­­МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ ТАТАРСТАН ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
«КАЗАНСКИЙ АВТОТРАНСПОРТНЫЙ ТЕХНИКУМ ИМ. А. П. ОБЫДЕННОВА»

Специальность 09.02.07 Информационные системы и программирование

(код, наименование)

**КУРСОВОЙ ПРОЕКТ**

**Тема: Проектирование приложение «Дистант»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: |  |  | Фирсов Дмитрий Алексеевич |
|  | (подпись, дата) |  | (Ф. И. О.) |
| Руководитель: |  |  | Макаров Давид Моррисович |
|  | (подпись, дата) |  | (Ф. И. О.) |

Казань 2024

**Содержание**

[Введение 3](#_Toc185340075)

[1 Анализ потребностей 5](#_Toc185340076)

[1.1 Целевая аудитория 5](#_Toc185340077)

[1.2 Функциональные требования 5](#_Toc185340078)

[1.2.1 Функциональные требования Студентов 6](#_Toc185340079)

[1.2.2 Функциональные требования Преподавателей 6](#_Toc185340080)

[1.2.3 Функциональные потребности Администраторов 7](#_Toc185340081)

[1.3 Нефункциональные требования 8](#_Toc185340082)

[1.3.1 Нефункциональные требования для студентов 9](#_Toc185340083)

[1.3.2 Нефункциональные требования для преподавателей 9](#_Toc185340084)

[1.3.3 Нефункциональные требования для администраторов 10](#_Toc185340085)

[2 Выбор стека технологий 11](#_Toc185340086)

[2.1 Примеры языков программирования 11](#_Toc185340087)

[2.2 Примеры СУБД 12](#_Toc185340088)

[2.3 Примеры фреймворков 12](#_Toc185340089)

[2.4 Обоснование выбора 13](#_Toc185340090)

[3 Разработка пользовательского интерфейса приложения 14](#_Toc185340091)

[3.1 Структура приложения 14](#_Toc185340092)

[3.2 Проектирование графического интерфейса в .NET 15](#_Toc185340093)

[3.2.1 Проектирование макета в Visual Studio 2022 16](#_Toc185340094)

[3.3 Интерфейс авторизации 23](#_Toc185340095)

[3.4 Интерфейс регистрации 25](#_Toc185340096)

[3.5 Интерфейс для студентов 28](#_Toc185340097)

[3.6 Интерфейс для преподавателей 31](#_Toc185340098)

[3.7 Интерфейс администраторов 33](#_Toc185340099)

[4 Проектирование базы данных 36](#_Toc185340100)

[4.1 Описание ER-диаграммы 36](#_Toc185340101)

[Заключение 38](#_Toc185340102)

[Список использованных источников 39](#_Toc185340103)

## Введение

В эпоху стремительного развития цифровых технологий дистанционное обучение стало одной из ключевых форм образовательного процесса. С его помощью обеспечивается доступ к знаниям для студентов, находящихся в разных уголках мира, и создаются гибкие условия для преподавателей и администраторов. На сегодняшний день многие образовательные учреждения сталкиваются с вызовами, связанными с организацией удалённого взаимодействия между участниками учебного процесса. Отсутствие универсальных инструментов для управления заданиями, обмена информацией и мониторинга успеваемости зачастую приводит к снижению качества обучения. Это подчёркивает важность создания специализированных приложений, которые объединяют функции взаимодействия, управления образовательными данными и предоставления обратной связи в рамках единой платформы.

Целью данного курсового проекта является проектирование приложения «Дистант», ориентированного на обеспечение эффективного дистанционного взаимодействия между студентами, преподавателями и администраторами образовательных учреждений.

Для достижения поставленной цели были определены следующие задачи:

Анализ потребностей целевой аудитории: выявление ключевых функций, необходимых для работы студентов, преподавателей и администраторов в дистанционном формате.

Определение требований к приложению: разработка функциональных и нефункциональных требований, включая удобство интерфейса, безопасность и производительность.

Проектирование пользовательского интерфейса: создание интуитивно понятного и функционального интерфейса для всех категорий пользователей.

Актуальность разработки приложения «Дистант» заключается в его способности повысить доступность и качество образовательного процесса, снизить нагрузку на преподавателей и администраторов, а также обеспечить удобный доступ к образовательным ресурсам для студентов.

# 1 Анализ потребностей

# 1.1 Целевая аудитория

Определение целевой аудитории для приложения «Дистант» позволяет понять, какие функции и возможности являются необходимыми для удовлетворения потребностей пользователей. Основные категории пользователей приложения включают в себя:

1. Студенты
   1. Цель: Получение знаний и навыков через онлайн-обучение и выполнение заданий.
   2. Потребности: Удобный интерфейс для проверки и отправки заданий.
2. Преподаватели
   1. Цель: Эффективная организация образовательного процесса и оценка выполненных заданий.
   2. Потребности: Инструменты для удобной проверки заданий.
3. Администрация образовательных учреждений
   1. Цель: Обеспечение качества образования и контроль за выполнением учебных планов.
   2. Потребности: Статистика по успеваемости, возможность анализа работы преподавателей, управление платформой.

Определение целевой аудитории для приложения «Дистант» позволяет понять, какие функции и возможности являются необходимыми для удовлетворения потребностей пользователей. Адаптация интерфейса и функционала приложения, основанная на характеристиках каждой группы, обеспечит более эффективное использование платформы и повысит удовлетворенность пользователей.

# **1.2 Функциональные **требования****

В этом разделе подробно описаны функциональные требования, сгруппированные по целевой аудитории, чтобы система эффективно обслуживала каждый тип пользователя.

# ****1.2.1**** Функциональные ****требования**** Студентов

Просмотр учебных материалов: студенты должны иметь возможность просматривать все пройденные учебные темы, включая текстовые материалы, видео лекции, презентации и другие мультимедийные ресурсы. Необходимо указать поддерживаемые форматы файлов (например, MP4, PDF, DOCX).

Навигация по учебным темам: должна быть удобная система навигации по темам, позволяющая студентам легко перемещаться между темами и разделами. Например, древовидная структура или карта курса.

Поиск по материалам: возможность поиска информации внутри учебных материалов.

Создание и отправка заданий: студенты должны иметь возможность создавать и отправлять задания. Поддержка различных типов файлов (например, DOCX, PDF, ZIP).

Подтверждение отправки: приложение должно подтверждать успешную отправку задания.

Отслеживание статуса: студенты должны видеть текущий статус своих заданий (проверяется, оценено).

Просмотр оценок: студенты должны иметь доступ к своим оценкам за выполненные задания.

Обратная связь: предоставление преподавателем подробной обратной связи по выполненным заданиям.

История оценок: возможность просмотра оценок за выполненные задания.

# ****1.2.2**** Функциональные ****требования**** Преподавателей

Создание учебных тем: выбор названия и подробного описания темы, включая цели обучения и необходимую информацию для студентов.

Структурирование контента: Возможность структурировать учебный курс на темы, разделы с логической последовательностью.

Загрузка материалов: загрузка различных типов учебных материалов: тексты, видео, аудио, презентации, документы, интерактивные задания и др., с указанием поддерживаемых форматов файлов.

Добавление инструкций и критериев оценки: возможность добавления подробных инструкций к заданиям, а также критериев оценки для объективной и прозрачной оценки.

Загрузка образцов и шаблонов: возможность загрузки образцов выполнения заданий и шаблонов для студентов.

Просмотр и оценка работ: удобный интерфейс для просмотра и оценки студенческих работ.

Предоставление обратной связи: возможность предоставления подробной обратной связи по каждой работе.

Выставление оценок: возможность выставлять оценки по заданиям.

# ****1.2.3**** Функциональные потребности Администраторов

Управление пользователями: возможность создавать новые учетные записи для преподавателей и студентов, назначая роли и права доступа.

Редактирование профилей: возможность редактировать существующие профили пользователей, включая личную информацию и роли.

Удаление учетных записей: возможность удалять учетные записи пользователей.

Управление ролями и правами доступа: возможность назначать различные роли пользователям (например, администратор, преподаватель, студент) с соответствующими правами доступа к функциям системы.

Восстановление паролей: возможность предоставлять пользователям инструменты для восстановления забытых паролей.

Управление учебным процессом: возможность просматривать список всех тем, созданных в системе.

Управление преподавателями: возможность назначать преподавателям предметы и группы.

# ****1.3 Нефункциональные требования****

Общие нефункциональные требования для студентов, преподавателей и администраторов дистанционного приложения фокусируются на аспектах, важных для всех пользователей, независимо от их роли. К ним относятся:

Надежность: приложение должно быть надежным и стабильным, обеспечивая бесперебойную работу и минимальное количество ошибок. Это включает в себя устойчивость к сбоям.

Безопасность: все пользователи должны быть защищены от несанкционированного доступа к данным и функциям. Это включает в себя безопасную аутентификацию и авторизацию, шифрование данных (как при передаче, так и в состоянии покоя), защиту от SQL-инъекций и других уязвимостей.

Производительность: хотя конкретные требования к производительности различаются для разных пользователей, общее требование — приемлемое время отклика приложения на все действия пользователей без чрезмерного потребления системных ресурсов.

Удобство использован: хотя конкретные аспекты удобства использования различаются, общее требование — интуитивно понятный пользовательский интерфейс, простая навигация и удобство работы с приложением для всех пользователей, независимо от уровня технической подготовки.

В следующих подпунктах рассмотрим подробнее нефункциональные требования по видам пользователей.

# 1.3.1 Нефункциональные требования для студентов

Простота и интуитивность интерфейса: студенты должны легко находить нужные учебные материалы, отправлять задания и просматривать оценки. Интерфейс должен соответствовать современным стандартам UX/UI, чтобы студенты могли легко находить нужные материалы, даже без предварительного обучения работе с приложением, также должен быть интуитивно понятным, даже для пользователей с минимальными техническими знаниями.

Надёжность и доступность информации: учебные материалы, задания и оценки должны быть доступны круглосуточно. Приложение должно сводить к минимуму сбои и технические проблемы.

Минимальные системные требования: приложение должно работать на устройствах с базовыми характеристиками, чтобы обеспечить доступ для всех студентов, независимо от их оборудования.

# 1.3.2 Нефункциональные требования для преподавателей

Интуитивный интерфейс: преподаватели должны быстро и легко выполнять задачи, такие как загрузка учебных материалов, проверка заданий и выставление оценок.

Стабильность работы: приложение должно функционировать без сбоев даже в пиковые нагрузки, например, во время массовой сдачи заданий.

Сохранение данных: внесённые преподавателями данные (оценки, комментарии) должны сохраняться автоматически, исключая их потерю.

Защита информации: только авторизованные преподаватели должны иметь доступ к данным студентов и их работам. Все данные защищены от утечек и несанкционированного доступа.

Производительность: приложение должно быстро обрабатывать запросы и обеспечивать доступ к данным больших групп студентов, включая курсы с несколькими потоками.

# 1.3.3 Нефункциональные требования для администраторов

Гибкость настройки: администраторы должны иметь возможность легко изменять параметры приложения, добавлять новых пользователей, управлять группами и курсами.

Защита административного доступа: доступ к административным функциям должен быть строго ограничен и защищён.

Обработка данных: приложение должно эффективно управлять большими объёмами данных, включая историю действий пользователей и статистику успеваемости.

# 2 Выбор стека технологий

Стек технологий - это совокупность инструментов, языков программирования, фреймворков и баз данных, используемых для создания программного обеспечения. В зависимости от типа приложения (веб, мобильное, настольное) и его особенностей, разработчики выбирают соответствующий стек технологий. Для приложения «Дистант» особое внимание уделяется инструментам, которые обеспечивают удобный интерфейс, стабильную работу с удалёнными данными и безопасность.

# 2.1 Примеры языков программирования

При разработке настольных приложений существует широкий выбор языков программирования. Рассмотрим наиболее популярные варианты:

* C#: Используется для создания приложений на платформе .NET. Подходит для разработки сложных настольных приложений с графическим интерфейсом. Обеспечивает удобную работу с базами данных и взаимодействие с удалёнными API.
* Java: Кроссплатформенный язык, который позволяет разрабатывать настольные и веб-приложения. Популярен благодаря широкому выбору фреймворков, таких как JavaFX и Swing.
* Python: Прост в изучении, используется для быстрого прототипирования. Подходит для настольных приложений с использованием библиотек, таких как PyQt или Tkinter.
* C++: Высокопроизводительный язык для создания системного ПО и настольных приложений. Требует больше времени для разработки и сложнее в освоении.

Выбор языка программирования зависит от задач проекта. Для «Дистант» оптимален C#, поскольку он позволяет эффективно разрабатывать настольные приложения с современным интерфейсом и функционалом.

# 2.2 Примеры СУБД

Система управления базами данных (СУБД) используется для хранения, управления и обработки данных приложения. Рассмотрим наиболее популярные СУБД:

* MySQL: Открытое и популярное решение для работы с реляционными базами данных. Высокая производительность и возможность интеграции с платформой .NET. Подходит для приложений с небольшими и средними объёмами данных.
* PostgreSQL: Надёжная и функциональная СУБД с поддержкой сложных запросов и транзакций. Предпочтительна для приложений, работающих с большими объёмами данных.
* Microsoft SQL Server: Оптимизирован для приложений на платформе .NET. Имеет встроенные инструменты для работы с данными и анализа.
* SQLite: Лёгкая СУБД, не требующая отдельного сервера. Подходит для приложений с небольшими базами данных.

Для приложения «Дистант» была выбрана MySQL благодаря её производительности, совместимости с платформой .NET и бесплатной лицензии.

# 2.3 Примеры фреймворков

Фреймворки упрощают разработку, предоставляя готовые инструменты и библиотеки для создания приложений. Рассмотрим фреймворки, которые могут быть полезны при разработке настольных приложений:

* .NET Framework: Подходит для создания приложений под Windows. Включает в себя обширный набор библиотек для работы с графическим интерфейсом, сетью и базами данных.
* .NET Core / .NET 6+: Кроссплатформенная версия .NET. Позволяет создавать приложения для Windows, Linux и macOS.
* Electron: Используется для разработки настольных приложений на основе веб-технологий (HTML, CSS, JavaScript). Применяется для кроссплатформенных решений.
* JavaFX: Фреймворк для создания настольных приложений на Java. Поддерживает создание гибкого графического интерфейса.

Для «Дистант» выбран .NET (6 и выше), который обеспечивает надёжность, кроссплатформенность и широкий функционал для создания удобного интерфейса и обработки данных.

