

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Кубанский государственный технологический университет»
(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности
Кафедра информационных систем и программирования
Направление подготовки 09.03.03 «Прикладная информатика»
Профиль «Разработка и внедрение прикладных информационных систем»

КУРСОВОЙ ПРОЕКТ

по дисциплине: Веб-технологии

на тему: «Система управления студенческими задачами и дедлайнами»

Выполнил студент группы 22-КБ-ПИ1 М.С. Мальцев

Допущен к защите 05.06.2025

Руководитель (нормоконтролер) работы  ассистент Р.И. Семкин

Защищен 05.06.2025 Оценка отлично

Члены комиссии:

 ст. преп. Н.В. Кушнир

 ст. преп. Ю.С. Вонарх

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

Направление подготовки 09.03.03 «Прикладная информатика»

Профиль «Разработка и внедрение прикладных информационных систем»

УТВЕРЖДАЮ

Заведующий кафедрой ИСП

 М.В. Янаева

« » 2025 г.

ЗАДАНИЕ на курсовой проект

Студенту М.С. Мальцеву группы 22-КБ-ПИ1 3 курса

Тема проекта: «Система управления студенческими задачами и дедлайнами»
(утверждена указанием директора института № 37 К7 от 13.02.2025г.)

План проекта:

1. Назначение и цели создания системы.
2. Описание предметной области.
3. Проектирование.
4. Реализация.

Объем проекта: 42 с.

- а) пояснительная записка
- б) Листинг программного кода

Рекомендуемая литература:

1. Основы технологий баз данных: учебное пособие / Б.А. Новиков, Е.А. Горшкова, Н.Г. Графеева; под ред. Е.В. Рогова. – 2-е изд. — М.: ДМК Пресс, 2020. – 582 с.
2. Брылева, А. А. Программные средства создания интернет-приложений : учебное пособие / А. А. Брылева. - Минск : РИПО, 2019. - 377 с.

Срок выполнения проекта: с «03» февраля 2025 г. по «02» июня 2025 г.

Срок защиты: «06» июня 2025 г.

Дата выдачи задания: «05» июня 2025 г.

Дата сдачи работы на кафедру: «05» июня 2025 г.

Руководитель (нормоконтролер) проекта:  ассистент Р.И. Семкин

Задание принял студент



М.С. Мальцев

Реферат

Пояснительная записка к курсовому проекту содержит: 42 страницы, 11 рисунков, 10 источников.

CQRS, ASP.NET Core, JWT-АУТЕНТИФИКАЦИЯ, ENTITY FRAMEWORK CORE, MEDIATR, FLUENTVALIDATION, POSTGRESQL, REACT, API, РОЛЕВАЯ АВТОРИЗАЦИЯ.

Цель курсового проекта заключается в разработке и реализации веб-приложения, предназначенного для управления учебными задачами студентов и повышения эффективности работы в учебных группах, с возможностью отслеживания дедлайнов, распределения задач и взаимодействия в рамках образовательного процесса.

Объект исследования — организация учебной деятельности студентов в условиях современного цифрового взаимодействия. Предмет исследования — методы цифровизации учебной коммуникации, механизмы управления задачами и структурирования образовательных обязанностей в групповой среде.

В ходе работы проведён анализ существующих подходов к управлению задачами в учебной среде, разработана и частично реализована система, обеспечивающая базовый функционал по созданию учебных групп, назначению и контролю задач, а также ведению коммуникации между участниками. Особое внимание уделено вопросам безопасности, масштабируемости и гибкости архитектуры приложения. Используются современные технологии построения серверной логики, централизованной обработки ошибок, валидации данных и разграничения прав доступа.

Результаты работы могут быть использованы в учебных заведениях и студенческих сообществах для повышения организованности, прозрачности и эффективности выполнения учебных заданий. Проект закладывает основу для дальнейшего расширения функционала, включая интеграцию календарей, напоминаний и систем оценки успеваемости.

Содержание

Введение	5
1 Нормативные ссылки	6
2 Анализ технических требований и уточнение спецификаций	7
2.1 Постановка задачи	7
2.2 Анализ требований к функционалу и безопасности	8
3 Проектирование структуры и компонентов программного продукта	10
3.1 Формализация задачи	10
3.2 Выбор архитектурного подхода и технологий	12
4 Разработка программы и анализ результатов	15
4.1 Обоснование выбора инструментов и технологий	15
4.2 Разработка серверной части (Backend)	17
4.3 Разработка клиентской части (Frontend)	18
4.4 Интеграция и взаимодействие компонентов	24
5. Тестирование и сопровождение программы	26
5.1. Обработка ошибок и логирование	26
5.2. Планы сопровождения и расширения функционала	27
Заключение	29
Список литературы	30
Приложение А. Листинг	32
Приложение Б. Тест на антиплагиат	37

Введение

Данная курсовая работа посвящена разработке многоуровневого веб-приложения для управления учебными задачами в студенческой среде. В рамках проекта реализована система с авторизацией, личными и групповыми чатами, системой задач, статусами выполнения и управлением ролями в группах.

Целью работы является создание современной, расширяемой и безопасной системы управления взаимодействием между студентами и организация задач в рамках учебного процесса, с возможностью групповой работы и контроля.

Для достижения этой цели были решены следующие задачи:

1. Анализ требований и проектирование архитектуры программного продукта.
2. Разработка клиентской части приложения с использованием React.
3. Реализация серверной части на ASP.NET Core с использованием принципов чистой архитектуры, CQRS и MediatR.
4. Интеграция с базой данных PostgreSQL с использованием Entity Framework Core.
5. Внедрение JWT-аутентификации, защиты маршрутов и разграничения прав доступа.

Первый раздел содержит перечень использованных нормативных документов и источников. Во втором разделе приводится постановка задачи и обзор архитектурных решений. В третьем — рассмотрена формализация задачи и структура системы. В четвёртом — реализация основных компонентов, описание клиентской и серверной частей, а также пример работы приложения. Заключительный раздел посвящён сопровождению и возможному расширению системы в будущем.

1 Нормативные ссылки

В данной пояснительной записке использованы ссылки на следующие стандарты:

- ГОСТ Р7.0.5-2008 СИБИД. Библиографическая ссылка. Общие требования и правила составления;
- ГОСТ Р1.5-2004. Стандарты национальные РФ. Правила построения, изложения, оформления и обозначения;
- ГОСТ 2.301-68 ЕСКД. Форматы;
- ГОСТ 7.82-2001 СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления;
- ГОСТ 7.12-93 СИБИД. Библиографическая запись. Сокращения слов на русском языке. Общие требования и правила;
- ГОСТ 7.9-95 СИБИД. Реферат и аннотация. Общие требования.

2 Анализ технических требований и уточнение спецификаций

2.1 Постановка задачи

Целью данного программного продукта является разработка системы управления учебными задачами (Study Task Manager), предназначенной для поддержки индивидуальной и групповой деятельности студентов в учебном процессе. Приложение должно обеспечить пользователям возможность эффективно организовывать учебные задачи, участвовать в групповых проектах, обмениваться сообщениями и управлять совместной работой.

Основные функции системы включают:

- Регистрация и аутентификация пользователей;
- Создание и ведение личных и групповых задач;
- Управление группами: создание, приглашение участников, распределение ролей;
- Ведение личных и групповых чатов;
- Назначение и отслеживание статусов задач;
- Ведение истории сообщений и активности пользователей;
- Безопасная работа с пользовательскими данными и защита информации.

Приложение должно иметь клиент-серверную архитектуру и обеспечивать взаимодействие между фронтендом и бэкендом через REST API. Система разрабатывается с учётом масштабируемости, модульности и возможности дальнейшего расширения функциональности.

