

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий  
Кафедра Программной инженерии  
Специальность 6-05-0612-01 «Программная инженерия»  
Направление специальности 6-05-0612-01 «Программная инженерия»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии  
программирования и стандарты проектирования»  
Тема «Система управления проектами»

Исполнитель  
Студентка 2 курса группы 6 Филипюк Илья Андреевич  
(Ф.И.О.)

Руководитель работы Мущук А. Н.  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой \_\_\_\_\_  
Председатель Смелов В.В.  
(подпись)

Минск 2025

## Оглавление

<b>ВВЕДЕНИЕ</b> .....	3
<b>1 Аналитический обзор прототипов и литературных источников</b> .....	4
1.1 Обзор аналогичных решений.....	4
1.2 Постановка задачи .....	7
<b>2 Анализ требований к программному средству и разработка функциональных требований</b>	8
2.1 Описание средств разработки.....	8
2.1.1 Язык программирования C#.....	9
2.1.2 Платформа .NET Framework .....	9
2.1.3 Entity Framework .....	9
2.1.4 Windows Presentation Foundation .....	10
2.1.5 MS SQL Server .....	11
2.2 Спецификация функциональных требований к программному средству .....	11
2.3 Диаграмма вариантов использования .....	11
<b>3 Проектирование программного средства</b> .....	12
3.1 Проектирование архитектуры приложения .....	13
3.2 Проектирование структуры базы данных .....	14
3.3 Структура проекта.....	16
3.4 Диаграмма классов .....	17
<b>4 Реализация программного средства</b> .....	17
4.1 Основные классы программного средства .....	18
4.1.1 Классы для реализации паттерна MVVM.....	18
<b>5. Тестирование, проверка на работоспособность, анализ полученных результатов</b> .....	19

## **ВВЕДЕНИЕ**

Тема разработки приложения «Система управления проектами» является актуальной и весьма востребованной в современном мире. В условиях динамичного развития бизнеса и роста числа проектов эффективное управление задачами, сроками и ресурсами становится ключевым фактором успешной работы команд и компаний. Автоматизация процессов планирования, контроля и взаимодействия между участниками проекта способствует повышению продуктивности, оптимизации рабочих процессов и достижению поставленных целей.

Анализ конкурентов позволяет выявить сильные и слабые стороны существующих приложений. В условиях усиливающейся конкуренции и возрастающих требований пользователей возникает потребность в инновационных и более удобных решениях, способных эффективно удовлетворять потребности клиентов и помогать им в достижении их целей.

Целью данного курсового проекта является разработка приложения для управления проектами, которое поможет пользователям эффективно планировать, отслеживать и организовывать свою работу. Основная задача приложения — предоставление удобного и интуитивно понятного инструмента для распределения задач и мониторинга прогресса, что позволит оптимизировать процессы и повысить продуктивность работы.

При разработке приложения для управления проектами учитываются несколько ключевых принципов, способствующих достижению поставленных целей. Во-первых, интерфейс должен быть интуитивно понятным и удобным, чтобы пользователи легко ориентировались в системе, независимо от их опыта работы с подобными инструментами. Во-вторых, приложение должно обеспечивать оперативное и эффективное управление задачами, сроками и ресурсами, позволяя пользователю оптимизировать рабочие процессы и минимизировать затраты времени.

После разработки и внедрения приложения для управления проектами пользователи смогут эффективно организовывать рабочие процессы, планировать задачи и контролировать сроки их выполнения. Это позволит оптимизировать управление проектами, повысить продуктивность и улучшить качество выполнения работ.

# 1 Аналитический обзор прототипов и литературных источников

## 1.1 Обзор аналогичных решений

Перед началом разработки собственного программного приложения важно провести детальный анализ существующих аналогов. Это позволит глубже понять потребности целевой аудитории, выявить актуальные тенденции и определить возможные пробелы на рынке. Исследование конкурентных решений помогает определить наиболее востребованные функции, а также выявить возможности для улучшения и внедрения инновационных подходов. Кроме того, анализ стратегий конкурентов, их методов привлечения клиентов и уровня обслуживания дает возможность сформировать уникальное предложение, которое будет выгодно отличаться от существующих решений. Такой подход повышает вероятность успешного запуска приложения и обеспечивает его соответствие ожиданиям пользователей.

В качестве первого аналога будет рассмотрено приложение «YouGile». На рисунке 1.1 представлен сайт системы управления проектами «YouGile»

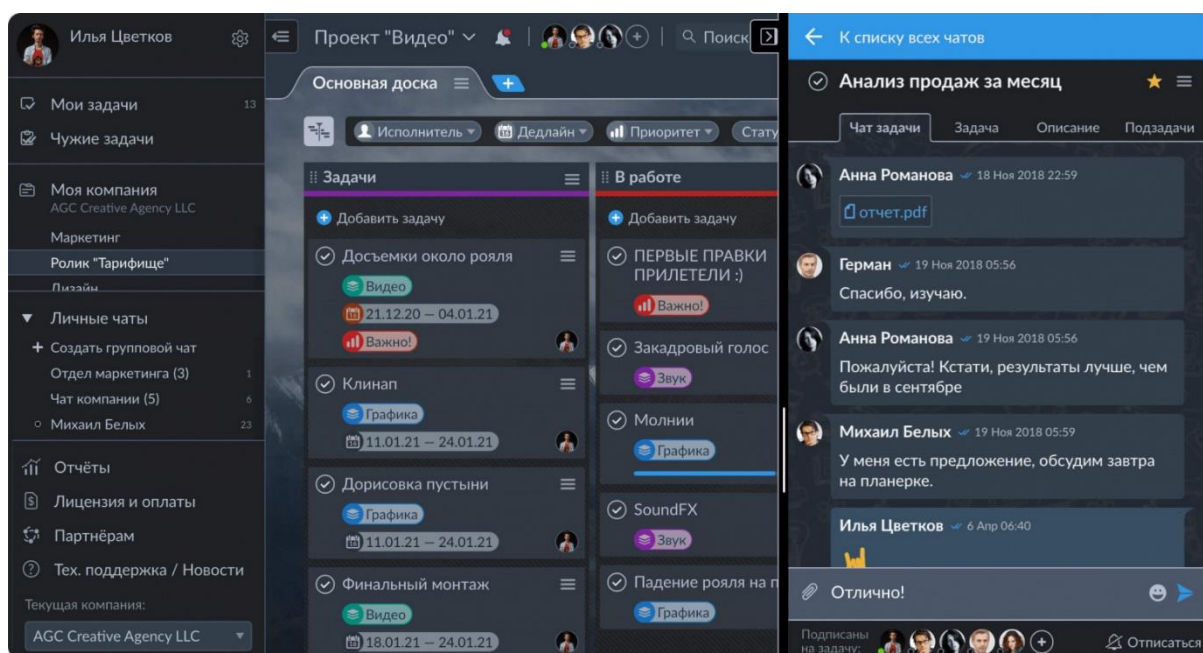


Рисунок 1.1 – Сайт системы управления проектами «YouGile»

Система не предоставляет возможности выбора методологии разработки, автоматически используя сочетание Agile и Scrum.

На рисунке 1.2 представлена страница вид доски задач.

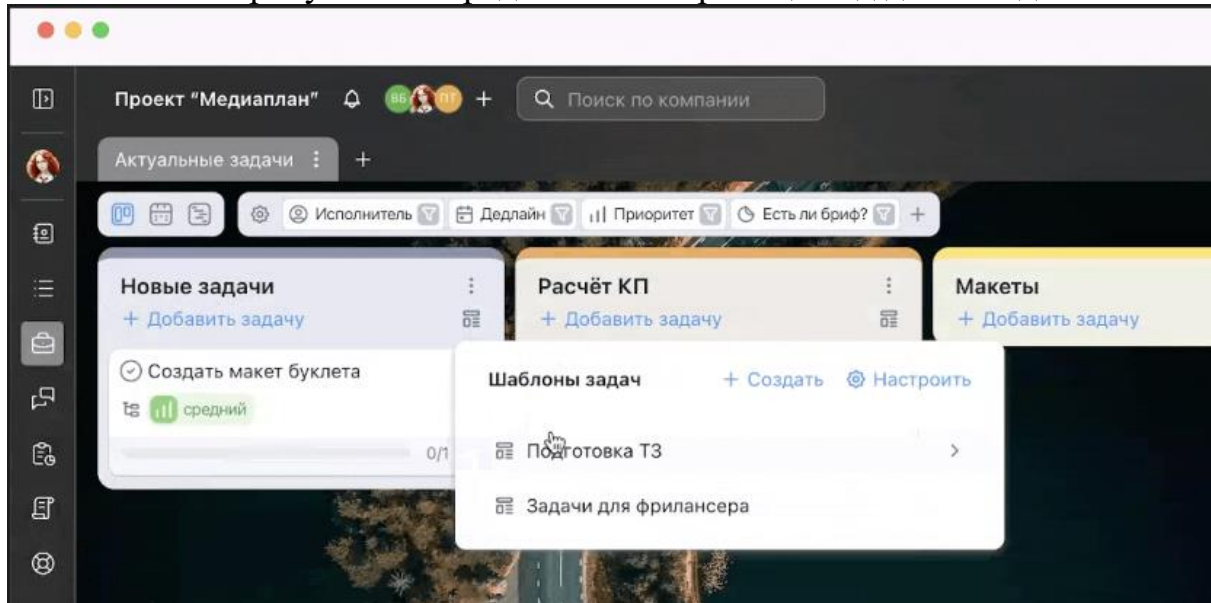


Рисунок 1.2 – Вид доски задач

Интерфейс приложения интуитивно понятен и легко настраивается под пользователя. Он самостоятельно определяет статусы для задач, однако этот процесс занимает некоторое время, которое можно было бы сократить, если бы была встроенная возможность выбора методологии разработки. Приоритеты задач определяются просто и логично, после чего они сортируются в соответствии с установленными значениями, что делает работу более удобной.

Взаимодействие с задачами в приложении представлено широкими возможностями. Пользователи могут создавать шаблоны для часто встречающихся задач, быстро добавлять новые, изменять статусы и удалять ненужные. Система категоризации задач удобна, однако отслеживание прогресса их выполнения организовано не самым удобным образом, что иногда приводит к задержкам в соблюдении сроков.