# 2.4 Обоснование выбора

Выбор стека технологий для разработки приложения играет ключевую роль в успешной реализации проекта. Для приложения «Дистант» в качестве стека технологий выбраны **язык программирования C#**, **платформа .NET**, и **СУБД MySQL**. Обоснование выбора основывается на характеристиках проекта, функциональных и нефункциональных требованиях, а также современных тенденциях разработки приложений.

Стек позволяет эффективно реализовать ключевые функции, такие как управление учебными материалами, отправка заданий, обработка оценок и генерация статистики. Использование C# и .NET гарантирует соблюдение высоких стандартов безопасности, включая защиту от SQL-инъекций, шифрование данных и безопасную аутентификацию пользователей. Разработчик уже имеет опыт использования этого стека, что позволяет сократить время на проектирование и реализацию. Благодаря этому стеку добавление новых функций или исправление ошибок будет происходить быстрее. С помощью C# и .NET можно легко расширять функционал приложения.

# 3 Разработка пользовательского интерфейса приложения

# 3.1 Структура приложения

Структура приложения представляет собой многомодульную архитектуру, разделенную на несколько ключевых компонентов, каждый из которых отвечает за выполнение определённых задач. Логическая организация системы разработана таким образом, чтобы удовлетворить потребности трёх типов пользователей: студентов, преподавателей и администраторов.

На Рисунке 1 представлена схема, описывающая иерархию разделов и взаимодействие между элементами приложения.

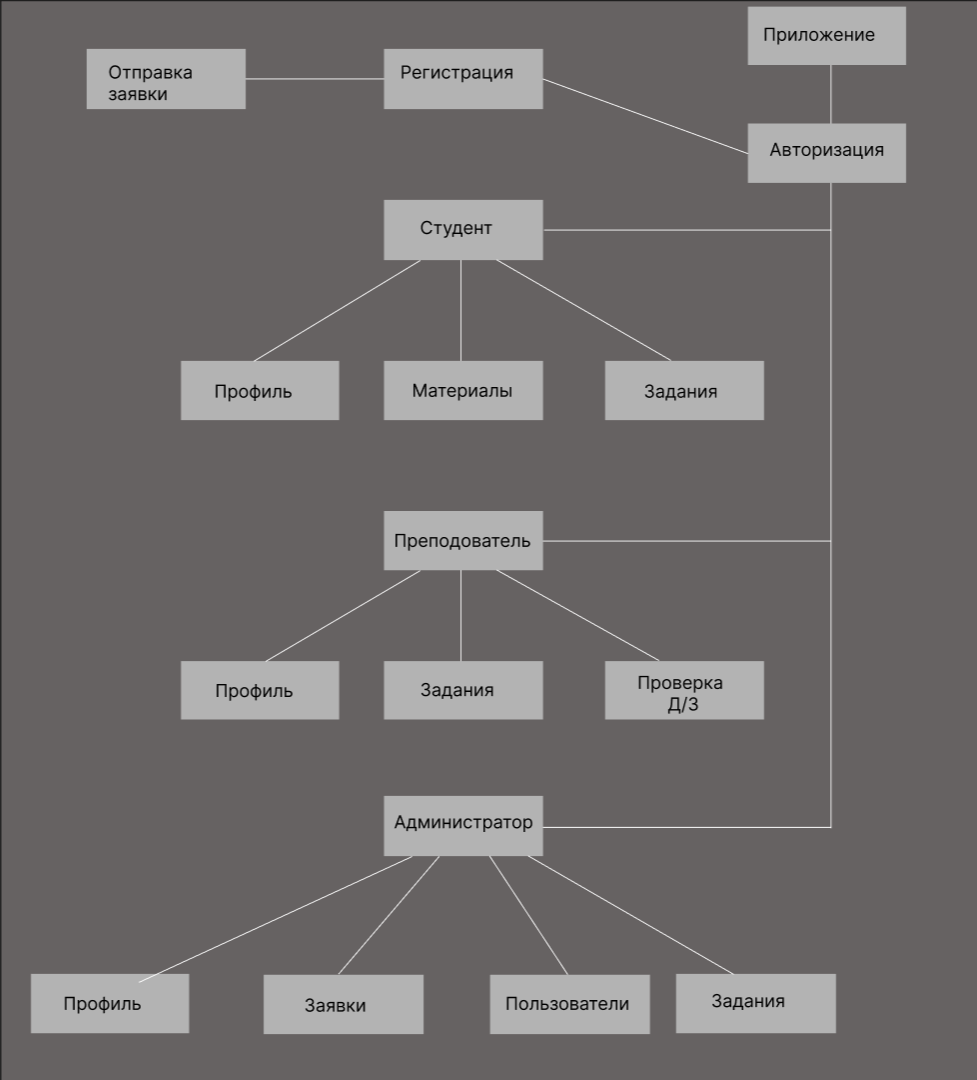


Рисунок 1 – Структура приложения

Пользователи взаимодействуют с приложением через свои интерфейсы, которые построены в зависимости от их роли. Авторизация определяет, какой функционал будет доступен каждому типу пользователя.

Каждый из представленных разделов обладает чётко определённым функционалом, что позволяет минимизировать сложность и повысить удобство использования системы.

Такая структура приложения позволяет эффективно распределять функционал между разными типами пользователей, обеспечивая удобство взаимодействия, чёткость логики и простоту дальнейшей модификации приложения.

# 3.2 Проектирование графического интерфейса в .NET

В платформе .NET для разработки графического интерфейса используются различные технологии, среди которых основными являются Windows Forms (WinForms) и Windows Presentation Foundation (WPF). Также часто применяются вспомогательные библиотеки, такие как MetroFramework и DevExpress, которые облегчают создание более сложных и визуально привлекательных интерфейсов. Однако в рамках данного проекта основное внимание уделяется технологии Windows Forms, которая предоставляет удобные и эффективные инструменты для создания пользовательских интерфейсов.

Windows Forms, или WinForms, представляет собой одну из базовых библиотек .NET для разработки настольных приложений с графическим интерфейсом. Она позволяет создавать интерфейсы с помощью различных элементов управления, таких как кнопки, текстовые поля, таблицы и другие компоненты, обеспечивая простоту в использовании и интуитивно понятный подход к разработке. Важно отметить, что WinForms ориентирован на создание классических приложений для операционных систем Windows, что делает его идеальным выбором для большинства приложений.

Технология WinForms использует несколько ключевых пространств имен. Одним из основных является System.Windows.Forms, которое отвечает за создание и работу с элементами управления, такими как кнопки, текстовые поля, списки, таблицы и другие визуальные компоненты. Это пространство имен предоставляет необходимые инструменты для проектирования и обработки пользовательского интерфейса. Важным элементом является также пространство имен System.Drawing, которое используется для работы с графическими элементами. Оно включает в себя функции для рисования фигур, обработки изображений, а также настройки цвета и шрифтов, что позволяет разработчикам придавать приложениям более разнообразный и красивый вид.

WinForms поддерживает два режима работы. Режим проектирования подразумевает использование визуального конструктора в среде разработки Visual Studio. В этом режиме разработчик может легко и быстро размещать элементы управления на форме, настраивать их параметры и свойства, а также устанавливать взаимодействие между ними. Этот подход значительно ускоряет процесс разработки и упрощает создание интерфейса. Режим выполнения, в свою очередь, касается непосредственно работы приложения, когда пользователи взаимодействуют с элементами управления в процессе работы программы. В этом режиме все элементы интерфейса становятся доступными для использования, и приложение реагирует на действия пользователей, такие как нажатие кнопок, ввод текста и другие события.

# 3.2.1 Проектирование макета в Visual Studio 2022

Для разработки графического интерфейса приложения на платформе .NET с использованием технологии Windows Forms, одной из самых удобных и мощных сред разработки является **Visual Studio 2022**. В этой среде разработки есть встроенные инструменты, которые упрощают создание интерфейсов.

Для проектирования макета приложения откроем Visual Studio 2022 и создадим проект для этого нажимаем на «Создание проекта».

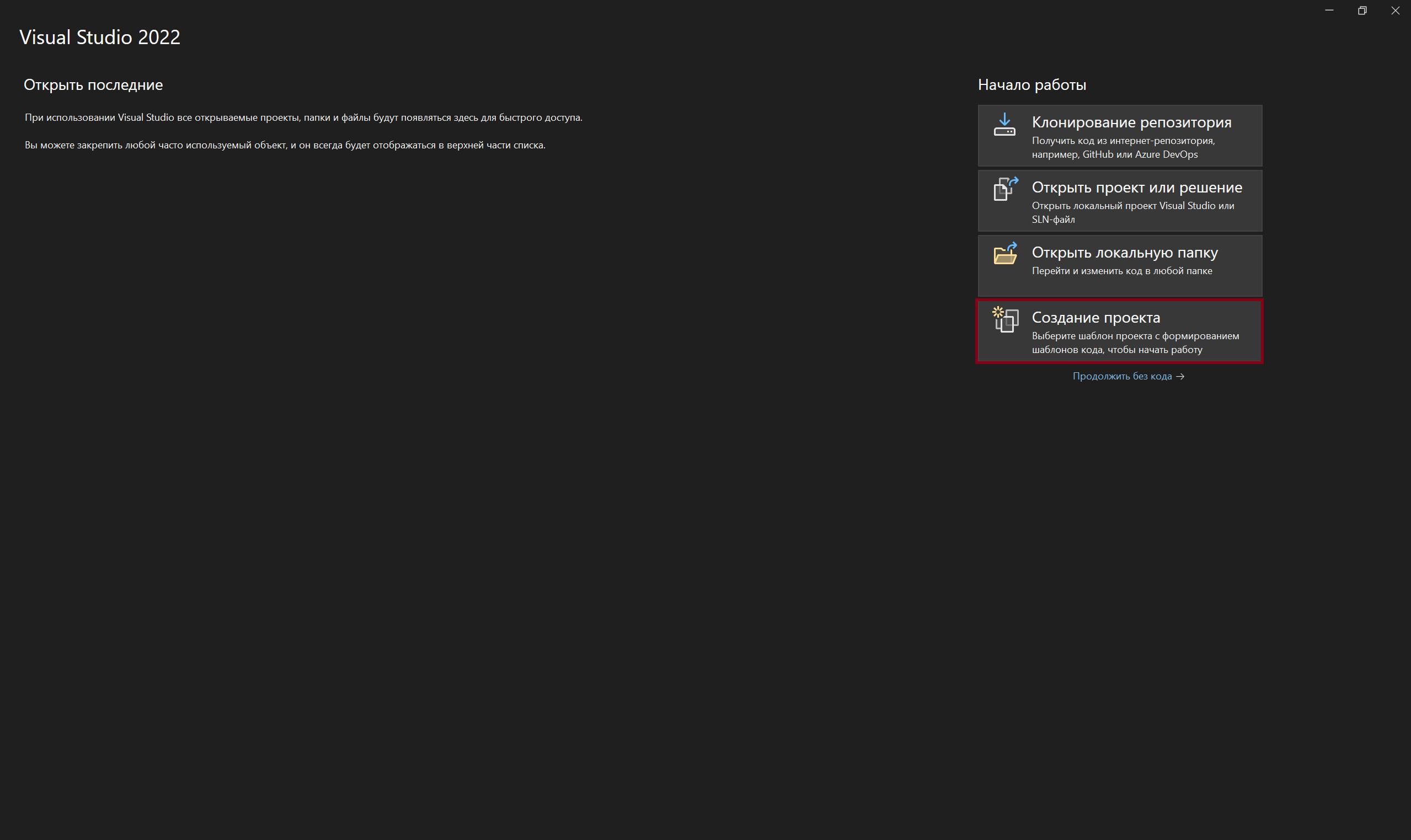


Рисунок 2 - Стартовое окно Visual Studio 2022

После этого открывается окно выбора шаблонов для нового проекта. В этом окне мы задаем параметры: выбираем язык программирования «**C#»**, платформу «**Windows»** и тип проекта «**Рабочий стол»**.

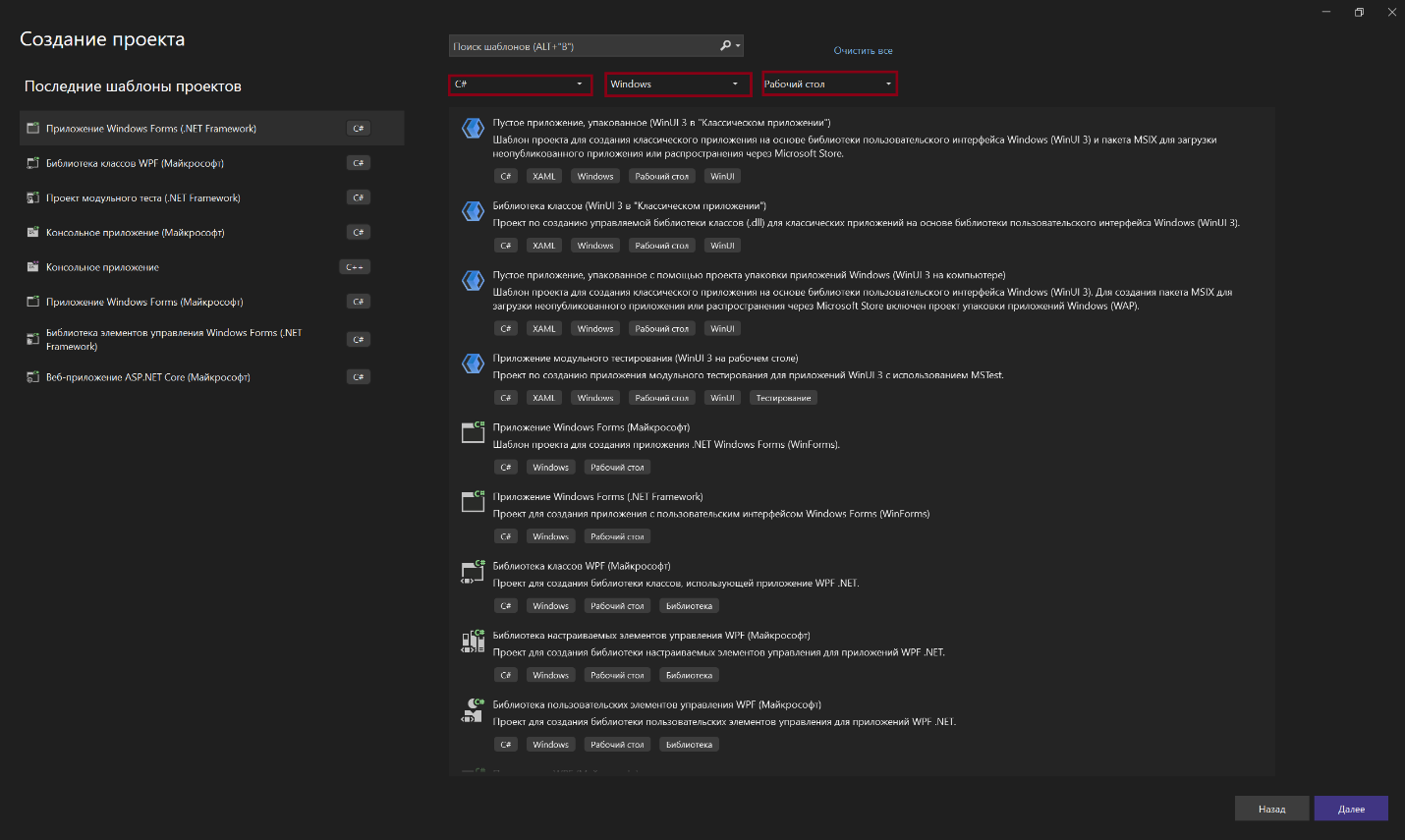


Рисунок 3 - Окно выбора шаблона проекта в Visual Studio 2022

Далее, из доступных шаблонов, выбираем «Приложение Windows Forms (.NET **Framework)»**.