В рамках проекта особое внимание уделяется:

- Интуитивно понятному пользовательскому интерфейсу;
- Безопасности хранения и передачи данных;
- Возможности работы в режиме реального времени (например, для сообщений и обновлений задач).

Пользователи системы — это студенты, преподаватели и администраторы. Приложение должно поддерживать авторизацию с разграничением прав доступа в зависимости от роли пользователя.

2.2 Анализ требований к функционалу и безопасности

Разработка системы управления учебными задачами предполагает активное взаимодействие с пользователем, поэтому важным этапом является анализ пользовательских сценариев и требований к интерфейсу. Понимание того, как конечный пользователь будет использовать систему, позволяет более точно определить структуру компонентов, навигацию и функциональные модули.

Основные пользовательские роли:

- Студент — основной пользователь, создающий личные и групповые задачи, участвующий в чатах и группах, отслеживающий статус выполнения задач.
- Преподаватель — может выступать как наблюдатель или координатор групповых задач, а также иметь доступ к дополнительной информации и аналитике.
- Администратор — управляет учетными записями, модерирует контент и следит за корректной работой системы.

Ключевые сценарии использования:

1. Регистрация и вход в систему — пользователь должен иметь возможность зарегистрироваться и войти в систему через защищённый интерфейс.
2. Работа с задачами — создание, редактирование, удаление, изменение статуса и установка дедлайнов.
3. Участие в группах — просмотр доступных групп, подача заявок, создание новых групп, приглашение участников.

4. Обмен сообщениями — реализация как личных чатов, так и групповых обсуждений с возможностью просмотра истории.

5. Управление ролями и правами доступа — разграничение действий в зависимости от роли в группе (участник, руководитель, админ).

6. Уведомления — система оповещений о новых сообщениях, изменениях в задачах и других событиях.

Требования к пользовательскому интерфейсу:

- Минимализм и понятная структура навигации;
- Поддержка адаптивной верстки для корректной работы на различных устройствах;
- Быстрая откликаемость (реакция интерфейса на действия пользователя в пределах 0.3–0.5 сек);
- Поддержка тёмной и светлой темы (по возможности);
- Использование стандартных UI-компонентов, обеспечивающих единообразный внешний вид и поведение.

Таким образом, анализ пользовательских сценариев позволяет сформировать четкие ориентиры при проектировании клиентской части и логики взаимодействия с сервером.

3 Проектирование структуры и компонентов программного продукта

3.1 Формализация задачи

Целью разработки является создание полнофункционального веб-приложения для управления учебными задачами, ориентированного на студентов, преподавателей и учебные группы. Приложение должно обеспечить эффективную коммуникацию между участниками, удобную постановку и контроль выполнения задач, а также поддержку групповой работы.

Общая формулировка задачи:

Разработать клиент-серверную систему, позволяющую:

- Пользователям регистрироваться, авторизовываться и взаимодействовать в рамках индивидуальных и групповых рабочих пространств;
- Управлять задачами: создавать, изменять, удалять, сортировать, фильтровать, назначать участникам;
- Организовывать группы с разной структурой ролей;
- Осуществлять коммуникацию между пользователями в виде личных и групповых чатов;
- Обеспечивать контроль доступа к функциональности на основе ролей и прав пользователей;
- Поддерживать масштабируемость и расширяемость архитектуры.

Основные функциональные компоненты:

1. Аутентификация и авторизация:

- Регистрация новых пользователей;
- Вход в систему с использованием JWT-токенов;
- Проверка прав доступа к защищённым маршрутам (ProtectedRoute).

2. Управление задачами:

- CRUD-операции (Create, Read, Update, Delete) для учебных задач;

- Назначение задач участникам групп;
- Установка сроков выполнения;
- Изменение статуса задачи (в процессе, выполнена и др.).

3. Работа с группами:

- Создание, редактирование, просмотр групп;
- Приглашение участников, удаление, управление ролями;
- Отображение задач, назначенных группе.

4. Мессенджер:

- Персональные и групповые чаты;
- Просмотр истории сообщений;
- Индикация непрочитанных сообщений.

5. Интерфейс администратора (возможность реализации):

- Управление пользователями;
- Просмотр системной информации и логов;
- Модерация контента.

Ограничения и допущения:

- Приложение реализуется в виде SPA (Single Page Application) с использованием библиотеки React.
- Серверная часть разрабатывается на ASP.NET Core 8.0 с использованием MediatR, Entity Framework Core и PostgreSQL.
- Основной тип клиентского устройства — браузер на ПК; мобильная адаптация минимальна.

Таким образом, формализация задачи позволяет выделить границы и структуру системы, определить её основные модули и взаимодействие между ними, что критически важно для дальнейшего архитектурного проектирования.

3.2 Выбор архитектурного подхода и технологий

При проектировании программного обеспечения была выбрана клиент-серверная архитектура с чётким разделением фронтенда и бэкенда, что обеспечивает модульность, масштабируемость и упрощает сопровождение приложения.

Архитектурный подход

Для реализации системы был применён подход «чистой архитектуры» (Clean Architecture), обеспечивающий:

- Ясное разделение слоёв (домен, приложение, инфраструктура, презентация);
- Инверсию зависимостей: бизнес-логика не зависит от фреймворков, UI или баз данных;
- Повышенную тестируемость и независимость компонентов;
- Возможность лёгкой замены внешних интерфейсов без изменения логики.

Основные слои приложения:

- Domain – описывает бизнес-сущности, интерфейсы, события и доменную модель;
- Application – реализует use-case'ы, команды, запросы, валидаторы и обработчики;
- Infrastructure – предоставляет реализацию внешних зависимостей (БД, авторизация и т.д.);
- WebAPI (Presentation) – реализует пользовательский API и конфигурацию приложения.

Используемые технологии

Backend (серверная часть):

- .NET 8 (ASP.NET Core) – основа для построения RESTful API;
- Entity Framework Core 9 – ORM для взаимодействия с базой данных;
- PostgreSQL – СУБД, используемая как основное хранилище данных;

- MediatR – реализация паттерна CQRS (Command and Query Responsibility Segregation);
 - FluentValidation – библиотека для валидации входных данных;
 - Serilog – система логирования, настроенная для вывода в файл и консоль;
 - Scrutor – библиотека для сканирования и автоматической регистрации зависимостей;
 - Quartz – фреймворк для планирования фоновых задач (опционально).
- Frontend (клиентская часть):
- React – JavaScript-библиотека для построения SPA;
 - React Router DOM – маршрутизация по страницам;
 - Fetch/Axios – для HTTP-запросов к API;
 - JWT – аутентификация и хранение токена доступа;
 - Tailwind CSS (опционально) – для стилизации компонентов.

Обоснование выбора

Выбор указанных технологий обусловлен следующими факторами:

- Надёжность и зрелость платформы .NET, активная поддержка и документация;
- Широкое сообщество и интеграция с современными практиками проектирования ПО;
- Гибкость React при построении динамических пользовательских интерфейсов;
- Поддержка масштабируемости и расширяемости как со стороны клиента, так и сервера;
- CQRS и MediatR упрощают управление зависимостями и делают архитектуру легко расширяемой.

На рисунке ниже (рисунок 3.1) иллюстрируется, как организованы данные, связи между таблицами, что важно для понимания архитектуры.