Отчётность в приложении отсутствует, что может создавать определённые сложности при анализе выполненных работ. Однако удобство использования компенсирует этот недостаток: интерфейс остаётся простым и интуитивно понятным, а визуализация рабочего процесса помогает легче ориентироваться в задачах.

К преимуществам можно отнести удобный интерфейс, возможность структурирования работы, большое количество встроенных функций и лёгкость в освоении и внедрении. Однако существуют и недостатки: система замкнута и не поддерживает интеграцию со сторонними сервисами и

облачными хранилищами. Пользователи не могут создавать собственные категории и вынуждены использовать стандартные, а также отсутствует возможность формирования отчетности, что ограничивает функциональность приложения.

Вторым аналогом будет приложение «Asana». Asana — это система управления проектами, которая предоставляет пользователям широкий функционал для эффективной организации работы. В приложении можно добавлять исполнителей и назначать им конкретные задачи, а также привлекать наблюдателей, которые смогут следить за ходом выполнения работ. Система предлагает расширенную бизнес-отчётность и возможность интеграции со сторонними сервисами, что делает её удобной для различных команд и проектов.

На рисунке 1.3 представлен вид главной страницы приложения.

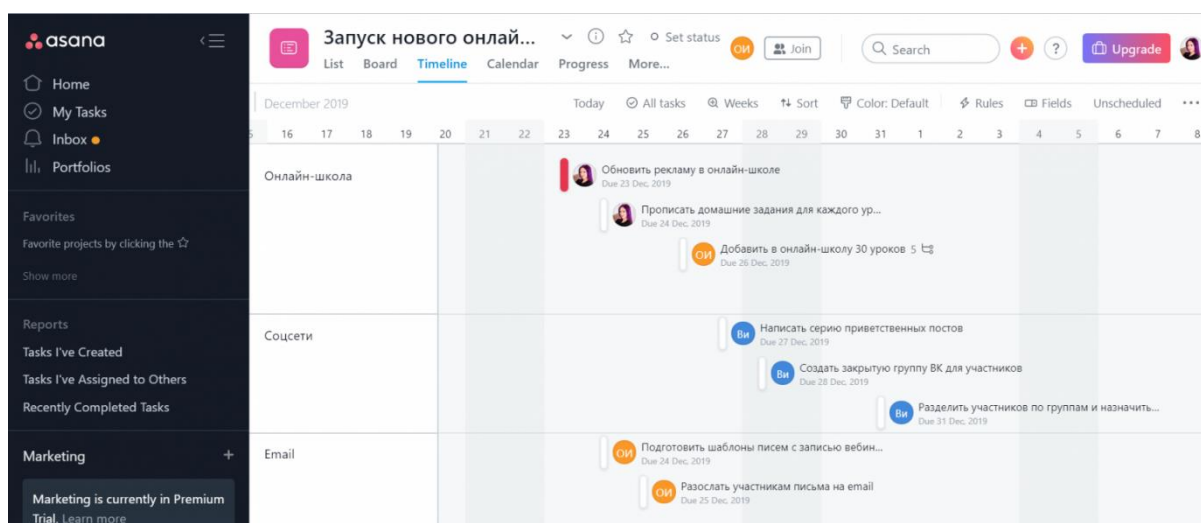


Рисунок 1.3 – Главная страница системы управления проектам «Asana»

Интерфейс Asana отличается простотой и удобством, не перегружен лишними элементами, что способствует легкому освоению. Интеграция в команду проходит без сложностей благодаря логичной структуризации интерфейса и удобному расположению элементов.

Среди преимуществ платформы можно выделить широкий функционал для исполнителей и наблюдателей, возможность интеграции с облачными хранилищами и сторонними сервисами, а также универсальность — Asana подходит как для крупных, так и для небольших команд. Однако у системы есть и недостатки: отсутствует иерархия сотрудников, что может быть неудобно для больших коллективов, сложность в расстановке приоритетов для задач и проектов, а также недостаточная структуризация задач при их большом количестве.

Третий аналог – «Jira». Это система управления проектами, которая предлагает пользователям широкий функционал для эффективного планирования и контроля задач. В ней отсутствуют ограничения по количеству пользователей, что делает её удобной для команд любого размера. Приложение позволяет выбирать методологию разработки, использовать мощный API, рассчитывать скорость выполнения задач и отслеживать прогресс работы.

На рисунке 1.4 представлена главная страница приложения «Jira».

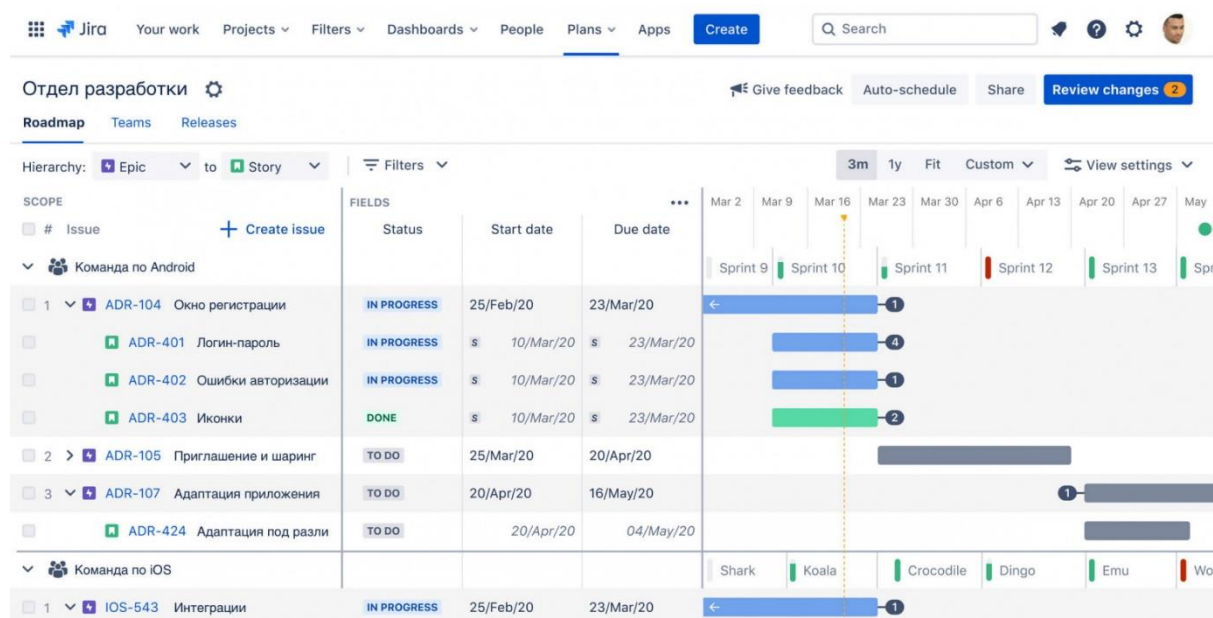


Рисунок 1.4– Система управления проектами «Jira»

Интерфейс Jira достаточно загружен, что может потребовать дополнительного времени на обучение, особенно для нетехнических команд. Частые изменения интерфейса требуют привыкания, но при этом система предлагает удобные инструменты для отслеживания сроков, хорошую структуризацию задач и проектов, а также продуманную систему иерархии сотрудников.

Среди преимуществ Jira можно выделить её универсальность, удобную визуализацию, широкий функционал и возможность интеграции со сторонними сервисами. Однако у системы есть и недостатки: сложный интерфейс, трудности с настройкой рабочих процессов и ограниченные возможности автоматизации. Несмотря на это, Jira остаётся популярным инструментом для управления проектами, особенно среди разработчиков и команд управления.

## 1.2 Постановка задачи

Исходя из анализа аналогичных решений из пункта 1.1, проектируемое программное средство должно обладать следующими характеристиками и функциональными возможностями:



- Интуитивно понятная навигация, чтобы пользователь смог проще и быстрее ориентироваться в приложении;
- Оптимальный стиль и размер шрифта, текст должен быть легко читаемым, без визуального напряжения;
- Цветовая палитра приложения не должна содержать много ярких цветов, чтобы не отвлекать пользователя;
- Настройка рабочих процессов в проектах должна быть простой и понятной;
- Задачи и проекты должны быть хорошо структурированы и отсортированы по важности (По ближайшим срокам, например);
- Должна быть возможность выбора методологии разработки по проекту.

Реализация этих функций обеспечит высокую продуктивность работы и положительный пользовательский опыт, способствуя успешному выполнению проектов.

## **2 Анализ требований к программному средству и разработка функциональных требований**

Анализ требований — это ключевой этап разработки программного обеспечения, определяющий, какие функции и возможности должны быть реализованы для удовлетворения потребностей пользователей и заказчиков. В ходе этого процесса проводится сбор, документирование, уточнение и систематизация требований, что помогает избежать недоразумений, ошибок и несоответствий на последующих этапах разработки.

Формирование функциональных требований играет важную роль, поскольку именно они определяют, какие возможности будут доступны пользователям при работе с приложением. Эти требования служат основой для проектирования архитектуры системы, ее дальнейшей реализации, а также являются критерием для тестирования и оценки готовности продукта.

В данном разделе пояснительной записки будет представлена информация о технических средствах, применяемых в процессе разработки, а также описание архитектуры системы.

### **2.1 Описание средств разработки**

Для создания приложения «Система управления проектами» были применены современные технологии и инструменты, обеспечивающие удобство использования, высокую производительность и надежность системы. В процессе разработки использовался язык программирования C# и платформа .NET Framework, а также технология Windows Presentation Foundation (WPF), предназначенная для построения клиентских приложений с графическим интерфейсом.



Для управления базами данных была выбрана реляционная система MS SQL Server, которая гарантирует высокую скорость обработки данных, безопасность и надежность, особенно при работе с большими объемами информации. Взаимодействие приложения с базой данных осуществляется через фреймворк Entity Framework, позволяющий разработчикам работать с данными, используя .NET-объекты, что упрощает процесс разработки и управления данными.

