

МИНИСТЕРСТВО ОБРАЗОВАНИЯ ОРЕНБУРГСКОЙ ОБЛАСТИ  
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«ОРЕНБУРГСКИЙ КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ»  
(ГАПОУ СПО ОКЭИ)

## КУРСОВОЙ ПРОЕКТ

*ОКЭИ 09.02.07. 4323. №15 ПЗ*  
(код документа)

*Разработка веб-приложения реал-тайм чата*

---

---

Количество листов 30

Дата готовности 28.12.2023

Руководитель Лукасян А.Д.

Разработал Мусеева Д.А.

Защищен \_\_\_\_\_ с оценкой \_\_\_\_\_  
(дата)

Оренбург 2023

## Содержание

Введение.....	3
1 Анализ предметной области.....	5
2 Проектирование приложения.....	6
3 Разработка программного обеспечения .....	8
3.1 Описание технологического стека разработки .....	8
3.2 Описание алгоритма работы .....	11
3.3 Описание интерфейса пользователя .....	12
4 Тестирование приложения .....	16
4.1 План тестирования .....	16
4.2 Оценка результатов проведения тестирования .....	17
Заключение .....	20
Список литературы .....	21
Приложение А (обязательное) Диаграмма прецедентов системы .....	22
Приложение Б (обязательное) Диаграмма последовательности системы .....	23
Приложение В (обязательное) Диаграмма классов системы.....	24
Приложение Г (обязательное) Листинг программного кода .....	25

					ОКЭИ 09.02.07. 4323. 15 ПЗ		
Изм.	Лист	№ докум.					
Разраб.		Мусеева Д.А.			Отчет	Лит.	Лист
Провер.		Гукасян А.Д.					Листов
							2
							37
						Отделение информационных технологий	

## Введение

В современном мире информационных технологий коммуникация играет важную роль в повседневной жизни людей. Вместе с тем, современные технологии поддержки общения также продолжают развиваться и совершенствоваться. Реал-тайм чаты становятся все более популярными средствами общения, обеспечивая мгновенный обмен сообщениями между людьми. Именно поэтому разработка реал-тайм чата с регистрацией и авторизацией пользователей является актуальным и значимым направлением в развитии Web-приложений.

Целью данного проекта является создание инновационного веб-приложения - реал-тайм чата, который обеспечит пользователей удобным и безопасным способом общения в режиме реального времени. Благодаря функционалу регистрации и авторизации, пользователи смогут создавать учетные записи, получать доступ к чату только после прохождения процесса авторизации. Основная задача проекта заключается в разработке функционала для передачи и приема сообщений в режиме реального времени, обеспечивая надежную и безопасную коммуникацию в веб-среде.

Реализация данного проекта возможна путем достижения ряда задач:

- разработка пользовательского интерфейса, который будет интуитивно понятен и удобен для пользователей. Он должен включать в себя элементы, необходимые для регистрации и авторизации пользователей, а также для обмена сообщениями в режиме реального времени;

- создание серверной части приложения, которая будет отвечать за обработку запросов пользователей, хранение сообщений и управление безопасностью данных;

- реализация механизмов регистрации и авторизации пользователей, включая защиту от несанкционированного доступа и обеспечение конфиденциальности данных;

- разработка алгоритмов передачи сообщений в режиме реального времени с возможностью обновления чата без необходимости обновления страницы;

- тестирование и отладка функциональности приложения для обеспечения его стабильной и безошибочной работы в различных ситуациях;

- внедрение и запуск реал-тайм чата на продуктивной среде, готовность к использованию реальными пользователями.

Полученные результаты этого проекта имеют значительную практическую значимость. Разработка реал-тайм чата с регистрацией и авторизацией пользователей позволит пользователям свободно и безопасно обмениваться сообщениями в режиме реального времени. Это полезно для различных сфер деятельности, включая коммерческие цели, образовательные учреждения, общественные организации и просто общение между друзьями. Благодаря созданному приложению, пользователи смогут устанавливать новые контакты,

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
						3
Изм	Лист	№ докум.	Подпись	Дата		

общаться, делиться информацией и идеями, повышая эффективность взаимодействия и сокращая временные задержки.

Для успешной реализации проекта необходимо использовать определенные ресурсы. Технические и программные ресурсы включают в себя следующее: выделенный сервер для хранения и обработки данных, базу данных для хранения учетных записей пользователей и сообщений, языки программирования и фреймворки для разработки клиентской и серверной частей приложения, а также различные инструменты для тестирования и отладки.

Кроме того, данное веб-приложение имеет перспективы расширения функционала. В будущем, можно реализовать следующие возможности: возможность отправки медиафайлов, добавление чат-ботов, страницы с более подробной информацией о пользователях.

В целом, разработка реал-тайм чата с регистрацией и авторизацией пользователей является актуальной и полезной задачей в сфере Web-приложений. Проект стремится создать инновационное приложение, которое обеспечит активное и безопасное общение пользователей в реальном времени. Этот проект имеет как теоретическую, так и практическую значимость, и мы уверены, что его результаты будут ценными и полезными для пользователей.

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		4

# 1 Анализ предметной области

Для проекта по разработке реал-тайм чата с регистрацией и авторизацией пользователей, необходимо провести анализ предметной области, чтобы полноценно понять проблемную сферу человеческой деятельности, которая нуждается в автоматизации или частичной автоматизации процессов.

Одним из ключевых аспектов предметной области является сфера коммуникации и общения. В современном мире люди активно используют сеть Интернет для обмена информацией и общения на различные темы. Однако, с увеличением потока информации и разнообразием коммуникационных каналов, возникает необходимость в создании эффективных инструментов, которые позволят пользователям организовать свое онлайн общение более удобным и безопасным способом.

Существуют различные сферы деятельности, которые могут испытывать потребность в таком реал-тайм чате. Например, в бизнес-среде, команды проектов, распределенные по разным географическим местоположениям, нуждаются в надежном инструменте для мгновенного обмена информацией, координации и принятия совместных решений. Также, в образовательной сфере, студентам и преподавателям может потребоваться платформа для онлайн общения и обсуждения учебных вопросов. Другие области, такие как общественные организации или проекты по развитию сообществ, также могут воспользоваться функционалом данного реал-тайм чата, чтобы объединить участников и обмениваться информацией.

В предметной области реал-тайм чата с регистрацией и авторизацией, основными сущностями являются пользователи. Каждый пользователь должен иметь уникальный идентификатор, а также возможность создать учетную запись и войти в систему с помощью авторизации. Кроме того, важно учитывать безопасность данных пользователей и обеспечить уровень конфиденциальности.

Отношения в предметной области включают в себя взаимодействие между пользователями внутри реал-тайм чата, передачу сообщений в режиме реального времени и установление связей между пользователями, например, при создании групповых чатов.

