображают этапы выполнения задачи. Карточки на списке представляют собой отдельные задачи или элементы работы, которые можно перемещать между списками для отслеживания прогресса.

Программное средство Trello предлагает несколько ключевых функциональных возможностей. Во-первых, пользователи могут создавать доски и настраивать их в соответствии с конкретными потребностями проекта или задачи. Во-вторых, они могут создавать списки на доске и добавлять карточки, которые содержат информацию о задаче, описание, сроки выполнения и другие детали.

Кроме того, Trello предоставляет возможность назначать ответственных лиц на задачи, устанавливать приоритеты, добавлять комментарии и вложения, а также устанавливать оповещения и сроки выполнения. Пользователи могут легко перемещать карточки между списками для обозначения текущего состояния задачи (например, "В ожидании", "В процессе", "Завершено").

Trello также предоставляет возможность сотрудничества и коммуникации между участниками команды. Пользователи могут добавлять других пользователей на доски и карточки, делиться комментариями, обсуждать задачи и отслеживать активности. Это способствует более эффективному взаимодействию и синхронизации работы внутри команды.

В качестве второго аналога рассмотрим **«YouGile»** Это программное средство для организации задач и управления проектами, которое предоставляет всесторонние инструменты для эффективного планирования, отслеживания и совместной работы над проектами. Оно разработано с учетом потребностей современных команд и предлагает интуитивно понятный интерфейс и широкий набор функциональных возможностей. Интерфейс приложения представлен на рисунке 1.2:



Рисунок 1.2 – Интерфейс программного средства «YouGile»

Одной из ключевых особенностей YouGile является его модульность и гибкость. Пользователи могут создавать проекты и определять собственную структуру задач, включая списки, подзадачи, теги и метки. Это позволяет адаптировать программное средство под конкретные потребности проекта или команды.

YouGile предлагает различные инструменты для планирования и отслеживания задач. Пользователи могут создавать задачи, устанавливать сроки выполнения, назначать ответственных лиц, устанавливать приоритеты и отмечать прогресс. Графическое представление досок и карточек позволяет легко визуализировать текущее состояние проекта и прогресс работы.

Кроме того, YouGile предоставляет возможности для коммуникации и совместной работы. Пользователи могут оставлять комментарии, обсуждать задачи и обмениваться файлами прямо в рамках программного средства. Также предусмотрены функции оповещений, которые помогают участникам команды быть в курсе изменений и обновлений проекта.

YouGile поддерживает гибкую систему управления доступом, позволяющую определить права и роли участников проекта. Это обеспечивает конфиденциальность и безопасность данных, а также позволяет гибко настраивать доступ к различным функциям и информации.

Важным преимуществом YouGile является его кросс-платформенность и мобильная поддержка. Пользователи могут получать доступ к своим проектам и задачам через веб-интерфейс или мобильные приложения, что позволяет управлять проектами в любое время и из любого места.

Также в качестве аналогов я выделил следующие приложения:

* Bitrix24;
* Pyrus;
* Jira;
* ClickUp;
* Wrike.

Все рассмотренные аналоги имеют схожую тематику и функционал, но у каждого есть отличия, что и делает каждое из них уникальным.

## Требования к проектируемому программному средству

Основываясь на анализе аналогичных решений из пункта 1.1, проектируемое программное средство должно включать следующие характеристики и функциональные возможности.

Оно должно предоставлять интуитивно понятный и визуально привлекательный интерфейс, который позволит пользователям создавать задачи и проекты. Графическое представление задач и их статусов должно быть четким и удобным для отслеживания прогресса.

Гибкость и настраиваемость являются важными аспектами. Пользователи должны иметь возможность настраивать структуру задач в соответствии с потребностями проекта, включая добавление задач и дополнительных сведений (прим. мультимедийных файлов). Гибкая система управления доступом должна позволять определить права и роли участников проекта.

Управление сроками и приоритетами также является важной функциональностью. Пользователи должны иметь возможность устанавливать сроки выполнения задач и определять их приоритеты. Программное средство должно предоставлять удобные инструменты для отслеживания сроков и уведомлять пользователей о предстоящих сроках.

Для обеспечения эффективной коммуникации и совместной работы, программное средство должно позволять пользователям оставлять комментарии, обсуждать задачи и обмениваться файлами в рамках программы. Оно также должно обеспечивать удобные инструменты для совместной работы и взаимодействия между участниками проекта.

Аналитика и отчетность представляют значимый аспект. Программное средство должно предоставлять аналитические возможности, такие как отчеты о выполнении задач, чтобы помочь в управлении проектами и принятии информированных решений.

Интеграция с другими инструментами, такими как электронная почта также является важной характеристикой. Программное средство должно обеспечивать возможность интеграции с различными популярными инструментами и сервисами, чтобы обеспечить более эффективный и синхронизированный рабочий процесс.

Уведомления и оповещения являются неотъемлемой частью программного средства. Оно должно предоставлять функциональность для отправки уведомлений и оповещений, чтобы информировать участников проекта о важных событиях, изменениях и обновлениях. Это помогает участникам оставаться в курсе и своевременно реагировать на изменения.

Суммируя, проектируемое программное средство должно обладать привлекательным интерфейсом, гибкой структурой задач, возможностью управления сроками и приоритетами, инструментами коммуникации и совместной работы, аналитическими возможностями, интеграцией с другими инструментами и функциональностью уведомлений. Учитывая эти характеристики, новое программное средство сможет эффективно поддерживать организацию задач и управление проектами.

# Анализ требований к программному средству и разработка 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. Этот раздел начнём с технических средств, которые использовались для разработки программного средства, после чего перейдём к архитектуре системы и диаграмме вариантов использования.

## Технические средства разработки

Для создания программного средства "TaskWave" были выбраны передовые технологии и инструменты, чтобы обеспечить высокое качество и удобство использования. В процессе разработки был использован язык программирования C#, платформа .Net Framework и инновационная технология Windows Presentation Foundation (WPF).

Организация работы с базой данных в приложении осуществляется с помощью мощного фреймворка Entity Framework, который обеспечивает эффективное взаимодействие с базой данных. База данных разработана с использованием Microsoft SQL Server, обеспечивая надежность и гибкость хранения данных.

Для создания привлекательного и интуитивно понятного пользовательского интерфейса программного средства "TaskWave" был использован Figma - инструмент для разработки макетов и прототипов. Это позволило создать эстетически привлекательный и удобный интерфейс, соответствующий современным требованиям пользователей.

### 2.1.1 Язык программирования C# и платформа .Net

C# - это не только объектно-ориентированный язык программирования, но и сокровищница возможностей для разработки мощных и гибких приложений в различных областях. Рожденный в конце 90-х годов прошлого века благодаря талантливым разработчикам из Microsoft, C# привносит свежесть и инновации в мир программирования.

Один из главных вдохновителей и архитекторов C# - Андерс Хейлсберг, чье имя тесно связано с идеями Turbo Pascal и Delphi. Под его руководством C# стал языком, способным воплотить амбициозные проекты и преобразить их в реальность.

А что же платформа .Net? Она представляет собой еще одно грандиозное достижение Microsoft. Сочетая в себе набор уникальных особенностей, .Net становится мощной платформой для создания приложений. Во-первых, она поддерживает не только C#, но и другие языки программирования и их диалекты, что расширяет возможности разработчиков. Во-вторых, .Net является переносимой платформой, способной работать на различных операционных системах благодаря своей актуальной версии, доступной для большинства платформ. В-третьих, платформа предоставляет разработчикам мощную библиотеку классов, которая является их надежным партнером в создании сложных приложений. И наконец, автоматическая сборка мусора в .Net облегчает жизнь разработчикам, позволяя им фокусироваться на создании функциональности, не отвлекаясь на управление памятью.

Таким образом, C# и платформа .Net представляют собой мощный союз, обеспечивающий разработчикам средства и среду для творчества и инноваций. Они открывают новые горизонты в программировании и являются надежными инструментами для создания современных и высокоэффективных приложений.

### 2.1.2 Entity Framework

Entity Framework - это инновационная объектно-ориентированная технология, основанная на фреймворке .Net, которая преображает работу с данными. Вместо традиционных таблиц, индексов и ключей, Entity Framework возвышает данные до уровня объектов, создавая удивительно элегантный уровень абстракции.

Основной концепцией, лежащей в основе Entity Framework, является сущность. Сущность представляет собой связанный с определенным объектом набор данных, открывая возможности для работы с данными на более высоком уровне абстракции. Забудьте о громоздких таблицах - здесь важны объекты и их связи.

Одной из ключевых особенностей Entity Framework является использование запросов LINQ, позволяющих легко и элегантно извлекать данные из базы данных. Это мощный инструмент, который дает разработчикам полный контроль над выборкой и манипуляцией данными.

Entity Framework предлагает три различных подхода для взаимодействия с базами данных: Database First, Model First и Code First. При разработке программного средства был выбран подход Database First, который позволяет начать с существующей базы данных и автоматически создать соответствующую модель объектов.

В итоге, благодаря Entity Framework, разработка программного средства становится не только эффективной, но и изящной. Эта технология поднимает работу с данными на новый уровень, обеспечивая удобство и гибкость взаимодействия с базами данных. Она становится надежным союзником разработчиков, помогая создавать высококачественные и масштабируемые приложения.

### 2.1.3 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) представляет собой изысканную структуру пользовательского интерфейса, которая вдохнула жизнь в настольные клиентские приложения. Эта платформа является истинным творческим пространством, предоставляющим разработчикам мощный инструментарий для создания впечатляющих и интерактивных пользовательских интерфейсов.

Одной из фундаментальных особенностей WPF является использование языка XAML - декларативного языка описания интерфейса, основанного на XML. XAML открывает двери в мир бесконечных возможностей, предоставляя панели, элементы управления, документы и графические фигуры. Это позволяет разработчикам создавать сложные макеты с несколькими страницами, воплощая свои идеи в прекрасные визуальные композиции.

WPF воплощает суть красоты и гибкости, предоставляя возможность создавать стили, темы и словари ресурсов для приложений. Благодаря этому, дизайнеры и разработчики могут сотворить неповторимые пользовательские элементы управления и преобразить шаблоны уже существующих. Каждый элемент интерфейса становится частью художественного произведения, где визуальные эффекты и анимации оживляют пользовательский опыт.

В итоге, WPF представляет собой гармоничное слияние технологии и искусства, где разработчики могут воплотить свои творческие видения в великолепные настольные приложения. Она подарит вам не только функциональность, но и эстетическое удовольствие, воплощая мир мультимедиа и анимации в пользовательском интерфейсе. WPF - это вдохновение для создания приложений, которые не только впечатляют функциональностью, но и приносят радость взаимодействия с пользователем.

### 2.1.4 Microsoft SQL Server

Microsoft SQL Server является великолепной системой управления реляционными базами данных, созданной с нежным вниманием к деталям корпорацией Microsoft. В ее сердце заложена мощь и гибкость, которая обеспечивает высочайшую скорость работы, надежность и безопасность данных, а также простоту использования и администрирования.

С верным спутником - языком запросов Transact-SQL, Microsoft SQL Server становится готовым рыцарем, способным исполнить все желания. Transact-SQL предлагает богатый набор операторов, охватывающих создание, изменение и удаление объектов баз данных, манипулирование данными, контроль доступа и управление транзакциями. Этот язык открывает двери в мир гибкости и функциональности, позволяя воплотить самые смелые идеи в работу с данными.

Microsoft SQL Server не только предлагает выдающуюся производительность и надежность, но и обладает впечатляющей системой шифрования данных, обеспечивая безопасность и конфиденциальность важной информации. Она становится надежным союзником в сохранении ценных данных и защите их от нежелательных глаз.

В наше время Microsoft SQL Server является великим лидером среди систем управления базами данных. Ее популярность и превосходство подтверждают ее путь к славе, ставя ее на передовом месте в мире баз данных. Благодаря своим превосходным возможностям и широкому применению, Microsoft SQL Server становится надежным союзником в решении самых сложных задач управления данными.

### 2.1.5 Figma

Figma — инновационный инструмент для дизайна и прототипирования, который позволяет командам с легкостью создавать, коллаборировать и делиться проектами. Это веб-приложение с широкими возможностями, которые делают его неотъемлемой частью рабочего процесса дизайнеров и разработчиков.

Одним из главных преимуществ Figma является его возможность работы в режиме реального времени. Благодаря этому, участники команды могут одновременно работать над проектом, видеть изменения других участников и общаться во время работы. Это позволяет значительно сократить время, затрачиваемое на коммуникацию и согласование изменений.

Figma также отличается простотой использования и интуитивным интерфейсом. Его инструменты и функции легко доступны даже для новичков, что делает процесс дизайна более эффективным и удобным. Благодаря мощным функциям макетирования и возможности создания интерактивных прототипов, Figma помогает дизайнерам и разработчикам визуализировать и испытать свои идеи перед реализацией.

Кроме того, Figma обеспечивает высокий уровень совместной работы. Все проекты и файлы хранятся в облаке, что позволяет легко делиться доступом и работать с коллегами и клиентами из любой точки мира. Команды могут вести обсуждения, оставлять комментарии и следить за изменениями, что способствует более эффективному и продуктивному процессу работы над проектами.

Figma стал незаменимым инструментом для дизайнеров и разработчиков, обеспечивая им возможность работать совместно, создавать потрясающие макеты и прототипы, а также сокращать время на коммуникацию и согласование. Благодаря своей мощности и удобству использования, Figma продолжает совершенствовать рабочий процесс команд и способствовать достижению великолепных результатов.

## Архитектура системы

Для курсового проекта был использован способ разработки программных модулей, основанный на архитектуре клиентского приложения. Приложение состоит из двух частей:

* Клиентская часть. В этом модуле осуществляется взаимодействие пользователя с электронным каталогом с помощью графического интерфейса. В клиентской части пользователь сможет получить полную информацию об объектах приложения, в зависимости от присвоенной ему роли (пользователь или администратор).
* Базы данных. В базе данных описываемого приложения хранится информация о пользователе, пользовательском каталоге, о системном каталоге, книгах и прогрессе пользователя.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь данного программного решения должен иметь возможность (тут рассмотрены сразу две роли: стандартный пользователь и управляющий проектами):

* регистрировать себя в приложении;
* входить в приложение, после ввода данных, необходимых для аутентификации;
* просматривать свои текущие задачи и проекты;
* просматривать отправленные и выполненные задачи;
* настраивать свой аккаунт;
* создавать групповые и локальные проекты;
* назначать ответственных для выполнения задач;
* редактировать проекты;
* при выполнении и отправки задач добавлять фотографии и файлы;
* просматривать уведомления;
* просматривать аналитику (статистику) по выполненным задачам;
* закрывать проекты;
* добавлять новые задачи.

Администратор имеет возможность:

* Поддерживать работу с базой данных;
* Поддерживать работу приложения;
* Контролировать конфиденциальность и безопасность личных данных пользователей;
* Осуществлять регистрацию пользователей с определённым статусом.

После проведения анализа были выявлены следующие функциональные требования:

* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* приложение должно предоставлять возможность получения информации по различным критериям

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

## Диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) является графическим средством моделирования, которое позволяет представить функциональность системы или программного продукта с точки зрения ее пользователей. Она помогает идентифицировать и описать различные сценарии использования системы, включая взаимодействие между актерами (пользователями) и самой системой.

Диаграмма вариантов использования состоит из актеров, вариантов использования (Use Case) и связей между ними. Актеры представляют различных пользователей или внешние системы, которые взаимодействуют с программой. Варианты использования представляют собой функциональные возможности системы, которые могут быть выполнены пользователями или другими актерами.

На диаграмме вариантов использования актеры обычно представлены в виде символов в форме человеческих фигур или прямоугольников с названиями, а варианты использования - в виде овалов с их названиями. Связи между актерами и вариантами использования обозначаются стрелками и указывают направление взаимодействия.