### **2.1.1 Язык программирования C#**

C# — это современный объектно-ориентированный язык программирования, созданный компанией Microsoft. Он широко используется для разработки приложений на платформе .NET, что делает его востребованным инструментом для создания веб-, мобильных и настольных программ. Благодаря компиляции в промежуточный язык C# обеспечивает высокую производительность. Его синтаксис прост и понятен, что значительно облегчает процесс написания и сопровождения кода. Кроме того, язык легко интегрируется с различными библиотеками и фреймворками, такими как ASP.NET и Entity Framework, что ускоряет разработку и расширяет функциональные возможности приложений.

### **2.1.2 Платформа .NET Framework**

.NET Framework — это программная платформа, созданная Microsoft, предназначенная для разработки и запуска приложений, преимущественно на операционной системе Windows. Она включает в себя среду выполнения, обеспечивающую работу программ, а также набор библиотек, необходимых для их создания. В состав .NET входят ключевые компоненты, такие как общая языковая среда (Common Language Runtime — CLR), библиотека классов .NET Framework и различные инструменты разработки, которые помогают программистам эффективно реализовывать свои проекты.

### **2.1.3 Entity Framework**

Entity Framework — это мощный инструмент для работы с базами данных в среде .NET, который значительно упрощает взаимодействие с данными. Он позволяет разработчикам работать с базами данных через объектно-ориентированную модель, избавляя их от необходимости писать сложные SQL-запросы. Благодаря поддержке автоматического создания и обновления схемы базы данных, разработчики могут легко вносить изменения в структуру данных без риска потери информации. Entity Framework совместим с различными системами управления базами данных, такими как MS SQL Server, PostgreSQL и MySQL, что делает его универсальным решением для различных проектов. Гибкость и расширяемость фреймворка позволяют

настраивать модели данных и использовать различные стратегии загрузки информации, что делает работу с ним удобной и эффективной.

Кроме того, Entity Framework поддерживает несколько режимов работы с данными, включая ленивую и жадную загрузку, что позволяет оптимизировать производительность приложения. Он также предоставляет мощные инструменты для работы с запросами, позволяя использовать LINQ для удобного и гибкого взаимодействия с данными. Благодаря встроенной поддержке миграций разработчики могут легко управлять изменениями в структуре базы данных, упрощая процесс обновления приложения.

Entity Framework широко применяется в корпоративных решениях, веб-приложениях и облачных сервисах, обеспечивая надежное и удобное управление данными. Его использование позволяет сократить время разработки, минимизировать ошибки при работе с базами данных и повысить общую эффективность программных решений.

#### **2.1.4 Windows Presentation Foundation**

Windows Presentation Foundation (WPF) — это мощная технология для создания графических интерфейсов приложений в среде .NET. Она разработана компанией Microsoft и предназначена для построения современных настольных приложений с богатым визуальным оформлением и интерактивными элементами.

Одной из ключевых особенностей WPF является использование языка разметки XAML, который позволяет описывать интерфейсы в декларативной форме, упрощая процесс разработки и отделяя визуальную часть приложения от логики. Благодаря этому разработчики могут легко изменять дизайн без необходимости вносить изменения в код программной части.

WPF поддерживает работу с двумерной и трехмерной графикой, анимацией, мультимедиа и стилями, что делает его мощным инструментом для создания сложных пользовательских интерфейсов. Важной особенностью является привязка данных, которая позволяет динамически обновлять элементы интерфейса в зависимости от изменений в данных.

Кроме того, WPF поддерживает привязку данных, что упрощает взаимодействие между пользовательским интерфейсом и бизнес-логикой. Это позволяет разработчикам легко обновлять интерфейсы в ответ на изменения данных, создавая более динамичные и отзывчивые приложения.

В целом, WPF — это универсальная платформа, которая сочетает в себе мощные инструменты для разработки и возможности для создания красивых и функциональных приложений, соответствующих современным требованиям пользователей.

### **2.1.5 MS SQL Server**

MS SQL Server — это реляционная система управления базами данных (СУБД), разработанная компанией Microsoft. Она предназначена для хранения, обработки и управления данными, обеспечивая высокую производительность, безопасность и масштабируемость.

Одной из ключевых особенностей MS SQL Server является поддержка транзакционной обработки данных, что гарантирует целостность и надежность информации. Система предлагает мощные инструменты для работы с запросами, включая язык SQL и расширенные возможности оптимизации выполнения запросов.

Кроме того, MS SQL Server предлагает широкий спектр инструментов для аналитики и обработки данных, включая поддержку хранимых процедур, триггеров и индексов, что позволяет оптимизировать работу с информацией. Система также включает средства мониторинга и управления производительностью, что делает её удобной для администраторов баз данных.

## **2.2 Спецификация функциональных требований к программному средству**

Программное средство должно предоставлять следующие функциональные возможности:

Для администратора:

- Выполнять авторизацию.
- Просматривать каталог товаров.
- Выполнять поисковые запросы, фильтрацию.
- Добавление/удаление новых проектов, пользователей, задач

Для клиента:

- Выполнять регистрацию и авторизацию.
- Свои задачи, проекты
- Выполнять поисковые запросы, фильтрацию.
- Взаимодействовать с доской задач

## **2.3 Диаграмма вариантов использования**

Диаграмма вариантов использования (Use Case Diagram) — это графический инструмент, предназначенный для моделирования функциональности системы с точки зрения её пользователей. Она позволяет определить различные сценарии взаимодействия, включая связь между пользователями (актерами) и самой системой.

Диаграмма включает в себя актеров, варианты использования (Use Case) и связи между ними. Актеры представляют собой пользователей или внешние системы, а варианты использования отражают доступные функциональные возможности. Визуально актеры изображаются в виде фигур, а варианты использования — в форме овалов, соединённых стрелками, обозначающими направление взаимодействия.

Использование диаграммы вариантов позволяет:

- визуализировать все возможные сценарии работы системы и понять, как пользователи взаимодействуют с ней;
- определить функциональные требования на основе реальных сценариев использования;
- обеспечить эффективное общение между разработчиками, заказчиками и другими заинтересованными сторонами.

Диаграмма вариантов использования для приложения «Система управления проектами» представлена в приложении А.

### **3 Проектирование программного средства**

Проектирование программного средства — это важный этап разработки, который включает в себя определение архитектуры, структуры и функциональности будущего приложения. На этом этапе разрабатываются модели данных, интерфейсы, алгоритмы и механизмы взаимодействия компонентов системы.

Процесс проектирования начинается с анализа требований, который позволяет определить ключевые функции и возможности программного средства. Затем создаются диаграммы, описывающие структуру системы, включая диаграммы классов, вариантов использования и последовательностей.

Выбор архитектуры играет решающую роль в проектировании, поскольку он определяет, как компоненты приложения будут взаимодействовать друг с другом. Важно учитывать принципы масштабируемости, надежности и безопасности, чтобы обеспечить стабильную работу системы.

Кроме того, проектирование включает разработку пользовательского интерфейса, который должен быть удобным и интуитивно понятным. Оптимизация взаимодействия с пользователем и продуманная навигация помогают повысить эффективность работы с приложением.

### 3.1 Проектирование архитектуры приложения

Архитектура программного обеспечения представляет собой совокупность ключевых решений, определяющих структуру и организацию системы. Она включает выбор компонентов, их интерфейсов, а также способы взаимодействия и интеграции в более крупные программные комплексы. Важную роль в архитектуре играет архитектурный стиль, который задает общую структуру системы, определяя элементы, их связи и принципы взаимодействия.

Для обеспечения высокого качества проектируемой системы применяются различные архитектурные паттерны. В данном проекте используется шаблон Model-View-ViewModel (MVVM), который включает три основных компонента: модель (Model), модель представления (ViewModel) и представление (View). Этот подход способствует четкому разделению логики приложения, обеспечивая удобство разработки, тестирования и поддержки.

Использование архитектурного шаблона MVVM способствует повышению модульности и облегчает повторное использование кода, а также упрощает процесс разработки и тестирования. Благодаря четкому разделению бизнес-логики и пользовательского интерфейса система становится более гибкой и масштабируемой, что позволяет ей легко адаптироваться к изменяющимся требованиям и расширяться в будущем.

На рисунке 3.1 представлена структура архитектуры MVVM.

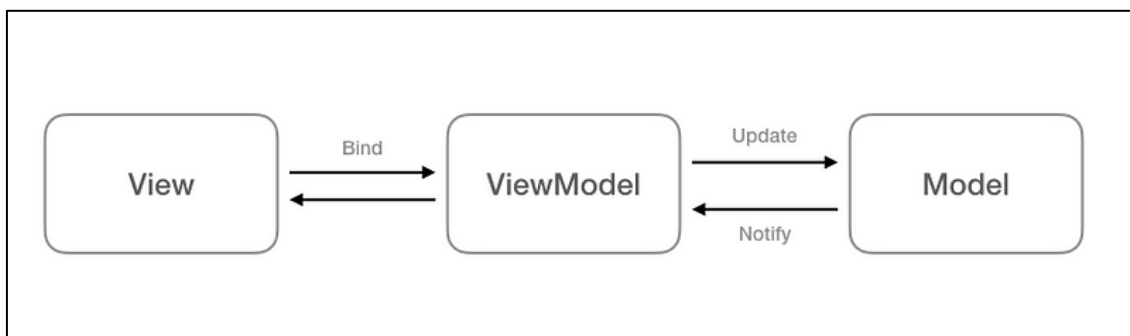


Рисунок 3.1 – Структура MVVM

**Model** — это компонент, который отвечает за работу с данными. Он представляет собой бизнес-логику приложения, включая обработку информации, взаимодействие с базой данных и выполнение вычислений. Model не зависит от пользовательского интерфейса и может быть использован в различных приложениях без изменений.

**View** — это визуальное представление данных, которое отображает информацию пользователю. Оно включает элементы интерфейса, такие как

кнопки, текстовые поля и списки. View реагирует на действия пользователя, но не содержит логики обработки данных — вся обработка выполняется в ViewModel.

**ViewModel** — это связующее звено между Model и View. Оно управляет данными, полученными из Model, и подготавливает их для отображения в View. ViewModel также обрабатывает пользовательские команды и обновляет данные в Model. Благодаря механизму привязки данных (Data Binding) ViewModel автоматически передает изменения в View, обеспечивая динамическое обновление интерфейса.

Также, в проекте применяются паттерн для работы с базой данных Repository.