Основными процессами в предметной области являются регистрация и авторизация пользователей, обмен сообщениями в режиме реального времени, обработка запросов пользователей, хранение и защита данных. Кроме того, важным процессом является разработка и поддержка пользовательского интерфейса, который обеспечит удобство и понятность взаимодействия пользователей с приложением.

Анализ предметной области проекта выявляет основные сущности, отношения и процессы, связанные с реализацией данного проекта и помогает определить требования для его успешной реализации.

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		5

## 2 Проектирование приложения

Функциональные требования к программному продукту описывают ожидаемое поведение системы при достижении определенных условий.

Функциональные требования в рамках реализации данного проекта:

Регистрация и авторизация:

— возможность регистрации новых пользователей с указанием уникального адреса электронной почты и пароля. После заполнения формы и проверки адреса электронной почты на уникальность, данные будут отправляться в базу данных, где создастся новая запись пользователя с его данными и уникальным идентификатором. В дальнейшем, данный идентификатор будет связывать запись пользователя с другими данными;

— авторизация уже зарегистрированных пользователей для доступа к чатам. При заполнении формы данные будут проходить проверку на правильность заполнения. При полном совпадении система будет давать пользователю токен доступа. В противном же случае, попросит заполнить форму корректными данными.

Интерфейс чата:

— возможность создания бесед (групповых чатов). В базе данных, в двух связанных внешним ключом таблицах, будут создаваться записи с информацией о названии беседы и об её участниках;

— отправка текстовых сообщений в реальном времени внутри чата. Данное соединение будет доступно с помощью использования websocket;

— отображение истории сообщений в чате. При отправке сообщения в чат, оно будет не только отображаться у всех пользователей, которые непосредственно являются участниками чата, но и так же отправляться в базу данных с указанием чата, которому это сообщение принадлежит, и времени его отправки.

Контакты:

— список пользователей отображает всех пользователей, зарегистрированных в системе (кроме аккаунта, который просматривает этот список). Данный запрос к базе данных возможен путем запроса всех пользователей, кроме того, у которого идентификатор совпадает с идентификатором пользователя, который непосредственно делает данный запрос к системе;

— при генерации списка пользователей система автоматически создает чаты с этими пользователями. Данная функция будет выполняться только в том случае, если чаты ранее не существовали.

Безопасность и конфиденциальность:

— защита данных пользователей. Для безопасности хранения данных в базе данных будет предусмотрено хэширование паролей;

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		6

— ограничение доступа к определенным функциям в зависимости от прав доступа пользователя. При наличии у пользователя прав доступа администратора, он сможет удалять других пользователей.

К нефункциональным требованиям программного продукта относятся те, которые она сможет продемонстрировать. Также к данному списку относятся ограничения.

Нефункциональные требования:

Производительность:

— высокая скорость отклика при отправке и получении сообщений в режиме реального времени. Данная характеристика может быть достигнута путем использования websocket. Также Node.js позволяет разрабатывать приложения, которые могут обрабатывать множество событий одновременно и предоставлять мгновенные обновления.

Удобство использования:

— интуитивный и простой в использовании интерфейс пользователя. Проектирование будет производиться таким образом, чтобы конечный продукт пользователь принимал как знакомый, т.е. с использованием общепринятых стандартов.

Масштабируемость:

— возможность расширения и масштабирования системы при увеличении числа пользователей и нагрузки.

Это основные требования к программному продукту для реализации реал-тайм чата с регистрацией и авторизацией пользователей. Они помогут создать удобное, безопасное и функциональное приложение, удовлетворяющее потребностям пользователей в области коммуникации и общения.

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
						7
Изм	Лист	№ докум.	Подпись	Дата		

## 3 Разработка программного обеспечения

### 3.1 Описание технологического стека разработки

Выбранный технологический стек для проекта по разработке реал-тайм чата с регистрацией и авторизацией пользователей обладает рядом преимуществ, которые делают его подходящим для данной задачи. Давайте рассмотрим каждый компонент технологического стека и его особенности.

Node.js:

- язык программирования: JavaScript;
- назначение: Node.js является средой выполнения JavaScript на сервере;
- особенности: Node.js обладает высокой производительностью, асинхронной природой и однопоточной архитектурой, что позволяет эффективно обрабатывать множество одновременных подключений в режиме реального времени.

React:

- язык программирования: JavaScript;
- назначение: React является JavaScript-библиотекой для построения пользовательских интерфейсов;
- особенности: React обеспечивает компонентный подход к разработке, что значительно упрощает создание и обслуживание сложных пользовательских интерфейсов. Он также обладает виртуальной DOM и эффективными механизмами обновления интерфейса, что повышает производительность при работе с реал-тайм данными.

Express:

- язык программирования: JavaScript;
- назначение: Express.js - это минималистичный веб-фреймворк для Node.js;
- особенности: Express.js обеспечивает простой и гибкий способ создания веб-сервера и обработки HTTP-запросов. Он предлагает удобную маршрутизацию, обработку сеансов, аутентификацию и авторизацию.

Sequelize:

- язык программирования: JavaScript;
- назначение: Sequelize - это ORM (Object-Relational Mapping) для Node.js, который облегчает взаимодействие с реляционными базами данных;
- особенности: Sequelize предоставляет удобные методы для создания и выполнения запросов к базе данных, а также для моделирования и миграции данных. Он облегчает работу с PostgreSQL и предоставляет интеграцию с Express.js.

Axios:

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		8



- язык программирования: JavaScript;
- назначение: Axios - это HTTP-клиент для выполнения запросов к серверу из JavaScript;

- особенности: Axios обладает простым и понятным интерфейсом, поддерживает Promise API и предоставляет широкий набор функций для работы с запросами и ответами, что делает его идеальным выбором для общения между клиентом и сервером.

#### JWT:

- язык программирования: JavaScript;
- назначение: JSON Web Tokens (JWT) - это открытый стандарт для аутентификации и авторизации;

- особенности: JWT позволяет создавать небольшие и самодостаточные токены, которые можно передавать между клиентом и сервером. Они безопасно содержат информацию о пользователе и могут использоваться для проверки подлинности и авторизации запросов.

#### MobX:

- язык программирования: JavaScript;
- назначение: MobX - это библиотека для управления состоянием приложения;

- особенности: MobX предоставляет простые и эффективные инструменты для управления состоянием приложения, что может быть особенно полезно в реал-тайм сценариях чата.

#### React-DOM и React-Router-DOM:

- язык программирования: JavaScript;
- назначение: React-DOM служит для рендеринга компонентов React на стороне клиента, а React-Router-DOM - для управления маршрутизацией приложения;

- особенности: React-DOM обладает быстрым и эффективным механизмом обновления виртуального DOM, а React-Router-DOM предоставляет инструменты для создания динамических и переиспользуемых маршрутов в React.