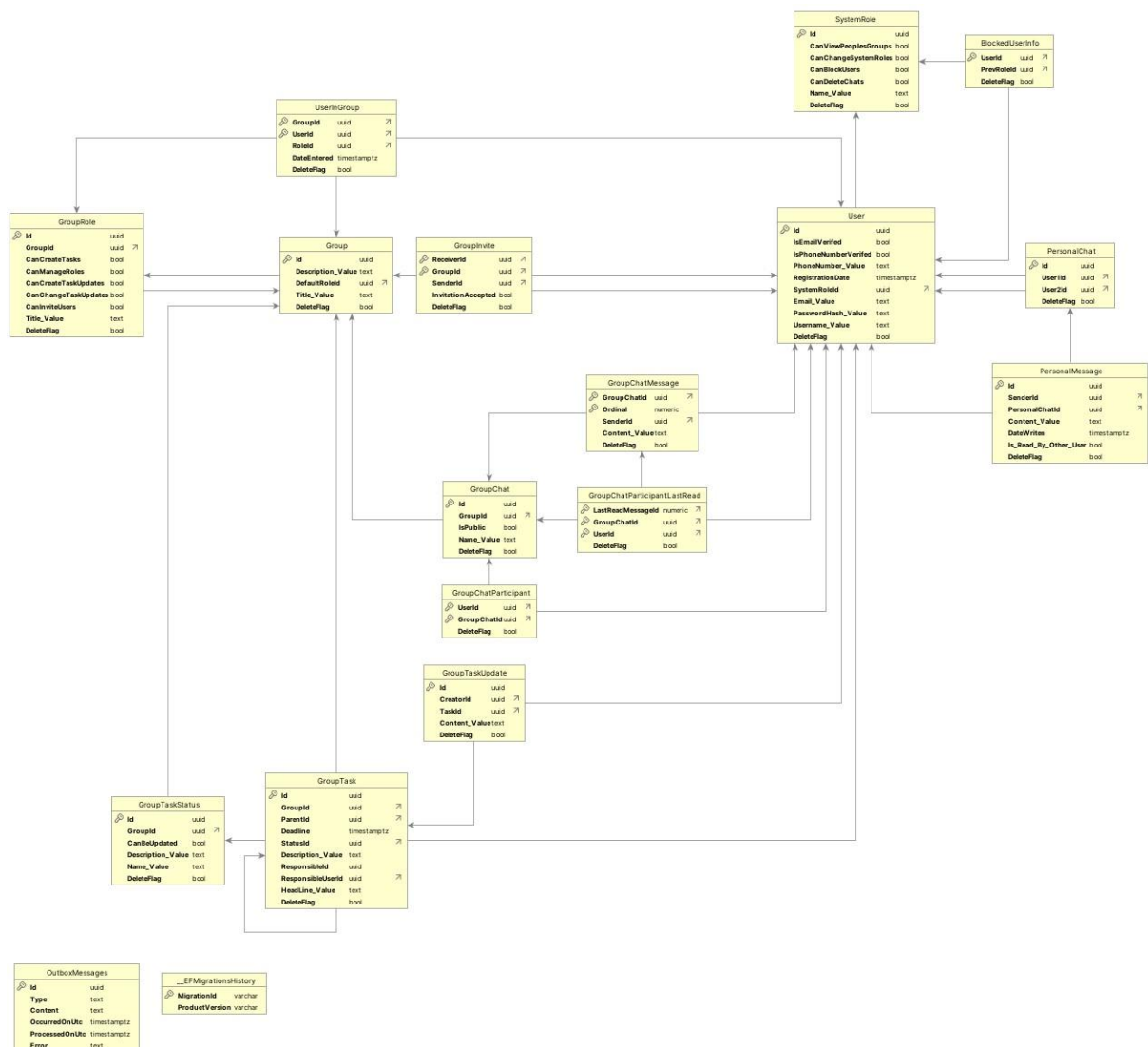


Рисунок 3.1 — Схема базы данных проекта

Таким образом, архитектура проекта ориентирована на долгосрочную поддержку и масштабирование, что позволяет легко адаптировать систему под новые бизнес-требования и технологические изменения.

4 Разработка программы и анализ результатов

4.1 Обоснование выбора инструментов и технологий

Выбор инструментов разработки был обусловлен требованиями к надёжности, расширяемости, удобству поддержки и скорости реализации. Ниже приведено обоснование выбора ключевых технологий и инструментов, использованных в проекте.

Язык и платформа разработки

C# и .NET 8 выбраны как основная технологическая платформа для серверной части. Это современный, активно развивающийся стек с высокой производительностью, встроенной поддержкой многопоточности, хорошей интеграцией с базами данных и средствами безопасности.

.NET предоставляет:

- встроенные механизмы авторизации и аутентификации;
- удобную работу с HTTP и JSON через ASP.NET Core Web API;
- поддержку middleware и фильтрации запросов;
- расширяемость и хорошую тестируемость компонентов.

Инструменты для работы с базой данных

Для хранения данных используется PostgreSQL – реляционная система управления базами данных, обеспечивающая:

- надёжность и отказоустойчивость;
- гибкую работу с транзакциями;
- поддержку полнотекстового поиска;
- расширенные функции безопасности.

Для взаимодействия с БД используется Entity Framework Core (EF Core) – объектно-реляционный маппер, который позволяет работать с данными через LINQ-запросы и обеспечивает миграции базы данных без необходимости писать SQL вручную.

Архитектурные библиотеки и паттерны

- MediatR: используется для реализации паттерна CQRS (разделение команд и запросов), что упрощает логику бизнес-операций, повышает модульность и облегчает сопровождение.
- FluentValidation: применён для валидации входных данных и команд, обеспечивая централизованный и читаемый способ проверки корректности пользовательского ввода.
- Serilog: мощная система логирования, позволяющая настраивать вывод в файл, консоль, систему мониторинга или базу данных, что критично при отладке и эксплуатации.

Инструменты фронтенда

Для реализации клиентской части использовался стек React:

- компонентная модель позволяет повторно использовать UI-элементы;
- библиотека React Router DOM используется для навигации по страницам SPA;
- поддержка JWT позволяет безопасно сохранять и передавать токены доступа;
- взаимодействие с сервером осуществляется с помощью fetch API или Axios;
- современная экосистема и богатый набор UI-библиотек.

Dev-инструменты и сопровождение

- Swagger (Swashbuckle) – для генерации документации по API, упрощающей тестирование и взаимодействие с сервером;
- SpaProxy – используется для локальной разработки, позволяя одновременно запускать клиент и сервер с удобной маршрутизацией запросов;
- Serilog + лог-файлы – позволяют отслеживать ошибки и поведение системы на всех этапах.

Таким образом, выбор всех технологий был направлен на создание надёжной, масштабируемой и сопровождаемой системы, соответствующей современным стандартам веб-разработки.

4.2 Разработка серверной части (Backend)

Серверная часть приложения реализована с использованием платформы ASP.NET Core 8.0, что обеспечивает высокую производительность, безопасность и масштабируемость. В проекте применены современные архитектурные подходы, такие как CQRS и MediatR, которые способствуют разделению ответственности и упрощают сопровождение кода.

В качестве базы данных используется PostgreSQL, взаимодействие с которой осуществляется через ORM Entity Framework Core 9.0, что позволяет эффективно работать с данными и упрощает миграции и поддержку схемы.

Для обеспечения безопасности реализована аутентификация и авторизация с помощью JWT-токенов (JSON Web Tokens), что позволяет безопасно управлять доступом пользователей к защищённым ресурсам API.

Валидация входящих данных осуществляется с помощью библиотеки FluentValidation, которая позволяет создавать удобные и расширяемые правила проверки. В случае ошибки валидации клиент получает подробное сообщение с описанием проблемы, что улучшает пользовательский опыт и снижает нагрузку на поддержку.

Для удобства и стандартизации разработки API интегрирован Swagger — инструмент для автоматической генерации документации и тестирования эндпоинтов. Ниже представлен пример интерфейса Swagger, демонстрирующий структуру и описание доступных методов API.