Паттерн **Repository** — это архитектурный шаблон проектирования, который используется для абстрагирования доступа к данным и управления ими. Он действует как промежуточный слой между бизнес-логикой приложения и системой хранения данных, обеспечивая централизованный способ работы с информацией. Основная цель паттерна Repository — отделить логику работы с данными от бизнес-логики, что делает код более чистым, удобным для тестирования и гибким в отношении изменений в системе хранения данных. Вместо того чтобы напрямую взаимодействовать с базой данных, приложение обращается к репозиторию, который управляет операциями чтения, записи, обновления и удаления данных.

### 3.2 Проектирование структуры базы данных

**База данных** — это организованная совокупность данных, хранящаяся в электронном виде и предназначенная для удобного хранения, управления и обработки информации. Базы данных используются в самых разных сферах, от бизнеса и медицины до социальных сетей и онлайн-магазинов. Они позволяют систематизировать данные, обеспечивать быстрый доступ к нужной информации и эффективно управлять большими объемами данных.

**Система управления базами данных (СУБД)** — это программное обеспечение, которое отвечает за создание, управление и использование баз данных. СУБД предоставляет инструменты для хранения, поиска, обновления и удаления данных, а также обеспечивает безопасность и целостность информации.

В данном проекте для создания и управления базой данных использовалась реляционная СУБД Microsoft SQL Server, которая предоставляет мощные средства для работы с данными и обеспечивает высокую производительность и безопасность.

База данных приложения «Система управления проектами» состоит из 4 таблиц: Users, Tasks, Projects, ProjectUsers.

Таблица Tasks хранит информацию о заданиях.

Таблица Users хранит информацию о пользователях.

Таблица Projects хранит информацию о проектах.

Таблица ProjectUsers номер проекта и номер пользователя, который над ним работает.

Схема базы данных PMSystem\_db изображена на рисунке 3.2.

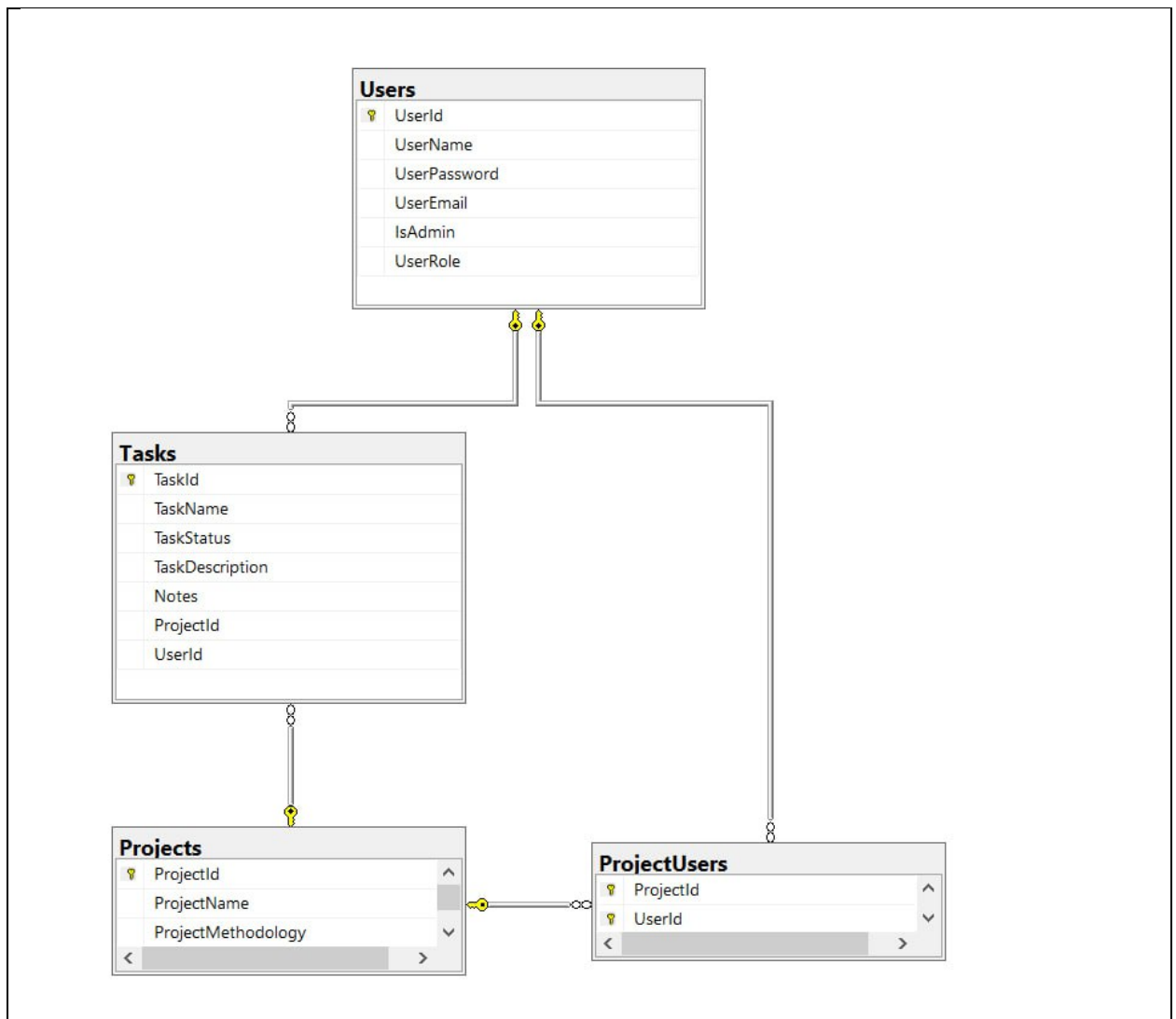


Рисунок 3.2 — Схема базы данных приложения

В приложении для работы с базой данных используется библиотека Entity Framework Core 9.0.4, которая предоставляет разработчикам удобные инструменты для взаимодействия с таблицами базы данных через объектно-ориентированную модель C#. В отличие от ADO.NET, здесь нет необходимости писать SQL-запросы вручную, поскольку для работы с данными применяется LINQ, а все преобразования выполняются библиотекой Entity Framework.



Существует три основных метода проектирования базы данных: Model-First, Database-First и Code-First. В данном проекте был выбран подход Code-First, который позволяет сначала определить модели данных в коде, а затем автоматически создать соответствующую структуру базы данных. Такой метод упрощает процесс разработки, поскольку разработчики могут сосредоточиться на бизнес-логике приложения, а не на ручном проектировании базы данных.

Code-First также облегчает тестирование и поддержку кода, позволяя создавать тестовые данные без сложных манипуляций с базой данных. Этот подход хорошо сочетается с принципами Agile-разработки, обеспечивая гибкость и возможность быстрого внесения изменений в структуру данных. Благодаря этому разработка становится более удобной и эффективной.

Использование Entity Framework Core в сочетании с Code-First делает работу с базой данных более удобной, гибкой и управляемой, что способствует повышению качества разработки и упрощает поддержку приложения.

### 3.3 Структура проекта

Структура проекта представлена на рисунке 3.3.

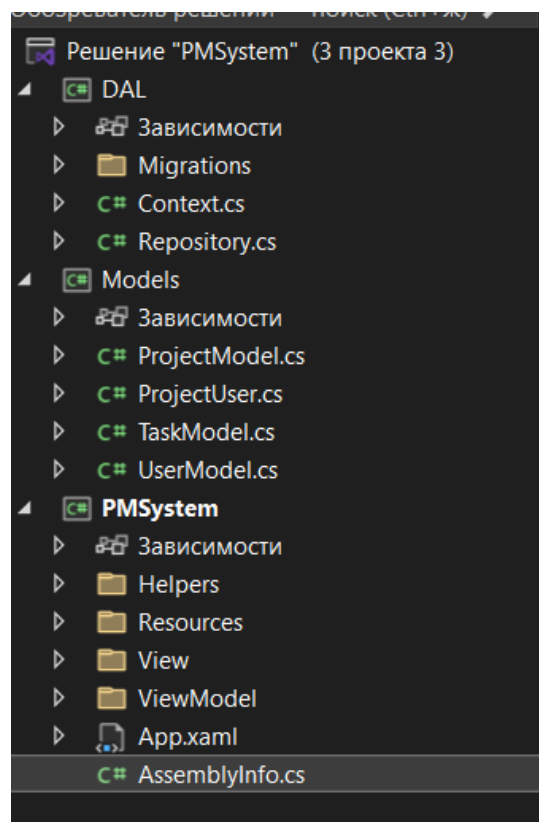


Рисунок 3.3 – Структура проекта

Папка «Migrations» хранит все классы миграций базы данных.

Папка «Views» содержит файлы с разметкой XAML, которые определяют пользовательский интерфейс (UI) приложения.

Папка «ViewModel» хранит классы, которые являются промежуточным слоем между моделью и представлением, обрабатывают данные и команды.

Папка «Resource» хранит в себе вспомогательные словари для локализации приложения.

Папка «Helpers» хранит вспомогательные классы, нужные для обработки команд из ViewModel и корректной обработки исключений.

Библиотека классов «Models» содержит в себе модели, на основе которых строится приложения (по паттерну MVVM) и база данных.

Файл «App.xaml» хранит общие ресурсы проекта.

### **3.4 Диаграмма классов**

При разработке программного продукта для планирования задач и управления проектами диаграмма классов играет важную роль в моделировании и проектировании системы. Она наглядно демонстрирует структуру приложения, отображая классы, их свойства и взаимосвязи.

В процессе создания программного обеспечения диаграмма классов помогает определить ключевые элементы системы, их характеристики и методы. Она позволяет визуализировать структуру данных, используемых в приложении, а также связи между различными компонентами. Кроме того, диаграмма способствует более глубокому пониманию концептуальной модели системы и служит основой для дальнейшего этапа разработки.

Диаграмма классов представлена в приложении Б.

## **4 Реализация программного средства**

На данном этапе осуществляется разработка программного обеспечения, реализующего предложенное техническое решение, а также сборка готового продукта. В ходе работы создаются программные интерфейсы, обеспечивающие взаимодействие между классами, методами и функциями.

Одним из ключевых аспектов является документирование классов, их атрибутов и методов, что способствует ясности и удобству дальнейшего использования. Процесс программирования включает написание исходного кода и отладку отдельных модулей проекта.