Диаграмма вариантов использования помогает команде разработки программного продукта понять, какие функциональности должны быть реализованы и как пользователи будут взаимодействовать с системой. Она также может использоваться в качестве средства коммуникации между разработчиками, заказчиками и другими заинтересованными сторонами для обсуждения требований и функциональности системы.

Преимущества использования диаграммы вариантов использования включают:

Визуализацию функциональности системы: Диаграмма вариантов использования помогает визуализировать все возможные сценарии использования системы, что помогает лучше понять ее функциональность и взаимодействие с пользователями.

Определение требований: Диаграмма вариантов использования помогает идентифицировать и описать функциональные требования к системе на основе сценариев использования.

Улучшение коммуникации: Диаграмма вариантов использования служит средством коммуникации между разработчиками, заказчиками и другими заинтересованными сторонами, позволяя им лучше понять ожидания от данного приложения.

UML-диаграмма вариантов использования для программного средства «TaskWave» представлена в приложении А.

# Проектирование программного средства

## Проектирования логической структуры базы данных

Для создания базы данных описываемого приложения использовалась система управления реляционными базами данных Microsoft SQL Server 2022.

База данных – это совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.

Система управления базами данных – совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами. База данных данного курсового проекта состоит из 10 таблиц. Схема базы данных изображена на рисунке 3.1

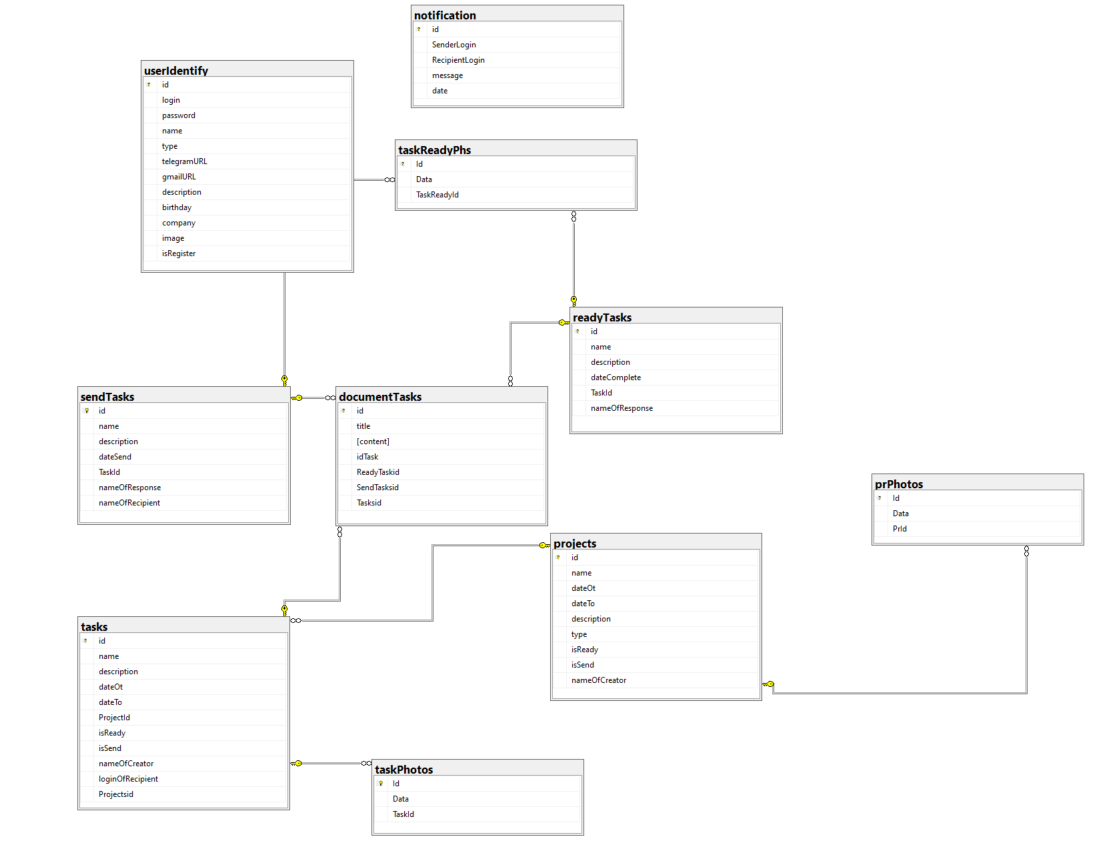


Рисунок 3.1 – Структура базы данных

Остановимся более подробно на некоторых основных таблицах и рассмотрим какие поля они содержат.

Таблица useridentify содержит информацию про всех зарегистрированных пользователей. Type определяет какой тип у пользователя, в зависимости от этого, при авторизации осуществляется открытие разных окон. Поле isRegister необходимо для пользователей с ролью «Руководитель проектов» - для регистрации данного аккаунта необходимо одобрени от администратора. На рисунке 3.2 представлена подробная информация про данную таблицу:

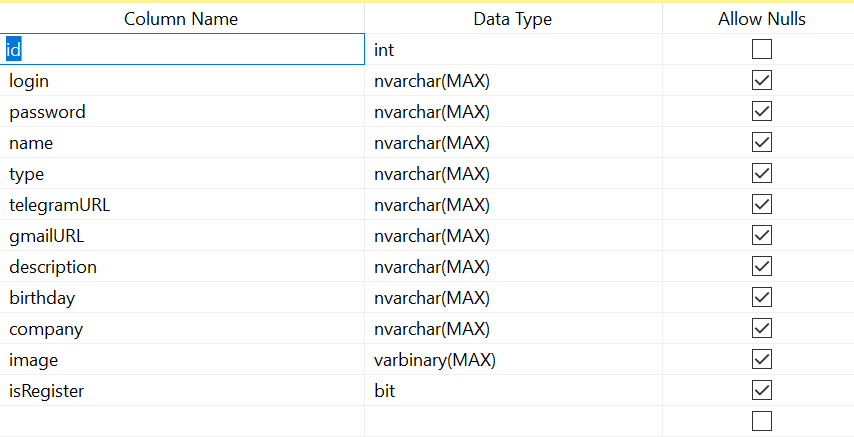


Рисунок 3.2 – Таблица userIdentify

Таблица projects содержит всю информацию про групповые и локальные проекты пользователей. На риснуке 3.3 представлена информация про эту таблицу:

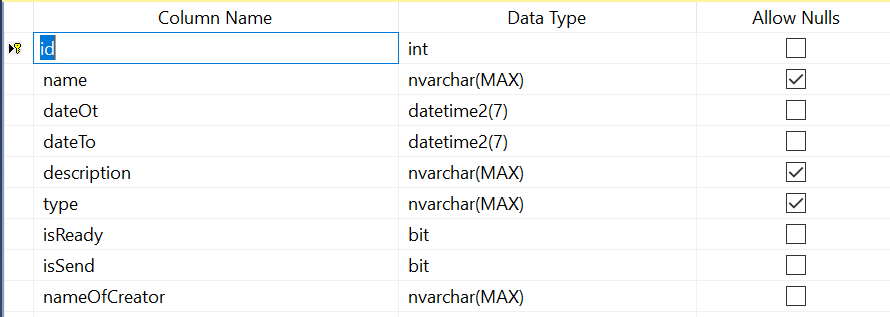


Рисунок 3.3 – Таблица projects

Таблица tasks содежит информацию про все задачи, включая отправленные и выполненные задачи. А поля ProjectId является вторичным ключем для связи с таблицей projects. Структура таблицы представлена на рисунке 3.4:

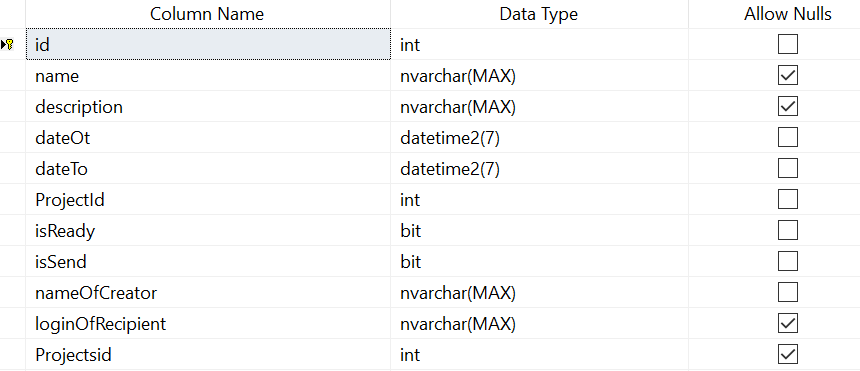


Рисунок 3.4 – Таблица tasks

Между таблицами реализованы связи один ко многим и многие ко многим. Связь один ко многим значит, что несколько строк дочерней таблицы зависят от одной строки родительской таблицы. Связь многие ко многим подразумевает, что многим записям одной таблицы может соответствовать множество записей другой. При это, для реализации, создается дополнительная техническая таблица с сохранением id для связи.

При работе с базой данных, нашим выбранным инструментом является Entity Framework Core 6. Этот фреймворк предоставляет нам множество значимых преимуществ. Мы можем работать с таблицами базы данных, как с обычными классами на языке C#, где столбцы таблиц становятся свойствами этих классов. Кроме того, синтаксис SQL-запросов заменяется на более удобный и интуитивно понятный LINQ. Entity Framework Core берет на себя ответственность по преобразованию нашего кода на C# в соответствующие SQL-инструкции.

В проектировании базы данных существуют три подхода, из которых мы выбрали подход Code-First. При использовании данного подхода мы отказываемся от использования модели EDMX и вместо этого вручную настраиваем классы C# в объектной модели данных. Code-First поддерживает как генерацию классов сущностей из существующей базы данных, так и создание базы данных из ранее созданной модели объектов C#. Это дает нам полный контроль над структурой базы данных и позволяет более гибко адаптироваться к требованиям проекта.

Использование Entity Framework Core и подхода Code-First позволяет нам более эффективно и удобно работать с базой данных, а также обеспечивает гибкость и контроль в процессе разработки и поддержки приложения.

## Проектирования структуры окон

Приложение включает в себя 28 окон и 4 страницы.

Стартовым окном является окно авторизации. Пользователь либо вводит данные и получает доступ к приложению, либо переходит на окно регистрации, после которого он также получает доступ к основному функционалу приложения (в случае регистрации под ролью участника проектов) или ожидает одобрение на регистрацию аккаунта от администратора (в случае регистрации под ролью руководителя проектов).

В зависимости от роли, пользователь переходит либо на окно администратора, либо стандартного пользователя, либо менеджера групповых проектов.

Боковое меню состоит из набора кнопок, которые относятся к определенным типам пользователей согласно их уровню доступа. Эти кнопки позволяют перейти на страницы. Далее рассмотрим варианты окон.

Окно стандартного пользователя дает доступ к стандартному функционалу приложения, пункты меню которого изображены на рисунке 3.5. У него представлены все базовые функции.

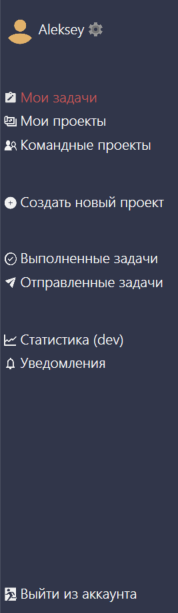


Рисунок 3.5 – Пункты меню для стандартного пользователя

При авторизации под логином и паролем администратора, можно попасть на страницу администрирования, пункты меню изображены на рисунке 3.6. Администратор имеет больше технических возможностей, чем остальные пользователи. Об этом свидетельствуют пункты меню, которые позволяют перейти на страницы с возможностью просмотра и редактирования таблиц БД, в том числе управления таблицами с информацией про пользователей.



Рисунок 3.6 – Пункты меню для администратора

И, наконец, окно руководителя проектов даёт доступ к дополнительному функционалу приложения, в отличии от стандартного пользователя, меню для такого пользователя изображено на рисунке 3.7.

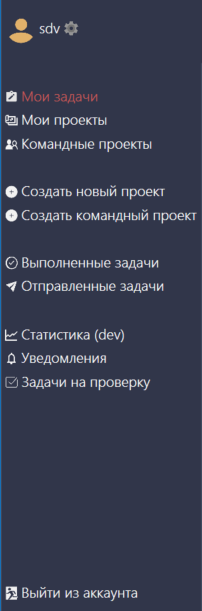


Рисунок 3.7 – Пункты меню для руководителя проектов

Таким образом, мы можем заметить, что для разных ролей пользователей существует разный функционал и, соответственно, разные окна и страницы.

## Проектирование архитектуры приложения

Архитектура программного обеспечения представляет собой совокупность ключевых решений, касающихся организации программной системы. Она включает выбор структурных элементов и их интерфейсов, определение их взаимодействия, а также объединение этих элементов в более крупные системы. Однако, центральную роль играет архитектурный стиль, который определяет всю организацию системы, включая элементы, их интерфейсы, их сотрудничество и способы их соединения.

Для удовлетворения требований проектируемой системы с точки зрения различных атрибутов качества, применяются различные архитектурные шаблоны, так называемые паттерны. В разрабатываемом нами приложении мы используем архитектурный шаблон Model-View-ViewModel (MVVM). Этот шаблон состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

На рисунке 3.8 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.8 – Структура шаблона MVVM

Модель описывает данные, используемые в приложении. В моделях может содержаться логика, связанная с этими данными, например, валидация свойств модели. В то же время модель не должна содержать логику, связанную с отображением данных и взаимодействием с визуальными элементами управления. Часто модель реализует интерфейс INotifyPropertyChanged, который позволяет системе автоматически обнаруживать изменения свойств модели и облегчает привязку к представлению, хотя сама модель не взаимодействует напрямую с представлением.

Представление, или View, определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Важным аспектом является эффективное проектирование пользовательского интерфейса, чтобы создать приятный и удобный опыт для конечного пользователя. Представление не обрабатывает события, а в основном выполняет действия посредством команд.

ViewModel, или модель представления, связывает модель и представление через механизм привязки данных. Если значения свойств модели изменяются, с использованием интерфейса INotifyPropertyChanged, данные в представлении автоматически обновляются, хотя сама модель и представление не прямо связаны. ViewModel также содержит логику получения данных из модели и обновления данных в модели.

Кроме архитектурного паттерна, использовались паттерны для работы с базой данных: это Repository и Unit of Work и паттерн Command для работы с командами.

Repository и Unit of Work являются распространенными паттернами проектирования, применяемыми при разработке приложений с использованием базы данных.

Паттерн Repository (Репозиторий) предоставляет абстракцию для работы с данными. Он представляет собой слой абстракции между приложением и базой данных, скрывая детали взаимодействия с хранилищем данных.

Паттерн Unit of Work (Рабочая единица) обеспечивает контроль над транзакционной целостностью и согласованностью операций с базой данных. Он объединяет несколько операций чтения и записи в единую транзакцию, позволяя гарантировать, что все операции будут успешно выполнены или отменены целиком.

## UML-диаграмма классов

В силу большего количество классов, я разбил диаграмму на несколько частей. Первая из которых изображена на рисунке 3.9 – это классы для реализации паттерна Command.



Рисунок 3.9 – UML-диаграмма классов для реализации команд

Как мы можем заметить, все классы наследуются от абстрактного класса «MyCommand», который, в свою очередь, реализует интерфейс ICommand. Более подробное описание данного класса на рисунке 3.10.

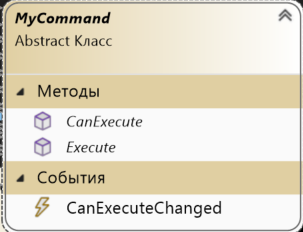


Рисунок 3.10 – Абстрактный класс MyCommand

Далее, на рисунке 3.11 изображены классы «ViewModel» - которые являются необходимыми классами для реализации паттерна MVVM.