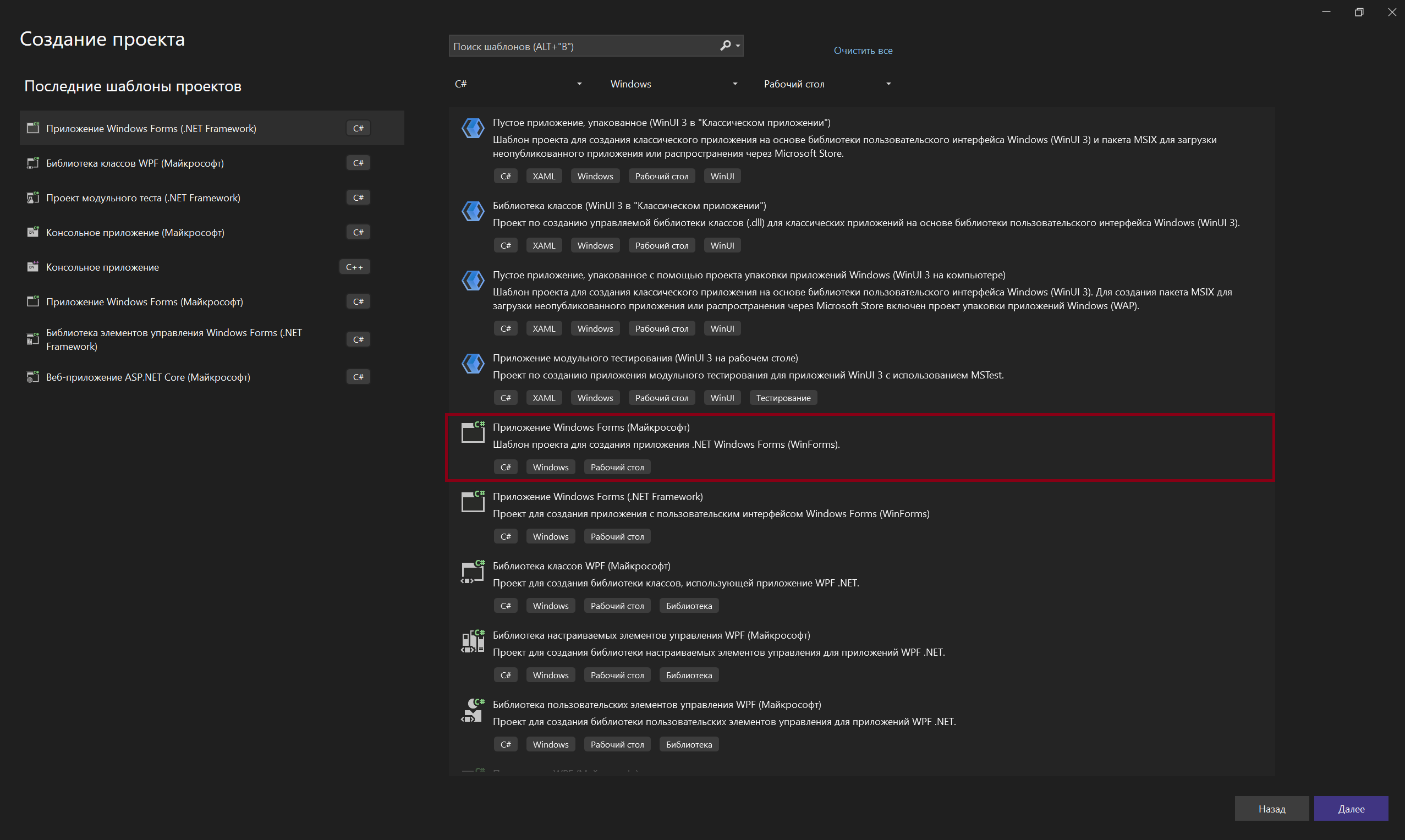


Рисунок 4 – Окно с выбором шаблона проекта Windows Forms

После выбора шаблона откроется окно, где нужно указать имя проекта, выбрать папку для его сохранения и задать имя решения. После заполнения всех полей нажимаем «Далее».

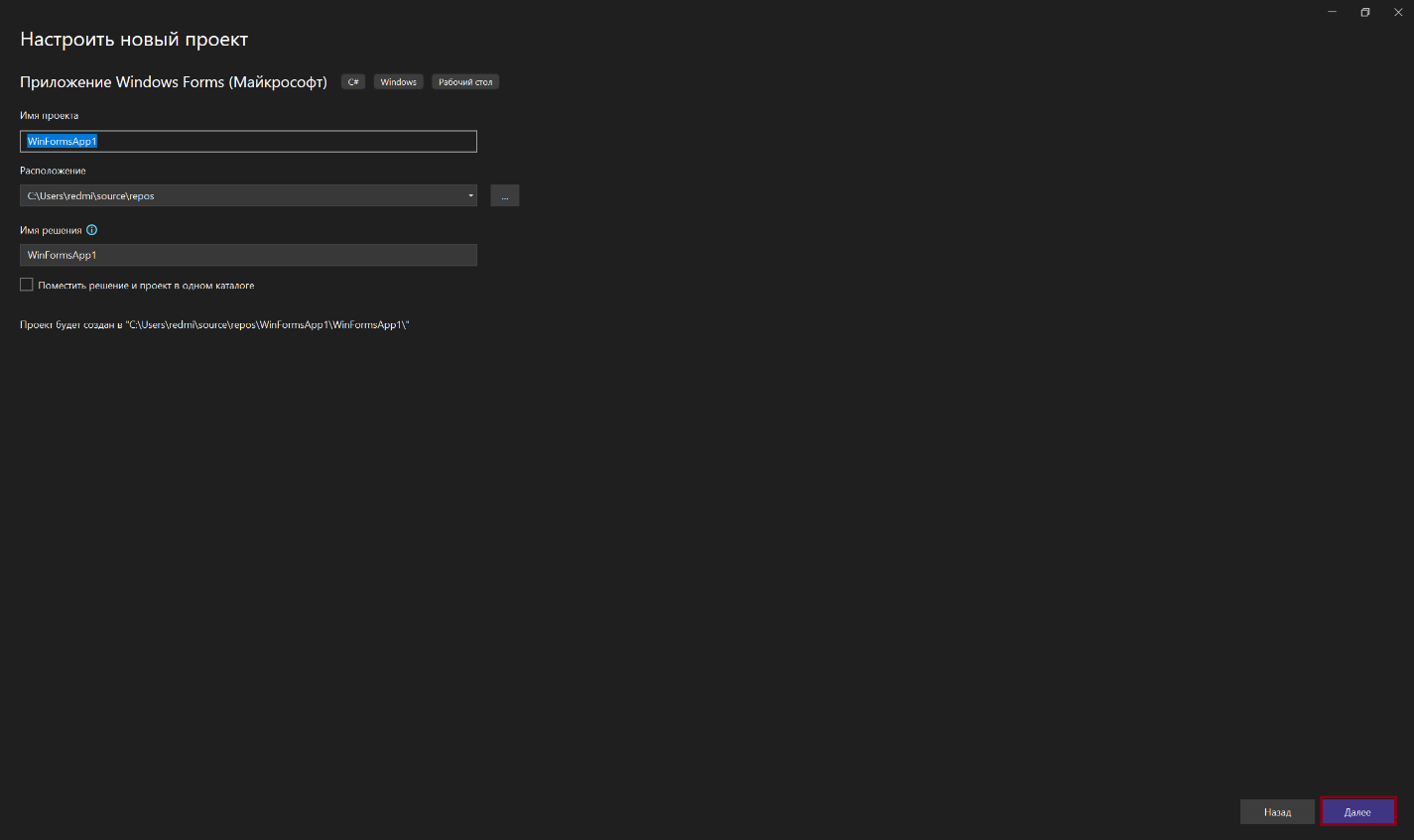


Рисунок 5 – Окно с Заданием имени проекта и его расположения

Далее выбираем платформу .NET 8.0 и нажимаем «Создать».

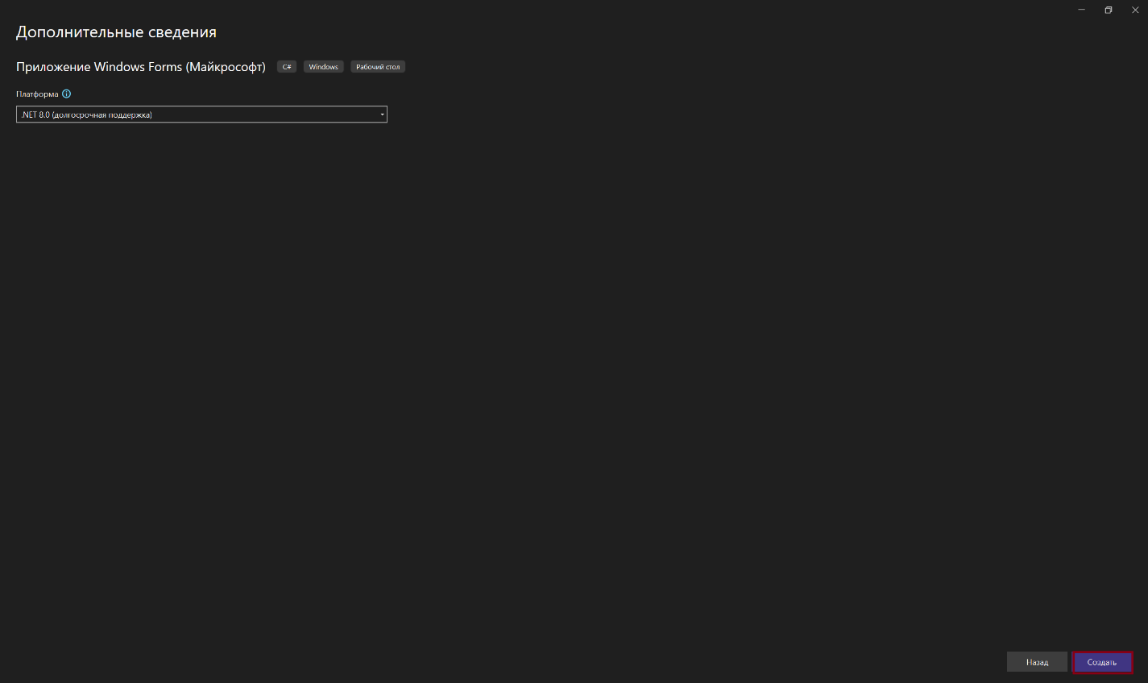


Рисунок 6 – Окно с выбором платформы

После завершения настройки проекта Visual Studio 2022 откроет основное рабочее пространство, где можно приступить к созданию пользовательского интерфейса. Чтобы добавить инструменты для работы с элементами интерфейса, в верхнем меню необходимо выбрать «Вид» → «Панель элементов».

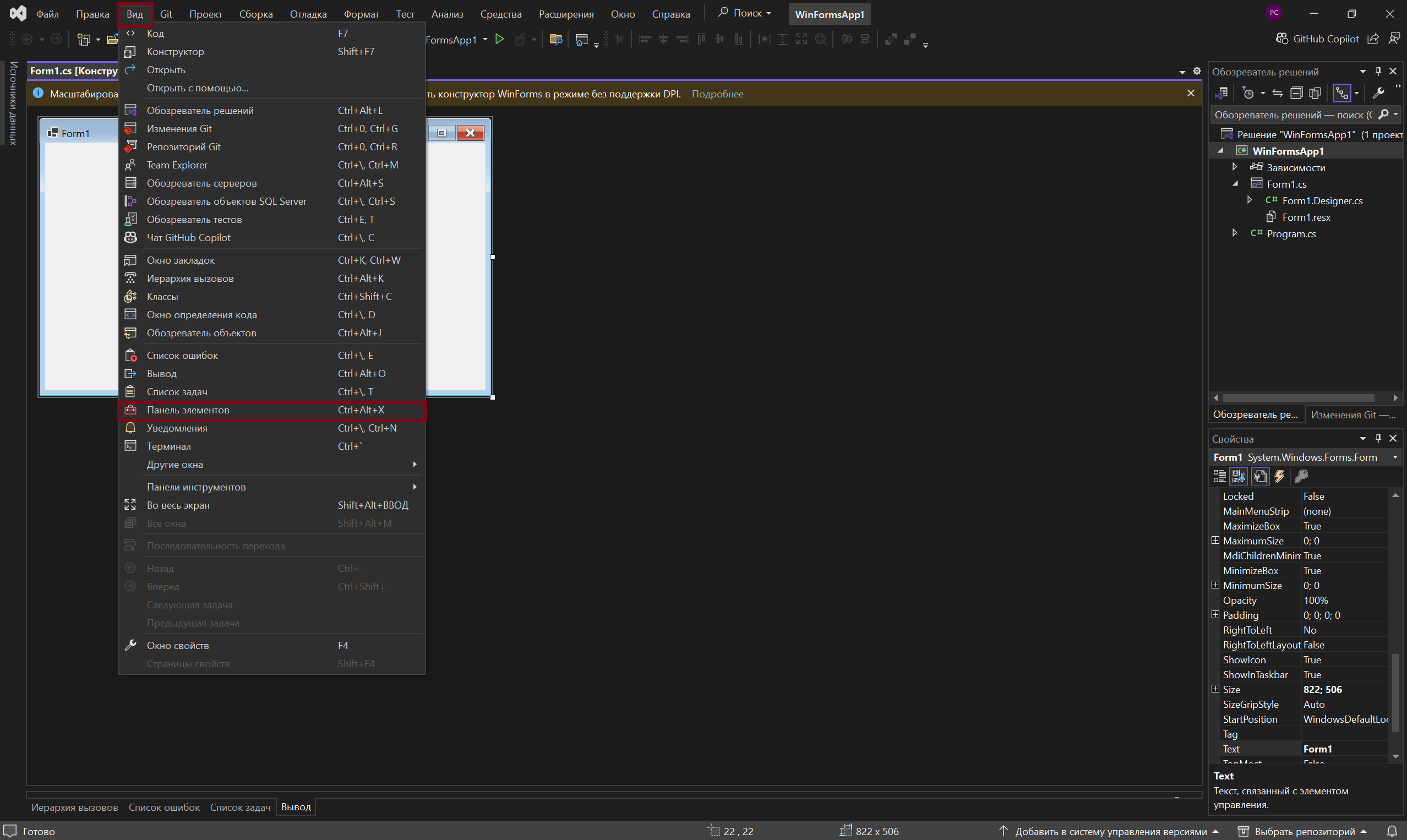


Рисунок 7 – Открытие панели элементов

В открывшемся окне с элементами интерфейса можно выбрать нужные компоненты, такие как кнопки, текстовые поля, списки и другие. Выбранные элементы просто перетаскиваются на форму **Form1**, которая служит основой для интерфейса. В окне с элементами интерфейса можно выбрать нужные компоненты, такие как кнопки, текстовые поля, списки и другие. Ниже представлена таблица с описанием основных элементов панели инструментов Visual Studio.