#### Bcrypt:

- язык программирования: JavaScript;
- назначение: Bcrypt - библиотека для хеширования паролей;
- особенности: Bcrypt предоставляет мощные функции хеширования паролей с солью, что обеспечивает безопасное хранение паролей пользователей в базе данных.

#### Dotenv:

- язык программирования: JavaScript;
- назначение: Dotenv - библиотека, которая загружает переменные окружения из файла .env;

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		9

— особенности: Dotenv помогает управлять конфиденциальной информацией и настройками приложения путем хранения их в защищенном файле .env, а не включая их в исходный код.

PG:

— язык программирования: JavaScript;

— назначение: PG (PostgreSQL) - это пакет для работы с PostgreSQL из JavaScript;

— особенности: PG облегчает взаимодействие с базой данных PostgreSQL, предоставляя набор функций для выполнения запросов, обработки результатов и управления соединением.

WS:

— язык программирования: JavaScript;

— назначение: WS - это библиотека для работы с WebSocket;

— особенности: WS предоставляет простой и эффективный способ создания серверных WebSocket и обмена данными в реальном времени между сервером и клиентами.

JWT-decode:

— язык программирования: JavaScript;

— назначение: jwt-decode - это библиотека JavaScript, которая позволяет декодировать и получать информацию из JSON Web Tokens (JWT);

— особенности: jwt-decode облегчает работу с JWT, предоставляя удобные функции для извлечения полезной нагрузки (payload) токена.

Выбранный технологический стек обладает следующими преимуществами:

— одноязычность: Весь стек разработки основан на JavaScript, что облегчает командную работу и повышает производительность разработчиков;

— широкая поддержка и активное сообщество: Все компоненты стека имеют активные сообщества разработчиков и постоянно обновляемую документацию, что помогает быстро решать проблемы и получать поддержку;

— гибкость и масштабируемость: Выбранные фреймворки и инструменты, такие как Express и Sequelize, обладают гибкостью и масштабируемостью, позволяя разрабатывать и масштабировать реал-тайм чат согласно потребностям проекта;

— безопасность: Использование библиотеки Bcrypt и JWT обеспечивает безопасность хранения паролей пользователей и безопасный обмен данными между клиентом и сервером;

— разработка в реальном времени: WebSocket-связь с помощью библиотеки WS позволяет обеспечить взаимодействие в режиме реального времени между клиентами и сервером.

Все эти факторы делают выбранный технологический стек подходящим для разработки реал-тайм чата с регистрацией и авторизацией пользователей, обеспечивая гибкость, безопасность и эффективность взаимодействия в реальном времени.

### 3.2 Описание алгоритма работы

Первоначальный этап разработки веб-приложения состоит из планирования и анализа системы. Планирование и анализ являются важными этапами, которые помогают успешно реализовать проект. Планирование позволяет определить цели и задачи веб-сайта, анализ же помогает понять потребности и предпочтения целевой аудитории. Определив цели и аудиторию сайта, можно перейти к проведению анализа конкурентов и требований рынка.

Следующий этап - проектирование и разработка дизайна. На этом этапе создаются макеты страниц, разрабатывается дизайн, учитывая корпоративный стиль и принципы UX/UI. Выбирается цветовая палитра, шрифты и другие дизайнерские элементы.

После этого необходимо выбрать технологический стек. Настраивается окружение для разработки с помощью Node.js и npm (Node.js Package Manager). Производится установка необходимых библиотек и фреймворков React, например, такие как react-router-dom для управления маршрутами в приложении.

Разработка фронтенда включает создание компонентов и страниц React, отображающих информацию о продуктах и услугах. Используются встроенные объекты управления состоянием, такие как useState, для эффективного управления данными в приложении. Взаимодействие с внешними данными, обращаясь к API через HTTP-запросы с помощью Fetch.

Далее нужно перейти к анализу организации базы данных в PostgreSQL. Создается схема базы данных и заполняется таблицы данными.

Переходя к разработке бэкенда, можно использовать Node.js и Express.js для создания серверного приложения. Нужно установить необходимые пакеты, такие как pg, nodemon, express. Создается основной файл сервера и определите маршруты для обработки HTTP-запросов.

Интегрировать серверное приложение с базой данных PostgreSQL можно, написав соответствующие SQL-запросы и скрипты. Используются библиотеки pg и sequelize для взаимодействия с PostgreSQL в Node.js.

Заключительным этапом является оптимизация кода для повышения производительности и быстрой загрузки страниц.

Далее осуществляется тестирование приложения, включая модульное, интеграционное и системное тестирование. Нужно удостовериться в безопасности приложения и его защите от потенциальных угроз. Также тестируется совместимость с различными браузерами и устройствами.

В приложении Б представлена диаграмма последовательности для демонстрации функционирования. Так как для общения необходимо хотя бы 2 человека, в диаграмме приведены Пользователь 1 и Пользователь 2 для демонстрации взаимодействия пользователей в системе.

Пользователь 1 ранее не был зарегистрирован в системе, именно поэтому проходит процесс регистрации путем ввода уникального адреса электронной

почты, пароля и имени. Далее, после успешной проверки системы на уникальность адреса электронной почты, пользователь получает сгенерированный токен, который позволит ему оставаться в системе.

В это время Пользователь 2 проходит процесс авторизации в системе, так как был ранее зарегистрирован. В данном случае, после ввода данных и отправки их на сервер, система проверяет, совпадает ли адрес электронной почты с тем, который был введен пользователем. Если да, в таком случае он проверяет совпадение пароля, соответствующего данному адресу электронной почты. После успешной проверки сервер генерирует токен и выдает его пользователю.

Пользователь 1 решает отправить сообщение Пользователю 2. После отправки система отправляет сообщение всем пользователям, у которых есть доступ к чату с данным идентификатором.

Пользователь 1 решает выйти из системы. Для этого системе необходимо удалить токен доступа для данного пользователя.

Администратор также должен пройти авторизацию перед тем, как получить доступ к системе. Данная роль дает привилегии – возможность удалять пользователей из базы данных системы. После корректного введения данных система находит нужную запись и удаляет данные о ней.

Для выхода из системы Администратор также должен сообщить серверу об этом для удаления токена доступа.

Данные хранятся в базе данных PostgreSQL. В ней содержатся 4 таблицы, представленные в Приложении В, которые связаны между собой внешними ключами.

### 3.3 Описание интерфейса пользователя

При создании интерфейса приложения мы руководствовались следующими принципами:

- простота использования: мы стремились создать простой и интуитивно понятный интерфейс, чтобы пользователи могли легко ориентироваться и выполнять необходимые действия;

- чистый дизайн: мы выбрали минималистичный дизайн, чтобы избежать излишней сложности и перегруженности интерфейса. Это помогает фокусироваться пользователям на основной функциональности приложения;

- система навигации: мы разработали логическую структуру навигации, чтобы пользователи могли легко находить необходимые разделы и функции. Меню или панели навигации размещены так, чтобы они были доступны в удобном месте и не мешали основному контенту;

- консистентность: мы старались использовать единые стили, шрифты, цвета и элементы дизайна на всех страницах приложения, чтобы создать единый и узнаваемый визуальный стиль;

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		12

— адаптивность: учитывая разнообразие устройств, на которых может работать приложение, мы сделали его адаптивным, чтобы интерфейс корректно отображался и работал на различных размерах экранов.