По завершении этого этапа будет создано полнофункциональное программное средство, готовое к эксплуатации.

## 4.1 Основные классы программного средства

В приложении создаётся большое количество классов, каждый из которых предназначен для выполнения определённых функций, что делает систему более модульной и гибкой. В этом разделе акцент будет на классах, отвечающих за реализацию паттерна MVVM, а также за работу с базой данных.

Архитектурный паттерн MVVM делит приложение на слои, обеспечивая четкое разделение между логикой представления и основной бизнес-логикой. Классы, управляющие взаимодействием с базой данных, играют ключевую роль в эффективном управлении данными. Они отвечают за такие задачи, как выполнение запросов, управление транзакциями и поддержка соединений с базой данных. Это позволяет приложению динамически загружать, обновлять и сохранять информацию, гарантируя её актуальность.

Эти компоненты важны для успешного функционирования приложения, поскольку они обеспечивают взаимодействие между интерфейсом и хранимыми данными, что способствует общей стабильности и эффективности системы.

### 4.1.1 Классы для реализации паттерна MVVM

Классы, реализующие ViewModel, как уже было сказано выше, отвечают за основную логику приложения, а также за логику взаимодействия с данными.

В классе LoginViewModel реализован необходимый функционал для входа пользователя в аккаунт, а также для проверки корректности введенных данных.

Основной метод класса — это метод Login, который запускается соответствующей командой, после чего начинается проверка на наличие такого пользователя в базе данных и на корректность введенной информации. Если проверка успешна, пользователь перенаправляется на главную страницу; в противном случае отображается сообщение об ошибке.

Для регистрации нового аккаунта в классе SignUpViewModel предусмотрен метод SignUp. При нажатии на соответствующую кнопку вызывается метод CanSignUp, который осуществляет проверку введенных данных на корректность. Данные проверяются на наличие ошибок, включая использование регулярных выражений для проверки email.

После успешного прохождения всех валидаций происходит попытка добавить нового пользователя в базу данных, и, при успехе, пользователь автоматически авторизуется и перенаправляется на главную страницу своего аккаунта.

Листинг реализации класса LoginViewModel и SignUpViewModel представлены в приложении В.

Далее, после перенаправления, пользователь попадает на главную страницу, за логику взаимодействия с которой отвечает класс MainViewModel. Он отвечает за корректное отображение страницы для простого пользователя, а также инструментов администратора. MainViewModel также генерирует с себе несколько классов поменьше(AddViewModel), которые используются администратором для добавления новых проектов, задач или пользователей.

Также присутствуют ещё классы MoreViewModel, которые необходимы при попытке получить подробную информация о задаче, пользователе или проекте.

Наконец, при перенаправлении пользователя непосредственно на страницу проекта, начинает работать ProjectInteractionViewModel, отвечающая за подгрузку данных, расстановку задач по своим колонкам и другие побочные действия на странице, включая изменения пользователем статуса задачи.

Также приложение взаимодействует с базой данных с помощью класса Context, наследуемого от класса DbContext. Также для более удобного и контролируемого взаимодействия с базой данных применяется паттерн Репозиторий, реализация которого размещена в соответствующем классе Repository.

## **5. Тестирование, проверка на работоспособность, анализ полученных результатов**

Тестирование приложения — это проведение проверки и оценки соответствия между реальным и ожидаемым поведением программы.

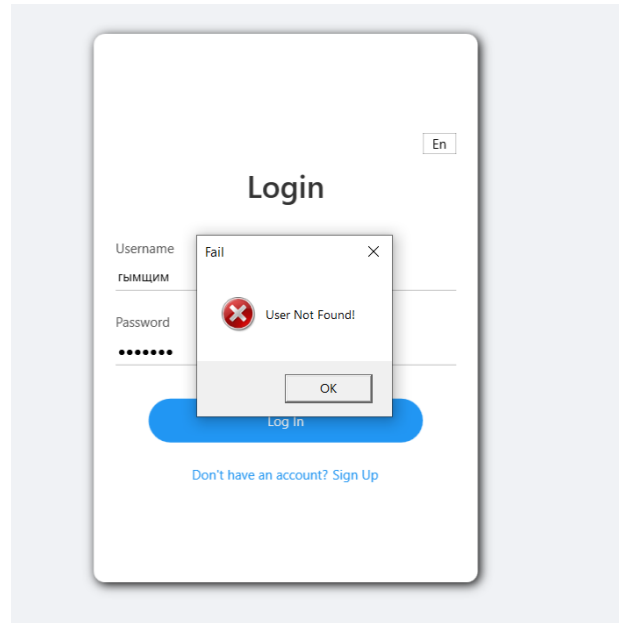
Цели тестирования:

- убедиться, что ПО отвечает заявленным требованиям;
- выявить ситуации, в которых поведение программы является неправильным, нежелательным или несоответствующим требованиям.

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

## 5.1 Тестирование 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.



Как видно, окно авторизации работает корректно и выдаёт соответствующее предупреждение.

Попробуем ввести некорректные данные в окне регистрации. На рисунке 5.2 демонстрируется результат, в котором выводится соответствующее сообщение.

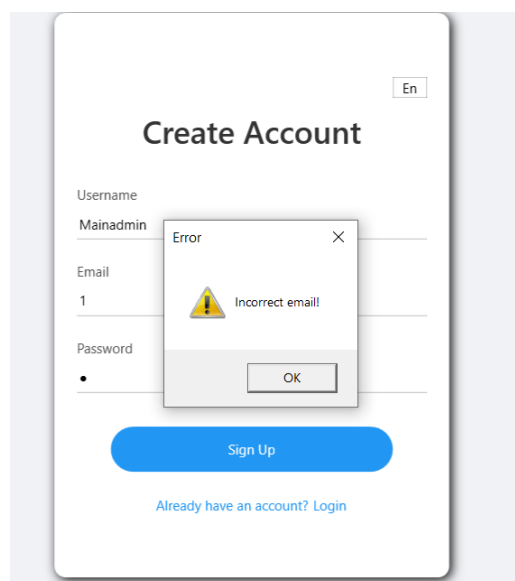


Рисунок 5.2 – Ошибка регистрации

Далее попробуем ввести некорректные или недоступные значения при добавлении новых элементов:

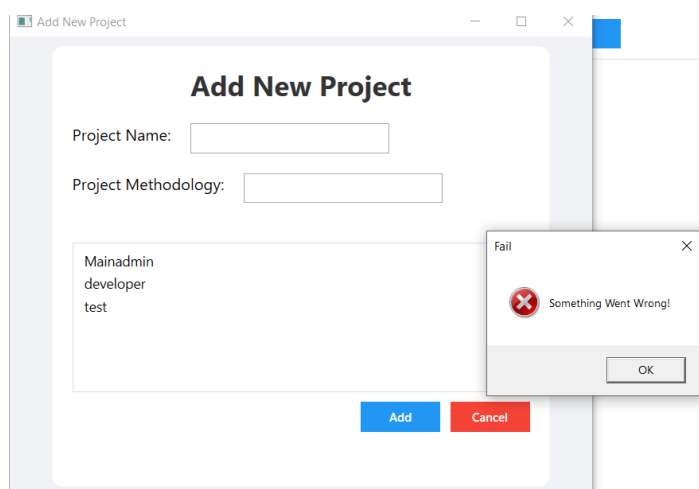


Рисунок 5.3 – Ошибка добавления проекта

Как видно, приложение распознаёт неправильные действия и выдаёт соответствующие предупреждения, одновременно предотвращая ошибочные действия и защищая данные, а также целостность самого приложения

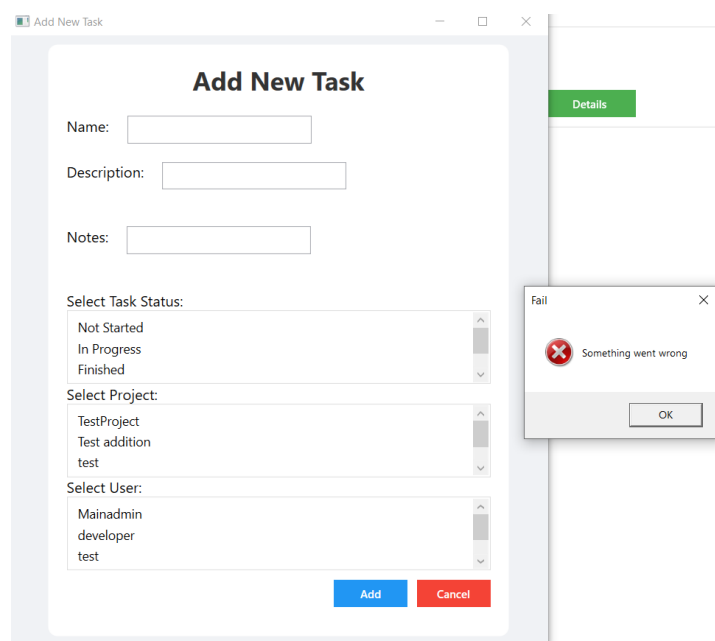
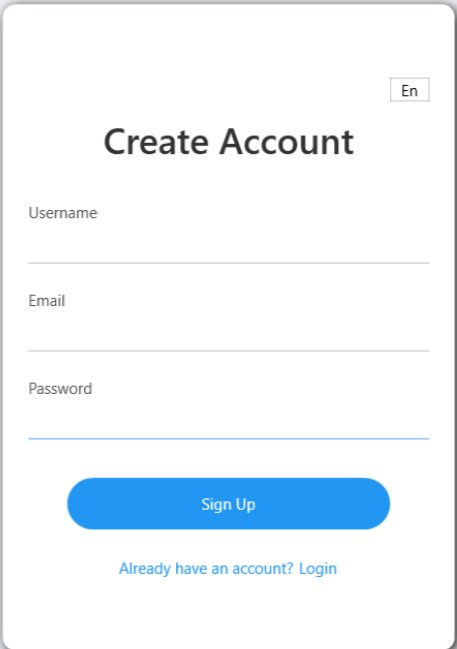


Рисунок 5.4 – Ошибка при добавлении задачи

## 6. Руководство по пользованию

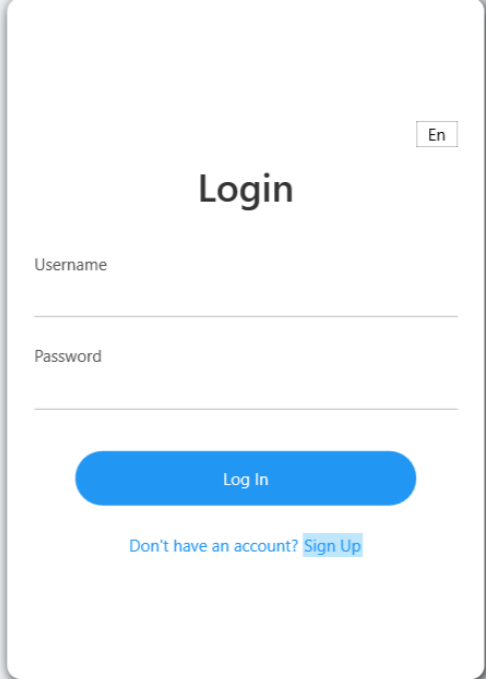
При запуске приложения открывается страница регистрации, на которой пользователь может либо зарегистрировать аккаунт, либо перейти на форму авторизации, если аккаунт уже имеется



The image shows a 'Create Account' registration form. It features a language selector 'En' in the top right corner. The title 'Create Account' is centered. Below the title are three input fields labeled 'Username', 'Email', and 'Password'. A blue 'Sign Up' button is positioned below the password field. At the bottom, there is a link that reads 'Already have an account? Login'.