Таблица 1 – Элементы панели инструментов в Visual Studio 2022

|  |  |
| --- | --- |
| Элемент | Описание |
| Button | Кнопка. Используется для выполнения действий по нажатию |
| Label | Метка. Отображает текстовую информацию на форме |
| TextBox | Поле ввода текста. Позволяет пользователю вводить текст |
| CheckBox | Флажок. Позволяет выбирать одно или несколько значений из списка |
| RadioButton | Переключатель. Используется для выбора одного значения из группы |
| ComboBox | |  | | --- | | Выпадающий список. Содержит список элементов, из которых можно выбрать один | |
| ListBox | Список. Позволяет отображать набор элементов для выбора. |
| PictureBox | Элемент для отображения изображений (файлы форматов PNG, JPG и др.) |
| Panel | Контейнер для группировки других элементов управления. |
| GroupBox | Элемент для объединения других элементов в логическую группу с заголовком |
| DataGridView | Таблица. Используется для отображения и редактирования данных в табличной форме |
| ProgressBar | Индикатор выполнения. Показывает прогресс выполнения задач. |
| TabControl | Панель вкладок. Позволяет создавать вкладки для организации контента |
| Timer | Таймер. Позволяет запускать код через заданные интервалы времени |
| DateTimePicker | Элемент для выбора даты и времени. |
| Splitter | Разделитель. Позволяет изменять размер соседних панелей |

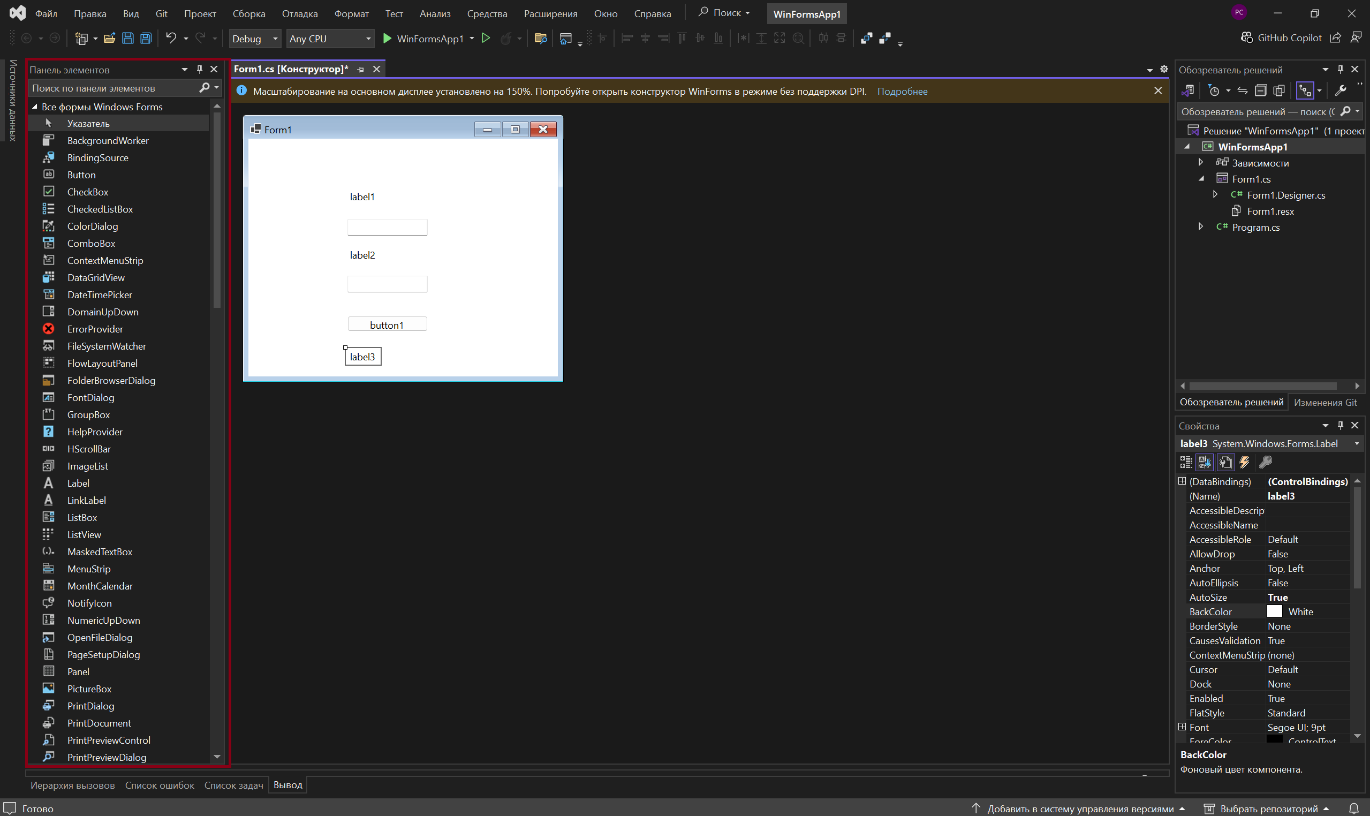


Рисунок 8 – Инструменты панели элементов для создания интерфейса

После того как элементы интерфейса размещены на форме, их можно настроить в окне «Свойства». Это окно позволяет гибко управлять характеристиками каждого элемента интерфейса. Благодаря этому, можно легко адаптировать элементы под нужды приложения, обеспечив их визуальную привлекательность и удобство использования.

Свойства в Visual Studio сгруппированы по категориям и позволяют разработчику быстро изменить любой аспект элемента интерфейса. Например, вы можете задать текст для кнопки, изменить цвет фона или текстовый шрифт, настроить выравнивание содержимого или указать параметры взаимодействия (такие как доступность или видимость).

Ниже представлена таблица, которая описывает основные свойства элементов в Visual Studio 2022 и их назначение.

Таблица 2 – Описание свойств элементов управления

|  |  |
| --- | --- |
| Свойство | Описание |
| Name | Уникальное имя элемента, используемое для обращения к нему в коде программы. |
| Text | Текст, отображаемый на элементе, например, текст кнопки или метки. |
| Size | Размер элемента, задается в виде ширины и высоты (в пикселях). |
| Location | Координаты положения элемента на форме, задаются значениями X и Y. |
| BackColor | Цвет фона элемента, например, кнопки или текстового поля. |

Продолжение таблицы 2

|  |  |
| --- | --- |
| ForeColor | Цвет текста или других элементов содержимого. |
| Font | Шрифт текста элемента, включая стиль (жирный, курсив), размер и тип шрифта. |
| Enabled | Определяет, доступен ли элемент для взаимодействия (значения true/false). |
| Visible | Указывает, отображается ли элемент на форме (true/false). |
| Anchor | Закрепляет элемент относительно краев формы, что позволяет ему менять положение при изменении размера формы. |
| Dock | Закрепляет элемент вдоль одного из краев формы или заполняет всю форму. |
| TabIndex | Устанавливает порядок переключения между элементами интерфейса при нажатии клавиши Tab. |
| TabStop | Определяет, можно ли выбрать элемент с помощью клавиши Tab (true/false). |
| TextAlign | Выравнивание текста внутри элемента, например, по центру или по краю. |
| BorderStyle | Устанавливает стиль границы элемента, например, FixedSingle, Fixed3D или None |
| Image | Изображение, которое отображается на элементе, например, на кнопке или PictureBox. |
| Cursor | Указывает тип курсора при наведении на элемент (например, стрелка или рука). |
| Maximum/Minimum | Устанавливает минимальное и максимальное значения для элементов, таких как ползунки или счетчики. |
| Value | Текущее значение элемента, например, для ProgressBar или NumericUpDown. |
| Multiline | Указывает, может ли элемент TextBox принимать текст в несколько строк. |
| PasswordChar | Символ, отображаемый вместо текста в TextBox, используемый для ввода пароля. |

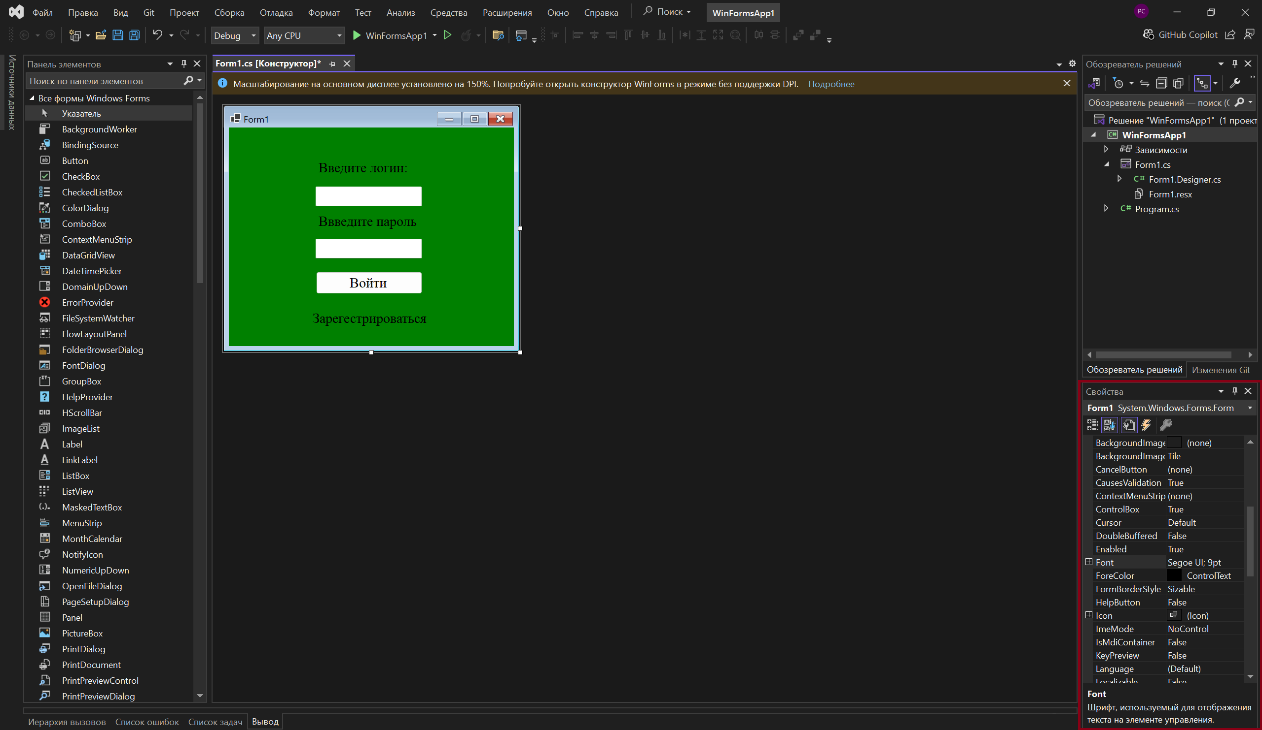


Рисунок 9 - Окно свойств элемента интерфейса

Когда интерфейс готов, можно запустить проект для проверки. Для этого нажимаем на зеленый треугольник в верхней части окна Visual Studio 2022. Откроется созданное приложение, где можно протестировать добавленные элементы