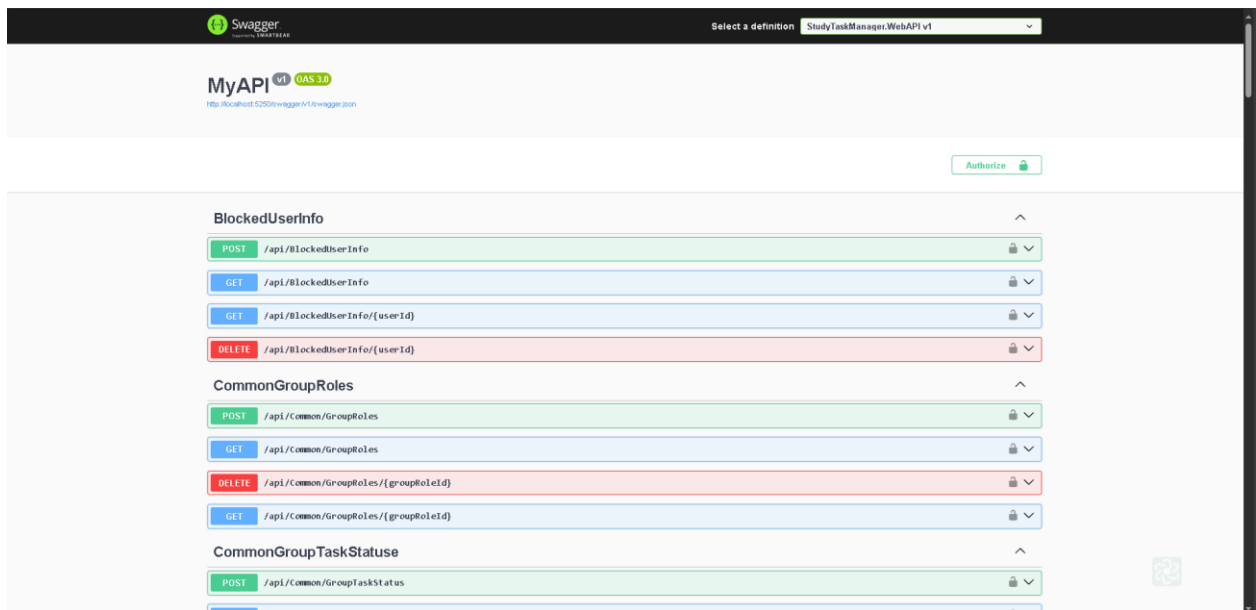


Рисунок 4.2 — Пример интерфейса Swagger UI для документации API

Логирование и мониторинг запросов реализованы с помощью Serilog, что позволяет эффективно отслеживать состояние приложения и оперативно выявлять ошибки.

Также в проекте настроена политика CORS, позволяющая взаимодействовать с фронтендом, размещённым на отдельном домене, без ограничений, обеспечивая при этом безопасность.

Таким образом, серверная часть обеспечивает надёжную, безопасную и масштабируемую основу для работы приложения, позволяя обрабатывать запросы пользователей, управлять данными и обеспечивать интеграцию с клиентской частью.

4.3 Разработка клиентской части (Frontend)

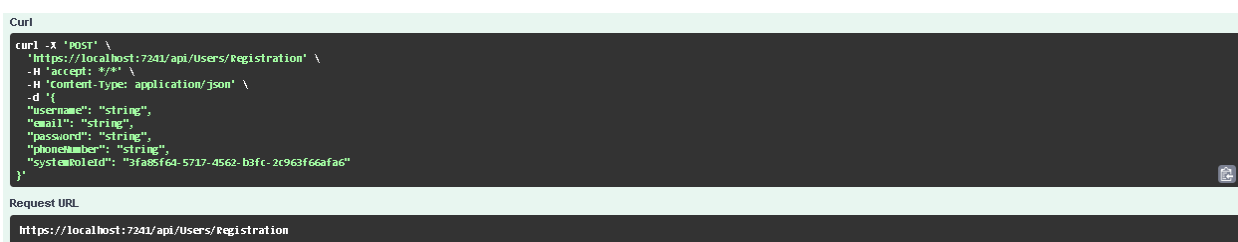
Клиентская часть приложения реализована с использованием библиотеки React — одного из самых популярных и мощных инструментов для создания современных веб-интерфейсов. React обеспечивает удобную компонентную архитектуру, позволяя создавать повторно используемые и легко поддерживаемые части интерфейса.

Для маршрутизации внутри приложения используется React Router, что позволяет реализовать многостраничную навигацию без перезагрузки страниц. В приложении настроены маршруты для страниц приветствия, регистрации, входа в систему, личных и групповых чатов, а также для управления группами.

Особое внимание уделено защите маршрутов — доступ к основной части приложения осуществляется только после успешной аутентификации пользователя, что реализовано через компонент ProtectedRoute.

Взаимодействие с сервером организовано через REST API с использованием стандартных методов HTTP. Для обработки ошибок и уведомлений пользователя реализованы соответствующие механизмы отображения сообщений об ошибках, например, при неверных данных на форме входа.

Валидация пользовательского ввода на стороне клиента проводится для улучшения UX и снижения количества некорректных запросов к серверу. Ниже представлен пример обработки ошибок валидации на форме, показывающий, как пользователь получает обратную связь при вводе недопустимых значений.



```
Curl
curl -X 'POST' \
  'https://localhost:7341/api/Users/Registration' \
  -H 'accept: */*' \
  -H 'content-type: application/json' \
  -d '{
    "username": "string",
    "email": "string",
    "password": "string",
    "phoneNumber": "string",
    "systemId": "3fa85f64-5717-4562-b3fc-2c963f66afae"
  }'
```

Request URL

https://localhost:7341/api/Users/Registration

Рисунок 4.3 — Пример входных данных

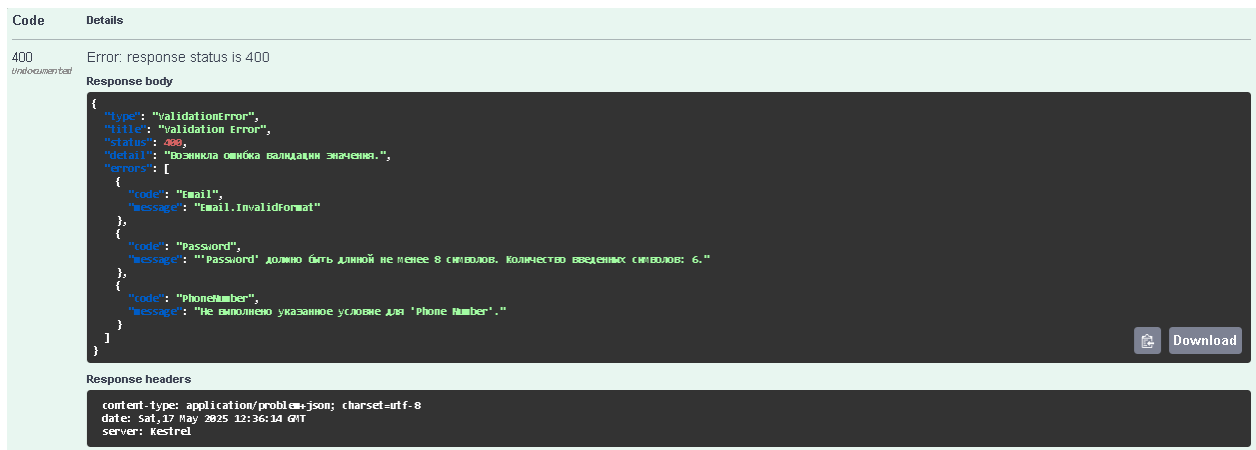


Рисунок 4.4 — Пример отображения сообщения об ошибке валидации на клиенте.