В нашем конкретном примере реализации интерфейса приложения основные приемы включают:

— центральная область чата: главная часть экрана выделена для отображения сообщений чата. Новые сообщения появляются снизу, а существующие сообщения прокручиваются вверх, чтобы облегчить чтение и взаимодействие;

— форма ввода сообщения: в нижней части экрана находится форма ввода, где пользователь может вводить текст своего сообщения;

— список пользователей: боковая панель может отображать список пользователей, для обеспечения связи и взаимодействия между участниками.

Страница вход, представленная на рисунке 1, содержит форму с полями для ввода электронной почты и пароля. В случае, если пользователь еще не зарегистрирован в системе, предлагается ссылка на страницу регистрации. При корректном заполнении полей, пользователь будет перенесен на страницу со списком чатов, в которых он состоит.

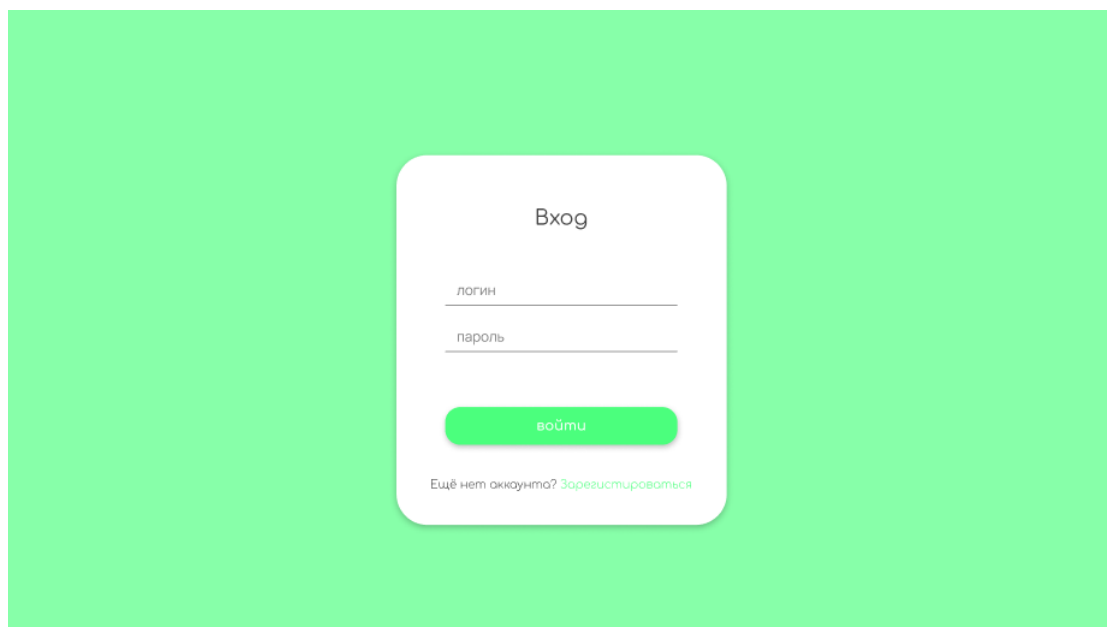
The image shows a login form titled "Вход" (Login) centered on a light blue background. The form is a white rounded rectangle containing two input fields labeled "логин" (login) and "пароль" (password). Below these fields is a blue button with the text "войти" (login). At the bottom of the form, there is a link that says "Ещё нет аккаунта? Зарегистрироваться" (Don't have an account? Register).

Рисунок 1 – Страница Вход

Форма для регистрации пользователя, представленная на рисунке 2, в отличие от формы входа, содержит в себе дополнительное поле для ввода имени пользователя. Также здесь пользователь может перейти на страницу Вход в случае, если у него уже есть ранее созданный аккаунт.

Рисунок 2 – Страница Регистрация

После успешной авторизации или регистрации пользователь попадает на страницу со списком чатов, представленной на рисунке 3, в которых он состоит. В левой части экрана представлен список пользователей, с которыми можно начать диалог. Внизу списка обозначается имя пользователя, в аккаунт которого был произведен вход. При клике на кнопку «Выйти» пользователь выйдет из аккаунта и его перенесет на страницу авторизации.

Рисунок 3 – Страница Чаты

При клике на имя другого пользователя или на диалог, будет открыта страница с чатом. Таким образом, станет доступна история сообщений с тем или

иным пользователем. В левой части экрана также остается доступной боковая панель со списком пользователей.

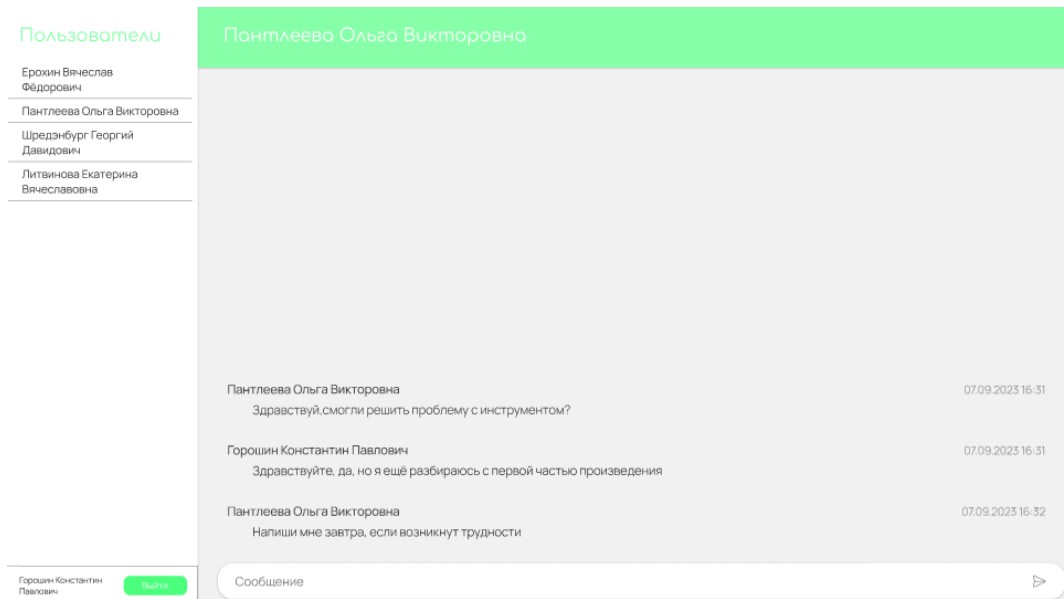


Рисунок 4 – Страница Чат

## 4 Тестирование приложения

### 4.1 План тестирования

Этапы контроля и приемки реал-тайм чата с авторизацией и регистрацией должны вести проверку многих аспектов программы. Несколько основных этапов контроля приведены и подробно описаны ниже.