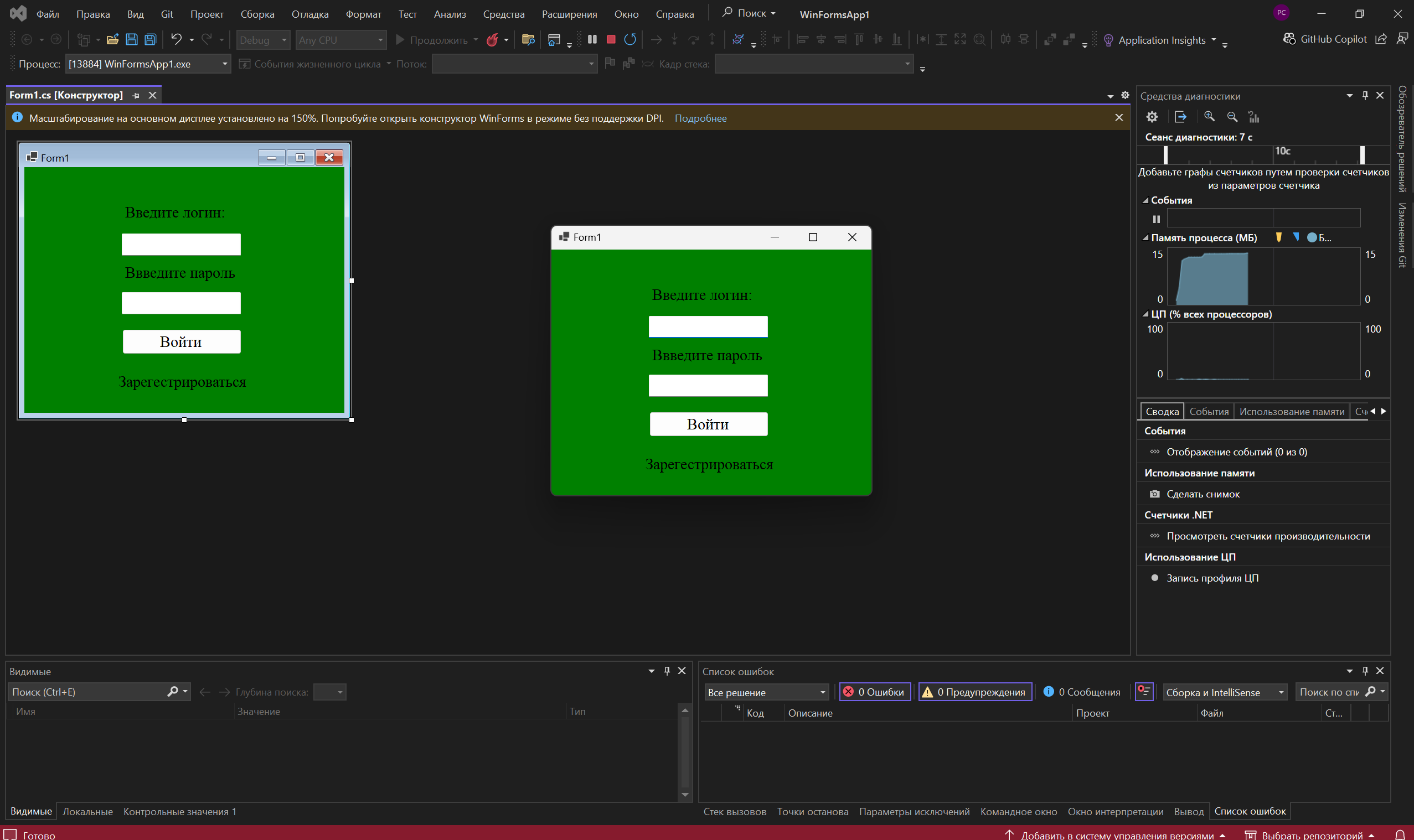


Рисунок 10 - Результат работы интерфейса в Visual Studio 2022

# 3.3 Интерфейс авторизации

В этом разделе описан интерфейс авторизации, который состоит из окна для входа, а также сообщений об ошибках в случае некорректного ввода данных.

На Рисунке 11 представлено основное окно для входа. Пользователь видит поле для ввода логина, поле для ввода пароля, кнопку «Войти», а также ссылку для перехода на страницу регистрации. Это окно позволяет пользователю войти в систему предоставляя свои учетные данные.

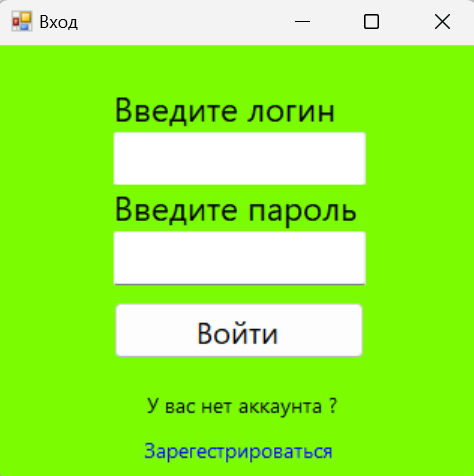


Рисунок 11 – Окно входа

Если пользователь пытается войти в систему, не заполнив поля логина и пароля, появляется предупреждение. На Рисунке 12 изображено сообщение об ошибке «Заполните все поля!». Это окно информирует пользователя о необходимости заполнить все обязательные поля перед входом в систему.

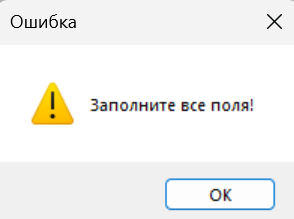


Рисунок 12 – Сообщение о не введении данных

В случае, если введенные данные некорректны, система выдает другое сообщение. Рисунок 4 демонстрирует окно с уведомлением «Неверный логин или пароль». Это окно информирует пользователя о том, что введенные данные не соответствуют учетным записям в базе данных.

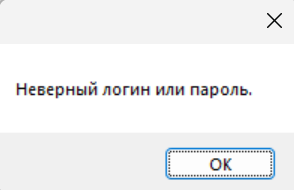


Рисунок 13 – Сообщение о введении некорректных данных

# 3.4 Интерфейс регистрации

Интерфейс регистрации позволяет новым пользователям создать учетную запись в системе. Окно регистрации содержит основные поля для ввода данных, такие как логин, пароль и подтверждение пароля. На форме также есть кнопка «Зарегистрироваться», которая запускает процесс проверки данных и создания учетной записи. На **Рисунке 1** представлено основное окно регистрации.

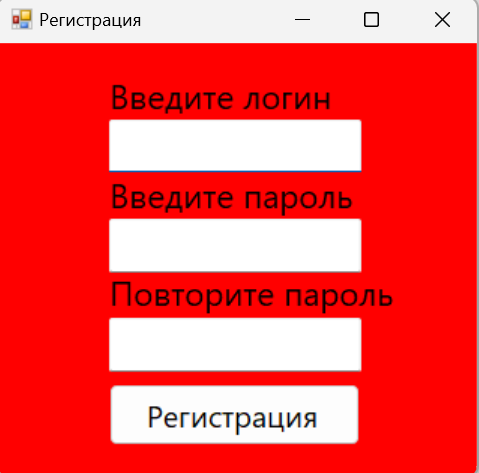


Рисунок 14 – Окно регистрации

Пользователь вводит данные в соответствующие поля и нажимает кнопку "Зарегистрироваться". Если какие-либо поля ввода остаются пустыми, система отображает предупреждающее сообщение: "Заполните все поля!", как показано на Рисунке 12. Это сообщение информирует пользователя о необходимости заполнения всех полей. Такой подход помогает избежать ошибок, связанных с неполным вводом данных, и делает процесс регистрации более понятным и удобным.

Если введенные пароли в полях "Пароль" и "Подтверждение пароля" не совпадают, появляется сообщение: **"Пароли не совпадают", как показано на Рисунке 15.** Это позволяет предотвратить создание учетной записи с неправильным паролем.

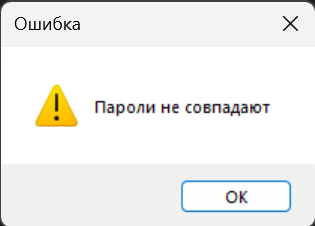


Рисунок 15 – Сообщение с ошибкой что пароли не совпадают

На Рисунке 16 показано сообщение о существующем логине. В случае, если пользователь пытается зарегистрироваться с уже занятым логином, система уведомляет: "Такой логин уже существует." Это исключает возможность дублирования учетных записей и помогает пользователю подобрать уникальный логин.

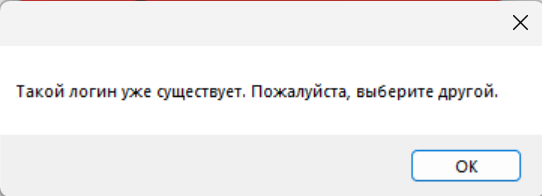


Рисунок 16 - Сообщение об ошибке дублирования логина

После успешного ввода всех данных и прохождения проверки система создает учетную запись и выводит сообщение: **"Регистрация успешно завершена".** На **Рисунке 17** продемонстрировано уведомление об успешной регистрации.

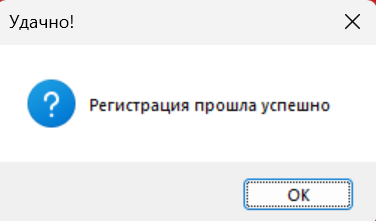


Рисунок 17 – Сообщение с успешной регистрацией

После успешной регистрации учетной записи, система автоматически открывает окно для подачи заявки на одобрение администраторами. Это окно предназначено для подтверждения регистрации и предоставления пользователю полного доступа к функционалу системы после проверки администраторами.

На Рисунке 18 представлено окно подачи заявки. Пользователь видит форму, в которой необходимо заполнить дополнительные данные, такие как ФИО, контактную информацию и другие сведения, требуемые системой. После заполнения всех полей пользователь нажимает кнопку "Подать заявку".

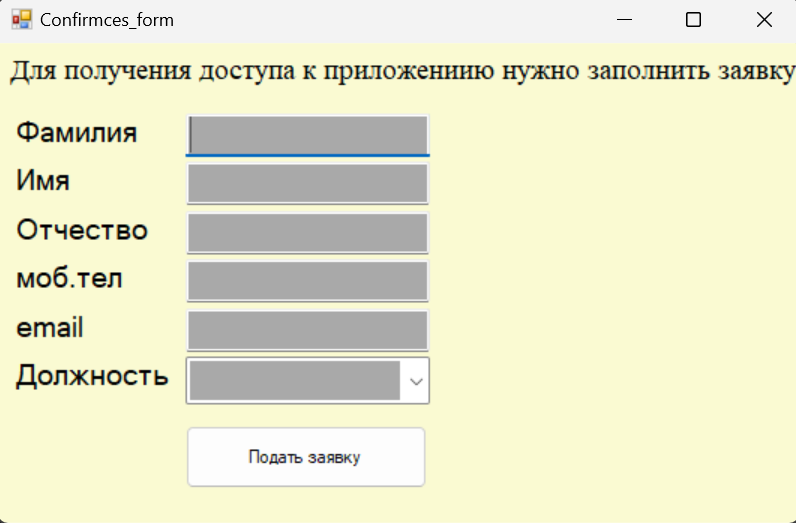


Рисунок 18 – Окно с подачей заявки

После успешной отправки заявки система выводит уведомление: "Заявка успешно отправлена. Ожидайте подтверждения администратора.". До одобрения заявки пользователь не может взаимодействовать приложением.

Однако, если введенные данные оказались некорректными, например, если поле с контактной информацией или обязательными данными осталось пустым, система отображает сообщение об ошибке, как показано на Рисунке 19. В этом случае пользователю предлагается проверить корректность ввода данных, и форма подачи заявки не будет отправлена до тех пор, пока все поля не будут правильно заполнены. Это помогает предотвратить ошибки и гарантирует, что только корректные данные попадут в систему.

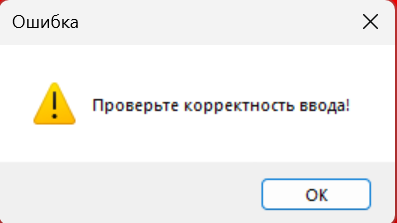


Рисунок 19 – Сообщение о проверке корректности ввода

Такой механизм подачи заявки позволяет администраторам контролировать процесс регистрации новых пользователей, исключая доступ к системе случайных или неподтвержденных учетных записей. Этот процесс также способствует улучшению безопасности и целенаправленному управлению доступом в систему.

# 3.5 Интерфейс для студентов

**Рисунок 20** представляет собой главный экран интерфейса студента. В верхней части окна расположены вкладки «Профиль», «Материалы» и «Задания», которые позволяют пользователю перемещаться по разделам приложения. В разделе «Профиль» отображаются персональные данные студента: имя, фамилия, контактная информация (почта и номер телефона), а также информация о группе и курсе. Также указаны данные о преподавателе, курсе и других предметах, которые изучает студент.

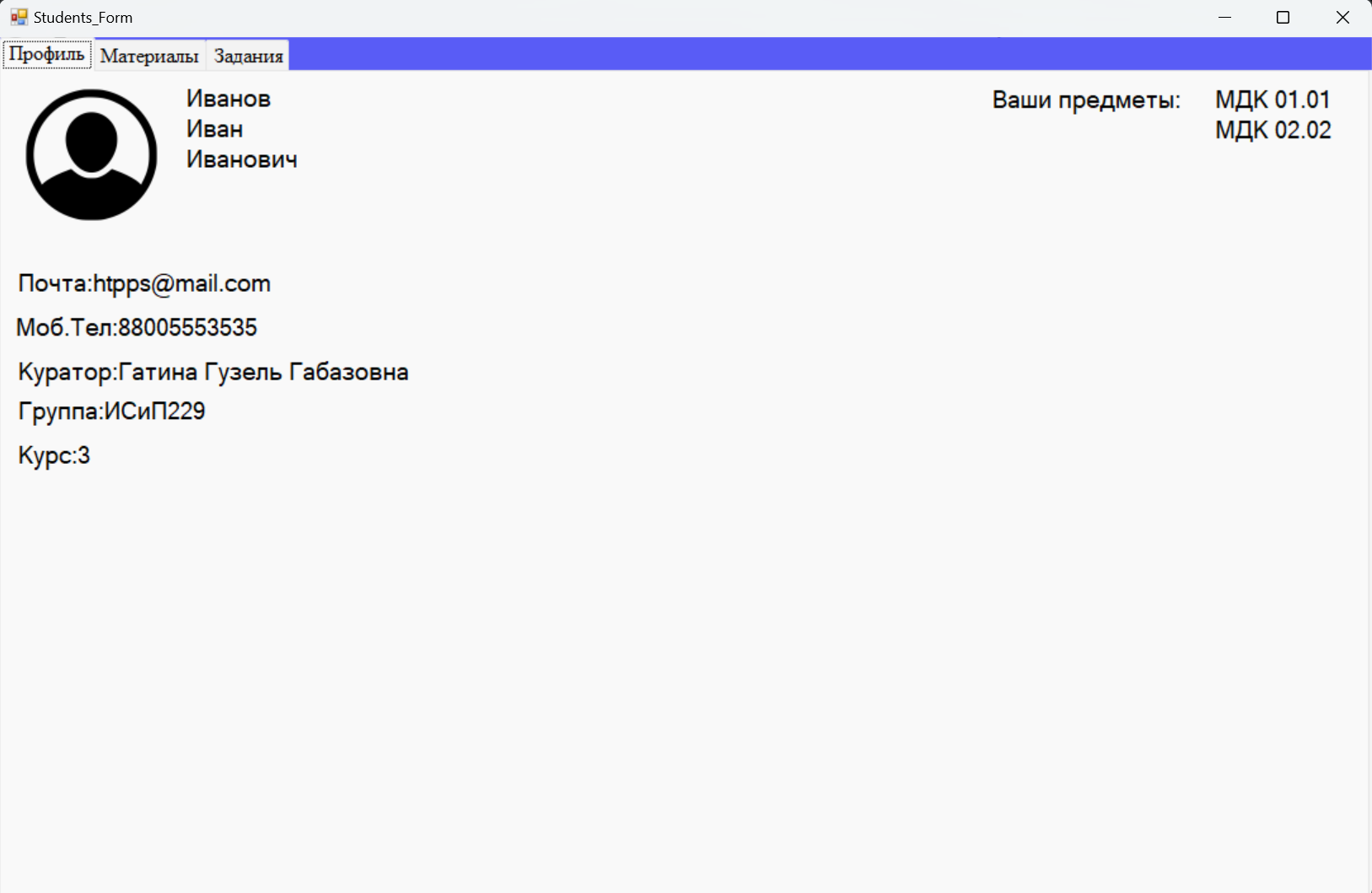


Рисунок 20 – Интерфейс профиля студентов

Рисунок 21 представляет собой экран "Материалы". На этом экране студенты могут выбрать предмет и тему, а также увидеть краткую информацию о выбранных материалах. В левой части экрана расположено окно с названиями тем. Также присутствует кнопка для скачивания материалов. Это позволяет студентам легко находить нужные ресурсы и скачивать их для дальнейшего изучения.