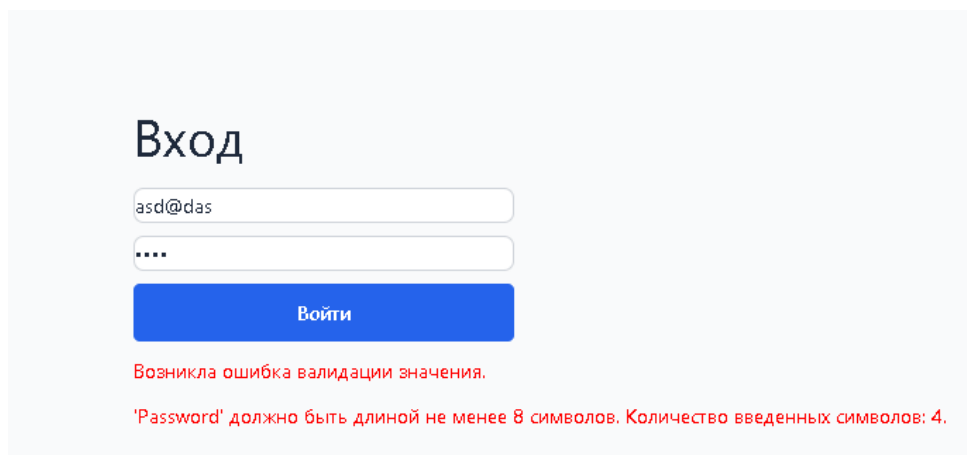


Рисунок 4.5 — Пример отображения ошибки валидации на форме входа

Интерфейс выполнен с использованием современного CSS и библиотек для стилизации, что обеспечивает адаптивность и удобство использования на разных устройствах.

Для организации навигации в приложении используется библиотека **React Router**, позволяющая создавать маршруты и управлять переходами между страницами без перезагрузки.

Ниже приведён фрагмент основного файла маршрутизации:

```
<Routes>
```

```
  <Route path="/" element={<WelcomePage />} />
```

```

<Route path="/login" element={<LoginPage />} />
<Route path="/register" element={<RegisterPage />} />

<Route element={<ProtectedRoute />}>
  <Route path="/home" element={<HomeLayout />}>
    <Route index element={<HomePage />} />
    <Route path="chats" element={<PersonalChatsPage />} />
    <Route path="chats/:idPersonalChat" element={<PersonalChatPage
/>} />

    <Route path="groups" element={<GroupsPage />} />
    <Route path="groups/:id" element={<GroupDetailsPage />} />
    <Route path="groups/create" element={<CreateGroupForm />} />

    <Route path="*" element={<NotFoundPage />} />
  </Route>
</Route>

```

Таким образом, маршрутизация позволяет разграничить доступ к страницам по правам пользователя — публичные страницы (WelcomePage, LoginPage, RegisterPage) доступны всем, а основные разделы приложения доступны только после авторизации, благодаря компоненту ProtectedRoute.

Для наглядного понимания приведены скриншоты ключевых страниц:

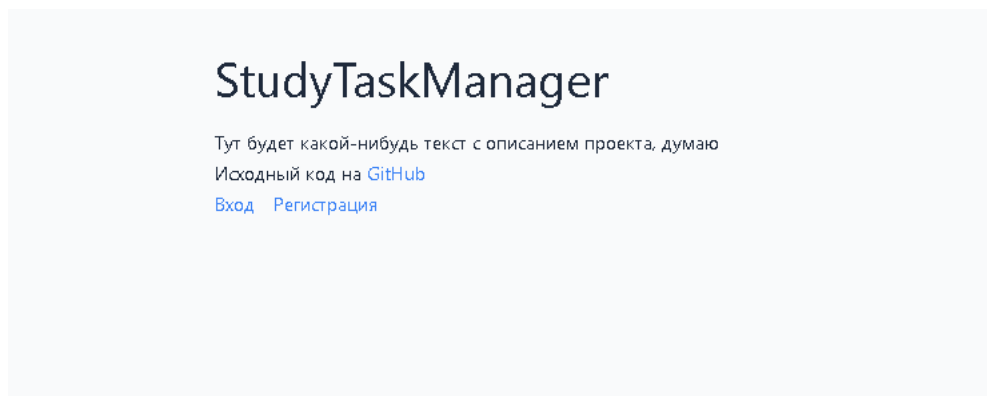


Рисунок 4.6 — Страница приветствия

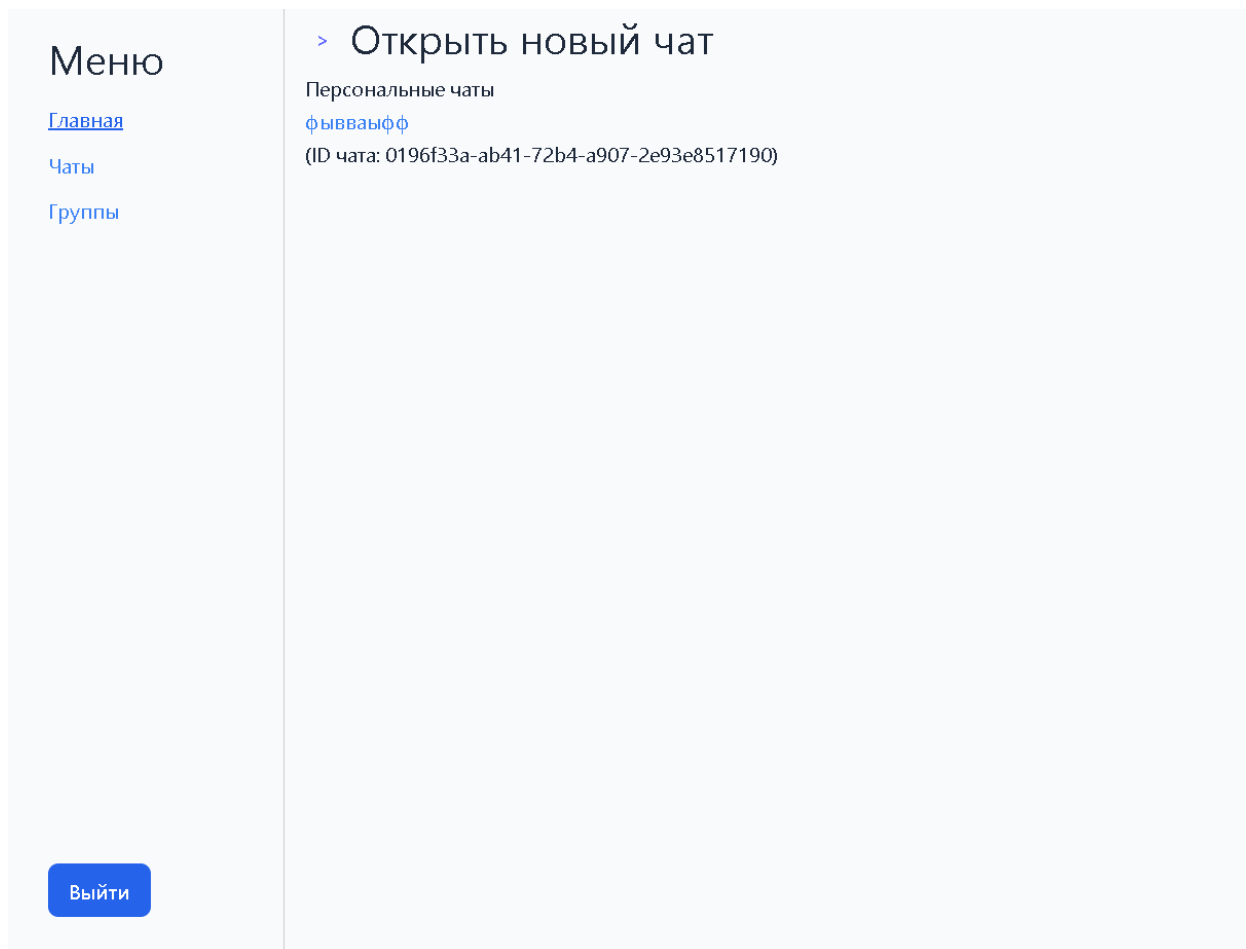


Рисунок 4.7 — Страница личных чатов

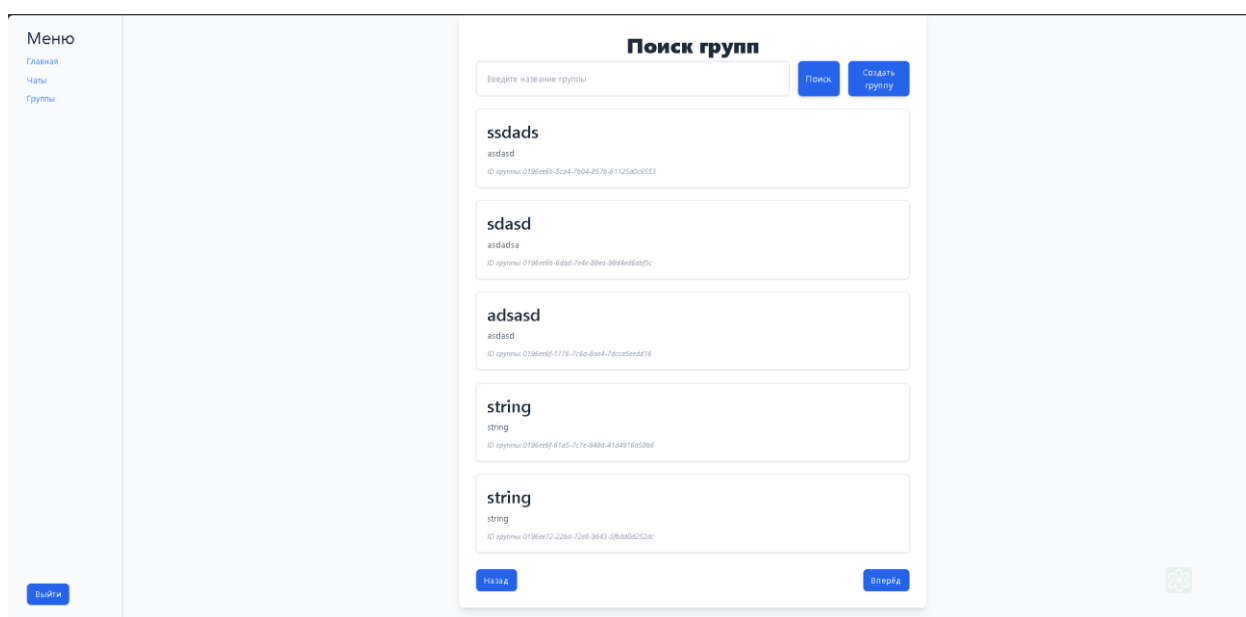


Рисунок 4.8 — Страница с группами

Меню

[Главная](#)

[Чаты](#)

[Группы](#)

Выйти

Создание новой группы

Название

Описание

Создать группу

Рисунок 4.9 — Страница создания группы

ssdads

asdasd

Создать новую задачу

Заголовок

04.05.2025 19:49

В процессе

Описание

+ Добавить задачу

Задачи группы

Задача1

описание1

Дедлайн: 04.05.2025, 19:49:00

В процессе

Удалить

Рисунок 4.10 — Страница группы

Таким образом, фронтенд часть приложения обеспечивает интуитивно понятное и отзывчивое взаимодействие пользователя с системой, поддерживая все необходимые функции для работы с чатами и группами.

4.4 Интеграция и взаимодействие компонентов

В данном разделе рассматривается процесс объединения серверной и клиентской частей приложения, а также организация взаимодействия различных модулей для обеспечения целостной и корректной работы системы.

Архитектура взаимодействия

Клиентская часть (Frontend), реализованная на React, взаимодействует с серверной частью (Backend), построенной на ASP.NET Core, посредством REST API. Все запросы на получение, создание, обновление и удаление данных осуществляются через HTTP, с передачей данных в формате JSON.

Для безопасного обмена данными используется механизм аутентификации через JWT (JSON Web Tokens). Токен передается в заголовках запросов и проверяется на сервере для подтверждения прав пользователя.

Основные сценарии интеграции

- **Аутентификация и авторизация:** Клиент отправляет данные для входа (логин и пароль) на сервер. При успешной проверке сервер возвращает JWT-токен, который клиент сохраняет и использует для последующих запросов.
- **Получение и отображение данных:** Клиент запрашивает списки чатов, групп, сообщений, используя идентификаторы пользователей и групп, получая данные от API и отображая их в соответствующих компонентах.
- **Обработка ошибок:** В случае ошибок (валидация, проблемы на сервере) сервер возвращает информативные сообщения с кодами состояния HTTP. Клиент обрабатывает их и отображает пользователю.

- Реализация защиты маршрутов: Защищённые страницы на клиенте доступны только при наличии валидного токена, что предотвращает несанкционированный доступ.

Технические решения для интеграции

- Использование библиотеки `axios` для удобного выполнения HTTP-запросов с поддержкой перехватчиков (`interceptors`), что позволяет автоматически добавлять токены авторизации и обрабатывать ошибки.

- Серверная часть содержит эндпоинты, документированные с помощью `Swagger`, что упрощает тестирование и интеграцию.

- Реализация единой модели ошибок и валидации данных как на стороне сервера (через `FluentValidation`), так и на клиенте (валидация форм `React-hook-form` или аналогичных).

5. Тестирование и сопровождение программы

5.1. Обработка ошибок и логирование

Важным аспектом обеспечения стабильности и поддержки программного продукта является эффективная обработка ошибок и ведение логов. Это позволяет своевременно выявлять проблемы, анализировать их причины и обеспечивать качественное сопровождение приложения.

Обработка ошибок

- На серверной стороне используется централизованное глобальное обработчик исключений, который перехватывает ошибки, возникающие при выполнении запросов. Он возвращает клиенту структурированный ответ с кодом ошибки и подробным сообщением.

- Валидация данных реализована с помощью библиотеки FluentValidation. При нарушении правил валидации клиент получает детализированные сообщения об ошибках, что улучшает UX и снижает нагрузку на сервер.

- На клиентской стороне ошибки отображаются пользователю в виде уведомлений или формовых подсказок, обеспечивая понятность происходящего.

Логирование

- Для логирования используется Serilog, настроенный на запись логов в консоль и файл с применением компактного формата. Это позволяет вести мониторинг событий, отслеживать успешные операции и ошибки.