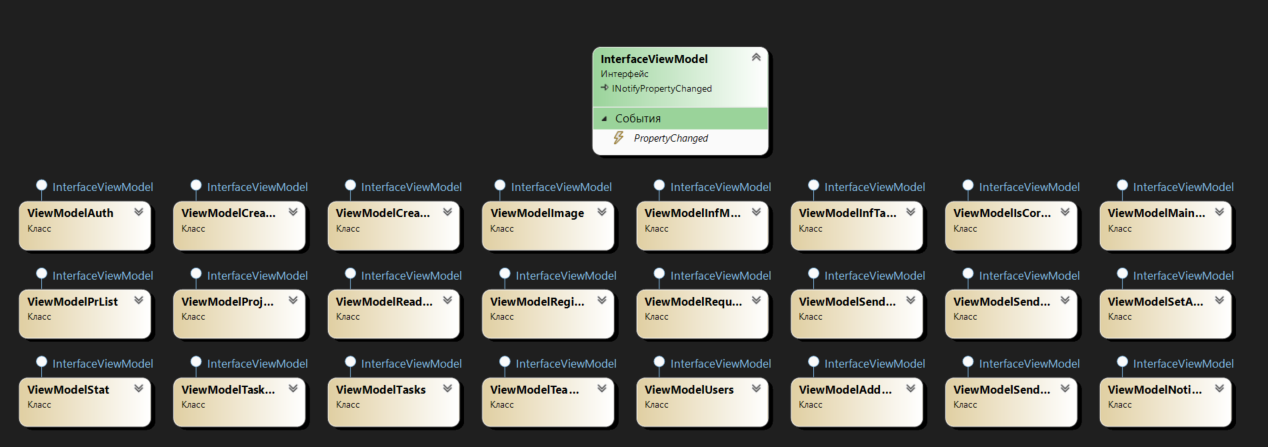


Рисунок 3.11 - UML-диаграмма классов ViewModel

Классы-обработчики для страниц и окон изображены на рисунке 3.12. Большой логики в приложения они не вносят, только соединяют dataContext страницы с моделью представления.



Рисунок 3.12 - UML-диаграмма классов-обработчиков

Так же присутствуют классы, которые используются в качестве сущностей для работы с Entity Framework, вспомогательные классы для реализации паттернов. Их UML-диаграмма представлена на рисунке 3.4.5



Рисунок 3.13 - UML-диаграмма дополнительных классов

В рамках разработки программного решения для организации задач и управления проектами, диаграмма классов является одним из ключевых инструментов моделирования и проектирования системы. Диаграмма классов представляет структуру системы, ее классы, атрибуты и отношения между классами.

В процессе проектирования программы для организации задач и управления проектами, диаграмма классов помогает определить основные классы, их свойства и методы. Она позволяет увидеть структуру данных, которые будут использоваться в программе, а также взаимосвязи между классами. Диаграмма классов также помогает лучше понять концептуальную модель системы и обеспечивает основу для дальнейшей разработки.

# Реализация программного средства

## Основные классы и методы программного средства

В данном разделе будут рассмотрены основные классы с точки зрения архитектуры программы. Естественно, что при разработке программного средства создаётся большое множество и других вспомогательных классов, но они не несут такой же большой ценности, как классы, описанные ниже.

Мы остановимся на классах:

* Для реализации паттерна MVVM;
* Для реализации паттерна Command;
* Для взаимодействия с базой данных.

Так же остановимся и на некоторых методах, которые использовались при разработке программы:

* Методы для работы с файлами;
* Методы для работы с электронной почтой;
* Методы для авторизации и регистрации пользователей.

## Классы для реализации паттерна MVVM

Как отмечалось в предыдущих разделах, при разработке приложения «TaskWave» использовался архитектурный паттерн MVVM. Для его реализации необходимо использовать следующие классы:

* ViewModel (модель представления);
* View (визуальный интерфейс);
* Model (модель данных).

На примере страницы MyTask рассмотрим пример использования вышеупомянтой модели. На рисунке 4.1 изображён фрагмент кода из xaml, демонстрирующий использование связывания для команды. В приложении Б можно ознакомиться с примером реализации xaml-кода для всей страницы.

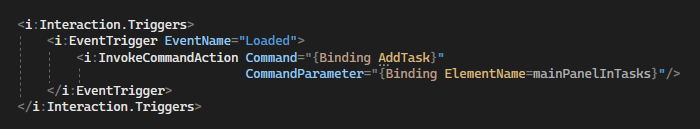


Рисунок 4.1 – XAML-код при связывании команд

Обращаю внимание, связывание команды AddTask происходит при помощи Binding, которое передаётся в свойство comand, а в ComandParameter передаётся параметр, в данном случае это элемент управления (далее ЭУ) StackPanel. Вызов данной команды происходит при помощи тригера событий при загрузке данной страницы (об этом свидетельсвтует событие Loaded).

На картинке 4.2 приводится пример модели представления для страницы MyTask. А в приложении В реализация модели представления для всей страницы. Отмечу, что все классы, реализующие ViewModel реализуют пользовательский интерфейс InterfaceViewModel(он рассмотрен в диаграмме классов в предыдущих разделах), который, в свою очередь, реализует интерфейс INotifyPropertyChancged, блягодаря чему получается использовать привязку данных и динамически обновлять данные в визуальном интерфейсе в случае их изменения. Пример реализации данной конструкции приведён на рисунке 4.3.



Рисунок 4.2 – Модель представления для страницы MyTask

Можем обратить внимание на реализацию метода OnPropertyChanged, который, как правило, реализовыван внутри свойств, что позволяет при передачи туда информации, уведомлять об этом систему, что позволит обновлять данные.

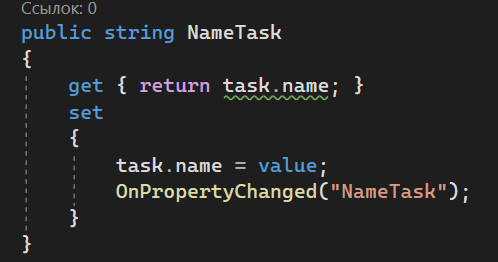


Рисунок 4.3 – Пример конструкции для обновления данных

И, наконец, модель данных представляет из себя обычный класс, объекты которого и будут использоваться на странице для отображения данных. На рисунке 4.4 приведён пример моего Tasks, который содержит информацию про задачи пользователей.

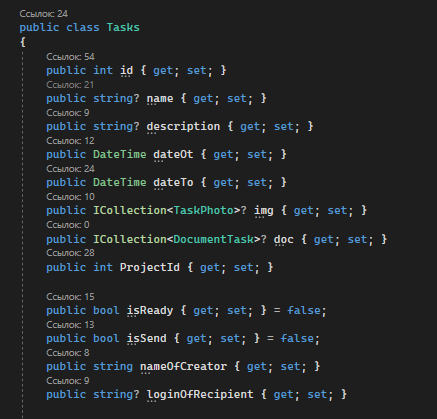


Рисунок 4.4 – Пример модели данных

Но стоит отметить, что наличие такой модели – это скорее из разряда «так надо». Данные со страницы могут привязываться в том числе и к локально созданным полям, которые, например, могут напрямую работать с базой данных.

## Классы для реализации паттерна Command

Как уже можно было заметить, паттерн MVVM вплотную взаимодействует с командами, которые, напомню, реализованы в данном случае с помощью паттерна Command.

Мною использовалось два варианта реализации команд. На рисунке 4.5 показана реализация команды CommandExit.

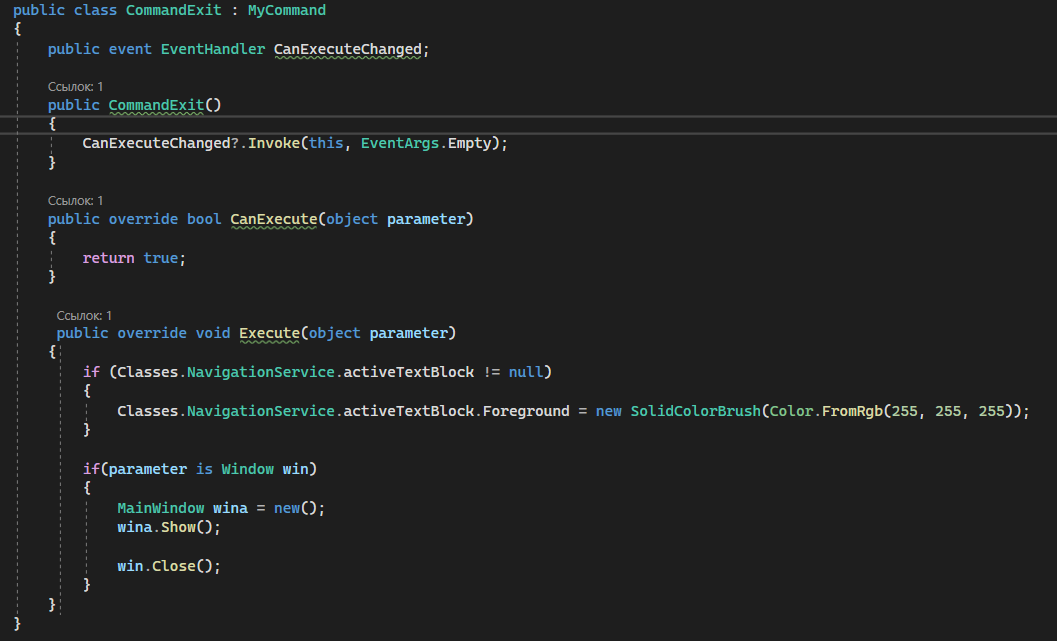


Рисунок 4.5 – Реализация команды CommandExit

Как можно было заметить, данная команда наследуется от абстрактного класса (рисунок 4.6) MyCommand, который реализует базовый интерфейс ICommand.

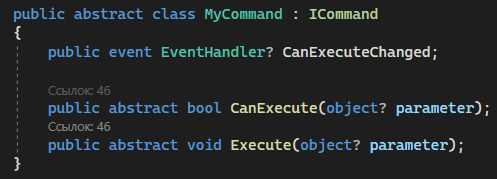


Рисунок 4.6 – Абстрактный класс MyCommand

Вторым вариантом реализации является создание команды, которая в качестве параметров принимает делегаты Action и Func. На рисунке 4.7 приводится пример реализации.

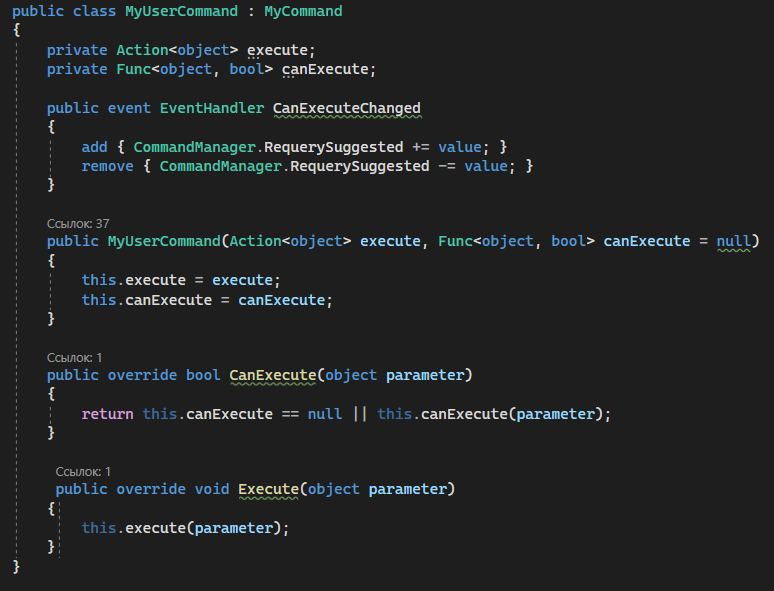


Рисунок 4.7 – реализация команды MyUserComand

После создания команд, необохимо с ними взаимодействовать. Для этого, в модели представления, в зависимости от способа создания команды у нас происходит вызов команд, который связан непосредственно с ЭУ в нашем визуальном интерфейсе. На рисунке 4.8 приведены два примера вызова команд.

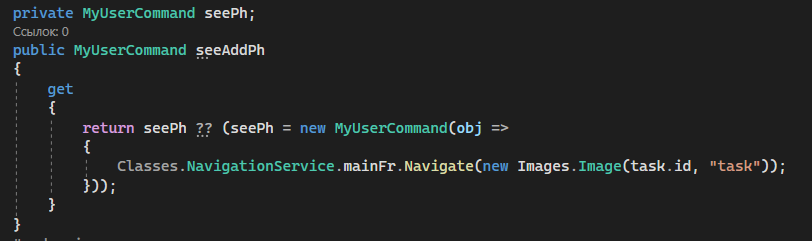


Рисунок 4.8 – Пример вызова команд

Команды можно связывать как для стандартных событий, например, нажатие на кнопку, так и для других соыбтий при помощи триггеров и бибилиотеки interactivity.

## Классы для реализации взаимодействия с базой данных

И наконец, нам осталось рассмотреть примеры классов, которые необходимы для работы с базой данных, взаимодействие с которой, происходит при помощи Entity Framework. Для работы с ним, нам необходимо было реализовать следующие классы:

* Сущность;
* Контекст данных;
* Репозиторий.

Сущность представляет из себя обычный класс, в котором создаются свойства – это и будут столбцы в нашей базе данных. Пример такого класса уже приводился выше на рисунке 4.4. В качестве контекста данных у нас выступпает класс, который наследуется от класса DbContext. В нём при помощи DbSet с указанием tempate-типа и происходит создание таблицы в нашей базе данных. В перегруженном методе OnConfiguring происходит подключение к БД. А в методе OnModelCreating – определение первичных и вторичных ключей. На рисунке 4.9 приводится реализация данного класса.

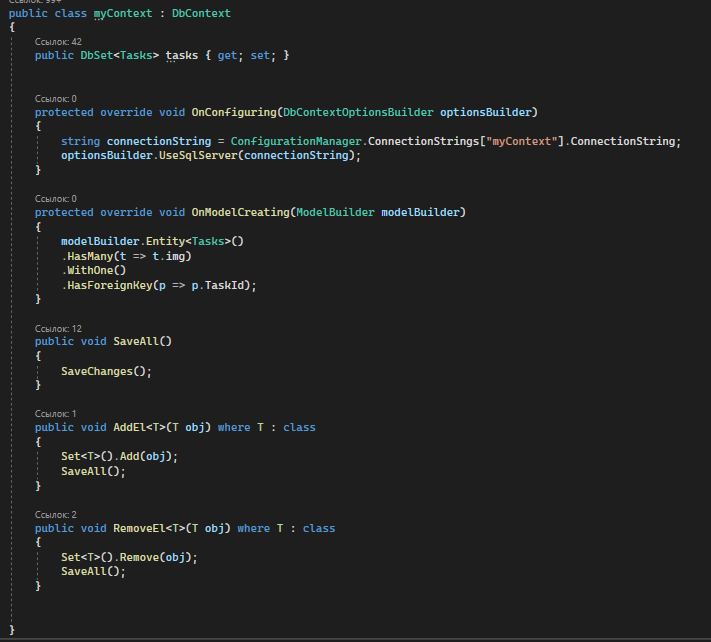


Рисунок 4.9 – Реализация класса myContext

В приложении Г сожно ознакомиться с листингом полного класса myContext. Также стоит обратить внимание на созданные методы схожие с SaveAll, которые являются, по большому счёту, реализацией паттерна Unit Of Work. Хотя, этот паттерн реализован и внутри класса DbContext.

На рисунке 4.10 изоражён класс, реализующий репоизторий. В данном случае рассмотрен пример на основе репоизтория для проектов – ProjectRepository.

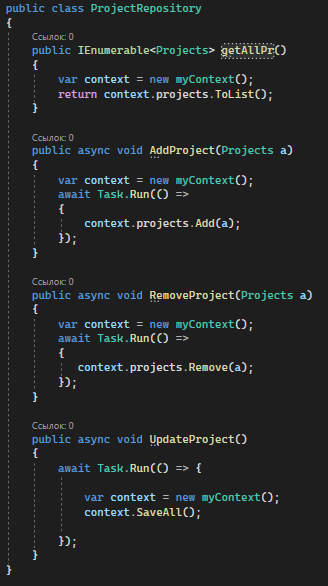


Рисунок 4.10 – Класс репозитория

Репозиторий, как уже отмечалось выше – это прослойка между взаимодействием пользователя и БД. Он позволяет добавить уровень абстрации, который скрывает возможность непосредственного взаимодействия с классом myContext.