Функциональный контроль. Проверка функциональности авторизации:

- регистрация нового пользователя с проверкой, что данные корректно сохраняются в базе данных пользователей;
- попытка входа в систему с используемыми при регистрации учетными данными и проверка, что пользователь успешно авторизуется;
- попытка входа в систему с неправильными учетными данными и проверка, что пользователь не может авторизоваться.

Проверка функциональности регистрации:

- заполнение формы регистрации с вводом корректных данных и проверка, что новый пользователь успешно создается в системе;
- заполнение формы регистрации с неправильными или неполными данными и проверка, что система не позволяет создать неправильного пользователя.

Проверка функциональности чата:

- отправка сообщения в чат и проверка, что оно отображается у всех пользователей, имеющих доступ к чату;
- получение сообщения в чате и проверка, что оно правильно отображается для текущего пользователя.

Security контроль. Проверка безопасности авторизации:

- попытка входа в систему с использованием неправильных учетных данных и проверка, что система не разрешает доступ.

Проверка защиты данных:

- проверка, что пароли хранятся в зашифрованном виде.

Проверка безопасности при обработке сообщений:

- проверка, что сообщения от пользователей отображаются только для авторизованных пользователей и не доступны через непосредственный URL доступ.

Performance контроль. Тестирование производительности системы при большой нагрузке:

- создание большого количества пользователей и проверка, что система продолжает корректно функционировать и обрабатывать запросы;
- проверка времени отклика системы при отправке и получении сообщений в чате при различном количестве активных пользователей.

Модульный контроль. Проверка корректности работы каждого модуля системы:

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		16



— проведение модульного тестирования каждого компонента, такого как модуль авторизации, модуль регистрации и модуль чата, для проверки их независимой работы и соответствия требованиям.

Интеграционный контроль. Проверка взаимодействия между компонентами системы:

— проведение интеграционного тестирования для проверки, что авторизация и регистрация успешно интегрированы с модулем чата и функционируют без ошибок.

Тестирование совместимости.

— проверка работоспособности в различных браузерах (Chrome, Firefox, Safari, Edge) и на их различных версиях.

Юзабилити тестирование сайта — это процесс оценки удобства использования веб-сайта пользователем. Оно направлено на выявление проблем, с которыми пользователи сталкиваются при взаимодействии с сайтом, и на предоставление рекомендаций по улучшению пользовательского опыта.

Для тестирования юзабилити сайта была выбрана методика «5-секундный тест». Его суть заключается в следующем: тестирующий показывает тестируемому скриншот или страницу сайта на 5 секунд, а затем, закрыв её, задаёт вопросы. Главное условие – тестируемый никогда не видел данный сайт.

Плюсы данного метода:

- скорость проведения;
- не требуется докупка дополнительного оборудования;
- простота.

Но данная методика имеет и свои минусы, такие как:

- для тестирования нескольких страниц веб-сайта потребуется большое количество людей;
- для выведения общей оценки потребуется проанализировать большой объем информации вручную.

## 4.2 Оценка результатов проведения тестирования

Тестирование веб-приложения имеет огромное значение в разработке и поддержке веб-проектов. Это обеспечивает получение качественного продукта. Тестирование помогает гарантировать, что веб-приложение работает так, как ожидается, и выполняет свои функции без ошибок. Также тестирование веб-приложения позволяет убедиться, что все функции и возможности приложения работают должным образом. Благодаря этому, пользователи смогут взаимодействовать с приложением в любой среде без проблем.

Функциональный контроль. Проверка функциональности авторизации:

- данные корректно сохраняются в базе данных пользователей при регистрации нового пользователя;

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		17

— пользователь успешно авторизуется при вводе данных, которые были указаны при регистрации;

— пользователь не может авторизоваться при попытке входа в систему с неправильными учетными данными.

Проверка функциональности регистрации:

— новый пользователь успешно создается в системе при корректном заполнении формы регистрации;

— система не позволяет создать пользователя при заполнении формы регистрации с неправильными или неполными данными.

Проверка функциональности чата:

— при отправке сообщения в чат оно отображается у всех пользователей, имеющих доступ к чату;

— при получении сообщения в чате оно правильно отображается для текущего пользователя.

Security контроль. Проверка безопасности авторизации:

— система не разрешает доступ при попытке входа в систему с использованием неправильных учетных данных.

Проверка защиты данных:

— пароли хранятся в зашифрованном виде.

Проверка безопасности при обработке сообщений:

— сообщения от пользователей отображаются только для авторизованных пользователей и не доступны через непосредственный URL доступ.

Performance контроль. Тестирование производительности системы при большой нагрузке:

— система продолжает корректно функционировать и обрабатывать запросы даже при большом количестве пользователей;

— маленькое время отклика системы при отправке и получении сообщений в чате при различном количестве активных пользователей.

Модульный контроль. Проверка корректности работы каждого модуля системы:

— было проведено модульное тестирование каждого компонента, таких как модуль авторизации, модуль регистрации и модуль чата, для проверки их независимой работы и соответствия требованиям.

Интеграционный контроль. Проверка взаимодействия между компонентами системы:

— авторизация и регистрация успешно интегрированы с модулем чата и функционируют без ошибок.

Тестирование совместимости.

— успешная проверка работоспособности в различных браузерах (Chrome, Firefox, Safari, Edge) и на их различных версиях.

Взаимодействие с бэкендом и API проходит успешно. Данные передаются согласованно между frontend и backend.

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		18

Для тестирования юзабилити продукта подбирались люди разных возрастов и профессий для того, чтобы максимально тщательно проанализировать возможности возникающих трудностей.

Далее приведён список вопросов, используемых в тестировании:

- «Для чего эта страница?»;
- «Что больше всего запомнилось?»;
- «По-вашему мнению, для кого этот сайт?»;
- «Что бы вы сделали дальше на этой странице?»;
- «Что вы ожидаете после выполнения этого действия?».

Для того, чтобы было удобнее вывести среднюю оценку среди всех испытуемых, было предпринято следующее: каждый из ответов на вопросы тестируемого оценивается от 1 до 10, где 1 – далеко от правды, а 10 – полное совпадение. Далее вычисляется среднее арифметическое анкеты, т.е. сумма всех ответов делится на 5 – их количество – и, в дальнейшем, для средней оценки всех анкет, также используется среднее арифметическое.

В таблице 1 приведены некоторые из результатов тестирования разных страниц реализованного веб-сайта. Отбор произошёл случайным образом.