- Логи содержат важные данные: временные метки, уровень важности, сообщения об ошибках и контекст выполнения. Это упрощает анализ инцидентов и поиск неисправностей.

```
[2025-05-20 19:42:58 INF] HTTP GET /api/Group/Take responded 200 in 41.8617 ms
[2025-05-20 19:42:59 INF] HTTP GET //index.html responded 404 in 2.6382 ms
[2025-05-20 19:42:59 INF] HTTP GET //index.html responded 404 in 0.1147 ms
[2025-05-20 19:43:00 INF] HTTP GET /api/Group/Take responded 200 in 6.3979 ms
[2025-05-20 19:43:00 INF] HTTP GET /api/Group/Take responded 200 in 2.7173 ms
[2025-05-20 19:43:07 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:43:08 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:43:15 INF] HTTP GET /api/users/me responded 200 in 103.7498 ms
[2025-05-20 19:43:15 INF] HTTP GET /api/users/me responded 200 in 3.1913 ms
[2025-05-20 19:43:15 WRN] The query uses a row limiting operator ('Skip'/'Take') without an 'OrderBy' operator. This may lead to unpredictable results. If the 'Distinct' operator is used
[2025-05-20 19:43:15 WRN] The query uses a row limiting operator ('Skip'/'Take') without an 'OrderBy' operator. This may lead to unpredictable results. If the 'Distinct' operator is used
[2025-05-20 19:43:15 INF] HTTP GET /api/Group/Take responded 200 in 124.9428 ms
[2025-05-20 19:43:15 INF] HTTP GET /api/Group/Take responded 200 in 42.9598 ms
[2025-05-20 19:43:17 INF] HTTP GET /api/users/me responded 200 in 1.3347 ms
[2025-05-20 19:43:17 INF] HTTP GET /api/users/me responded 200 in 0.4131 ms
[2025-05-20 19:43:17 INF] HTTP GET /api/Group/Take responded 200 in 5.1824 ms
[2025-05-20 19:43:17 INF] HTTP GET /api/Group/Take responded 200 in 2.6572 ms
[2025-05-20 19:43:46 INF] HTTP GET /api/Group/GetAll responded 200 in 14.5247 ms
[2025-05-20 19:52:00 INF] HTTP GET /api/Users/All responded 200 in 175.0143 ms
[2025-05-20 19:52:31 INF] HTTP GET /api/Group/Take responded 400 in 8.6784 ms
[2025-05-20 19:52:34 INF] HTTP GET /api/Group/Take responded 200 in 3.1606 ms
[2025-05-20 19:52:52 INF] HTTP GET /api/Common/GroupRoles responded 200 in 30.3820 ms
[2025-05-20 19:54:15 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:54:16 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:54:32 INF] HTTP GET /api/Group/GetAll responded 200 in 164.7727 ms
[2025-05-20 19:55:11 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:55:12 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:55:19 INF] HTTP GET /api/users/me responded 200 in 80.9857 ms
[2025-05-20 19:55:19 INF] HTTP GET /api/users/me responded 200 in 2.6249 ms
[2025-05-20 19:55:19 WRN] The query uses a row limiting operator ('Skip'/'Take') without an 'OrderBy' operator. This may lead to unpredictable results. If the 'Distinct' operator is used
[2025-05-20 19:55:19 WRN] The query uses a row limiting operator ('Skip'/'Take') without an 'OrderBy' operator. This may lead to unpredictable results. If the 'Distinct' operator is used
[2025-05-20 19:55:19 INF] HTTP GET /api/Group/Take responded 200 in 134.4130 ms
[2025-05-20 19:55:19 INF] HTTP GET /api/Group/Take responded 200 in 49.8546 ms
[2025-05-20 19:55:30 INF] HTTP GET /api/Group/0196ee6b-5ca4-7b04-857b-61125a0c6553 responded 400 in 16.8215 ms
[2025-05-20 19:55:30 INF] HTTP GET /api/Group/0196ee6b-5ca4-7b04-857b-61125a0c6553 responded 400 in 1.7809 ms
[2025-05-20 19:56:46 INF] HTTP GET //index.html responded 404 in 0.4200 ms
[2025-05-20 19:57:59 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:58:00 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 19:58:15 INF] HTTP GET /api/users/me responded 200 in 77.2739 ms
[2025-05-20 19:58:15 INF] HTTP GET /api/users/me responded 200 in 3.2022 ms
[2025-05-20 19:58:15 INF] HTTP GET /api/Group/0196ee6b-5ca4-7b04-857b-61125a0c6553 responded 200 in 166.9682 ms
[2025-05-20 19:58:15 INF] HTTP GET /api/Group/0196ee6b-5ca4-7b04-857b-61125a0c6553 responded 200 in 43.7250 ms
[2025-05-20 20:08:28 WRN] The query uses a row limiting operator ('Skip'/'Take') without an 'OrderBy' operator. This may lead to unpredictable results. If the 'Distinct' operator is used
[2025-05-20 20:08:28 WRN] The query uses a row limiting operator ('Skip'/'Take') without an 'OrderBy' operator. This may lead to unpredictable results. If the 'Distinct' operator is used
[2025-05-20 20:08:28 INF] HTTP GET /api/Group/Take responded 200 in 72.8557 ms
[2025-05-20 20:08:28 INF] HTTP GET /api/Group/Take responded 200 in 2.4991 ms
[2025-05-20 20:26:32 INF] HTTP GET /api/Users/Take responded 401 in 4.9530 ms
[2025-05-20 20:26:32 INF] HTTP GET /api/Users/me responded 401 in 4.9652 ms
[2025-05-20 20:26:32 INF] HTTP GET /api/Users/Take responded 401 in 0.8318 ms
[2025-05-20 20:26:32 INF] HTTP GET /api/Users/me responded 401 in 0.4789 ms
[2025-05-20 20:26:48 INF] HTTP GET /api/Group/Take responded 200 in 50.6738 ms
[2025-05-20 20:26:48 INF] HTTP GET /api/Group/Take responded 200 in 2.3093 ms
[2025-05-20 20:28:19 INF] HTTP GET /api/users/me responded 401 in 0.5056 ms
[2025-05-20 20:28:19 INF] HTTP GET /api/users/me responded 401 in 0.5380 ms
[2025-05-20 20:29:31 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
[2025-05-20 20:29:32 WRN] The WebRootPath was not found: C:\Users\defaultuser0\source\repos\Sadaly\StudyTaskManager\StudyTaskManager\studytaskmanager.webapi\studytaskmanager.webapi.Server
```

Рисунок 5.11 — Пример лог-файла

5.2. Планы сопровождения и расширения функционала

Для успешного функционирования и развития программного продукта важно организовать качественное сопровождение и регулярно обновлять функционал с учётом изменяющихся требований и пожеланий пользователей.

Сопровождение

- Мониторинг работоспособности — постоянное отслеживание состояния приложения с использованием логирования и инструментов мониторинга.
- Обработка ошибок и багов — оперативное реагирование на сообщения об ошибках, их анализ и исправление.
- Обновление зависимостей — регулярное обновление библиотек и фреймворков для поддержания безопасности и стабильности.

- Резервное копирование данных — обеспечение сохранности данных через регулярные бэкапы.

Расширение функционала

- Добавление новых возможностей — реализация новых функций по запросам пользователей или в соответствии с развитием бизнеса.

- Оптимизация производительности — улучшение скорости работы и масштабируемости приложения.

- Интеграция с внешними сервисами — подключение новых API и сервисов для расширения возможностей.

- Адаптация к новым стандартам безопасности — обновление механизмов аутентификации и защиты данных.

План развития

В дальнейшем планируется:

- Внедрить систему уведомлений в реальном времени.
- Расширить возможности группового чата и управление ролями.
- Добавить мобильное приложение для удобного доступа.
- Реализовать аналитические отчёты по активности пользователей.

Регулярное сопровождение и развитие помогут поддерживать высокий уровень качества и удовлетворять потребности пользователей на долгосрочной основе.

Заключение

В данном курсовом проекте была решена задача разработки программного продукта для управления задачами и коммуникациями в учебном процессе. В ходе работы были изучены и проанализированы технические требования, уточнены спецификации, а также выбраны оптимальные архитектурные подходы и технологии для реализации системы.

Была построена структурированная архитектура, включающая серверную и клиентскую части, обеспечивающая удобство использования и масштабируемость. Реализация программного продукта выполнена с применением современных инструментов и библиотек, таких как React для фронтенда и ASP.NET Core для бэкенда, что позволило создать надёжное и функциональное приложение.