## Методы для работы с файлами

В разработанном программном средстве есть возможность работать с файлами: загружать их и открывать. Поддерживаются следующие форматы файлов:

* Фотографии в большинстве популярных форматах;
* Файлы word;
* Файлы excel;
* Файлы pdf;
* Файлы txt.

Кроме тех типов файлов, которые были отмечены – можно загружать и любые другие. Но непосредственно внутри приложения открывать их будет нельзя.

При добавлениии файла в приложении, в базе данных сохраняется его расширение и данные в виде байт. Функция, которая отвечает за добавление файлов представлена в листинге в приложении Д.

При необходимости открыть файл, данный файл будет создан во временоной папке windows temp, название будет соответсвовать изначальному, а внутри он заполнится данными, которые будут конвертированы из байт. Листинг представлен в приложении Е.

## Методы для работы электронной почтой

В приложении «TaskWave» при регистрации пользователей с ролью управляющего проекта необходимо получить ответ от администратора, который может как одобрить, так и запретить регистрацию. Соответствующий ответ будет отправлен пользователю на электронную почту. В листинге 4.1 приводится реализация данного метода для отправки сообщения.

string senderEmail = "TaskWave@outlook.com";

string senderPassword = "TWPassword";

var message = new MimeMessage();

message.From.Add(new MailboxAddress("Отправитель", senderEmail));

message.To.Add(new MailboxAddress("Получатель", recipientEmail));

message.Subject = "Письмо от приложения TaskWave";

message.Body = new TextPart("plain")

{

Text = "Привет, мы рады сообщить, что администрация подтвердила статус вашего аккаунта. Это значит, что теперь вы можете создавать " +

"новые групповые проекты и руководить ими. Желаем удачи!"

};

using (var client = new MailKit.Net.Smtp.SmtpClient())

{

client.Connect("smtp-mail.outlook.com", 587, SecureSocketOptions.StartTls);

client.Authenticate(senderEmail, senderPassword);

client.Send(message);

client.Disconnect(true);

}

message.Dispose();

Листинг 4.1 – Реализация метода для отправки сообщения

Для отправки сообщений мы указываем пароль и логин электронной почты отправителя и данные получателя. Также необохимо указать порт для отправки и SMTP-сервер.

## Методы для регистрации и авторизации пользователей

При запуске приложения пользователь первым делом попадает в окно авторизации, в случае ввода корректных данных, он попадает дальше в зависимости от своей роли. Листинг модели представления для авторизации приводится в приложении Ж.

При отсутствии аккаунта пользователь может зарегистрироваться. Причем на выбор предлагается две роли: управляющий проектов и участник проектов. В приложении З приводится пример реализации модели представления.

# Тестирование, проверка работоспособности и анализ полученных результатов

## Тестировании регистрации и авторизации

Начнём тестирование с авторизации. Попробуем первым делом ввести неверные данные. В результате этих действий будет выведено соответствующее сообщение, которое демонстрируется на рисунке 5.1. Сообщение будет исчезать через 3 секунды после появления.

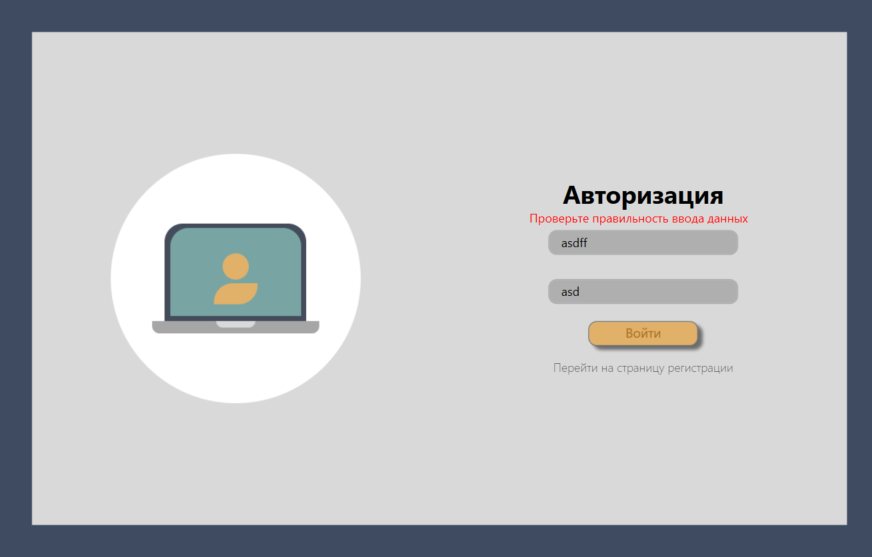


Рисунок 5.1 – Ошибка авторизации

Следующим этапом попробуем оставить все поля для заполнения пустыми. В результате данных действий выполнится анимация, которая в течении 3 секунд будет подсвечивать не заполненные поля. Демонстрируется это на рисунке 5.2.

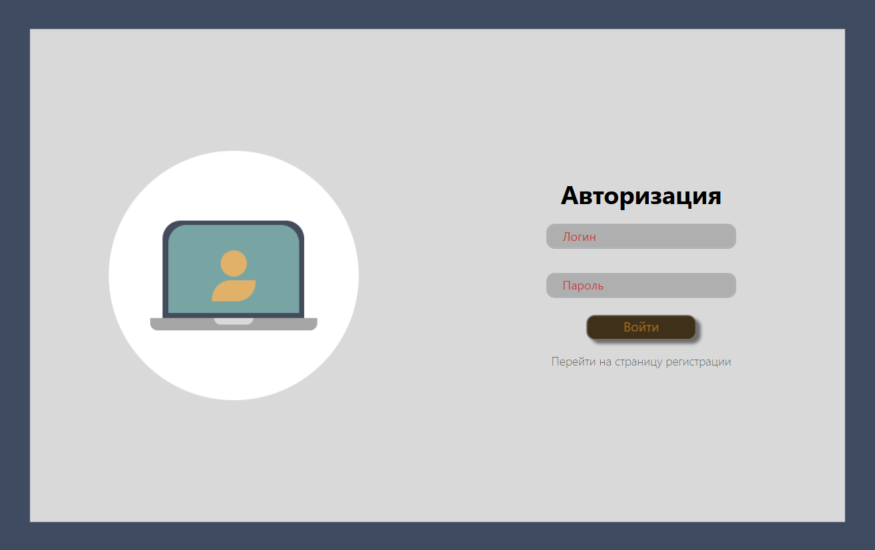


Рисунок 5.2 – Ошибка при заполнении полей

Как мы видим, окно авторизации работает корректно.

Попробуем при регистрации указать логин пользователя, который уже существует. На рисунке 5.3 демонстрируется результат, в котором выводится соответствующее сообщение.

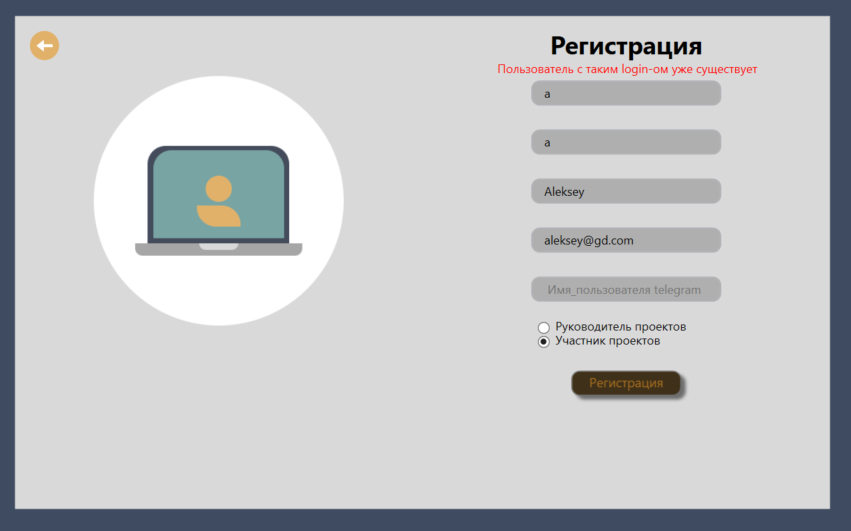


Рисунок 5.3 – Ошибка регистрации

Далее укажем неверный адрес электронной почты. На рисунке 5.4 показано окно, которое будет показано.

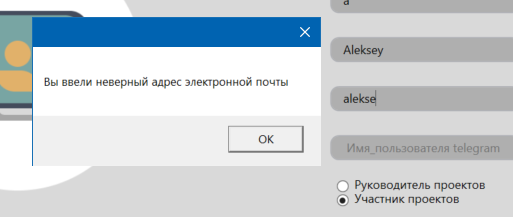


Рисунок 5.4 – Ошибка ввода электронной почты

Таким образом, можно сделать вывод, что окна для авторизации и регистрации обрабатывают все основные случаи возникновения ошибок.

## Тестирование заполнения формы для создания задачи

В основе приложения лежит возможность создания проектов, которые строятся на создании задач. Т.е. каждый проект разбивается на определенное их количество. Непосредственно перед созданием задач, нам необходимо создать проект, указав его описание, название и конечную дату для выполнения. После этого у нас осуществляется переход на страницу создания задач. Мы будем рассматривать создания групповых проектов (т.е. для выполнения задачи мы можем назначить любого пользователя).

Первым делом, попробуем создать проект без задач, результат отображён на картинке 5.5.

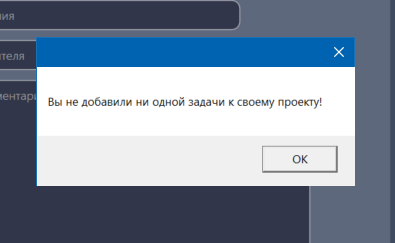


Рисунок 5.5 – Ошибка при отсутствии задач

Далее попробуем создать задачу с датой, которая уже прошла. Результат данной операции на рисунке 5.6.

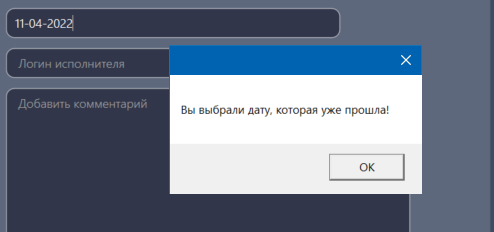


Рисунок 5.6 – Ошибка при заполнении даты

Попробуем ввести дату, которая будет идти после обозначенной даты для окончания всего проекта, а также в место даты ввести строковое значение. На рисунке 5.7 показана оба этих результата.

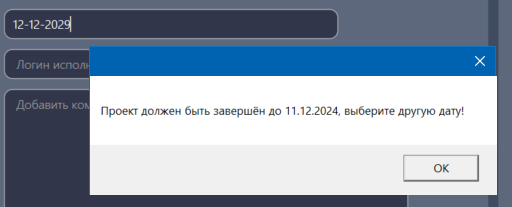
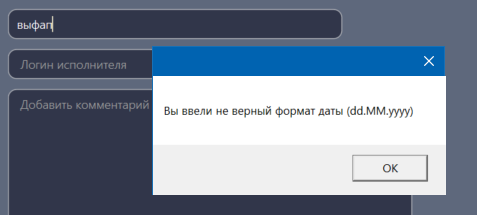


Рисунок 5.7 – Сообщения при неправильном вводе даты

Поскольку это создание проектов для команды, мы для выполнения задачи попробуем ввести неверный логин пользователя. Будет выведено сообщение, которое показано на рисунке 5.8.

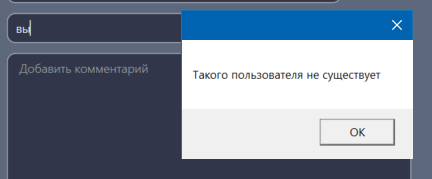


Рисунок 5.5 – Сообщения с оповещением про ошибку ввода логина пользователя

В случае, если мы оставим все поля пустыми и захотим создать задачу, то выполнится анимация, аналогичная той, которая происходит при авторизации.

# Руководство по использованию

При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы (рисунок 6.1).

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Регистрация» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После успешной регистрации, вам нужно будет вернутся в окно авторизации и ввести ваши данные, указанные при регистрации.

Нажав на соответствующие кнопки в окне авторизации, у вас появится окно, где нужно будет ввести данные для входа в учётную запись, через которую производится вход.

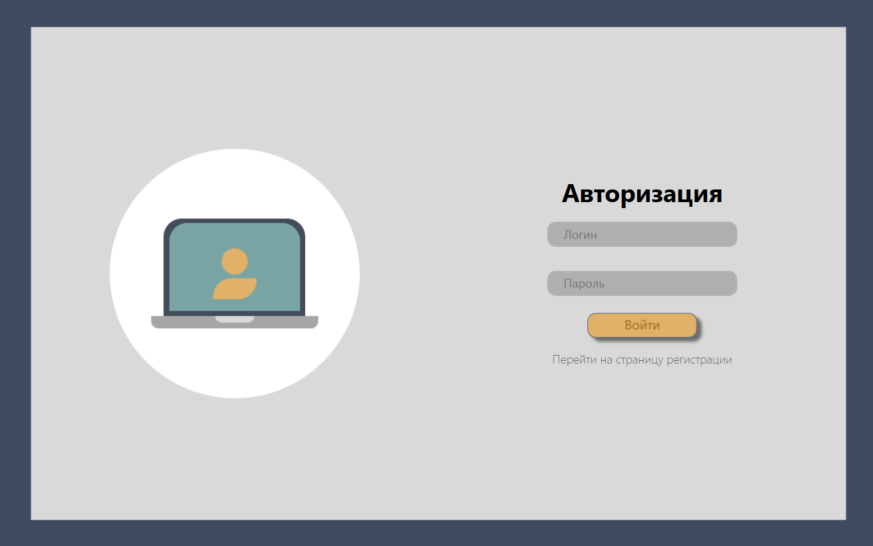


Рисунок 6.1 – Окно авторизации

После успешного входа, мы попадаем в главное окно (на примере пользователя с ролью управляющего проектов), которое изображено на рисунке 6.2.

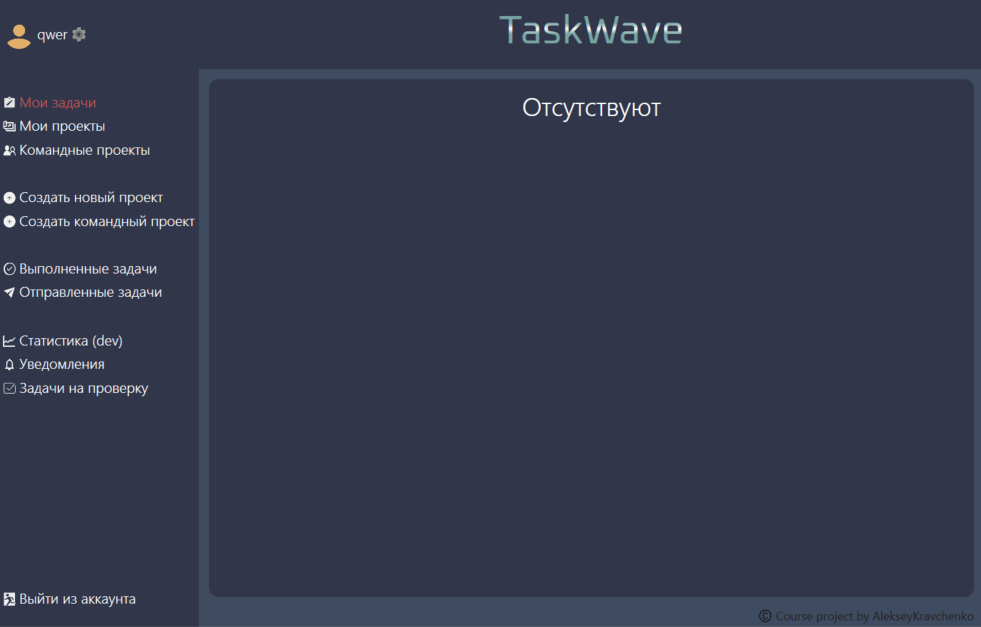


Рисунок 6.2 – Стартовое окно после авторизации

Справа располагается меню, которое является интуитивно понятным.