Таблица 1 – Выборочные результаты тестирования юзабилити сайта

№ п/п тестируемого	Номер вопроса					Среднее арифм-е
	1	2	3	4	5	
1	10	10	10	8	10	9,6
2	10	10	10	10	10	10
3	9	8	9	10	10	9,2
4	10	10	9	10	10	9,8
5	9	10	10	10	10	9,8
6	8	10	10	8	10	9,2
7	10	9	8	10	8	9,0
8	9	8	9	9	10	9,0
9	9	9	9	10	9	9,2
10	9	10	10	10	10	9,8

Проведя анализ результатов, все тестируемые получили среднюю оценку за анкету выше 9 баллов из 10 возможных, что является высоким показателем понятности пользовательского интерфейса.

## Заключение

В рамках нашего проекта мы успешно достигли поставленной цели - создания инновационного веб-приложения в виде реал-тайм чата. Результатом нашей работы стало полноценное приложение, способное обеспечивать коммуникацию между пользователями в режиме реального времени.

В первую очередь, мы уделили огромное внимание разработке пользователям дружелюбного пользовательского интерфейса. Нашей целью было создать интерфейс, который будет интуитивно понятен и удобен для всех категорий пользователей. Мы включили в него необходимые элементы для регистрации и авторизации пользователей, а также для обмена сообщениями в режиме реального времени.

Однако, успешная работа пользовательского интерфейса требует надежной и безопасной серверной части приложения. Мы уделили большое внимание созданию этой части, которая отвечает за обработку запросов пользователей, хранение сообщений и управление безопасностью данных. Благодаря скрупулезной работе, серверная часть приложения обладает высокой производительностью и надежностью. Мы также реализовали механизмы регистрации и авторизации пользователей, обеспечив защиту от несанкционированного доступа и обеспечивая конфиденциальность данных. Это позволяет пользователям чувствовать себя комфортно и уверенно при использовании нашего приложения.

Одной из наших ключевых задач было обеспечение передачи сообщений в режиме реального времени с возможностью обновления чата без необходимости обновления страницы. Мы провели глубокий анализ и разработали эффективные алгоритмы передачи сообщений, которые позволяют пользователям обмениваться мгновенными сообщениями без задержек и простоев. Благодаря этому, чат обновляется плавно и без сбоев, создавая комфортное пользовательское впечатление.

Чтобы обеспечить стабильную и безошибочную работу нашего приложения, мы провели обширное тестирование и отладку функциональности в различных ситуациях. Мы проверили работу приложения на разных устройствах и операционных системах, а также с различными объемами пользовательской нагрузки. Тестирование помогло нам выявить и исправить возможные проблемы и ошибки, гарантируя стабильное функционирование приложения.

Завершая наш проект, хочется отметить, что мы гордимся результатом нашей работы. Мы создали инновационное веб-приложение - реал-тайм чат, которое обеспечивает быструю, безопасную и удобную коммуникацию между пользователями. Мы уверены, что наше приложение будет полезным и позволит пользователям эффективно общаться и делиться информацией. Мы готовы продолжать развивать и совершенствовать наше приложение в будущем, чтобы удовлетворить потребности и ожидания наших пользователей.

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		20

## Список литературы

- 1 Async/await. — Текст: электронный // learn.javascript.ru: [сайт]. — Режим доступа: <https://learn.javascript.ru/async-await>
- 2 HTTP-запросы: структура, методы, строка статуса и коды состояния. — Текст: электронный // selectel.ru: [сайт]. — Режим доступа: <https://selectel.ru/blog/http-request/>
- 3 Introduction to JSON Web Tokens. — Текст: электронный // jwt.io: [сайт]. — Режим доступа: <https://jwt.io/introduction>
- 4 LocalStorage, sessionStorage. — Текст: электронный // learn.javascript.ru: [сайт]. — Режим доступа: <https://learn.javascript.ru/localstorage>
- 5 Middleware (Усилители). — Текст: электронный // reactdev.ru: [сайт]. — Режим доступа: <https://reactdev.ru/libs/redux/react-redux/middleware-usiliteli/>
- 6 Route. — Текст: электронный // reactrouter.com: [сайт]. — Режим доступа: <https://reactrouter.com/en/main/route/route>
- 7 Sequelize v6. — Текст: электронный // sequelize.org: [сайт]. — Режим доступа: <https://sequelize.org/docs/v6/>
- 8 Использование Node.js и веб-сокетов для создания сервиса чата. — Текст: электронный // code.tutsplus.com: [сайт]. — Режим доступа: <https://code.tutsplus.com/ru/-nodejs-----net-34482t>
- 9 Основы JavaScript. — Текст: электронный // learn.javascript.ru: [сайт]. — Режим доступа: <https://learn.javascript.ru/first-steps>
- 10 Учебное пособие по диаграммам последовательностей: полное руководство с примерами. — Текст: электронный // creately.com: [сайт]. — Режим доступа: <https://creately.com/blog/ru/диаграмма/учебное-пособие-по-последовательной>

## Приложение А (обязательное)

### Диаграмма прецедентов системы

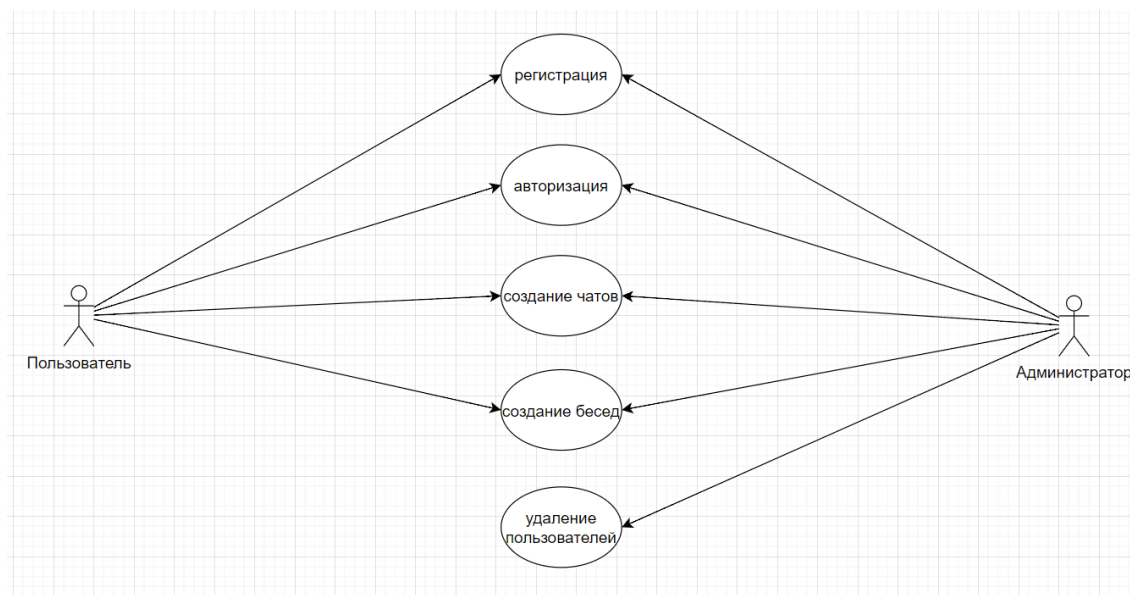


Рисунок А1 - Диаграмма вариантов использования (прецедентов)