Рисунок 6.1 – Форма регистрации

Отсюда пользователь может перейти на форму авторизации, где, введя данные аккаунта, попадёт на главную страницу.



The image shows a 'Login' authorization form. It features a language selector 'En' in the top right corner. The title 'Login' is centered. Below the title are two input fields labeled 'Username' and 'Password'. A blue 'Log In' button is positioned below the password field. At the bottom, there is a link that reads 'Don't have an account? Sign Up'.

Рисунок 6.2 – Страница авторизации



После регистрации или авторизации пользователь попадает на главную страницу.

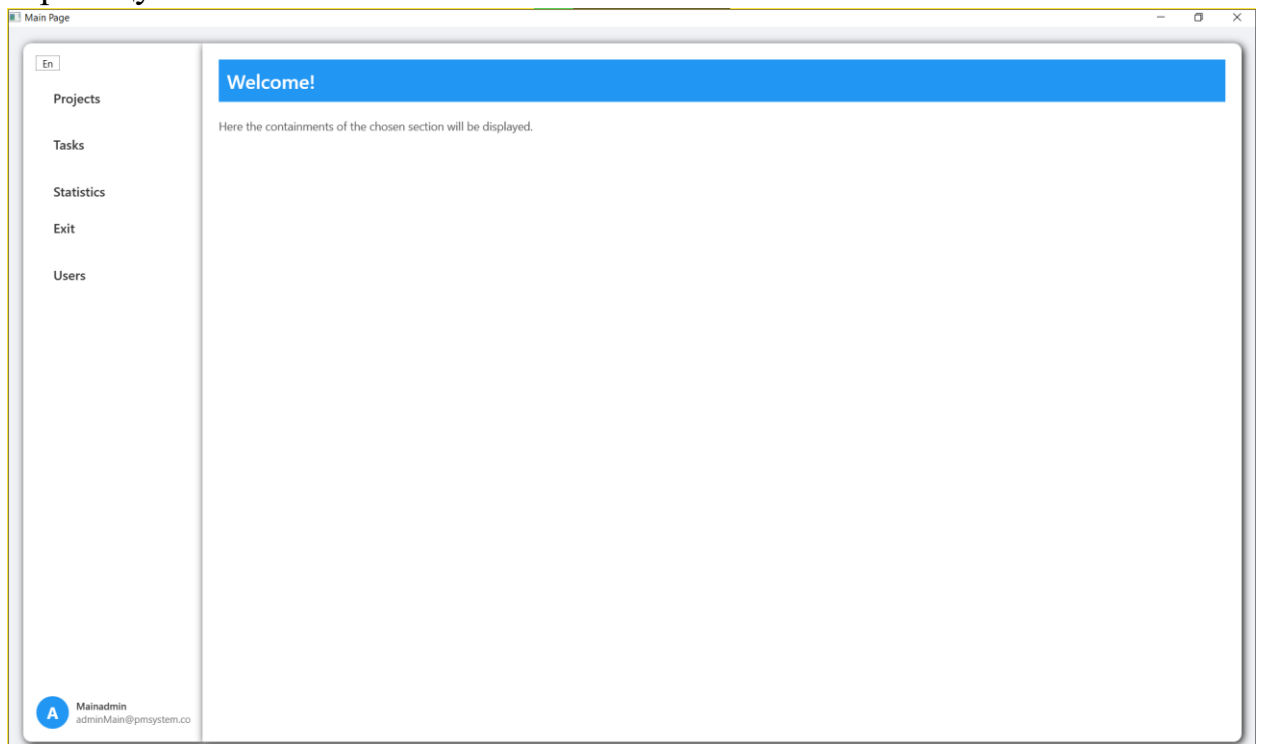


Рисунок 6.3 – Главная страница

Здесь пользователь может просматривать все свои проекты, задачи , а также персональную статистику и профиль.

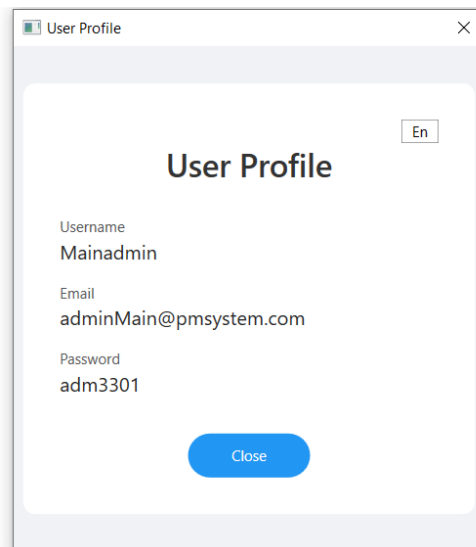


Рисунок 6.4 – Просмотр профиля

Также при просмотре проектов пользователь может перейти на страницу каждого из них и , по сути, воспользоваться основным функционалом приложения.



Рисунок 6.5 – Страница одного из выбранных проектов

Здесь пользователь может видеть все свои задачи и взаимодействовать с ними, изменять их статус и добавлять заметки.

## ЗАКЛЮЧЕНИЕ

Данное курсовое проектное задание было выполнено с целью разработки программного средства, которое позволяет эффективно и просто управлять своими задачами и проектами, а также администратору управлять базой данных и обеспечивать техническую поддержку.

В процессе выполнения проекта были достигнуты следующие результаты:

- разработано интуитивно понятное пользовательское приложение с удобным интерфейсом для просмотра и работы с задачами и проектами;
- реализована функциональность взаимодействия, добавления, удаления и изменения элементов;
- разработана административная часть для управления проектами, задачами и пользователями;
- произведено тестирование программного средства для проверки его функциональности и корректности работы.

Для реализации приложения был выбран язык программирования C# и технология Windows Presentation Foundation (WPF), обеспечивающая разработку удобного и современного пользовательского интерфейса. В качестве базы данных была использована MS SQL Server.

В процессе разработки программного средства был применен паттерн проектирования MVVM (Model-View-ViewModel). MVVM является одним из наиболее популярных паттернов для разработки пользовательских интерфейсов в технологии WPF.

В процессе разработки программного средства ViewModel была использована для связи пользовательского интерфейса с бизнес-логикой и данными. Благодаря MVVM, разработка стала более структурированной и управляемой, а код стал более поддерживаемым и расширяемым.

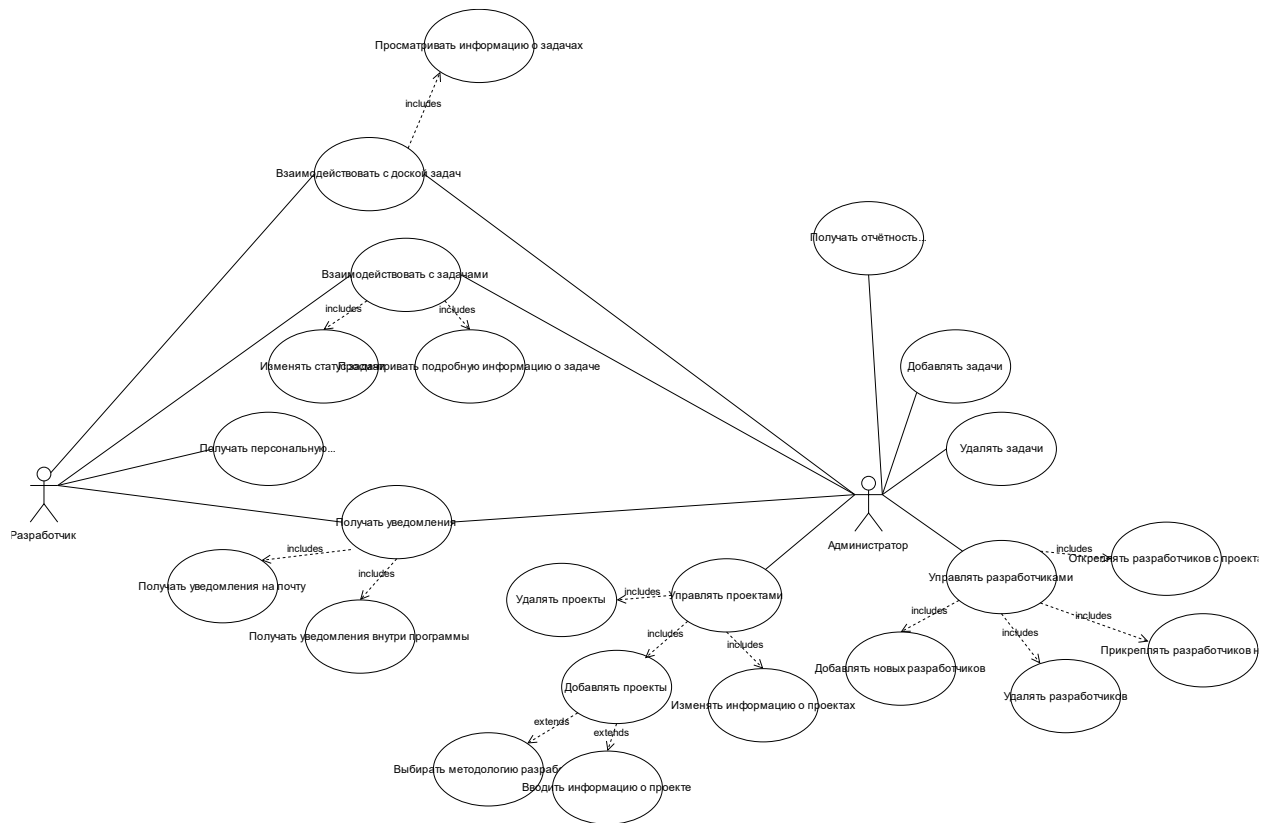
Таким образом, применение паттерна MVVM в программном средстве обеспечило эффективное управление данными и логикой приложения, улучшенную архитектуру и обеспечило высокую отзывчивость и удобство использования пользовательского интерфейса.