Лишь сделаю краткое описание каждого из пунктов меню:

* Мои задачи – располагается список всех задач, которые закреплены за текущим пользователем;
* Мои проекты – там можно просмотреть актуальные проекты;
* Командные проекты – аналогично моим проектом, только там будут располагаться те, которые создаются в команде;
* Создать новый проект – при переходе в этот раздел меню, пользователь может создать свой новый личный проект;
* Создать групповой проект – пользователь может создать проект, который будет реализован в рамках взаимодействия с другими пользователями;
* Выполненные задачи – это список задач, который пользователь успешно завершил;
* Отправленные задачи – список задач, которые пользователь отправил на проверку другому пользователю, который ее назначил;
* Статистика – там демонстрируется график с зависимостью выполненных задач и дней;
* Уведомления – раздел, в котором можно просмотреть актуальные сообщения, отправленные пользователю в ходе выполнения задач;
* Задачи на проверку – список задач, которые другие пользователи отправили данному на проверку;
* Выйти из аккаунта – перейти в окно авторизации.

Существует возможность изменения данных про свой личный аккаунт, данный интерфейс демонстрируется на рисунке 6.3.

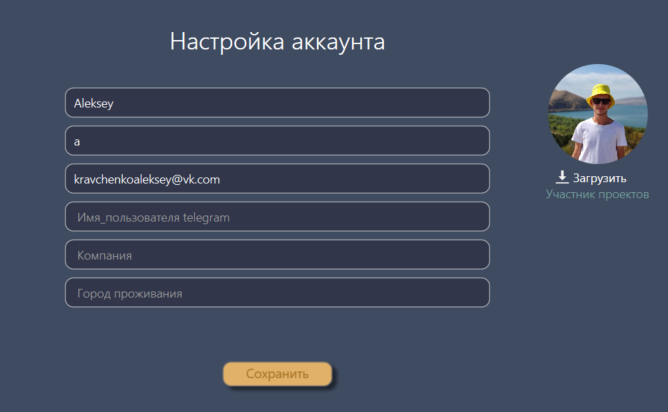


Рисунок 6.3 – Страница редактирования личных данных

Пользователь может изменить свой логин, имя, адрес электронной почты, а также добавить информацию про страну проживания и компанию, в которой он работает. В дальнейших версиях приложения можно будет добавить возможность объединения пользователей по странам проживания и компаниям. Так же пользователь может изменить свою фотографию, загрузив её в любом формате.

Далее рассмотрим пример создания проекта для личных целей. Нам необходимо перейти во вкладку «Создать новый проект», после чего нам откроется окно, которое демонстрируется на рисунке 6.4.

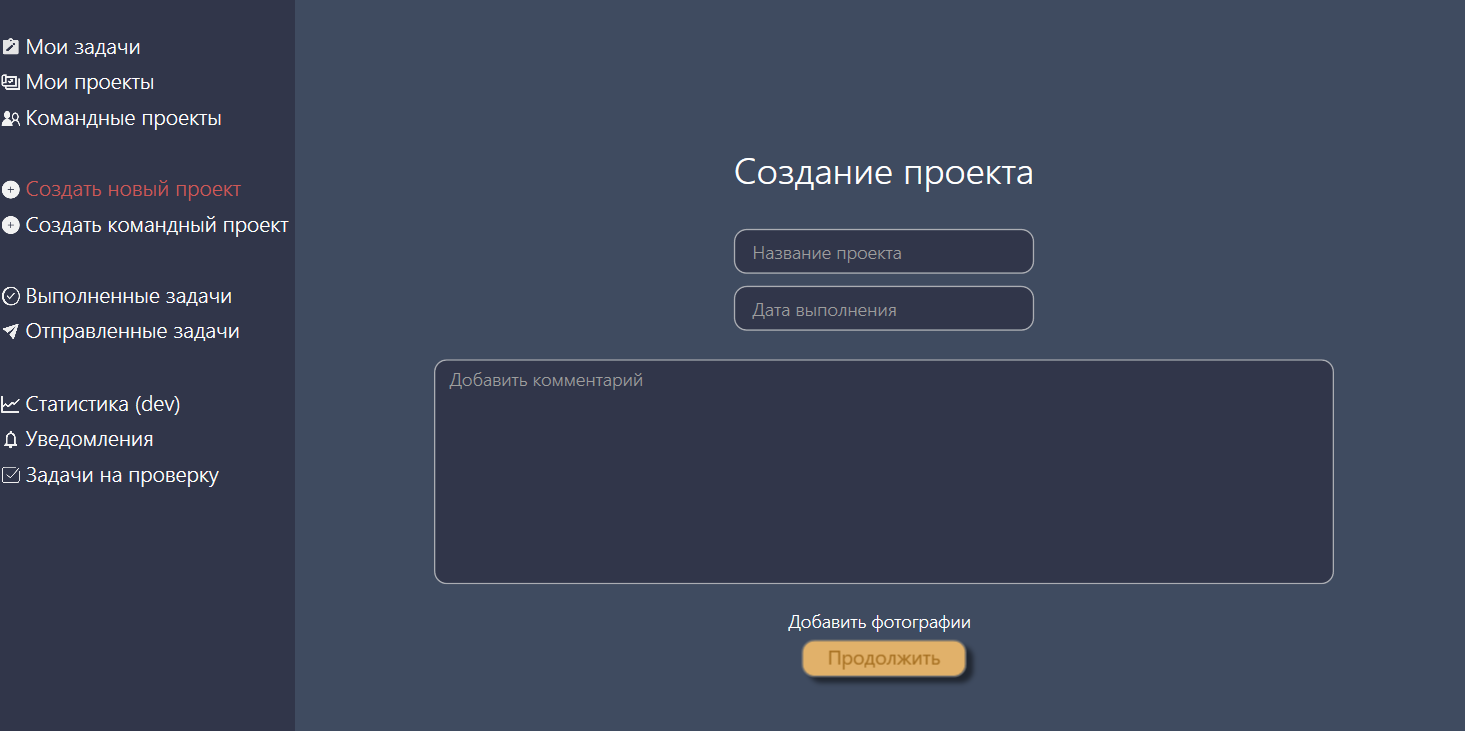


Рисунок 6.4 – Форма для создания проекта

После заполнения данных про проект (название, дата выполнения и описание), прикреплении фотографий (к пожеланию) и нажатии на кнопку «Продолжить», у нас появляется окно, в котором необходимо добавить задачи к текущему проекту. Данное окно показано на рисунке 6.5.

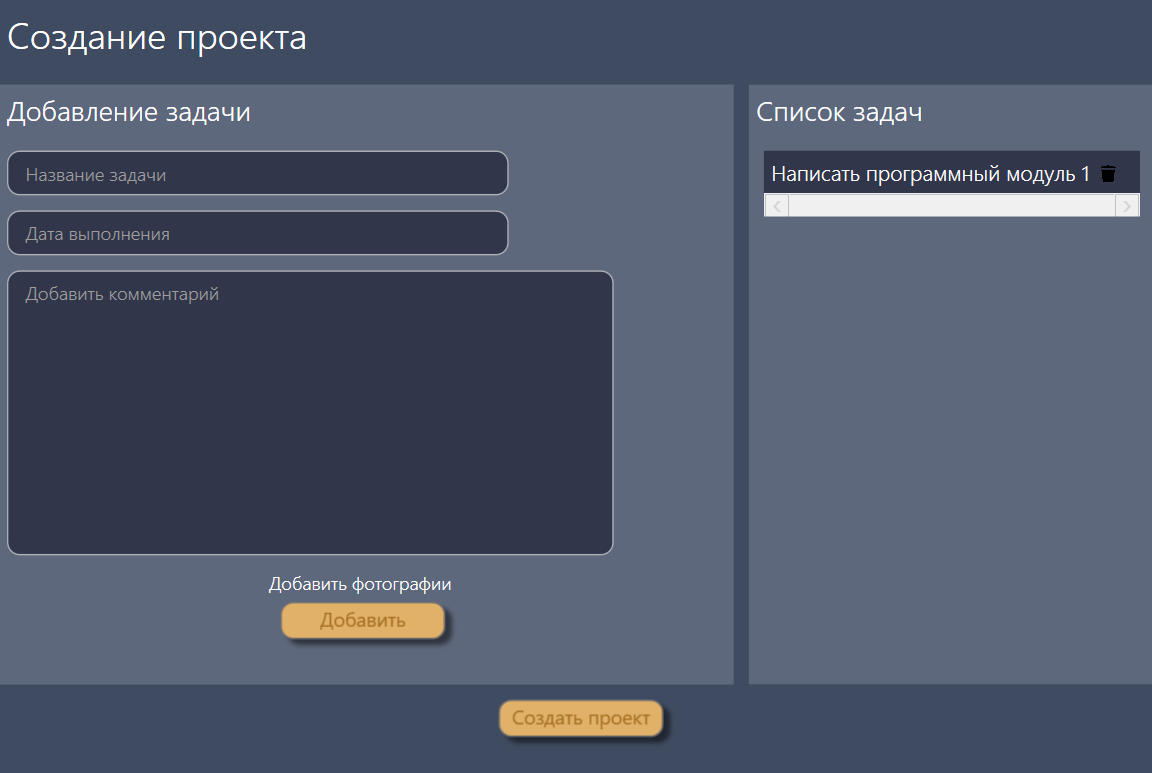


Рисунок 6.5 – Форма для создания задач

При добавлении новой задачи – она отображается у нас в панели справа, в последствии, ее можно удалить. Слева у нас поля для заполнения информации про текущую задачу. При нажатии на добавить – задача будет добавлена. После того, как мы добавим все необходимые задачи, необходимо нажать «Создать проект» и проект будет создан.

После создания задач и проектов, их можно будет просмотреть в соответствующих пунктах меню. На рисунке 6.6 демонстрируется панель со списком всех задач и страница, которая открывается при нажатии на кнопку «Подробнее».

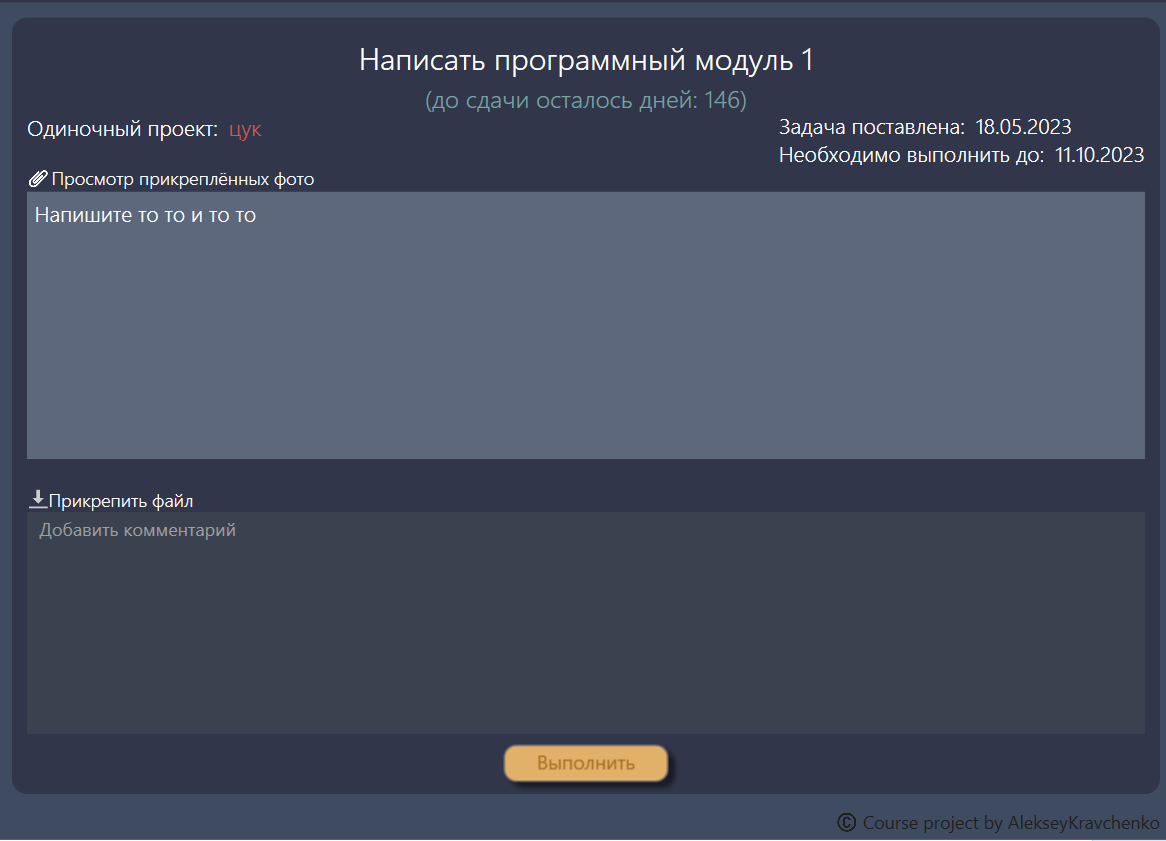
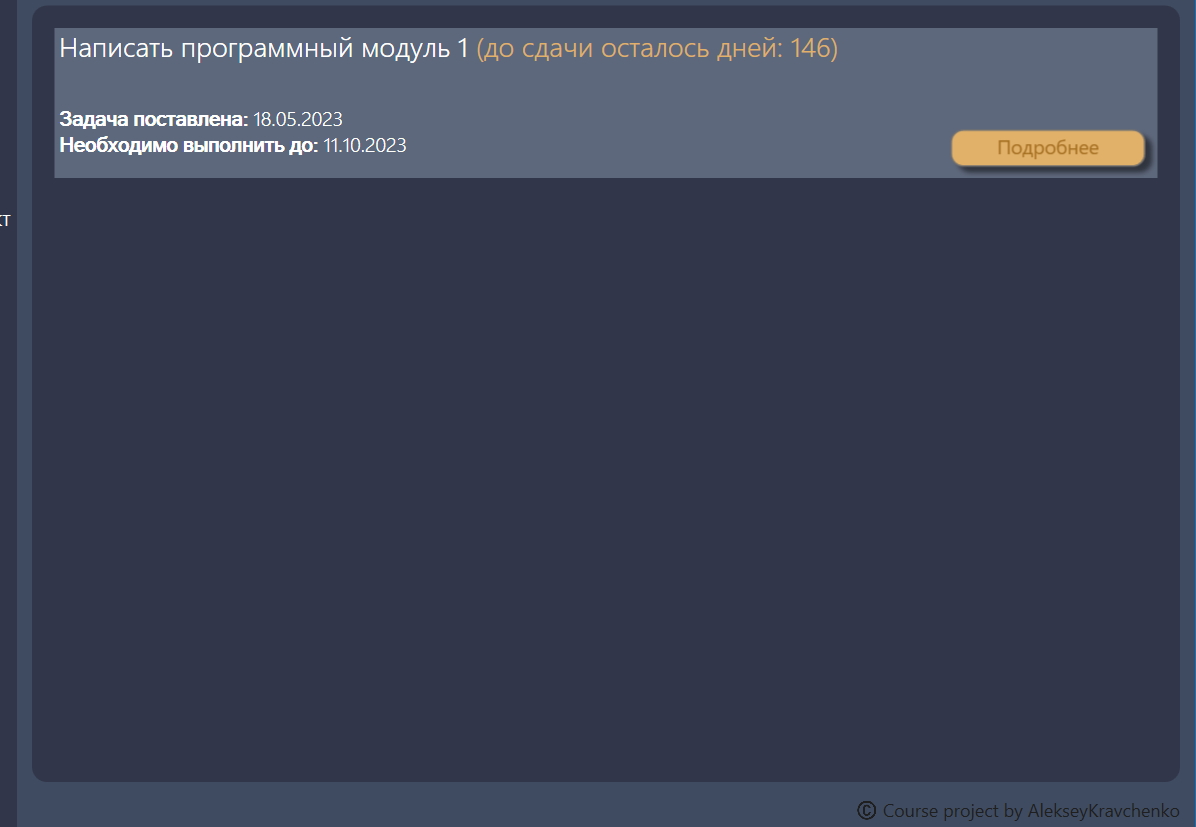


Рисунок 6.6 – Просмотр задач

Далее, пользователь может выполнить задачу (если это личный проект) и отправить на проверку, если это групповой проект. Эти задачи будут отображаться во вкладках «Выполненные задачи» и «Отправленные задачи» соответственно. При отправлении также можно прикрепить файлы и добавить комментарий, который получит пользователь, назначивший данную задачу.

