

Министерство науки и высшего образования РФ Федеральное государственное бюджетное образовательное учреждение высшего образования «Белгородский государственный технологический университет им. В.Г. Шухова»

## **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**«Разработка front-end Web-приложения – учебной среды с чатами и AI-анализом кода лабораторных работ»**

**Автор работы:** Бондаренко Сергей Владимирович, студент группы ПВ-212

**Руководитель:** Мельников Антон Борисович, руководитель департамента автоматизации бизнеса ООО «Технологии надежности»

Белгород 2025 г.

**Цель:** улучшить учебный процесс преподавателей и студентов через разработку front-end Web-приложения, автоматизирующего проверку заданий и упрощающего коммуникацию.

### Задачи:

- 1 Провести анализ существующих образовательных решений.
- 2 Определить архитектуру Web-приложения и технологический стек.
- 3 Разработать пользовательский интерфейс.
- 4 Реализовать модули для управления учебными структурами, заданиями и системой общения.
- 5 Интегрировать модуль автоматической проверки решений с использованием ИИ.
- 6 Провести тестирование бизнес-логики Web-приложения.

## Анализ существующих образовательных решений

Проблема заключается в том, что Google Classroom, Microsoft Teams for Education и Moodle решают задачи управления заданиями. VK, Telegram и Viber предназначены только для общения. А CodeSignal и Codility реализуют исключительно AI-анализ кода.

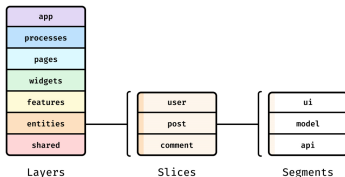
Из-за разделения функций между разными сервисами преподаватели и студенты вынуждены постоянно переключаться между несколькими приложениями, что создаёт неудобства и снижает эффективность работы.

Аналоги/Функции	Система заданий	Общение	AI-анализ
Google Classroom	+	—	—
MS Teams	+	+	—
Moodle	+	—	—
CodeSignal	—	—	+
Codility	—	—	+

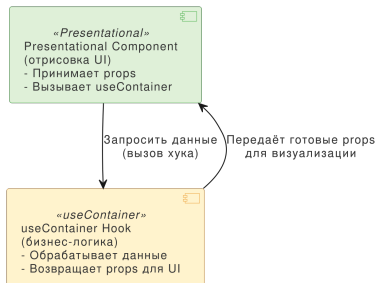


- **Next.js** — используется для серверного рендеринга (SSR), маршрутизации и повышения производительности клиентского приложения.
- **React** — обеспечивает декларативный подход к построению компонентного пользовательского интерфейса.
- **TypeScript** — добавляет статическую типизацию, улучшает читаемость и сопровождаемость кода.

# Архитектура Web-приложения



FSD-архитектура



Контейнерно-презентационный подход

- Feature-Sliced Design (FSD) — архитектурный подход, основанный на разделении приложения на функциональные срезы и уровни, что упрощает масштабирование и сопровождение.
- Container/Presentational Components — паттерн, разделяющий компоненты на логические и визуальные.

# Модуль аутентификации и авторизации

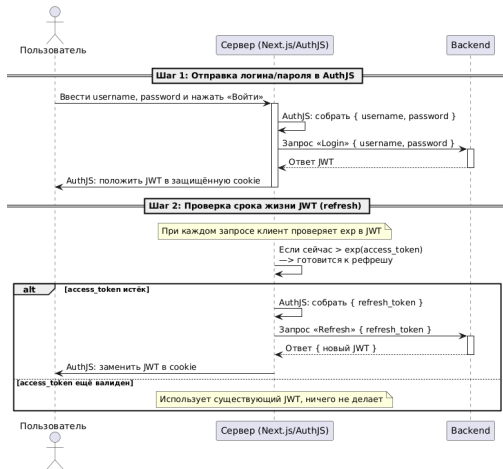


Диаграмма авторизации и аутентификации



Вход

Email \*

besmilev@mail.ru

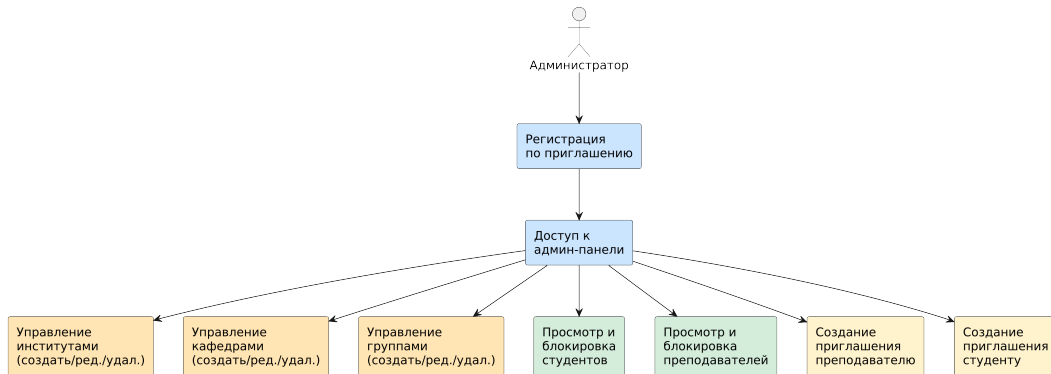
Пароль \*

.....