Изм	Лист	№ докум.	Подпись	Дата

ОКЭИ 09.02.07. 4323. 15 ПЗ

Лист

22

Приложение Б  
(обязательное)

Диаграмма последовательности системы

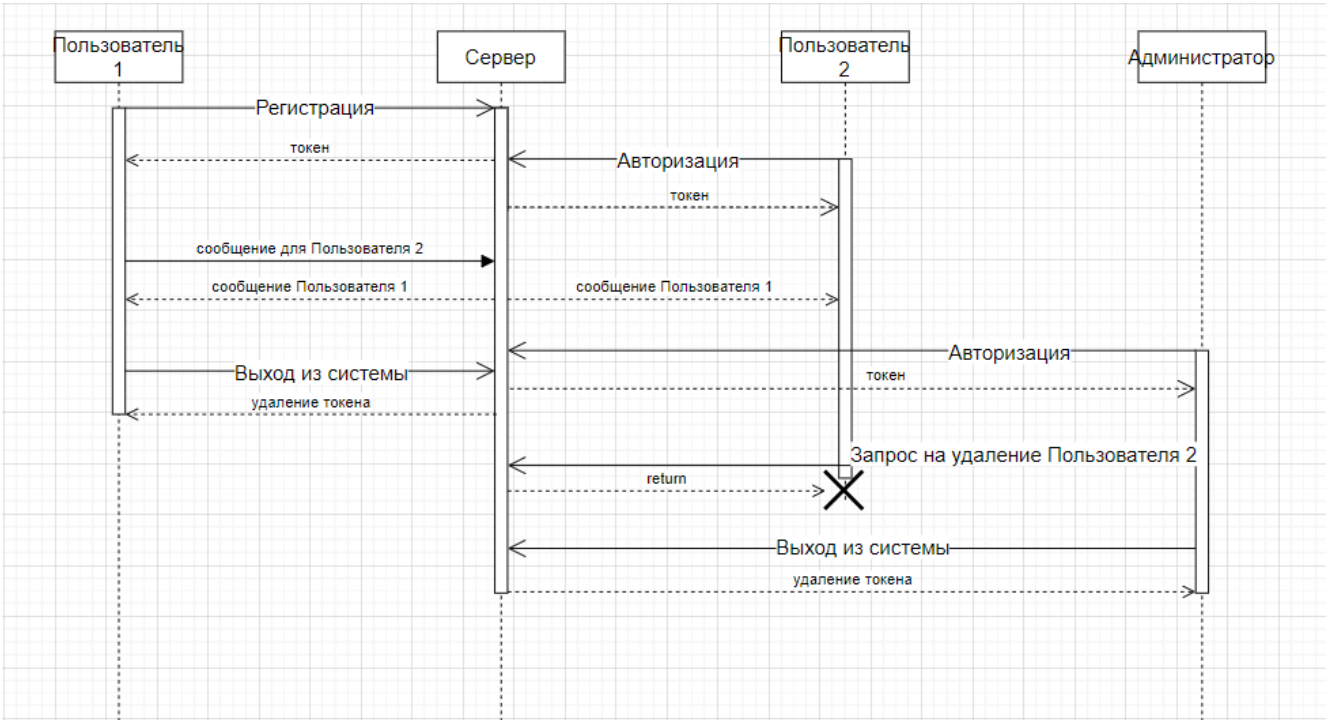


Рисунок Б1 - Диаграмма последовательности

Приложение В  
(обязательное)

Диаграмма классов системы

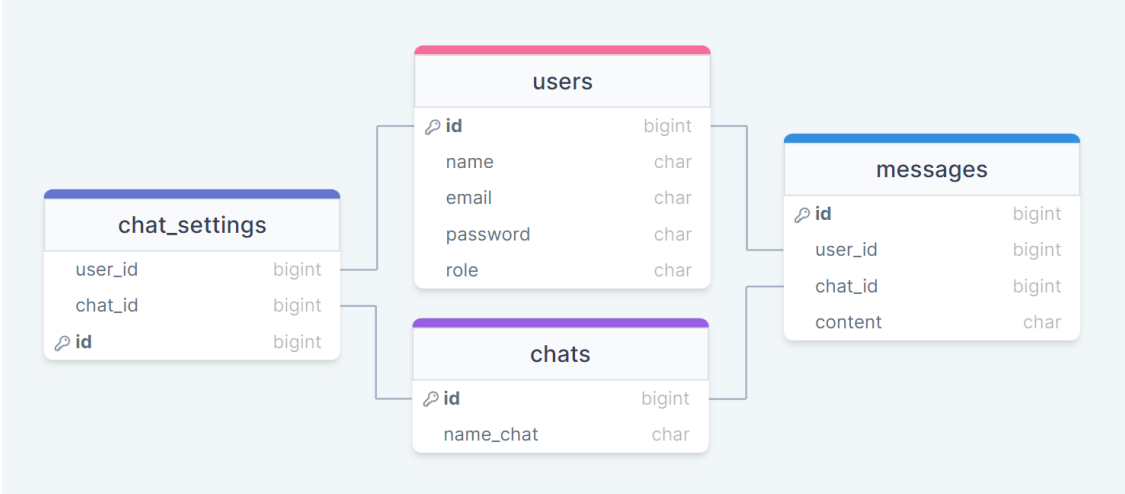


Рисунок В1 - Диаграмма классов



## Приложение Г (обязательное)

### Листинг программного кода

index.js:

```
import express, { json } from "express";
import WebSocket, { WebSocketServer } from "ws";
import sequelize from "../db.js";
import models from "../models/models.js";
import cors from "cors";
import router from "../routes/index.js";
import errorHandler from "../middleware/ErrorHandlerMiddleware.js";

const app = express()

app.use(cors())
app.use(express.json())
app.use('/api', router)

const start = async () => {
  try {
    await sequelize.authenticate() // аутентификация пользователя
    await sequelize.sync() // сверка состояния бд со схемой данных в models

    const server = app.listen(5000, () => console.log('Сервер стартовал на порту номер 5000'))

    const wss = new WebSocketServer({ server })

    wss.on('connection', function connection(ws) {
      ws.on('message', function (message) {
        const parsedMessage = JSON.parse(message)
        let idRoom
        let formattedMessages
        switch (parsedMessage.event) {
          case 'connection':
            idRoom = parsedMessage.idRoom;
            ws.id = idRoom;
            const messageHistory = models.Message.findAll({ where: { chatId: idRoom }, order: [['createdAt', 'DESC']] })
            messageHistory.then(messages => {
              formattedMessages = messages.map(({ id, content }) => ({ id, content }));
              console.log(formattedMessages);
            })
            wss.clients.forEach((client) => {
```