Можно было заметить, что при разработке пользовательского интерфейса, особый акцент был направлен на проработку интуитивно понятного интерфейса для любой аудитории в любом возрасте.

# **ЗАКЛЮЧЕНИЕ**

Таким образом, разработанная программа для организации задач и управления проектами является важным и актуальным направлением в современном бизнес-мире. Проанализировав существующие методы и инструменты управления, становится ясно, что требования современных проектов выходят за рамки их возможностей.

Однако, этот курсовой проект стремится преодолеть эти ограничения и создать инновационное программное решение, способное эффективно организовывать и управлять задачами и проектами. Главной целью программы стало установление прозрачности, согласованности и улучшение управленческих процессов, а также повышение эффективности и результативности командной работы.

Основой разработки программы лежат принципы, направленные на достижение этих целей. Во-первых, приоритетом является универсальность и гибкость программного решения, чтобы оно могло применяться в различных отраслях и типах проектов. Это позволит организациям адаптировать программу под свои уникальные потребности и требования, обеспечивая более точное управление и контроль над задачами и проектами.

Во-вторых, уделяется особое внимание удобству использования и интуитивно понятному интерфейсу программы. При разработке приложения было стремление создать интуитивно понятное и легкое в освоении программное решение, чтобы пользователи могли быстро овладеть его функциональностью и максимально эффективно использовать его в своей работе.

Создание такой программы для организации задач и управления проектами открывает перед организациями новые горизонты возможностей. Она способствует повышению продуктивности, сокращению времени выполнения проектов и улучшению коммуникации между участниками.

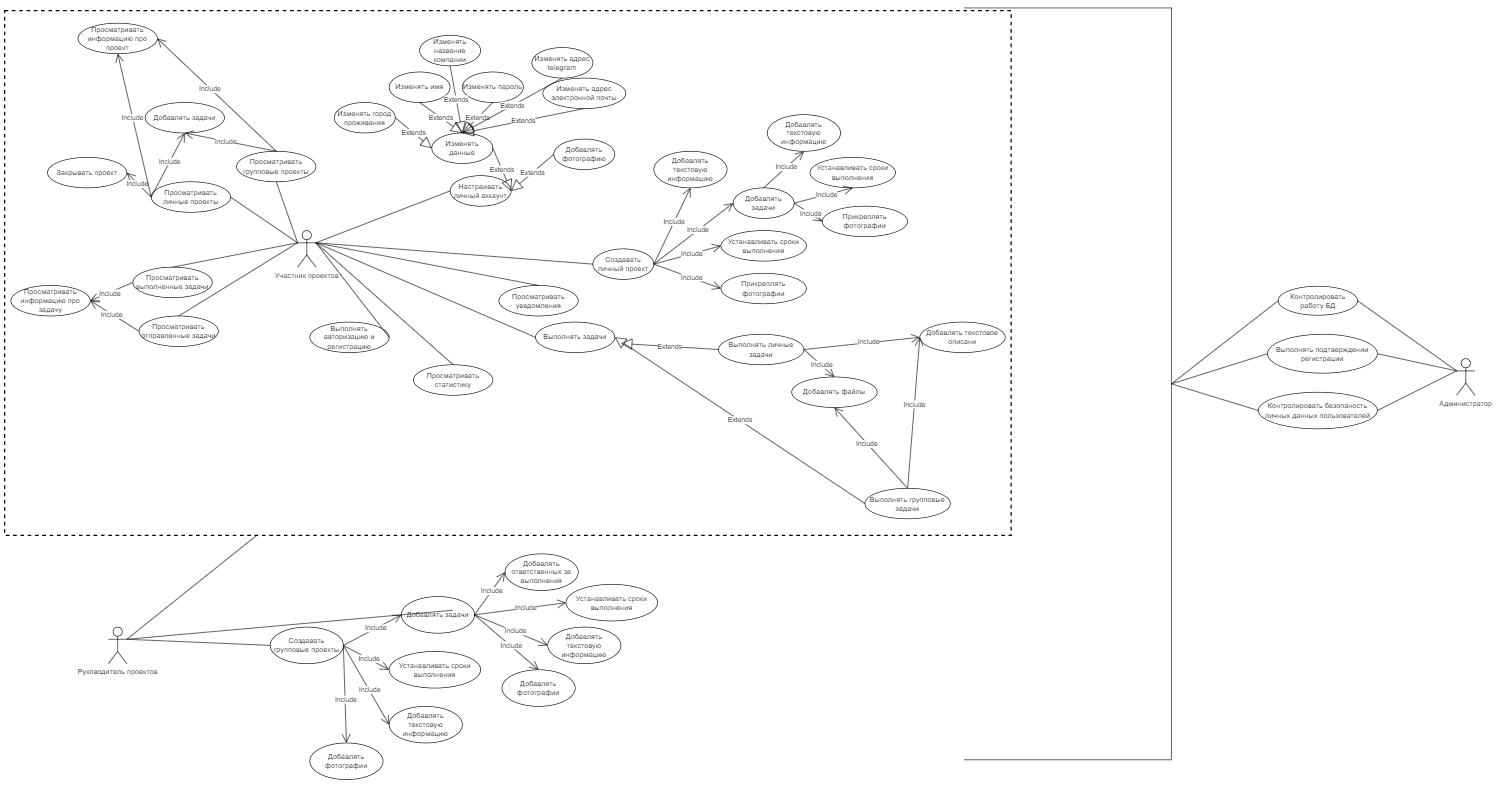
В итоге, разработка и реализация этого программного решения принесет значительную пользу для организаций, помогая им достигать своих целей более эффективно и успешно. Все эти факторы делают программу для организации задач и управления проектами неотъемлемой частью современного бизнеса, способствуя его развитию и достижению высоких результатов.

# Список использованных источников

1. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.04.2023
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 23.04.2023
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 28.04.2023
5. Руководство по XAML // [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/xaml/index.htm– Дата доступа: 20.04.2023
6. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/ – Дата доступа 23.04.2023
7. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

# ПРИЛОЖЕНИЕ А

Диаграмма вариантов использования



# ПРИЛОЖЕНИЕ Б

Листинг – XAML-код для страницы CheckTask

<Page x:Class="TaskWave.Pages.Manager.SendTaskCheck.SendTaskCheck"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:TaskWave.Pages.Manager.SendTaskCheck" xmlns:i="http://schemas.microsoft.com/expression/2010/interactivity"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="SendTaskCheck">

<Border Style="{StaticResource BorderTask}">

<ScrollViewer VerticalScrollBarVisibility="Auto">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="0.7\*"/>

</Grid.RowDefinitions>

<StackPanel x:Name="mainPanelInProject" Margin="10 10 10 0" Grid.Row="0">

<TextBlock Text="{Binding NameTask}" Style="{StaticResource TB\_H1}"/>

<TextBlock HorizontalAlignment="Center">

<TextBlock Text="{Binding DayEnd}" Style="{StaticResource TB\_H2}"/>

</TextBlock>

<StackPanel>

<StackPanel HorizontalAlignment="Left">

<TextBlock>

<TextBlock Text="{Binding GroupPrOrNot}" Style="{StaticResource TB\_H3TypePr}"/>

<TextBlock Text="{Binding namePr}" Style="{StaticResource TB\_H3TypePrValue}"/>

</TextBlock>

</StackPanel>

<StackPanel Style="{StaticResource rightPanel}">

<TextBlock Text="{Binding date}" Style="{StaticResource TB\_H3TypePr}"/>

<TextBlock/>

</StackPanel>

</StackPanel>

<TextBlock>

<i:Interaction.Triggers>

<i:EventTrigger EventName="MouseDown">

<i:InvokeCommandAction Command="{Binding seeAddPh}" />

</i:EventTrigger>

</i:Interaction.Triggers>

<Image Style="{StaticResource iconAttach}"/>

<TextBlock Text="Просмотр прикреплённых фото" Style="{StaticResource TB\_SeePhoto}" Cursor="{StaticResource cur}"/>

</TextBlock>

</StackPanel>

<ScrollViewer Style="{StaticResource ScrollViewer}">

<TextBlock Style="{StaticResource TBWithOpis}" Text="{Binding description}"/>

</ScrollViewer>

<TextBlock Style="{StaticResource TBBut}" Margin="10 7 0 0">

<Image Style="{StaticResource iconAttach}" />

<TextBlock Text="Просмотр отправленных файлов" Style="{StaticResource TB\_SeePhoto}" Cursor="{StaticResource cur}">

<i:Interaction.Triggers>

<i:EventTrigger EventName="MouseDown">

<i:InvokeCommandAction Command="{Binding OpenFiles}" />

</i:EventTrigger>

</i:Interaction.Triggers>

</TextBlock>

</TextBlock>

<ScrollViewer Grid.Row="1" Margin="10 25 10 50" VerticalScrollBarVisibility="Auto">

<TextBlock Style="{StaticResource TBWithOpis}" Text="{Binding OurResponse}" Background="#3B424F"/>

</ScrollViewer>

<Button Grid.Row="1" Style="{StaticResource StartBtn}" Content="Принять" Command="{Binding Accept}" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="0 0 120 8"/>

<Button Grid.Row="1" Style="{StaticResource StartBtn}" Content="Не принять" Command="{Binding NoAccept}" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="120 0 0 8"/>

</Grid>

</ScrollViewer>

</Border>

</Page>

# ПРИЛОЖЕНИЕ В

Листинг – Модель представления для страницы CheckTask

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using TaskWave.Classes;

using TaskWave.Commands;

using TaskWave.DataBase;

using TaskWave.Pages.Manager.TaskForCorrect;

using TaskWave.Pages.ReadyTask;

using TaskWave.Pages.SnadartUser.SendTask;

namespace TaskWave.Pages.Manager.SendTaskCheck

{

public class ViewModelSendTaskCheck : InterfaceViewModel

{

public event PropertyChangedEventHandler? PropertyChanged;

protected void OnPropertyChanged(string name)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

handler(this, new PropertyChangedEventArgs(name));

}

}

Classes.SendTasks sendTask;

myContext context;

public ViewModelSendTaskCheck(int id)

{

context = new();

int b = 0;

foreach (var a in context.sendTasks)

{

if (a.TaskId == id)

{

b++;

sendTask = a;

nameTask = a.name;

break;

}

}

}

#region fields

private string \_DayEnd;

public string DayEnd

{

get

{

return "Задача выполнена: " + sendTask.nameOfResponse;

}

set

{

OnPropertyChanged(nameof(DayEnd));

}

}

private string nameTask { get; set; }

public string groupOrNotBtn

{

get

{

return "Вернуться назад";

}

}

public string DescResponse

{

get

{

return sendTask.description;

}

set

{

sendTask.description = value;

OnPropertyChanged("DescResponse");

}

}

public string NameTask

{

get

{

return sendTask.name;

}

set

{

sendTask.name = value;

OnPropertyChanged("NameTask");

}

}

public string GroupPrOrNot

{

get

{

foreach (var a in context.tasks.ToList())

{

if (sendTask.TaskId == a.id)

{

foreach (var b in context.projects.ToList())

{

if (a.ProjectId == b.id && b.type == "team")

{

return "Групповой проект: ";

}

}

}

}

return "Одиночный проект: ";

}

set

{

OnPropertyChanged("GroupPrOrNot");

}

}

public string namePr

{

get

{

foreach (var a in context.tasks.ToList())

{

if (sendTask.TaskId == a.id)

{

foreach (var b in context.projects.ToList())

{

if (a.ProjectId == b.id)

{

return b.name;

}

}

}

}

return "";

}

set

{

OnPropertyChanged("namePr");

}

}

public string date

{

get

{

return "Задача отправлена: " + Convert.ToString(sendTask.dateSend.Date.ToString("dd.MM.yyyy"));

}

set

{

OnPropertyChanged("dateTo");

}

}

public string description

{

get

{

foreach (var a in context.tasks)

{

if (a.id == sendTask.TaskId)

return a.description;

}

return "";

}

}

public string OurResponse

{

get

{

return sendTask.description;

}

}

#endregion

#region commands

private static TextBox tb;

private MyUserCommand \_NoAccept;

public MyUserCommand NoAccept

{

get

{

return \_NoAccept ?? (\_NoAccept = new MyUserCommand(obj =>

{

Window dialog;

dialog = new Window

{

Title = "Ввод информации",

MinWidth = 500,

MinHeight = 315,

MaxHeight = 315,

MaxWidth = 500,

Cursor = new System.Windows.Input.Cursor("D:\\studing\\4\_semestr\\Course\_project\\Cursor\\Red Neon\\normal\_select.cur"),

WindowStartupLocation = WindowStartupLocation.CenterOwner,

};

dialog.Content = CreateDialogContent(dialog);

dialog.ShowDialog();

int id2 = 0;

foreach (var a in context.tasks)

{

if (a.id == sendTask.TaskId)

{

id2 = a.id;

a.isSend = false;

a.isReady = false;

break;

}

}

context.SaveChanges();

foreach(var a in context.sendTasks)

{

if(a.id == sendTask.id)

{

Notifications notf = new Notifications();

notf.date = DateTime.Now;

notf.message = tb.Text;

notf.SenderLogin = activeUser.user.login;

notf.RecipientLogin = a.nameOfResponse;

context.notification.Add(notf);

context.sendTasks.Remove(a);

break;

}

}

context.SaveChanges();

MessageBox.Show("Вы не приняли задачу!");

Classes.NavigationService.mainFr.Navigate(new CheckTask());

}));

}

}

private StackPanel CreateDialogContent(Window win)

{

var stackPanel = new StackPanel() { Width = 485, Height = 300, Background = new SolidColorBrush((System.Windows.Media.Color)System.Windows.Media.ColorConverter.ConvertFromString("#444C5C")) };

var image = new System.Windows.Controls.Image { Width = 70, VerticalAlignment = VerticalAlignment.Top, HorizontalAlignment = HorizontalAlignment.Right, Margin = new Thickness(5), Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\book-registr.png", UriKind.RelativeOrAbsolute)) };

var textBlock = new TextBlock { Text = "Введите сообщение для пользователя", Margin = new Thickness(7, 0, 0, 0), FontSize = 12, Style = (Style)Application.Current.Resources["nameAcc"] };

var border = new Border { CornerRadius = new CornerRadius(7), Background = System.Windows.Media.Brushes.White, Margin = new Thickness(5), Height = 100 };

var textBox = new TextBox { Margin = new Thickness(5), BorderThickness = new Thickness(0) };

var button = new Button { Style = (Style)Application.Current.Resources["StartBtn"], Content = "Ок" };

tb = textBox;

button.Click += (sender, e) =>

{

win.Close();

};

stackPanel.Children.Add(image);

stackPanel.Children.Add(textBlock);

stackPanel.Children.Add(border);

border.Child = textBox;

stackPanel.Children.Add(button);

return stackPanel;

}

private MyUserCommand \_Accept;

public MyUserCommand Accept

{

get

{

return \_Accept ?? (\_Accept = new MyUserCommand(obj =>

{

int id2 = 0;

foreach(var a in context.tasks)