Особое внимание уделялось вопросам безопасности, валидации данных и обработке ошибок, что гарантирует стабильную работу и защиту данных пользователей. Разработаны и реализованы механизмы аутентификации и авторизации, обеспечивающие контроль доступа.

Проведённое тестирование и анализ результатов подтвердили корректность функционирования системы и её соответствие поставленным требованиям. Планы по сопровождению и расширению функционала предусматривают дальнейшее улучшение продукта, что обеспечит его актуальность и востребованность.

Таким образом, поставленные цели были успешно достигнуты, а реализованное решение может быть использовано как основа для дальнейшего развития и внедрения в учебный процесс.

Ознакомится с исходным кодом можно по ссылке:
<https://github.com/Sadaly/StudyTaskManager>.

Список литературы

1. Кулаичев А.П. Методы и средства комплексного статистического анализа данных: учеб. пособие / А.П. Кулаичев. М. 2021 — 484 с.
2. Объектно-ориентированное программирование анализ и дизайн: Методическое пособие / В. В. Мухортов, В. Ю. Рылов. – Новосибирск 2022 – 108 с.
3. Приемы объектно-ориентированного проектирования. Паттерны проектирования: Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. — СПб: Питер, 2023 — 368 с.: ил. (Серия «Библиотека программиста»).
4. Балдин К. В. Математическое программирование / Балдин К.В., Брызгалов Н.А., Рукоусев А.В., - 2-е изд. - Москва : Дашков и К, 2022. - 218 с.: ISBN 978-5-394-01457-4. - Текст: электронный. - URL: <https://znanium.com/bookread2.php?book=415097>
5. Аттетков А. В. Методы оптимизации: Учебное пособие / А.В. Аттетков, В.С. Зарубин, А.Н. Канатников. - М.: ИЦ РИОР: НИЦ Инфра-М, 2023. - 270 с.: ил.; - (Высшее образование: Бакалавриат). - ISBN 978-5-16-103309-8. - Текст: электронный. - URL: <https://znanium.com/bookread2.php?book=1002733>
6. Карманов В. Г. Математическое программирование [Электронный ресурс] : Учебное пособие / В. Г. Карманов. - 6-е изд., испр. - Москва : ФИЗМАТЛИТ, 2021. - 264 с. - ISBN 978-5-9221-0983-3. - Текст : электронный. - URL: <https://znanium.com/bookread2.php?book=544747>
7. Каштанов В. А. Исследование операций (линейное программирование и стохастические модели) : учебник / В.А. Каштанов, О.Б. Зайцева. — Москва : КУРС, 2021. - 256 с. - ISBN 978-5-906818-78-2. - Текст : электронный. - URL: <https://znanium.com/bookread2.php?book=1017099>
8. Microsoft Docs. ASP.NET Core Documentation [Электронный ресурс] <https://learn.microsoft.com/en-us/aspnet/core/>

9. React Official Documentation [Электронный ресурс]
<https://reactjs.org/docs/getting-started.html>

10 RFC 6749 - The OAuth 2.0 Authorization Framework [Электронный ресурс] <https://tools.ietf.org/html/rfc6749>

Приложение А. Листинг

```
using StudyTaskManager.Persistence;
using Serilog;
using Microsoft.EntityFrameworkCore;
using FluentValidation;
using StudyTaskManager.Application.Behaviors;
using MediatR;
using StudyTaskManager.Persistence.Interceptors;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using StudyTaskManager.WebAPI.OptionsSetup;
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);
var configuration = builder.Configuration;

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at
// https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddCors(options =>
{
    options.AddDefaultPolicy(policy =>
    {
        policy.WithOrigins("https://localhost:62284");
        policy.AllowAnyHeader();
        policy.AllowAnyMethod();
    });
});
```



```

        policy.AllowCredentials();
    });
}
);

```

builder

```

.Services
.Scan(
    selector => selector
    .FromAssemblies(
        StudyTaskManager.Infrastructure.AssemblyReference.Assembly,
        StudyTaskManager.Persistence.AssemblyReference.Assembly)
    .AddClasses(false)
    .AsImplementedInterfaces()
    .WithScopedLifetime());

```

```

builder.Services.AddMediatR(cfg                                     =>
    cfg.RegisterServicesFromAssembly(StudyTaskManager.Application.AssemblyRef
    erence.Assembly));

```

```

builder.Services.AddScoped(typeof(IPipelineBehavior<,>),
    typeof(ValidationPipelineBehavior<,>));

```

```

builder.Services.AddValidatorsFromAssembly(StudyTaskManager.Application.As
    semblyReference.Assembly,
    includeInternalTypes: true);

```

```

builder.Services.AddSwaggerGen(opt =>
{
    opt.SwaggerDoc("v1", new OpenApiInfo { Title = "MyAPI", Version = "v1" });

```

```

opt.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme
{
    In = ParameterLocation.Header,
    Description = "Please enter token",
    Name = "Authorization",
    Type = SecuritySchemeType.Http,
    BearerFormat = "JWT",
    Scheme = "bearer"
});
opt.AddSecurityRequirement(new OpenApiSecurityRequirement
{
    {
        new OpenApiSecurityScheme
        {
            Reference = new OpenApiReference
            {
                Type=ReferenceType.SecurityScheme,
                Id="Bearer"
            }
        },
        new string[]{}
    }
});
});

```

```

string? connectionString =
builder.Configuration.GetConnectionString("Database");

```

```

builder.Services.AddSingleton<ConvertDomainEventsToOutboxMessagesIntercep
tor>();

```

```

builder.Services.AddDbContext<AppDbContext>(
    (sp, optionsBuilder) =>
    {
        var interceptor =
sp.GetService<ConvertDomainEventsToOutboxMessagesInterceptor>();

        if (interceptor != null)
            optionsBuilder.UseNpgsql(connectionString)
                .AddInterceptors(interceptor);
    });

builder.Host.UseSerilog((context, configuration) =>
    configuration.ReadFrom.Configuration(context.Configuration));

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer();

builder.Services.ConfigureOptions<JwtOptionsSetup>();
builder.Services.ConfigureOptions<JwtBearerOptionsSetup>();

var app = builder.Build();

app.UseDefaultFiles();
app.UseStaticFiles();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();

```

```

    app.UseSwaggerUI();
}
app.UseSerilogRequestLogging();

app.UseCors();

app.UseHttpsRedirection();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.MapFallbackToFile("/index.html");

using (var scope = app.Services.CreateScope())
{
    var dbContext =
scope.ServiceProvider.GetRequiredService<AppDbContext>();
    DbInitializer.Initialize(dbContext);
}

app.Run();

```

Приложение Б. Тест на антиплагиат

88

usevg.antiplagiат.ru/report/summary/1

Антиплагиат

ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

sk

ПОЛЬЗОВАТЕЛЬ

plm12112004@mail.ru

расширить документ

МЕНЮ

ru

ТАРИФ

Free

кабинет

БАЛЛЫ

0

Цитирование

0%

Заминствования

3.27%

Оригинальность

96.73%

полный отчет

краткий отчет

история отчетов

Имя исходного файла

Кисловуа_Махсев_22-кв-Р11.pdf

Автор(ы) документа

Махсев

Название документа

Кисловуа_Махсев_22-кв-Р11

Тип документа

Курсовая работа

Свойства документа

Структура документа

Текстовые метрики

Параметры проверки

Статистика по документу

Создать ссылку

Выгрузить

Распечатать

Сохранить изменения

Отменить изменения

главная

история обновлений

помощь

контакты

Сайт для корпоративных клиентов

Пользовательское соглашение

Соглашение об обработке персональных данных

АО "Антиплагиат" 2005-2022 © Все права защищены