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ докум.	Подпись	Дата		25

```

        if (client.id == idRoom) {
            formattedMessages.map(
                client.send(
                    JSON.stringify(
                        { event: 'history', id, content }
                    )))
        }
    })
    console.log(`Подключение установлено ${idRoom}`);
    break;
    case 'message':
        // отправка сообщений в бд
        const { id, chat_id, content } = parsedMessage
        const newMessage = models.Message.create({ userId: id,
chatId: chat_id, content: content })
        wss.clients.forEach(function each(client) {
            if (client.id == idRoom) {
                client.send(JSON.stringify({ event: 'message', data:
newMessage })))
            }
        });
        break;
    }
})
})

} catch (error) {
    console.log(error);
}

}

start()

app.use(errorHandler) // обязательно должно быть в конце!

```

models.js:

```

import sequelize from "../db.js";
import { DataTypes } from "sequelize";

const User = sequelize.define('user', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    name: {type: DataTypes.STRING, allowNull: false},
    email: {type: DataTypes.STRING, allowNull: false, unique: true},
    pass: {type: DataTypes.STRING, allowNull: false},
    role: {type: DataTypes.STRING, defaultValue: "USER", allowNull: false},
});

const Chat = sequelize.define('chat', {
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

```

Изм	Лист	№ докум.	Подпись	Дата

ОКЭИ 09.02.07. 4323. 15 ПЗ

Лист

26

```

    chat_name: {type: DataTypes.STRING, allowNull: false},
    type: {type: DataTypes.STRING, allowNull: false}
  });

const Message = sequelize.define('message', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
  content: {type: DataTypes.STRING}
});

const Chat_settings = sequelize.define('chat_settings', {
  id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
});

User.belongsToMany(Chat, {through: Chat_settings})
Chat.belongsToMany(User, {through: Chat_settings})

User.hasMany(Message)
Message.belongsTo(User)

Chat.hasMany(Message)
Message.belongsTo(Chat)

export default {User, Chat, Chat_settings, Message}

```

mainController.js:

```

import { config } from 'dotenv';
config();

import models from "../models/models.js";
import ApiError from "../error/ApiError.js"
import bcrypt from "bcrypt";
import jwt from 'jsonwebtoken';

const User = models.User;
const Chat = models.Chat;
const Chat_settings = models.Chat_settings;
const Message = models.Message;

const generateJWT = (id, name, email, role) => {
  return jwt.sign(
    { id, name, email, role }, // содержит
    process.env.SECRET_KEY, // ключ шифрования
    { expiresIn: '24h' } // время действия
  )
}

class MainController {
  // Регистрация
  async registration(req, res, next) {

```

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
Изм	Лист	№ док.им.	Подпись	Дата		27

```

const { name, email, pass, role } = req.body

if (!email || !pass) {
  return next(ApiError.badRequest("Некорректные данные"))
}

const candidate = await User.findOne({ where: { email } })

if (candidate) {
  return next(ApiError.badRequest("Пользователь с таким email уже существует"))
}

const hashPassword = await bcrypt.hash(pass, 5)
const user = await User.create({ name, email, role, pass: hashPassword })
const token = generateJWT(user.id, user.email, user.name, user.role)

return res.json({ token })
}

// Авторизация
async login(req, res, next) {
  const { email, pass } = req.body
  const user = await User.findOne({ where: { email } })
  if (!user) {
    return next(ApiError.internal("Пользователь не найден"))
  }

  // Проверка правильности пароля - сравнение введенного пароля и пароля с БД
  let comparePassword = bcrypt.compareSync(pass, user.pass)
  if (!comparePassword) {
    return next(ApiError.internal("Указан неверный пароль"))
  }

  const token = generateJWT(user.id, user.email, user.name, user.role)

  res.json({ token })
}

// Проверка вошел ли пользователь в систему
async check(req, res, next) {
  const token = generateJWT(req.user.id, req.user.email, req.user.name, req.user.role)
  res.json({ token })
}

async userId(req, res, next) {
  const { id } = req.user;
  res.json({ id })
}

```

```

    }
  }

  export default new MainController()

```

#### CheckRoleMiddleware.js:

```

import jwt from "jsonwebtoken"

export default function CheckRoleMiddleware(role) {
  return function (req, res, next) {
    if (req.method === "OPTIONS") {
      next()
    }
    try {
      const token = req.headers.authorization.split(' ')[1]
      if (!token) {
        return res.status(401).json({ message: "Пользователь не авторизован" })
      }
      const decoded = jwt.verify(token, process.env.SECRET_KEY)
      if (decoded.role !== role) {
        res.status(403).json({ message: "Нет доступа" })
      }
      // в поле user добавили данные, которые вытащили из токена
      // и во всех функциях этот user будет доступен
      req.user = decoded
      next() // вызов следующего в цепочке middleware
    } catch (error) {
      res.status(401).json({ message: "Пользователь не авторизован" })
    }
  }
}

```

#### http.js:

```

import axios from "axios";

const $host = axios.create({
  baseURL: 'http://localhost:5000/'
})

const $authHost = axios.create({
  baseURL: 'http://localhost:5000/'
})

const authInterceptor = config => {
  config.headers.authorization = `Bearer ${localStorage.getItem('token')}`
  return config
}

$authHost.interceptors.request.use(authInterceptor)

```

Изм	Лист	№ докум.	Подпись	Дата

ОКЭИ 09.02.07. 4323. 15 ПЗ

Лист

29

```
export {
  $host,
  $authHost
}
```

userAPI.js:

```
import { $authHost, $host } from "../http";
import { jwtDecode } from "jwt-decode";

export const registration = async (name, email, pass) => {
  const { data } = await $host.post('api/registration', { name, email, pass })
  localStorage.setItem('token', data.token)
  return jwtDecode(data.token)
}

export const login = async (email, pass) => {
  const { data } = await $host.post('api/login', { email, pass })
  localStorage.setItem('token', data.token)
  return jwtDecode(data.token)
}

export const check = async () => {
  if (localStorage.getItem('token')) {
    const { data } = await $authHost.get('api/auth')
    localStorage.setItem('token', data.token)
    return jwtDecode(data.token)
  }
}

export const userId = async () => {
  const { data } = await $authHost.get('api/userId')
  return data.id
}
```

					ОКЭИ 09.02.07. 4323. 15 ПЗ	Лист
						30
Изм	Лист	№ докум.	Подпись	Дата		