{

if(a.id == sendTask.TaskId)

{

id2 = a.id;

a.isSend = false;

a.isReady = true;

break;

}

}

context.SaveChanges();

Classes.ReadyTask readyTask = new();

foreach (var a in context.sendTasks)

{

if(a.TaskId == id2)

{

Notifications notf = new Notifications();

notf.date = DateTime.Now;

notf.message = "Ваша задача " + a.name + " принята";

notf.SenderLogin = activeUser.user.login;

notf.RecipientLogin = a.nameOfResponse;

context.notification.Add(notf);

context.sendTasks.Remove(a);

readyTask.dateComplete = DateTime.Now;

readyTask.description = a.description;

readyTask.TaskId = a.TaskId;

readyTask.nameOfResponse = a.nameOfResponse;

readyTask.name = a.name;

readyTask.doc = a.doc;

readyTask.img = a.img;

break;

}

}

context.SaveChanges();

context.readyTasks.Add(readyTask);

context.SaveChanges();

MessageBox.Show("Вы приняли задачу!");

Classes.NavigationService.mainFr.Navigate(new CheckTask());

}));

}

}

private MyUserCommand seePh;

public MyUserCommand seeAddPh

{

get

{

return seePh ?? (seePh = new MyUserCommand(obj =>

{

foreach (var a in context.tasks)

{

if (a.id == sendTask.TaskId)

{

Classes.NavigationService.mainFr.Navigate(new SnadartUser.Images.Image(a.id, "task", "check"));

break;

}

}

}));

}

}

private MyUserCommand openFiles;

public MyUserCommand OpenFiles

{

get

{

return openFiles ?? (openFiles = new MyUserCommand(obj =>

{

Classes.NavigationService.mainFr.Navigate(new SnadartUser.AddFiles.AddFiiles("sendTask", sendTask.id, true));

}));

}

}

#endregion

}

}

# ПРИЛОЖЕНИЕ Г

Листинг – Класс MyContext

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using TaskWave.Classes;

namespace TaskWave.DataBase

{

public class myContext : DbContext

{

public DbSet<User> userIdentify { get; set; }

public DbSet<Projects> projects { get; set; }

public DbSet<Tasks> tasks { get; set; }

public DbSet<ReadyTask> readyTasks { get; set; }

public DbSet<SendTasks> sendTasks { get; set; }

public DbSet<PrPhoto> prPhotos { get; set; }

public DbSet<TaskPhoto> taskPhotos { get; set; }

public DbSet<TaskReadyPh> taskReadyPhs { get; set; }

public DbSet<DocumentTask> documentTasks { get; set; }

public DbSet <Notifications> notification { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

string connectionString = ConfigurationManager.ConnectionStrings["myContext"].ConnectionString;

optionsBuilder.UseSqlServer(connectionString);

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<User>().HasKey(e => e.id);

modelBuilder.Entity<Notifications>().HasKey(e => e.id);

modelBuilder.Entity<Projects>()

.HasMany(p => p.images)

.WithOne()

.HasForeignKey(i => i.PrId);

modelBuilder.Entity<Tasks>()

.HasMany(t => t.img)

.WithOne()

.HasForeignKey(p => p.TaskId);

modelBuilder.Entity<ReadyTask>()

.HasMany(t => t.img)

.WithOne()

.HasForeignKey(p => p.TaskReadyId);

modelBuilder.Entity<SendTasks>()

.HasMany(t => t.img)

.WithOne()

.HasForeignKey(p => p.TaskReadyId);

}

public void SaveAll()

{

SaveChanges();

}

public void AddEl<T>(T obj) where T : class

{

Set<T>().Add(obj);

SaveAll();

}

public void RemoveEl<T>(T obj) where T : class

{

Set<T>().Remove(obj);

SaveAll();

}

}

}

# ПРИЛОЖЕНИЕ Д

Листинг – Метод для добавления файлов

//добавление документов в БД с открытием окна для выбора файлов

private MyUserCommand addDoc;

public MyUserCommand AddDoc

{

get

{

return addDoc ?? (addDoc = new MyUserCommand(obj =>

{

try {

var openFileDialog = new Microsoft.Win32.OpenFileDialog();

if (openFileDialog.ShowDialog() == true)

{

string Path1 = openFileDialog.FileName;

var document = new DocumentTask

{

title = Path.GetFileName(Path1),

content = File.ReadAllBytes(Path1),

idTask = task.id

};

string extension = Path.GetExtension(Path1);

Image image = new Image();

image.Width = 15;

image.Margin = new Thickness(2, 10, 0, -1);

switch (extension)

{

case ".doc":

case ".docx":

// файл Microsoft Word

image.Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\microsoft-word-90.png"));

break;

case ".xls":

case ".xlsx":

// файл Microsoft Excel

image.Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\microsoft-excel-90.png"));

break;

case ".pdf":

// файл Adobe Acrobat Reader

image.Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\pdf-90.png"));

break;

case ".jpg":

case ".jpeg":

case ".png":

case ".bmp":

image.Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\image-90.png"));

break;

case ".txt":

image.Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\txt-90.png"));

break;

default:

image.Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\file-90.png"));

break;

}

// Создаем textBlock

TextBlock textBlock = new TextBlock();

textBlock.Margin = new Thickness(5, 3, 0, 0);

textBlock.Inlines.Add(new Run(Path.GetFileName(Path1)));

textBlock.Foreground = Brushes.White;

textBlock.Inlines.Add(new InlineUIContainer() { Child = image });

textBlock.Cursor = new Cursor("D:\\studing\\4\_semestr\\Course\_project\\Cursor\\Red Neon\\normal\_select.cur");

// Создаем крестик

TextBlock closeButton = new TextBlock();

closeButton.Text = "x";

closeButton.Width = 16;

closeButton.Height = 16;

closeButton.Margin = new Thickness(2);

closeButton.Foreground = Brushes.White;

closeButton.Tag = (Path.GetFileName(Path1));

closeButton.MouseLeftButtonUp += CloseButton\_Click; // Подписываемся на событие клика мышки

closeButton.Cursor = new Cursor("D:\\studing\\4\_semestr\\Course\_project\\Cursor\\Red Neon\\link\_select.cur");

// Добавляем крестик рядом с textBlock

StackPanel container = new StackPanel();

container.Orientation = Orientation.Horizontal;

container.Children.Add(textBlock);

container.Children.Add(closeButton);

container.Name = "a" + (Path.GetFileName(Path1)).Replace(".", "");

textBl.Children.Add(container);

documents.Add(document);

}

}

catch(Exception)

{

MessageBox.Show("Ошибка при добавлении данного файла, попробуйте изменить название и попробовать ещё раз!");

}

}));

}

}

# ПРИЛОЖЕНИЕ Е

Листинг – Метод для открытия файлов

private void OpenFile(string fileName, byte[] fileContent)

{

// Создаем временный файл в папке TEMP

string tempFilePath = Path.Combine(Path.GetTempPath(), fileName);

File.WriteAllBytes(tempFilePath, fileContent);

// Определяем тип файла и открываем его

string extension = Path.GetExtension(fileName).ToLower();

switch (extension)

{

case ".doc":

case ".docx":

// Открываем файл с помощью Microsoft Word

var wordPath = @"C:\Program Files (x86)\Microsoft Office\root\Office16\WINWORD.EXE";

Process.Start(new ProcessStartInfo

{

FileName = wordPath,

Arguments = tempFilePath,

UseShellExecute = false,

RedirectStandardOutput = true,

CreateNoWindow = true

});

break;

case ".xls":

case ".xlsx":

// Открываем файл с помощью Microsoft Excel

var excelPath = @"C:\Program Files (x86)\Microsoft Office\root\Office16\EXCEL.EXE";

Process.Start(new ProcessStartInfo

{

FileName = excelPath,

Arguments = tempFilePath,

UseShellExecute = false,

RedirectStandardOutput = true,

CreateNoWindow = true

});

break;

case ".pdf":

// Открываем файл с помощью Adobe Acrobat Reader

var pdfPath = @"C:\Program Files (x86)\Adobe\Acrobat DC\Acrobat\Acrobat.exe";

Process.Start(new ProcessStartInfo

{

FileName = pdfPath,

Arguments = $"/A \"page=1\" \"{tempFilePath}\"",

UseShellExecute = false

});

break;

case ".jpg":

case ".jpeg":

case ".png":

case ".bmp":

BitmapImage bitmapImage = new BitmapImage();

using (MemoryStream ms = new MemoryStream(fileContent))

{

bitmapImage.BeginInit();

bitmapImage.CacheOption = BitmapCacheOption.OnLoad;

bitmapImage.StreamSource = ms;

bitmapImage.EndInit();

}

System.Windows.Controls.Image image = new System.Windows.Controls.Image();

image.Source = bitmapImage;

image.Stretch = Stretch.Uniform;

Window window = new Window();

window.Title = "Image";

window.Icon = new BitmapImage(new Uri(@"D:\studing\4\_semestr\Course\_project\image\logo.png"));

window.Background = Brushes.Transparent;

window.Content = image;

window.ShowDialog();

break;

case ".txt":

Process.Start("C:\\Users\\aleks\\Notepad++\\notepad++.exe", tempFilePath);

break;

default:

// Открываем файл с помощью проводника Windows

Process.Start("explorer.exe", $"/select,\"{tempFilePath}\"");

break;

}

}

# ПРИЛОЖЕНИЕ Ж

Листинг – Модель представления для окна авторизации

using Firebase.Auth;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media.Animation;

using System.Windows.Media;

using System.Windows.Threading;

using TaskWave.Commands;

using TaskWave.DataBase;

using TaskWave.Forms;

using System.Text;

using System.Security.Cryptography;

using TaskWave.Forms.MainWinManager;

using TaskWave.Classes;

using TaskWave.Forms.MainWinAdmin;

namespace TaskWave.Pages.StartModel.Auth

{

public class ViewModelAuth : InterfaceViewModel

{

public string[] arr = { "\u00A0Пароль", "\u00A0Логин" };

public event PropertyChangedEventHandler? PropertyChanged;

protected void OnPropertyChanged(string name)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

handler(this, new PropertyChangedEventArgs(name));

}

}

public ViewModelAuth()

{

myContext context = new myContext();

bool isRegistr = false;

foreach(var a in context.userIdentify)

{

if(a.type == "admin")

{

isRegistr = true;

break;

}

}

if(!isRegistr)

{

Classes.User user = new();

user.type = "admin";

user.name = "Aleksey\_admin";

user.login = "admin";

user.password = HashPassword("admin");

user.gmailURL = "alekseykravchenko120@gmail.com";

user.telegramURL = "https://t.me/A1ek7eey";

context.userIdentify.Add(user);

context.SaveChanges();

}

}

private string login = "\u00A0Логин", password = "\u00A0Пароль";

#region fields

public string Login

{

get { return login; }

set

{

login = value;

OnPropertyChanged("Login");

}

}

public string Password

{

get { return password; }

set

{

password = value;

OnPropertyChanged("Password");

}

}

#endregion

#region commands

private InputFocusCommand? \_inputFocus;

public InputFocusCommand InputFocus

{

get

{

if (\_inputFocus == null)

\_inputFocus = new InputFocusCommand();

return \_inputFocus;

}

}

private InputLostFocusCommand? \_inputLostFocus;

public InputLostFocusCommand InputLostFocus

{

get

{

if( \_inputLostFocus == null)

\_inputLostFocus = new InputLostFocusCommand();

return \_inputLostFocus;

}

}

private ClickRegisterCommand? \_clickRegister;

public ClickRegisterCommand ClickRegister

{

get

{

return \_clickRegister ?? (\_clickRegister = new ClickRegisterCommand(obj =>

{

Classes.NavigationService.\_frame.Navigate(new Registration());

}));

}

}

private LogInCommand? \_logIn;

public LogInCommand LogIn

{

get

{

return \_logIn ?? (\_logIn = new LogInCommand(obj =>

{

var textboxes = FindTextBoxes(Classes.NavigationService.panel3);

if (this.Login== "\u00A0Логин" || this.password == "\u00A0Пароль")

{

foreach (var textBox in textboxes)

{

int i = 0;

while (i < arr.Length)

{

if (arr[i] == textBox.Text)

{

SolidColorBrush redBrush = new SolidColorBrush(Colors.Red);

SolidColorBrush blackBrush = new SolidColorBrush(Color.FromArgb(128, 0, 0, 0));

// Создаем анимацию изменения цвета

ColorAnimation animation = new ColorAnimation();

animation.From = ((SolidColorBrush)textBox.Foreground).Color;

animation.To = redBrush.Color;

animation.Duration = new Duration(TimeSpan.FromSeconds(1));

// Запускаем анимацию

Storyboard storyboard = new Storyboard();

storyboard.Children.Add(animation);

Storyboard.SetTarget(animation, textBox);

Storyboard.SetTargetProperty(animation, new PropertyPath("(TextBox.Foreground).(SolidColorBrush.Color)"));

storyboard.Begin();

// Возвращаем цвет текста через 1 секунду

DispatcherTimer timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += (s, args) =>

{

textBox.Foreground = blackBrush;

timer.Stop();

};

timer.Start();

break;

}

i++;

}

}

}

else

{

myContext context = new myContext();

context.Database.EnsureCreated(); // Создаем базу данных, если ее нет

foreach (var a in context.userIdentify)

{

Classes.activeUser.passw = this.password;

if (a.login == login && a.password == HashPassword(password) && a.isRegister!=false)

{

myContext context2 = new myContext();

foreach (var b in context2.userIdentify)

{

if (b.login == this.login)

{

Classes.activeUser.user = b;

}

}

if(a.type == "standart")

{

MainWinUser us = new MainWinUser();

Classes.NavigationService.\_mainWindow.Close();

us.WindowStartupLocation = WindowStartupLocation.CenterScreen; // установка позиции запуска окна по центру экрана

us.Show();

return;

}

else if(a.type == "manager")

{

MainWindowManager us = new MainWindowManager();

Classes.NavigationService.\_mainWindow.Close();

us.WindowStartupLocation = WindowStartupLocation.CenterScreen; // установка позиции запуска окна по центру экрана

us.Show();

return;

}

else if (a.type == "admin")

{

MainWinAdmin us = new ();

Classes.NavigationService.\_mainWindow.Close();

us.WindowStartupLocation = WindowStartupLocation.CenterScreen; // установка позиции запуска окна по центру экрана

us.Show();

return;

}

}

}

TextBlock label = new TextBlock

{

Text = "Проверьте правильность ввода данных",

FontSize = 12,

FontFamily = new FontFamily("Montserrat"),

Margin = new Thickness(90, 0, 0, -8), // отступы

Foreground = Brushes.Red,

Name = "error"// цвет текста

};

//DoubleAnimation animation = new DoubleAnimation

//{

// From = 1.0,

// To = 0,

// Duration = TimeSpan.FromSeconds(2.5)

//};

var errorBlock2 = Classes.NavigationService.panel3.Children.OfType<TextBlock>().FirstOrDefault(b => b.Name == "error");

foreach (var textBox in textboxes)

{

if (textBox.Name == "inputLoginAuth" && errorBlock2 == null)

{

TextBox tb = textBox as TextBox;

Classes.NavigationService.panel3.Children.Insert(Classes.NavigationService.panel3.Children.IndexOf(tb), label);

}

}

//label.BeginAnimation(TextBlock.OpacityProperty, animation);

// Устанавливаем таймер на 3 секунды

DispatcherTimer timer2 = new DispatcherTimer();

timer2.Interval = TimeSpan.FromSeconds(3);

// создаем таймер

timer2.Tick += (sender, args) =>

{

// ищем TextBlock с именем "error" и удаляем его

var errorBlock = Classes.NavigationService.panel3.Children.OfType<TextBlock>().FirstOrDefault(b => b.Name == "error");

if (errorBlock != null)

{

// Удаляем TextBlock из родительского контейнера

Classes.NavigationService.panel3.Children.Remove(label);

}

timer2.Stop(); // останавливаем таймер

};

timer2.Start(); // запускаем таймер

return;

}

}));

}

}

private string HashPassword(string password)

{

using (var sha256 = SHA256.Create())

{

var hashedBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));

return Convert.ToBase64String(hashedBytes);

}

}

public static IEnumerable<TextBox> FindTextBoxes(DependencyObject parent)

{

List<TextBox> textboxes = new List<TextBox>();

for (int i = 0; i < VisualTreeHelper.GetChildrenCount(parent); i++)

{

var child = VisualTreeHelper.GetChild(parent, i);

if (child is TextBox textbox)

{

textboxes.Add(textbox);

}

else

{

textboxes.AddRange(FindTextBoxes(child));

}

}

return textboxes;

}

#endregion

}

}

# ПРИЛОЖЕНИЕ З

Листинг – Модель представления для окна регистрации

using Azure;

using Firebase.Auth;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Reflection.Metadata;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Media.Imaging;

using System.Windows.Threading;

using TaskWave.Commands;

using TaskWave.DataBase;

using TaskWave.Forms;

namespace TaskWave.Pages.StartModel.Register

{

public class ViewModelRegister : InterfaceViewModel

{

Classes.User user;

String[] arr =

{

"\u00A0Логин", "\u00A0Пароль", "\u00A0Имя", "\u00A0Почтовый адрес", "\u00A0Поделитесь информацией о вашей компании или проекте, чтобы мы могли познакомиться с вашей деятельностью более подробно..."