В результате выполнения данного курсового проекта было создано функциональное и удобное программное средство. Программа имеет потенциал для дальнейшего развития и расширения функциональности в соответствии с потребностями пользователей.

### Список использованных источников

1. Microsoft Visual Studio [Электронный ресурс] – [https://ru.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) – Дата доступа 23.04.2024
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: <https://metanit.com/sharp/tutorial/> – Дата доступа: 23.04.2024
3. Работа с Entity Framework Core [Электронный ресурс] – <https://professorweb.ru/my/entity-framework/6/level1/> – Дата доступа 26.04.2024
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/wpf/> – Дата доступа: 25.04.2024
5. Руководство по XAML // [Электронный ресурс]. – Режим доступа: <https://www.tutorialspoint.com/xaml/index.htm> – Дата доступа: 25.04.2024

# Приложение А



## Приложение В

```

<Window x:Class="PMSystem.View.LogInView"
    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"
    mc:Ignorable="d"
    d:DesignHeight="500" d:DesignWidth="800">

    <Grid Background="#f0f2f5">
        <Border Width="350"
            Height="500"
            Background="White"
            CornerRadius="10"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Padding="20">
            <Border.Effect>
                <DropShadowEffect BlurRadius="15" ShadowDepth="3"
Color="#66000000"/>
            </Border.Effect>
            <StackPanel Orientation="Vertical" VerticalAlignment="Center">
                <Button
                    Content="En"
                    Width="30"
                    HorizontalAlignment="Right"
                    Background="White"
                    BorderThickness="0.5"
                    Cursor="Hand" Click="LangChngeClick"
                >

                </Button>
                <TextBlock Text="{DynamicResource LoginTitle}" FontSize="28"
FontWeight="SemiBold" Foreground="#333" HorizontalAlignment="Center" Margin="0,10"/>

                <TextBlock Text="{DynamicResource UsernameLabel}" Margin="0,20,0,5"
Foreground="#555"/>
                <TextBox Text="{Binding Username,
UpdateSourceTrigger=PropertyChanged}"
                    BorderBrush="#ccc" BorderThickness="0,0,0,1"
Background="Transparent" Padding="0,5"/>

                <TextBlock Text="{DynamicResource PasswordLabel}" Margin="0,20,0,5"
Foreground="#555"/>
                <PasswordBox x:Name="PasswordBox"
                    PasswordChanged="PasswordBox_PasswordChanged"
                    BorderBrush="#ccc" BorderThickness="0,0,0,1"
                    Background="Transparent" Padding="0,5"/>

                <Button Content="{DynamicResource LogInText}"
                    Command="{Binding LoginCommand}"
                    Background="#2196F3"
                    Foreground="White"
                    Margin="0,30,0,10"
                    Height="40"
                    BorderThickness="0"
                    Cursor="Hand"
                    Width="250"
                    HorizontalAlignment="Center">
                <Button.Style>
                    <Style TargetType="Button">
                        <Setter Property="Template">
                            <Setter.Value>
                                <ControlTemplate TargetType="Button">

```

```

                                <Border x:Name="border"
Background="{TemplateBinding Background}" CornerRadius="20" Height="40" Width="250">
                                <ContentPresenter
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                                </Border>
                                <ControlTemplate.Triggers>
                                <Trigger Property="IsMouseOver"
Value="True">
                                <Setter TargetName="border"
Property="Background" Value="#1976D2"/>
                                </Trigger>
                                </ControlTemplate.Triggers>
                                </ControlTemplate>
                                </Setter.Value>
                                </Setter>
                                </Style>
                                </Button.Style>
                                </Button>

                                <TextBlock HorizontalAlignment="Center" Margin="0,10,0,0"
Cursor="Hand" Foreground="#2196F3">
                                <Run Text="{DynamicResource DontHaveAccountText}"/>
                                <Button Command="{Binding GoToSignUpCommand}"
Content="{DynamicResource SignUpText}"
Background="White"
BorderThickness="0"
Foreground="#2196F3"
Margin="0,0,0,-4"
Cursor="Hand"
>
                                </Button>
                                </TextBlock>

                                </StackPanel>
                                </Border>
                                </Grid>
</Window>

```

Листинг В1 – Разметка страницы авторизации

```

<Window x:Class=" PMSystem.View.SignUpView"
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"
mc:Ignorable="d"
d:DesignHeight="500" d:DesignWidth="800"
xmlns:local="clr-namespace:PMSystem.View">

    <Grid Background="#f0f2f5">
        <Border Width="350"
Height="500"
Background="White"
CornerRadius="10"
HorizontalAlignment="Center"
VerticalAlignment="Center"
Padding="20">
            <Border.Effect>
                <DropShadowEffect BlurRadius="15" ShadowDepth="3"
Color="#66000000"/>
            </Border.Effect>
            <StackPanel Orientation="Vertical" VerticalAlignment="Center">
                <Button
Content="En"

```



```

        Width="30"
        Background="White"
        BorderThickness="0.5"
        Cursor="Hand"
        HorizontalAlignment="Right" Click="LangChangeClick"
    >

</Button>
<TextBlock Text="{DynamicResource SignUpTitle}" FontSize="28"
FontWeight="SemiBold" Foreground="#333" HorizontalAlignment="Center" Margin="0,10"/>

    <TextBlock Text="{DynamicResource UsernameLabel}" Margin="0,20,0,5"
Foreground="#555"/>
    <TextBox Text="{Binding Username,
UpdateSourceTrigger=PropertyChanged}" BorderBrush="#ccc" BorderThickness="0,0,0,1"
Background="Transparent" Padding="0,5"/>

    <TextBlock Text="{DynamicResource EmailLabel}" Margin="0,20,0,5"
Foreground="#555"/>
    <TextBox Text="{Binding Email, UpdateSourceTrigger=PropertyChanged}"
BorderBrush="#ccc" BorderThickness="0,0,0,1" Background="Transparent"
Padding="0,5"/>

    <TextBlock Text="{DynamicResource PasswordLabel}" Margin="0,20,0,5"
Foreground="#555"/>
    <PasswordBox x:Name="PasswordBox"
        PasswordChanged="PasswordBox_PasswordChanged"
        BorderBrush="#ccc" BorderThickness="0,0,0,1"
Background="Transparent" Padding="0,5"/>

    <Button Content="{DynamicResource SignUpText}"
        Command="{Binding SignUpCommand}"
        Background="#2196F3"
        Foreground="White"
        Margin="0,30,0,10"
        Height="40"
        BorderThickness="0"

        Width="250"
        HorizontalAlignment="Center">
    <Button.Style>
        <Style TargetType="Button">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border x:Name="border"
Background="{TemplateBinding Background}" CornerRadius="20" Height="40" Width="250">
                            <ContentPresenter
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver"
Value="True">
                                <Setter TargetName="border"
Property="Background" Value="#1976D2"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Button.Style>
</Button>

```

```

        <TextBlock HorizontalAlignment="Center" Margin="0,10,0,0"
Cursor="Hand" Foreground="#2196F3" >
            <Run Text="{DynamicResource AlreadyHaveAccount}"/>
            <Button
                Command="{Binding GoToLoginCommand}"
Content="{DynamicResource LoginTitle}" Margin="0,0,0,-4"
                Background="White" BorderThickness="0" Foreground="#2196F3"
                HorizontalAlignment="Center" Cursor="Hand"
Click="Button_Click" >

                </Button>
            </TextBlock>

        </StackPanel>
    </Border>
</Grid>
</Window>

```

## Листинг В2 – Разметка страницы регистрации

```

<Window x:Class="PMSystem.View.MainView"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mdl ="clr-namespace:Models;assembly=Models"

    Title="{DynamicResource MainPageTitle}" Height="500" Width="800"
Background="#f0f2f5" WindowStartupLocation="CenterScreen">

    <!-- Стиль для кнопок боковой панели -->

    <Window.Resources>

        <DropShadowEffect x:Key="DropShadowEffect" BlurRadius="15" ShadowDepth="3"
Color="#66000000"/>

        <Style x:Key="SidePanelButtonStyle" TargetType="Button">
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border x:Name="border" Background="{TemplateBinding
Background}" CornerRadius="8" Padding="{TemplateBinding Padding}">
                            <ContentPresenter HorizontalAlignment="Left"
VerticalAlignment="Center" Margin="10,0"/>
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter TargetName="border" Property="Background"
Value="#2196F3"/>
                                <Setter Property="Foreground" Value="White"/>
                            </Trigger>
                            <Trigger Property="IsEnabled" Value="False">

```

```

        <Setter TargetName="border" Property="Background"
Value="#1976D2"/>

        <Setter Property="Foreground" Value="White"/>
    </Trigger>
</ControlTemplate.Triggers>
</ControlTemplate>
    </Setter.Value>
</Setter>
    <Setter Property="Foreground" Value="#333333"/>
    <Setter Property="Cursor" Value="Hand"/>
    <Setter Property="FontWeight" Value="SemiBold"/>
    <Setter Property="FontSize" Value="16"/>
</Style>

    <DataTemplate x:Key="ProjectCardTemplate" DataType="{x:Type
mdl:ProjectModel}">

        <Border BorderBrush="#E0E0E0" BorderThickness="1" CornerRadius="10"
Margin="10" Padding="10">

            <StackPanel>

                <TextBlock Text="{Binding ProjectName}" FontWeight="Bold"
FontSize="16"/>

                <TextBlock Text="{Binding ProjectMethodology}" FontSize="14"
Foreground="#555555"/>

            </StackPanel>

        </Border>

    </DataTemplate>
</Window.Resources>

<Grid Margin="20" >
    <Grid.ColumnDefinitions>