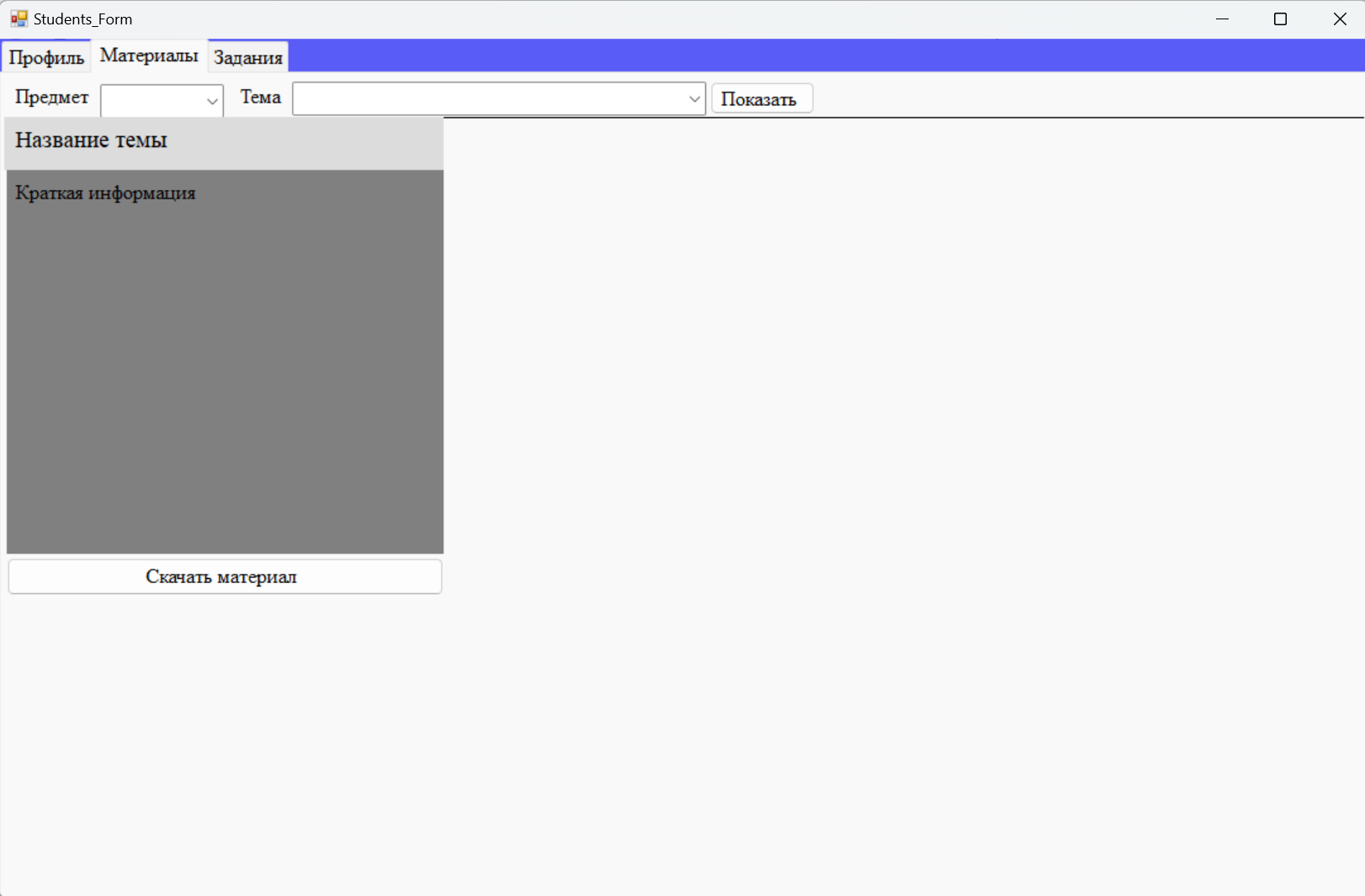


Рисунок 21 - Интерфейс материалов студента

На рисунке 22 представлен интерфейс, позволяющий студентам отправлять выполненные задания. Основные элементы интерфейса включают: выбор предмета и дата отправленного задания. Форма отправки задания представлена с кнопкой выбора файла для загрузки и отправки ответа. Основная цель данного интерфейса — обеспечить студентам простой и удобный способ загрузки и отправки выполненных работ, с указанием статуса выполнения.

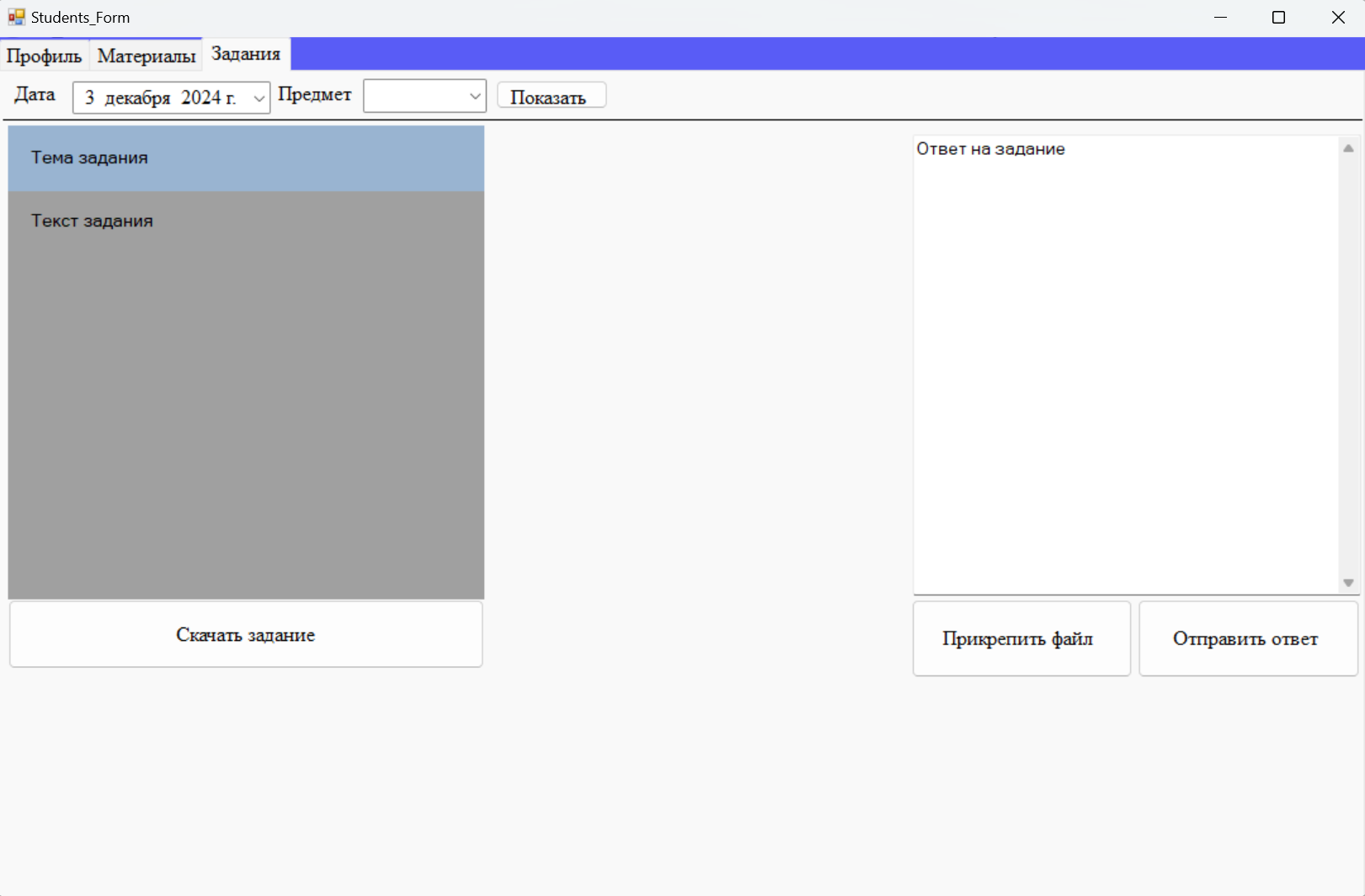


Рисунок 22 - Интерфейс заданий студентов

# 3.6 Интерфейс для преподавателей

На рисунке 23 изображен профиль преподавателей в котором отображается контактная информация, предметы и группы. Также добавлены кнопки для добавления и удаление групп и предметов.

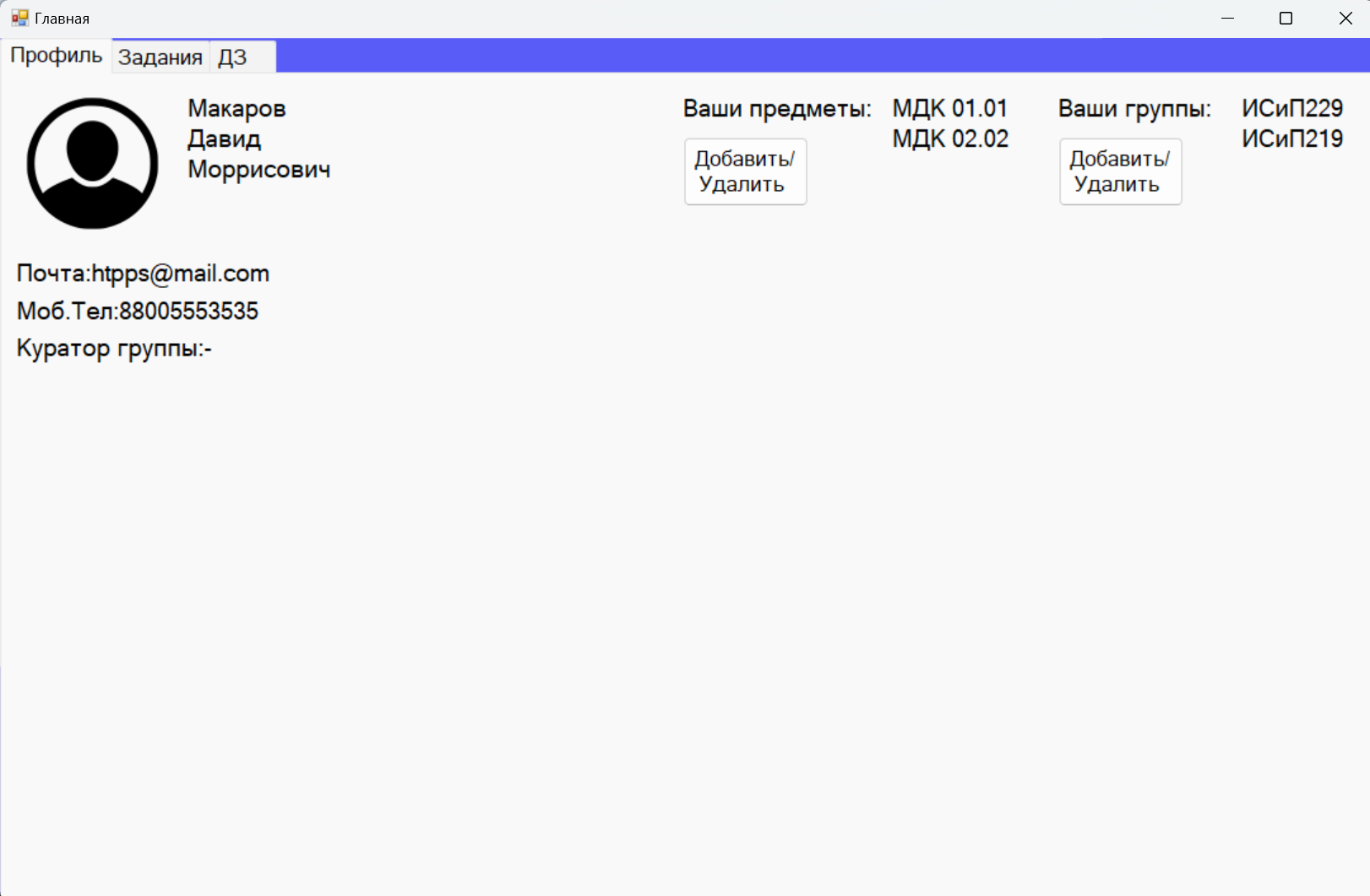


Рисунок 23 – Интерфейс профиля преподавателя

На рисунке 24 изображено окно с созданием учебных тем и заданий. Для прикрепления файлов присутствуют специальные кнопки. Для отправки задания используется кнопка отправить