};

public event PropertyChangedEventHandler? PropertyChanged;

protected void OnPropertyChanged(string name)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

handler(this, new PropertyChangedEventArgs(name));

}

}

public ViewModelRegister(Classes.User u)

{

this.user = u;

user.description = "\u00A0Поделитесь информацией о вашей компании или проекте, чтобы мы могли познакомиться с вашей деятельностью более подробно...";

}

#region fileds

private string hiddenPassword = "\u00A0Пароль";

public string HiddenPassword

{

get { return hiddenPassword; }

set

{

hiddenPassword = value;

OnPropertyChanged("HiddenPassword");

}

}

public string FieldWithInf

{

get { return user.description; }

set

{

user.description = value;

OnPropertyChanged("FieldWithInf");

}

}

public string Login

{

get { return user.login; }

set {

user.login = value;

OnPropertyChanged(nameof(Login));

}

}

public string Password

{

get { return user.password; }

set

{

user.password = value;

OnPropertyChanged(nameof(Password));

}

}

public string Name

{

get { return user.name; }

set

{

user.name = value;

OnPropertyChanged(nameof(Name));

}

}

public string Gmail

{

get { return user.gmailURL; }

set

{

try

{

if (value == "")

{

user.gmailURL = "";

OnPropertyChanged(nameof(Gmail));

return;

}

if (value == "\u00A0Почтовый адрес")

{

user.gmailURL = "\u00A0Почтовый адрес";

OnPropertyChanged(nameof(Gmail));

return;

}

if (!IsValidEmail(value))

{

OnPropertyChanged(nameof(Gmail));

throw new Exception();

}

else

{

user.gmailURL = value;

}

OnPropertyChanged(nameof(Gmail));

}

catch(Exception ex)

{

MessageBox.Show("Вы ввели неверный адрес электронной почты");

user.gmailURL = "";

OnPropertyChanged(nameof(Gmail));

}

}

}

public string Telegram

{

get { return user.telegramURL; }

set

{

user.telegramURL = value;

OnPropertyChanged(nameof(Telegram));

}

}

#endregion

#region commands

public static bool IsValidEmail(string email)

{

string pattern = @"^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$";

Regex regex = new Regex(pattern);

return regex.IsMatch(email);

}

private BackToAuth \_backToAuth;

public BackToAuth BackToAuth

{

get

{

if(\_backToAuth == null)

\_backToAuth = new BackToAuth();

return \_backToAuth;

}

}

private InputFocusCommand \_inputFocusCommand;

public InputFocusCommand InputFocusCommand

{

get

{

if(\_inputFocusCommand == null)

\_inputFocusCommand = new InputFocusCommand();

return \_inputFocusCommand;

}

}

private InputLostFocusCommand \_inputLostFocus;

public InputLostFocusCommand InputLostFocus

{

get

{

if(\_inputLostFocus == null)

\_inputLostFocus = new InputLostFocusCommand();

return \_inputLostFocus;

}

}

private AddFieldForManagerCommand \_addFieldForManagerCommand;

public AddFieldForManagerCommand addFieldForManagerCommand

{

get

{

if (\_addFieldForManagerCommand == null)

\_addFieldForManagerCommand = new AddFieldForManagerCommand();

return \_addFieldForManagerCommand;

}

}

private RemoveFieldManCommand \_removeFieldManCommand;

public RemoveFieldManCommand RemoveFieldManCommand

{

get {

if(\_removeFieldManCommand == null)

\_removeFieldManCommand = new RemoveFieldManCommand();

return \_removeFieldManCommand;

}

}

private ColorIs? \_colorIs;

public ColorIs ColorIs

{

get

{

if(\_colorIs == null)

\_colorIs = new ColorIs();

return \_colorIs;

}

}

private RegistrationCommand \_registr;

public RegistrationCommand Registr

{

get

{

return \_registr ?? (\_registr = new RegistrationCommand(obj =>

{

if (user.login == "\u00A0Логин" || user.password == "\u00A0Пароль" || user.name == "\u00A0Имя" ||

user.gmailURL == "\u00A0Почтовый адрес")

{

var textboxes = FindTextBoxes(Classes.NavigationService.panel);

foreach (var textBox in textboxes)

{

int i = 0;

while (i < arr.Length)

{

if (arr[i] == textBox.Text)

{

SolidColorBrush redBrush = new SolidColorBrush(Colors.Red);

SolidColorBrush blackBrush = new SolidColorBrush(Color.FromArgb(128, 0, 0, 0));

// Создаем анимацию изменения цвета

ColorAnimation animation = new ColorAnimation();

animation.From = ((SolidColorBrush)textBox.Foreground).Color;

animation.To = redBrush.Color;

animation.Duration = new Duration(TimeSpan.FromSeconds(1));

// Запускаем анимацию

Storyboard storyboard = new Storyboard();

storyboard.Children.Add(animation);

Storyboard.SetTarget(animation, textBox);

Storyboard.SetTargetProperty(animation, new PropertyPath("(TextBox.Foreground).(SolidColorBrush.Color)"));

storyboard.Begin();

// Возвращаем цвет текста через 1 секунду

DispatcherTimer timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += (s, args) =>

{

textBox.Foreground = blackBrush;

timer.Stop();

};

timer.Start();

break;

}

i++;

}

}

}

else

{

user.type = "standart";

UserRepository repo = new();

myContext cont = new();

myContext context = new();

context.Database.EnsureCreated(); // Создаем базу данных, если ее нет

foreach (var a in cont.userIdentify)

{

if(a.login == user.login)

{

TextBlock label = new TextBlock

{

Text = "Пользователь с таким login-ом уже существует",

FontSize = 12,

FontFamily = new FontFamily("Montserrat"),

Margin = new Thickness(75, 0, 0, -8), // отступы

Foreground = Brushes.Red,

Name = "error"// цвет текста

};

//DoubleAnimation animation = new DoubleAnimation

//{

// From = 1.0,

// To = 0,

// Duration = TimeSpan.FromSeconds(2.5)

//};

var textboxes = FindTextBoxes(Classes.NavigationService.panel);

var errorBlock2 = Classes.NavigationService.panel.Children.OfType<TextBlock>().FirstOrDefault(b => b.Name == "error");

foreach (var textBox in textboxes)

{

if(textBox.Name == "inputLoginRegister" && errorBlock2 == null)

{

TextBox tb = textBox as TextBox;

Classes.NavigationService.panel.Children.Insert(Classes.NavigationService.panel.Children.IndexOf(tb), label);

}

}

//label.BeginAnimation(TextBlock.OpacityProperty, animation);

// Устанавливаем таймер на 3 секунды

DispatcherTimer timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(3);

// создаем таймер

timer.Tick += (sender, args) =>

{

// ищем TextBlock с именем "error" и удаляем его

var errorBlock = Classes.NavigationService.panel.Children.OfType<TextBlock>().FirstOrDefault(b => b.Name == "error");

if (errorBlock != null)

{

// Удаляем TextBlock из родительского контейнера

Classes.NavigationService.panel.Children.Remove(label);

}

timer.Stop(); // останавливаем таймер

};

timer.Start(); // запускаем таймер

return;

}

}

repo.AddUser(user);

Classes.activeUser.user = user;

Classes.activeUser.passw = user.password;

Classes.NavigationService.registrUser = new Classes.User("\u00A0Логин", "\u00A0Пароль", "\u00A0Имя", "none", "\u00A0Имя\_пользователя telegram", "\u00A0Почтовый адрес", "-");

user = Classes.NavigationService.registrUser;

Forms.MainWinUser win = new();

Classes.NavigationService.\_mainWindow.Close();

win.WindowStartupLocation = WindowStartupLocation.CenterScreen; // установка позиции запуска окна по центру экрана

win.Show();

}

}));

}

}

private RegistrationCommand \_registrManager;

public RegistrationCommand RegistrManager

{

get

{

return \_registrManager ?? (\_registrManager = new RegistrationCommand(obj =>

{

if (user.login == "\u00A0Логин" || user.password == "\u00A0Пароль" || user.name == "\u00A0Имя" ||

user.gmailURL == "\u00A0Почтовый адрес" || user.description == "\u00A0Поделитесь информацией о вашей компании или проекте, чтобы мы могли познакомиться с вашей деятельностью более подробно...")

{

var textboxes = FindTextBoxes(Classes.NavigationService.panel2);

foreach (var textBox in textboxes)

{

int i = 0;

while (i < arr.Length)

{

if (arr[i] == textBox.Text)

{

SolidColorBrush redBrush = new SolidColorBrush(Colors.Red);

SolidColorBrush blackBrush = new SolidColorBrush(Color.FromArgb(128, 0, 0, 0));

// Создаем анимацию изменения цвета

ColorAnimation animation = new ColorAnimation();

animation.From = ((SolidColorBrush)textBox.Foreground).Color;

animation.To = redBrush.Color;

animation.Duration = new Duration(TimeSpan.FromSeconds(1));

// Запускаем анимацию

Storyboard storyboard = new Storyboard();

storyboard.Children.Add(animation);

Storyboard.SetTarget(animation, textBox);

Storyboard.SetTargetProperty(animation, new PropertyPath("(TextBox.Foreground).(SolidColorBrush.Color)"));

storyboard.Begin();

// Возвращаем цвет текста через 1 секунду

DispatcherTimer timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += (s, args) =>

{

textBox.Foreground = blackBrush;

timer.Stop();

};

timer.Start();

break;

}

i++;

}

}

}

else

{

user.type = "manager";

UserRepository repo = new();

myContext cont = new();

myContext context = new();

context.Database.EnsureCreated(); // Создаем базу данных, если ее нет

foreach (var a in cont.userIdentify)

{

if (a.login == user.login)

{

TextBlock label = new TextBlock

{

Text = "Пользователь с таким login-ом уже существует",

FontSize = 12,

FontFamily = new FontFamily("Montserrat"),

Margin = new Thickness(75, 0, 0, -8), // отступы

Foreground = Brushes.Red,

Name = "error"// цвет текста

};

var textboxes = FindTextBoxes(Classes.NavigationService.panel2);

var errorBlock2 = Classes.NavigationService.panel2.Children.OfType<TextBlock>().FirstOrDefault(b => b.Name == "error");

foreach (var textBox in textboxes)

{

if (textBox.Name == "inputLoginRegister" && errorBlock2 == null)

{

TextBox tb = textBox as TextBox;

Classes.NavigationService.panel2.Children.Insert(Classes.NavigationService.panel2.Children.IndexOf(tb), label);

}

}

// Устанавливаем таймер на 3 секунды

DispatcherTimer timer2 = new DispatcherTimer();

timer2.Interval = TimeSpan.FromSeconds(3);

// создаем таймер

timer2.Tick += (sender, args) =>

{

// ищем TextBlock с именем "error" и удаляем его

var errorBlock = Classes.NavigationService.panel2.Children.OfType<TextBlock>().FirstOrDefault(b => b.Name == "error");

if (errorBlock != null)

{

// Удаляем TextBlock из родительского контейнера

Classes.NavigationService.panel2.Children.Remove(label);

}

timer2.Stop(); // останавливаем таймер

};

timer2.Start(); // запускаем таймер

return;

}

}

user.isRegister = false;

repo.AddUser(user);

Window dialog;

dialog = new Window

{

Title = "Ввод информации",

MinWidth = 500,

MinHeight = 315,

MaxHeight = 315,

MaxWidth = 500,

Cursor = new System.Windows.Input.Cursor("D:\\studing\\4\_semestr\\Course\_project\\Cursor\\Red Neon\\normal\_select.cur"),

WindowStartupLocation = WindowStartupLocation.CenterOwner,

};

dialog.Content = CreateDialogContent(dialog);

dialog.ShowDialog();

Classes.NavigationService.registrUser = new Classes.User("\u00A0Логин", "\u00A0Пароль", "\u00A0Имя", "none", "\u00A0Имя\_пользователя telegram", "\u00A0Почтовый адрес", "-");

user = Classes.NavigationService.registrUser;

Classes.NavigationService.\_frame.Navigate(new Authorization());

//Classes.activeUser.user = user;

//Classes.activeUser.passw = user.password;

//Classes.NavigationService.registrUser = new Classes.User("\u00A0Логин", "\u00A0\u00A0Пароль", "\u00A0Имя", "none", "\u00A0Имя\_пользователя telegram", "\u00A0Почтовый адрес", "-");

//user = Classes.NavigationService.registrUser;

//Forms.MainWinManager.MainWindowManager win = new();

//Classes.NavigationService.\_mainWindow.Close();

//win.WindowStartupLocation = WindowStartupLocation.CenterScreen; // установка позиции запуска окна по центру экрана

//win.Show();

}

}));

}

}

private StackPanel CreateDialogContent(Window win)

{

var stackPanel = new StackPanel() { Width = 485, Height = 300, Background = new SolidColorBrush((System.Windows.Media.Color)System.Windows.Media.ColorConverter.ConvertFromString("#444C5C")) };

var image = new System.Windows.Controls.Image { Width = 70, VerticalAlignment = VerticalAlignment.Top, HorizontalAlignment = HorizontalAlignment.Right, Margin = new Thickness(5), Source = new BitmapImage(new Uri("D:\\studing\\4\_semestr\\Course\_project\\image\\book-registr.png", UriKind.RelativeOrAbsolute)) };

var border = new Border { CornerRadius = new CornerRadius(7), Background = System.Windows.Media.Brushes.White, Margin = new Thickness(5), Height = 100 };

var textBlock = new TextBlock { Margin = new Thickness(5),TextWrapping = TextWrapping.Wrap, Text= "Уважаемый "+user.name +", для подтверждения вашего статуса руководителя совместных проектов, мы отправили ваш запрос администрации. Они свяжутся с вами в ближайшее время." };

var button = new Button { Style = (Style)Application.Current.Resources["StartBtn"], Content = "Ок" };

button.Click += (sender, e) =>

{

win.Close();

};

stackPanel.Children.Add(image);

stackPanel.Children.Add(border);

border.Child = textBlock;

stackPanel.Children.Add(button);

return stackPanel;

}

public static IEnumerable<TextBox> FindTextBoxes(DependencyObject parent)

{

List<TextBox> textboxes = new List<TextBox>();

for (int i = 0; i < VisualTreeHelper.GetChildrenCount(parent); i++)

{

var child = VisualTreeHelper.GetChild(parent, i);

if (child is TextBox textbox)

{

textboxes.Add(textbox);

}

else

{

textboxes.AddRange(FindTextBoxes(child));

}

}

return textboxes;

}

#endregion

}

}