```

```

        <ColumnDefinition Width="220"/>
        <ColumnDefinition Width="*/>
    </Grid.ColumnDefinitions>

    <!-- Боковая панель -->
    <Border Grid.Column="0" Background="White" CornerRadius="10,0,0,10"
Padding="15"
        Effect="{DynamicResource DropShadowEffect}">
        <DockPanel LastChildFill="True" >
            <StackPanel DockPanel.Dock="Top">
                <!-- Кнопки боковой панели -->
                <Button
                    Content="En"
                    Background="Transparent"
                    HorizontalAlignment="Left"
                    Margin="0,0,0,10"
                    Cursor="Hand"
                    BorderThickness="0.5"
                    Width="30"
                    Click="LangChangeClick">
                </Button>
                <Button Content="{DynamicResource ProjectButton}"
Margin="0,0,0,12" Padding="12" Background="Transparent" BorderThickness="0.5"
Foreground="#333333" FontWeight="SemiBold" Cursor="Hand"
                    Command="{Binding ShowProjectsCommand}"
Style="{DynamicResource SidePanelButtonStyle}" Click="ProjectsClick"
x:Name="ProjectsButton"/>
                <Button Content="{DynamicResource TaskButton}" Margin="0,0,0,12"
Padding="12" Background="Transparent" BorderThickness="0.5" Foreground="#333333"
FontWeight="SemiBold" Cursor="Hand"
                    Command="{Binding ShowTasksCommand}"
Style="{DynamicResource SidePanelButtonStyle}" Click="TasksClick"
Name="TasksButton"/>
                <Button Content="{DynamicResource StatisticsButton}"
Margin="0,0,0,0" Padding="12" Background="Transparent" BorderThickness="0.5"
Foreground="#333333" FontWeight="SemiBold" Cursor="Hand"
                    Style="{DynamicResource SidePanelButtonStyle}"
Click="StatisticsClick" Name="StatisticsButton"/>
                <Button
                    Content="{DynamicResource ExitButton}"

```

```

        Style="{DynamicResource SidePanelButtonStyle}"
        Margin="0,0,0,12" Padding="12" Background="Transparent"
BorderThickness="0.5" Foreground="#333333" FontWeight="SemiBold" Cursor="Hand"
        Command="{Binding ExitCommand}">

</Button>
<Button
    Content="{DynamicResource UserButton}"
    Style="{DynamicResource SidePanelButtonStyle}"
    Command="{Binding ShowUsersCommand}"
    Margin="0,0,0,12" Padding="12" Background="Transparent"
BorderThickness="0.5" Foreground="#333333" FontWeight="SemiBold" Cursor="Hand"
    Visibility="Hidden"
    x:Name="UsersButton" Click="UsersClick">

</Button>

</StackPanel>
<StackPanel DockPanel.Dock="Bottom" Orientation="Horizontal"
Margin="0" VerticalAlignment="Bottom" HorizontalAlignment="Left" >
    <Button Background="Transparent" Cursor="Hand"
BorderThickness="0" Click="ElipseClick">
        <Grid Width="40" Height="40">

            <Ellipse Fill="#2196F3"/>
            <TextBlock x:Name="IconText" Text="U" Foreground="White"
FontWeight="Bold" FontSize="20"
                HorizontalAlignment="Center" VerticalAlignment="Center"
TextAlignment="Center"/>
        </Grid>
    </Button>

    <StackPanel Margin="8,0,0,0" VerticalAlignment="Center">
        <TextBlock Text="{Binding User.UserName}"
FontWeight="SemiBold" Foreground="#333333" TextWrapping="NoWrap"/>
        <TextBlock Text="{Binding User.UserEmail}" FontSize="12"
Foreground="#777777" TextWrapping="NoWrap"/>
    </StackPanel>
</StackPanel>

```

```

        </DockPanel>

</Border>

<!-- Основной контент -->
    <Border Grid.Column="1" Background="White" CornerRadius="0,10,10,0"
Padding="20"
        Effect="{DynamicResource DropShadowEffect}">
        <StackPanel>
            <StackPanel Name="MainTitle">
                <TextBlock x:Name="WelcomeTextBlock" Text="{DynamicResource
WelcomeText}" Background="#2196F3" FontSize="24" FontWeight="SemiBold"
Foreground="White" Margin="0,0,0,20" Padding="10" />
                <TextBlock x:Name="SideTextBlock" Text="{DynamicResource
SideText}" FontSize="14" Foreground="#555555" TextWrapping="Wrap"/>
            </StackPanel>

            <StackPanel Orientation="Horizontal" HorizontalAlignment="Center"
Name="SearchBlock" Visibility="Collapsed">
                <Button Content="{DynamicResource AddButtonTitle}"
                    Cursor="Hand"
                    Width="50"
                    Height="30"
                    Background="#2196F3" Foreground="White"
BorderThickness="0"
                    Name="AddProjectButton"
                    Command="{Binding AddNewProjectCommand}"
                    Visibility="Collapsed"
                    Margin="0,0,20,0"/>

                <Button Content="{DynamicResource AddButtonTitle}"
                    Cursor="Hand"
                    Width="50"
                    Height="30"
                    Background="#2196F3" Foreground="White"
BorderThickness="0"
                    Name="AddTaskButton"
                    Command="{Binding ShowAddTaskCommand}"

```

```

        Visibility="Collapsed"
        Margin="0,0,20,0"/>

        <Button Content="{DynamicResource AddButtonTitle}"
            Cursor="Hand"
            Width="50"
            Height="30"
            Background="#2196F3" Foreground="White"
BorderThickness="0"

            Name="AddUserButton"
            Command="{Binding ShowAddUser}"
            Visibility="Collapsed"
            Margin="0,0,20,0"/>

        <TextBox Text="{Binding
FilterText,UpdateSourceTrigger=PropertyChanged}" x:Name="SearchBar"
Visibility="Visible" Width="300" Height="30" Margin="0,10,0,10" Padding="5"
VerticalAlignment="Top" />

        <Button Command="{Binding SearchCommand}" Cursor="Hand"
Content="{DynamicResource SearchButtonTitle}" Width="50" Height="30"
Margin="30,0,0,0" Background="#2196F3" Foreground="White"
BorderThickness="0"></Button>

    </StackPanel>

    <ScrollViewer Name="scroll" Visibility="Hidden">

        <ItemsControl Name="ContentControl">

            <ItemsControl.Resources>

                <DataTemplate DataType="{x:Type mdl:ProjectModel}">

                    <Border Width="300" BorderBrush="#E0E0E0"
BorderThickness="1" CornerRadius="10" Margin="10" Padding="10" Background="White">

                        <StackPanel Width="300">

                            <TextBlock Text="{Binding ProjectName}"
FontWeight="Bold" FontSize="18" Foreground="#333333" Margin="0,0,0,5"/>

                            <TextBlock Text="{Binding
ProjectMethodology}" FontSize="14" Foreground="#777777" Margin="0,0,0,10"/>

                            <Button CommandParameter="{Binding}"
Command="{Binding DataContext.ShowProjectInterCommand,RelativeSource={RelativeSource

```



```

AncestorType={x:Type ItemsControl}}}" Content="{DynamicResource ProjectMoreButton}"
Width="100" Height="30" Background="#2196F3" Foreground="White"
FontWeight="SemiBold" Cursor="Hand" BorderThickness="0"/>

        </StackPanel>

    </Border>

</DataTemplate>

<DataTemplate DataType="{x:Type mdl:TaskModel}">
    <Border Width="300" BorderBrush="#E0E0E0"
BorderThickness="1" CornerRadius="10" Margin="10" Padding="10" Background="White">

        <StackPanel>

            <TextBlock Text="{Binding TaskName}"
FontWeight="Bold" FontSize="18" Foreground="#333333" Margin="0,0,0,5"/>

            <TextBlock Text="{Binding TaskDescription}"
FontSize="14" Foreground="#777777" Margin="0,0,0,10"/>

            <Button CommandParameter="{Binding}"
Command="{Binding DataContext.ShowTaskDetailsCommand,RelativeSource={RelativeSource
AncestorType={x:Type ItemsControl}}}" Content="{DynamicResource TasksDetailsButton}"
Width="100" Height="30" Background="#4CAF50" Foreground="White"
FontWeight="SemiBold" Cursor="Hand" BorderThickness="0"/>

        </StackPanel>

    </Border>

</DataTemplate>

<DataTemplate DataType="{x:Type mdl:UserModel}">
    <Border Width="300" BorderBrush="#E0E0E0"
BorderThickness="1" CornerRadius="10" Margin="10" Padding="10" Background="White">

        <StackPanel>

            <TextBlock Text="{Binding UserName}"
FontWeight="Bold" FontSize="18" Foreground="#333333" Margin="0,0,0,5"/>

            <TextBlock Text="{Binding UserEmail}"
FontSize="14" Foreground="#555555" Margin="0,0,0,5"/>

```

```

<TextBlock Text="{Binding UserRole}"
FontSize="14" Foreground="#777777"/>

<Button Content="{DynamicResource
UserInfoButton}" Command="{Binding DataContext.
ShowDetailedUser,RelativeSource={RelativeSource AncestorType={x:Type
ItemsControl}}}" CommandParameter="{Binding}" Width="100" Height="30"
Background="#2196F3" Foreground="White" FontWeight="SemiBold" Cursor="Hand"
BorderThickness="0"/>

</StackPanel>

</Border>
</DataTemplate>
</ItemsControl.Resources>
<ItemsControl.ItemsPanel>
<ItemsPanelTemplate>
<WrapPanel />
</ItemsPanelTemplate>
</ItemsControl.ItemsPanel>
</ItemsControl>
</ScrollViewer>
</StackPanel>

</Border>
</Grid>

</Window>

```

Листинг В3 – Разметка главной страницы