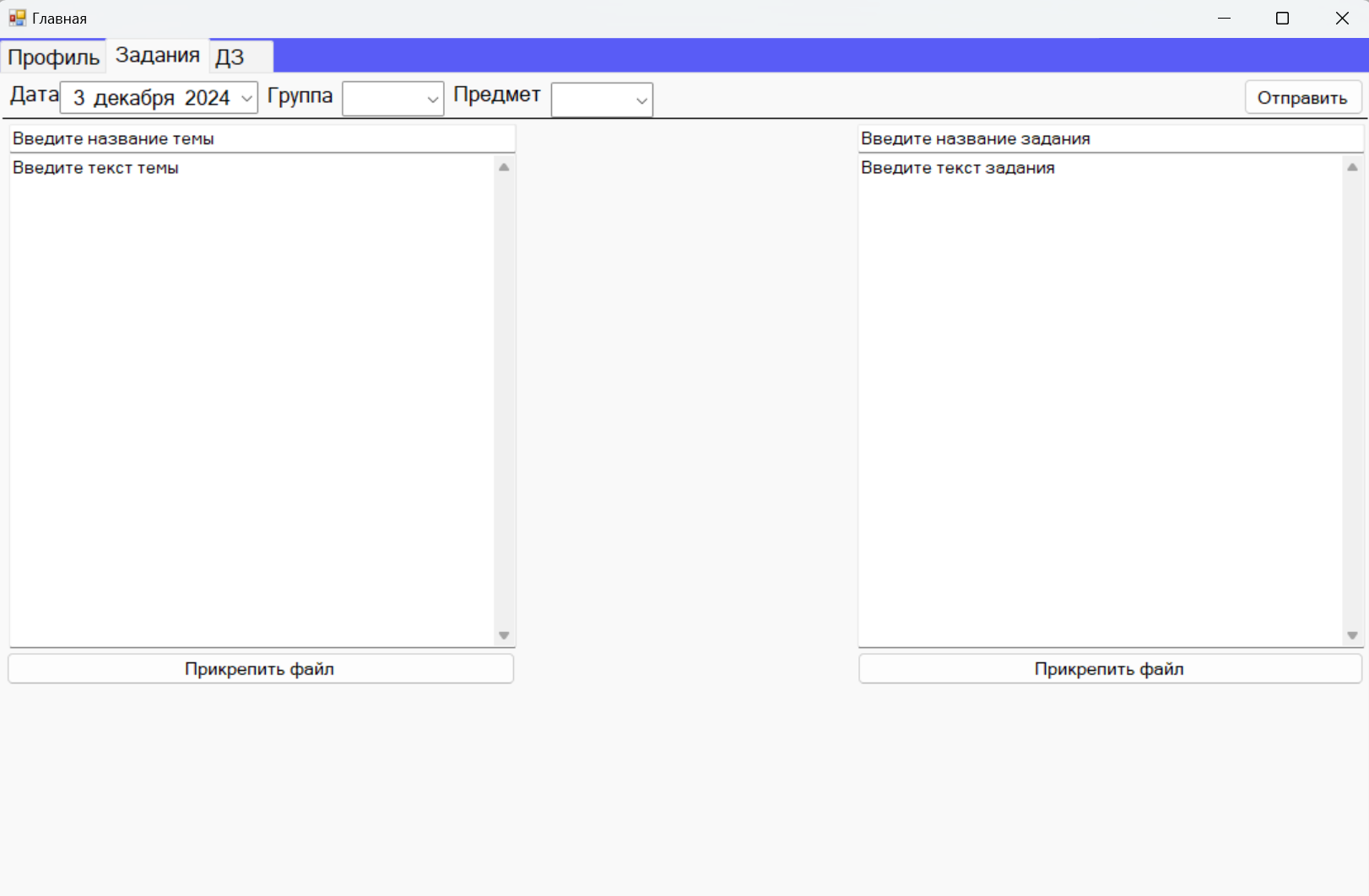


Рисунок 24 – Интерфейс с отправкой ДЗ и учебного материала

На Рисунке 25 изображен интерфейс с проверкой задания в котором нужно будет выбрать группу предмет и тему задания для появления списка студентов и выставления оценок. В правой части отображается тема задания и краткий ответ на задания можно добавить комментарий на ответ студентов.

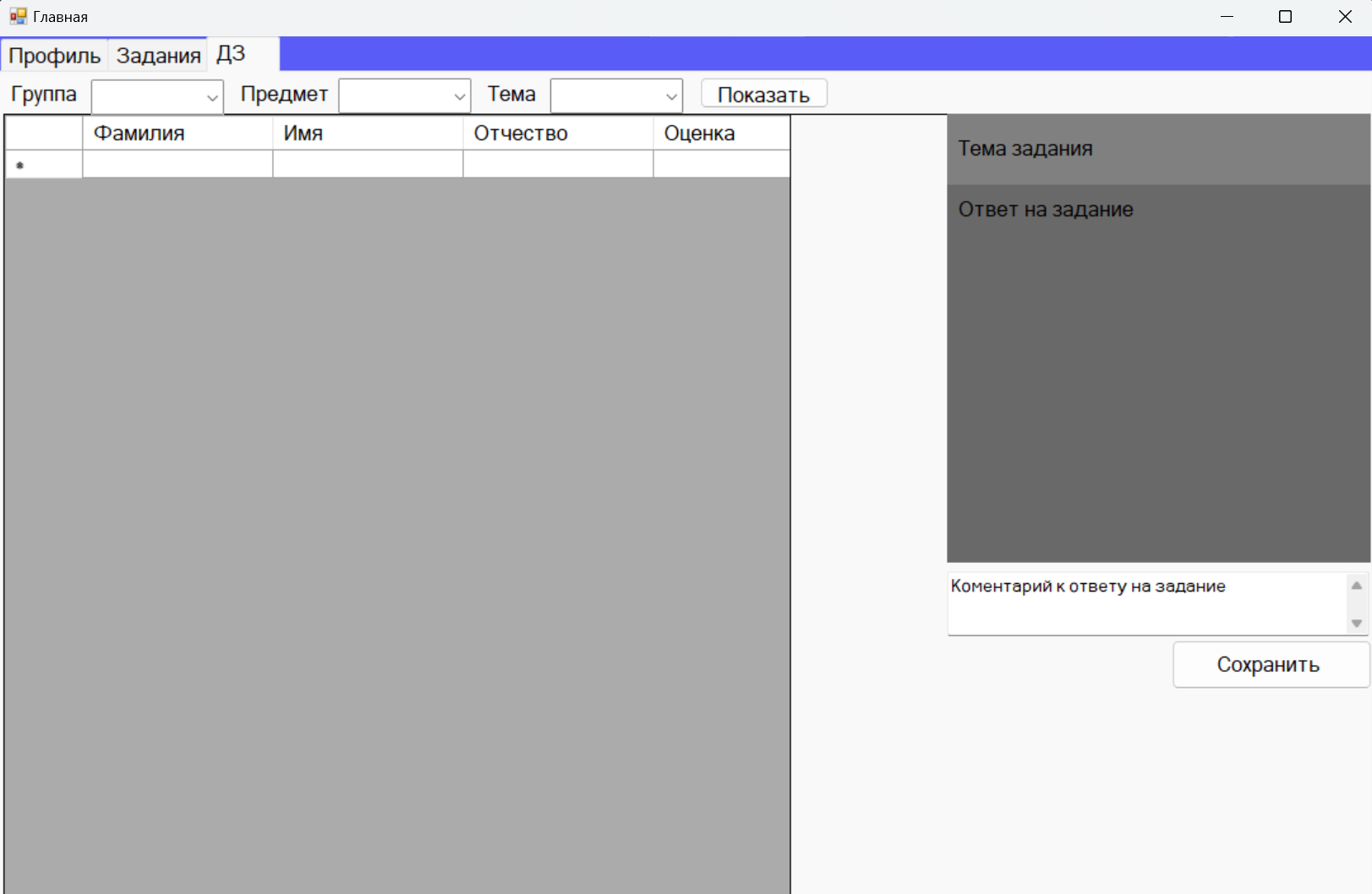


Рисунок 25 – Интерфейс с проверкой и выставления оценок

# 3.7 Интерфейс администраторов

На рисунке 26 изображен интерфейс профиля администратора.

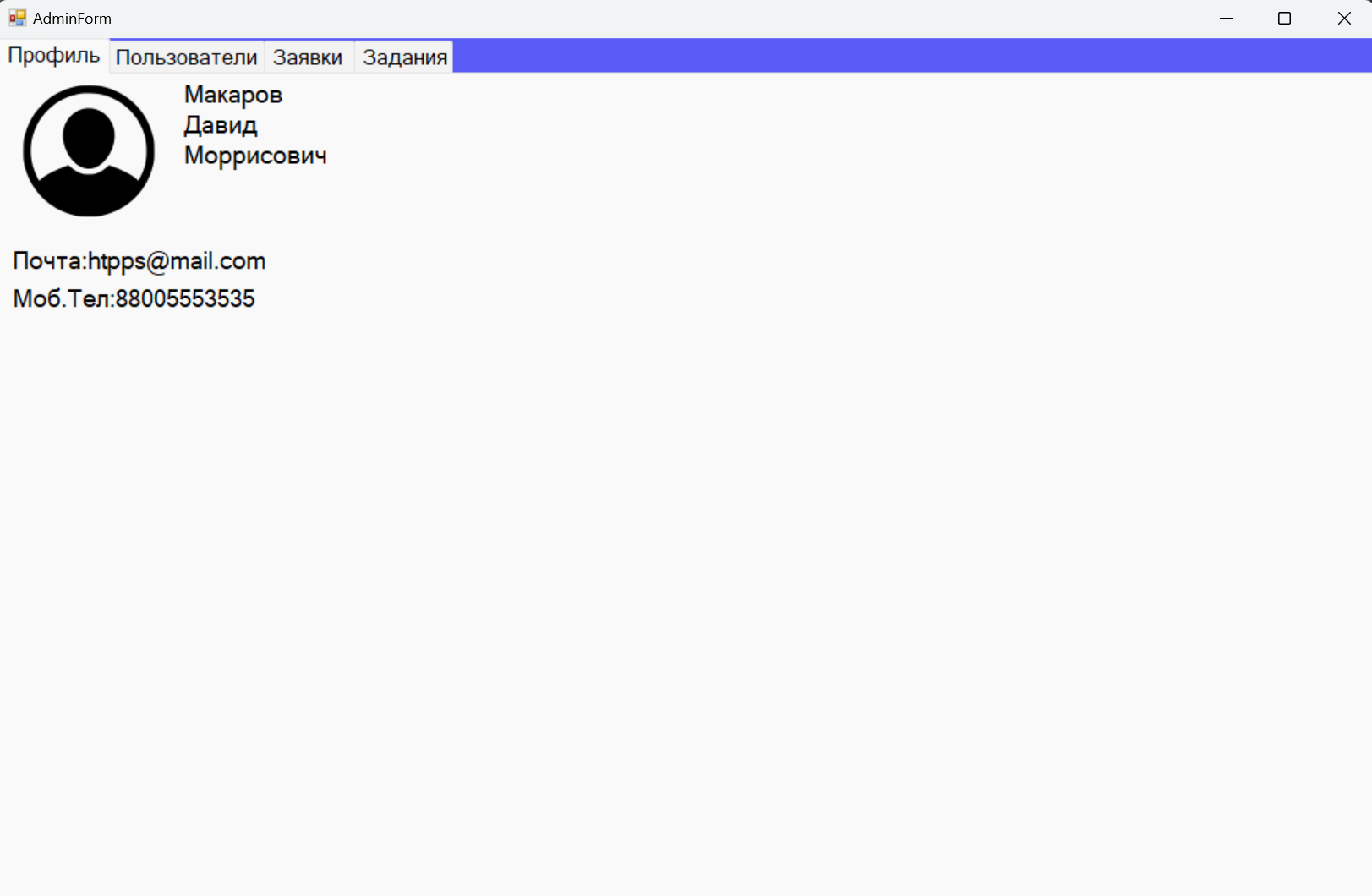


Рисунок 26 – Интерфейс профиля администратора

На рисунке 27 изображен интерфейс с списком всех пользователей и возможности удаления пользователей.

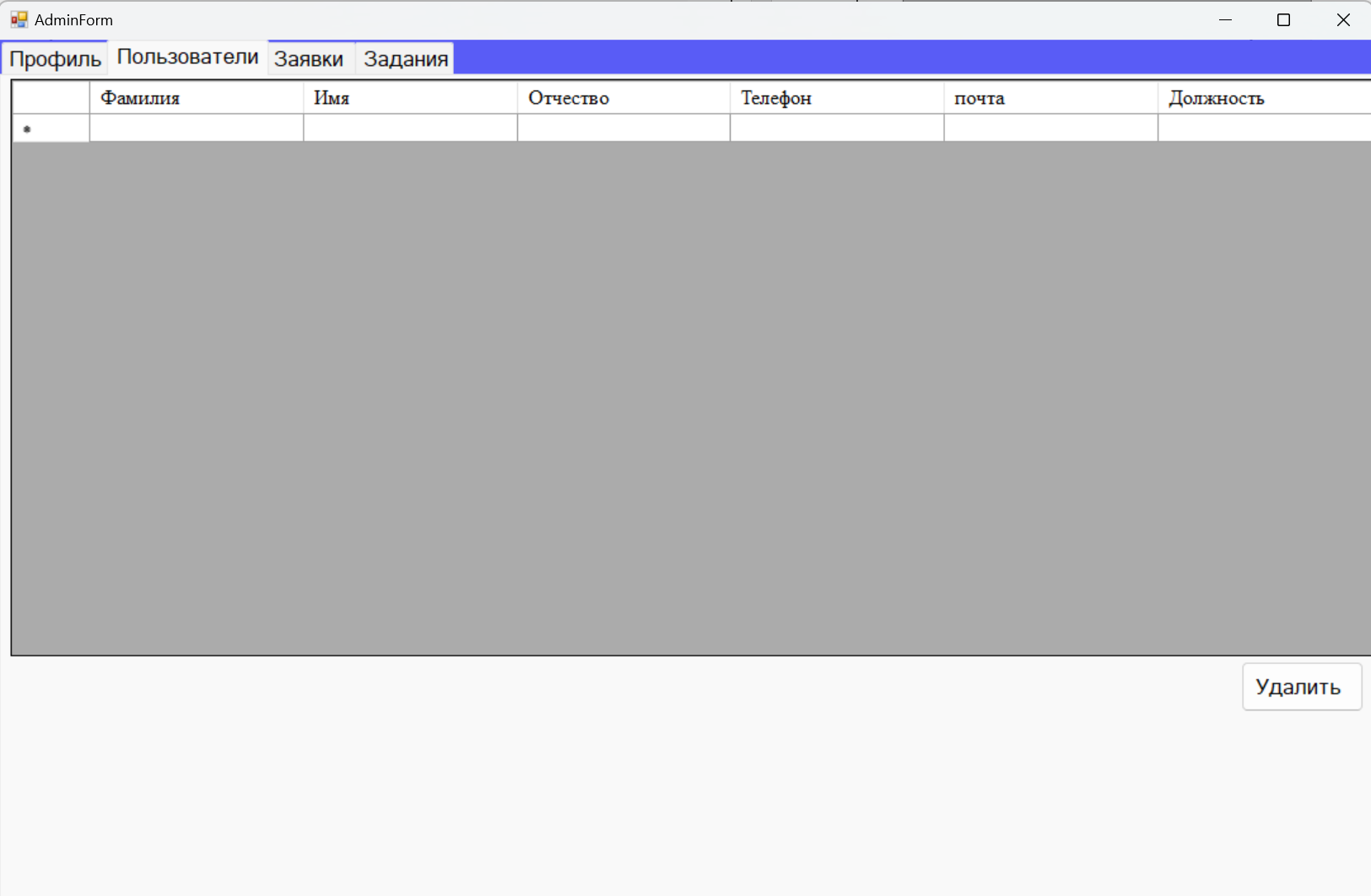


Рисунок 27 – Интерфейс для просмотра и удаления пользователей

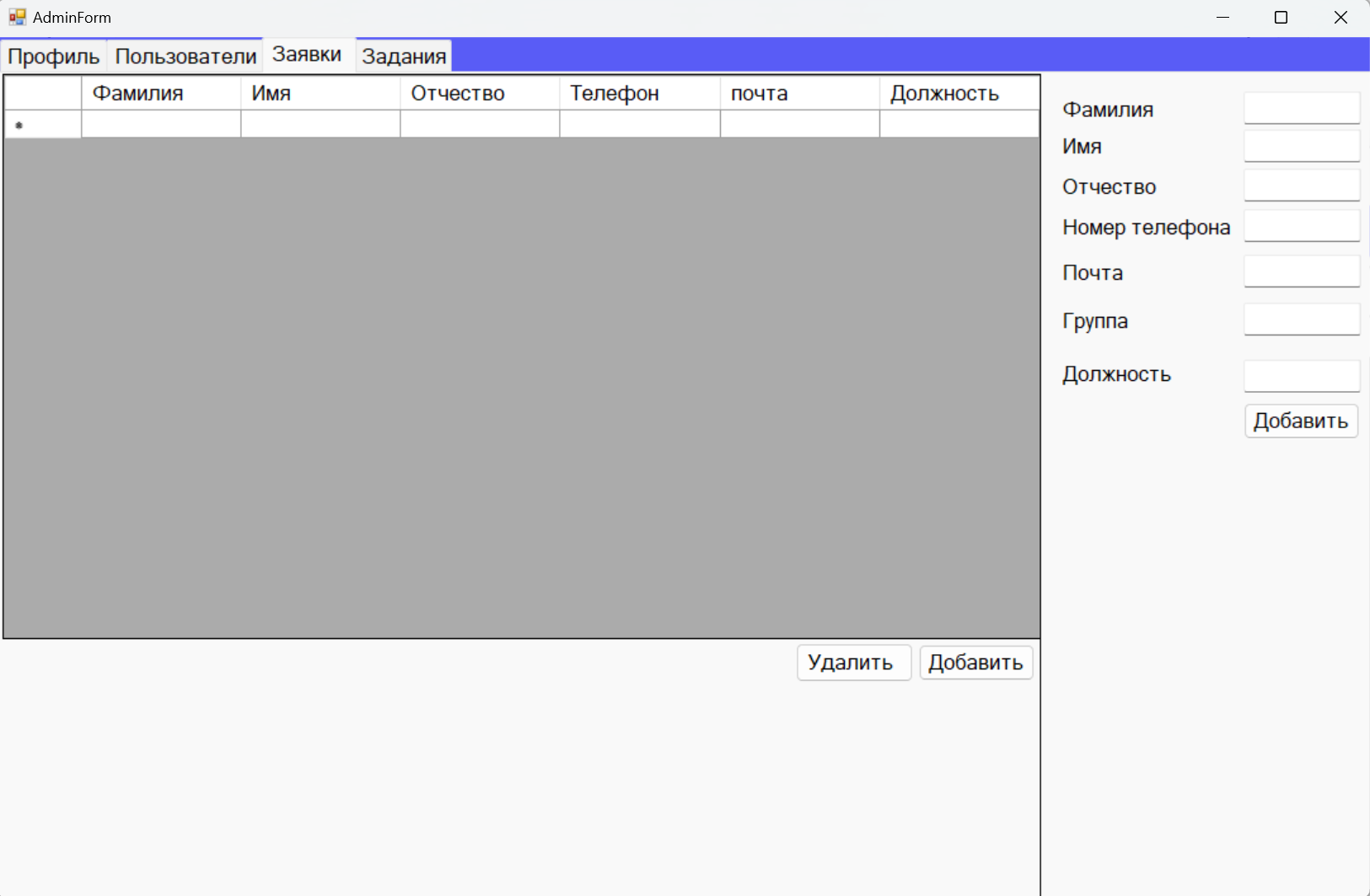
На Рисунке 28 изображен интерфейс с поданными заявками для доступа к приложению с помощью кнопок можно добавить пользователя либо удалить заявку для отказано заявки.

Рисунок 28 – Интерфейс с поданным заявками для доступа к приложению

На рисунке 29 изображен интерфейс с просмотром ответов студентов на задания и просмотра выставленных оценок.

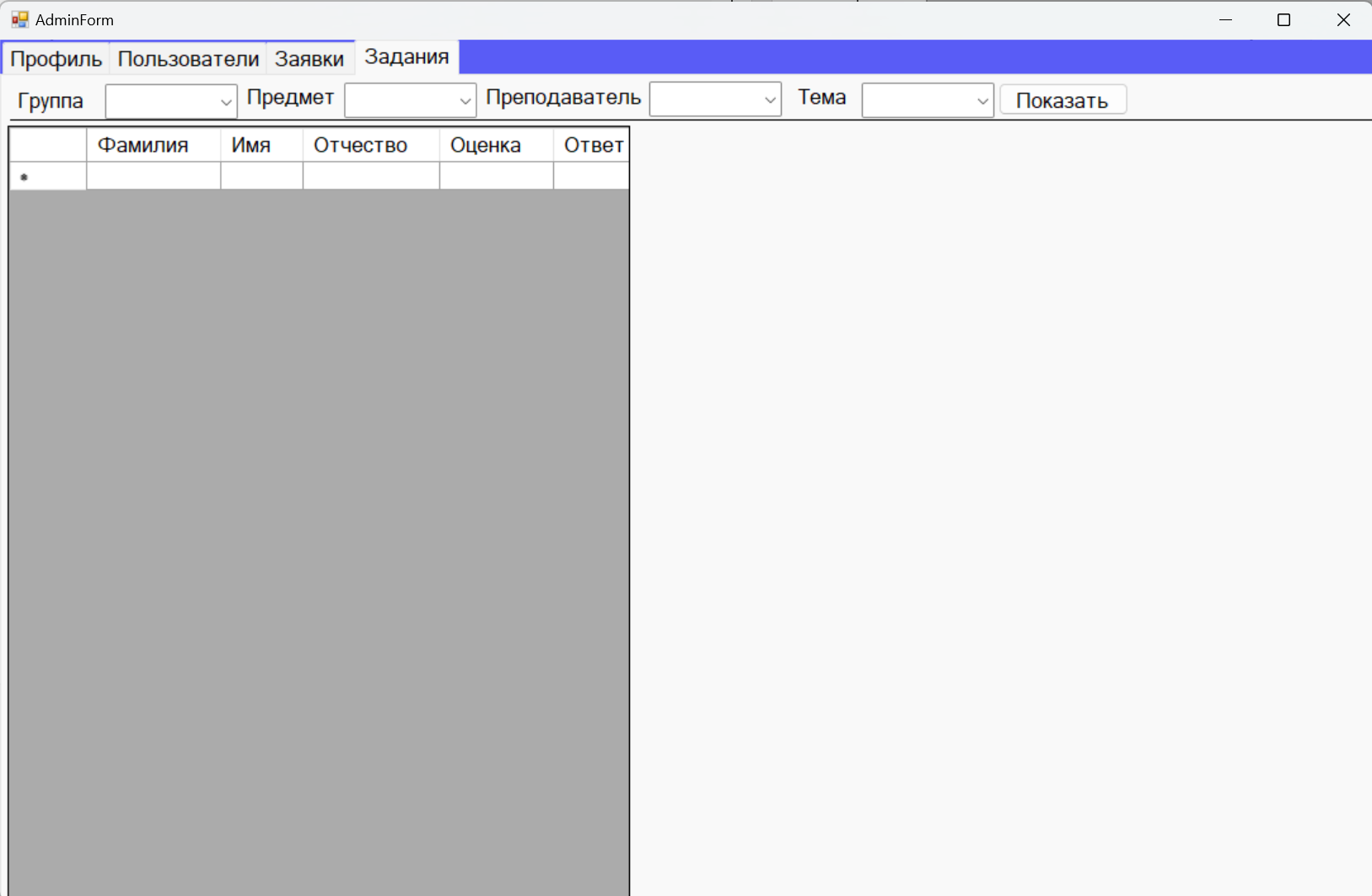


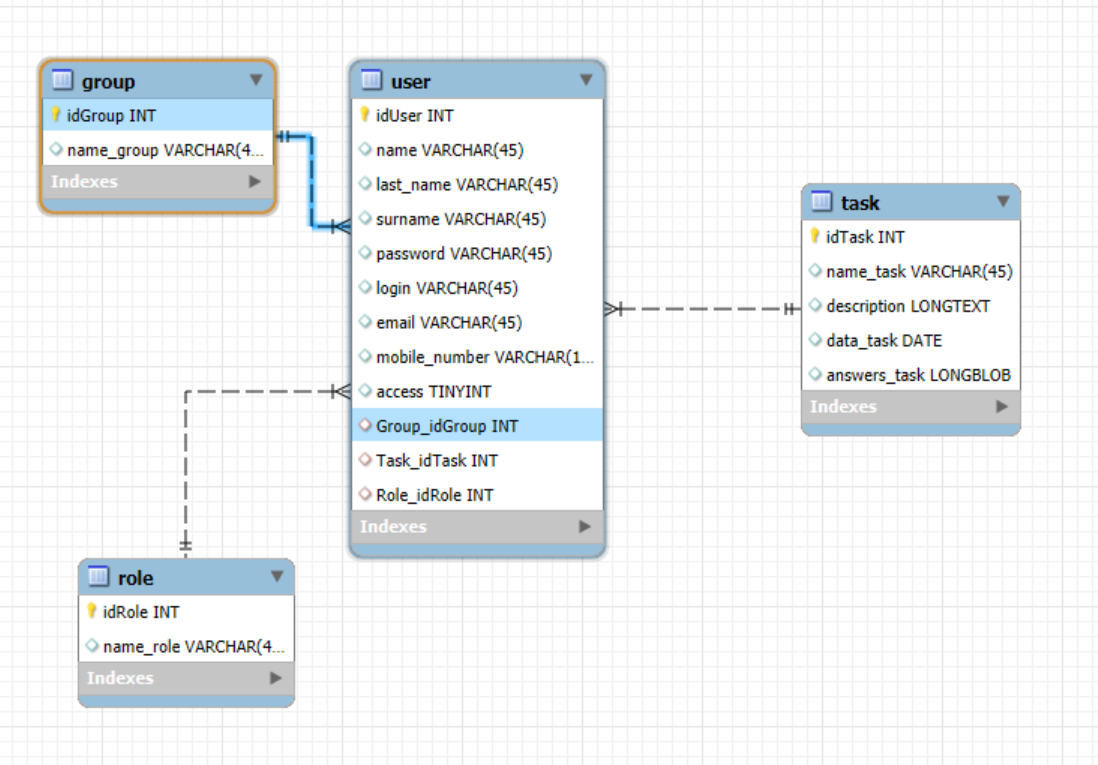
Рисунок 29 – Интерфейс с просмотром выполненных заданий и оценок

# 4 Проектирование базы данных

Проектирование базы данных является важнейшим этапом разработки приложения, так как от него зависит эффективность работы системы, а также возможность обработки и хранения данных. В данном разделе будет рассмотрена структура базы данных, а также представлена **ER-диаграмма** (Entity-Relationship Diagram), описывающая взаимосвязи между сущностями.

# 4.1 Описание ER-диаграммы

Описание: ER-диаграмма (диаграмма «сущность-связь») — это графическое представление структуры базы данных. Она показывает основные сущности (таблицы) и связи между ними. На этой ER-диаграмме мы видим ключевые сущности: «Пользователь» (User), «Роль» (Role), «Задание» (Task), «Группа» (Group), а также ряд связующих таблиц для реализации отношений «многие ко многим».

Рисунок 30 – ER-диаграмма базы данных

# Заключение

В рамках данного курсового проекта была успешно достигнута цель по проектированию приложения «Дистант», ориентированного на обеспечение эффективного дистанционного взаимодействия между студентами, преподавателями и администраторами образовательных учреждений. Для достижения этой цели последовательно были решены все поставленные задачи: проведен анализ потребностей целевой аудитории, выявлены ключевые функции, необходимые для работы в дистанционном формате, определены функциональные и нефункциональные требования к приложению, включая удобство интерфейса, безопасность и производительность, и, наконец, спроектирован интуитивно понятный и функциональный пользовательский интерфейс для всех категорий пользователей.

# Список использованных источников

1. Шмидт, А. В. Проектирование пользовательских интерфейсов: теория и практика. Москва: Вилма, 2019.
2. Петров, И. С. Основы разработки программных приложений. Санкт-Петербург: БХВ-Петербург, 2020.
3. Дьяков, С. П. Технологии баз данных для разработчиков. Москва: Наука, 2018.
4. Бенедиктов, В. А. Разработка и тестирование программных приложений. Киев: Наукова думка, 2017.
5. Козлов, М. И. Технологии и инструменты для создания приложений на платформе .NET. Минск: Технопресс, 2021.
6. Основы программирования на C#. Udemy. Доступно на: <https://www.udemy.com>
7. Проектирование интерфейсов с использованием .NET. Coursera. Доступно на: <https://www.coursera.org>
8. Основы работы с MySQL для разработчиков. LearnSQL. Доступно на: <https://learnsql.com>
9. Stack Overflow. Ответы на вопросы по .NET и программированию. Доступно на: <https://stackoverflow.com>
10. Habr. Статьи о разработке приложений и UI/UX дизайне. Доступно на: <https://habr.com>
11. Томпсон, М. Проектирование баз данных: теории и методы. Москва: Альфа, 2019.
12. Мур, Л. Практическое руководство по .NET Framework. Москва: Синергия, 2018.
13. Веб-ресурс Microsoft. Документация по .NET (2021). Доступно на: <https://docs.microsoft.com/ru-ru/dotnet>
14. О'Рейли, Т. C# для начинающих. Санкт-Петербург: Питер, 2021.м