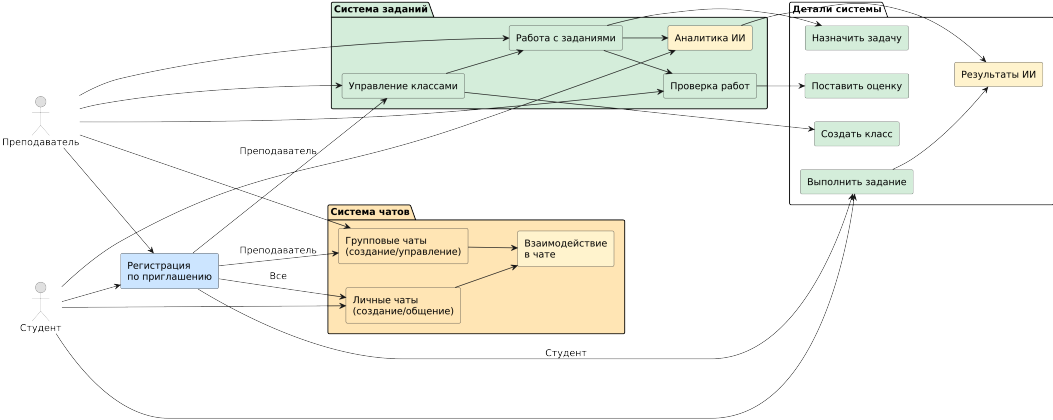
Войти

Интерфейс входа

# Диаграмма взаимодействия администратора с web-приложением



# Диаграмма взаимодействия студента и преподавателя с web-приложением





# Процесс создания задачи

[← Назад](#)

## Создать задачу

Название

Крайний срок сдачи

Описание

Первое задание посвящено ознакомлению со средой разработки CLINE.

Количество баллов

10 ^ v

Анализ лабораторной работы с помощью ИИ

Запрос для анализа

Проверить файл на выполнение задания.

Прикрепить

Файл

Добавить

Задание.pdf 244 KB 256

Интерфейс создания задачи

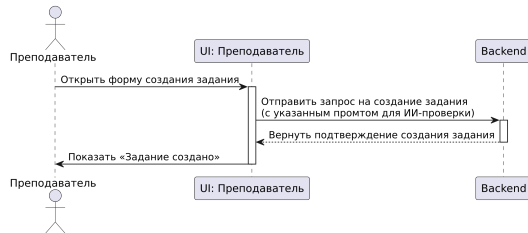
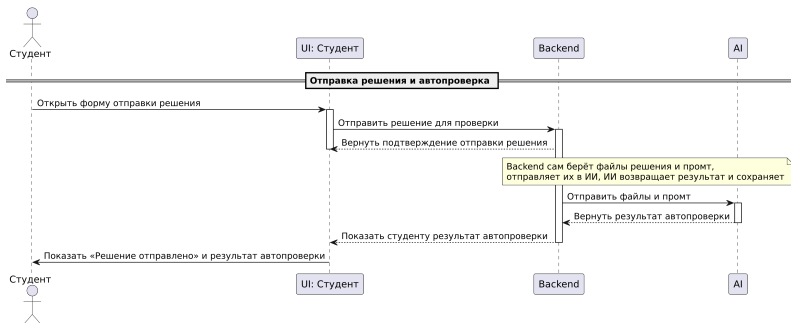
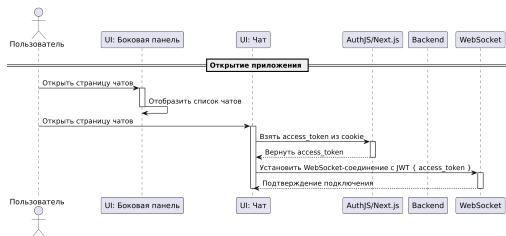


Диаграмма создания задачи

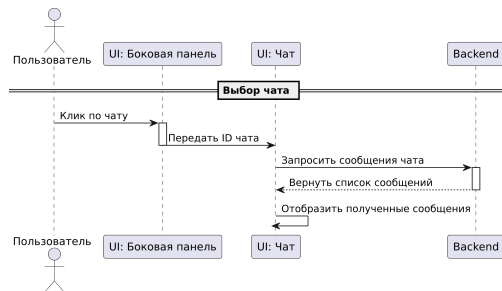
# Диаграмма взаимодействия студента с заданием



# Диаграммы взаимодействия пользователя с чатами

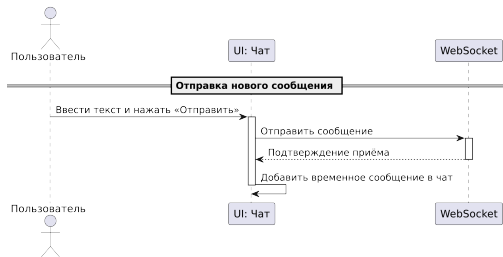


Подключение и инициализация чата

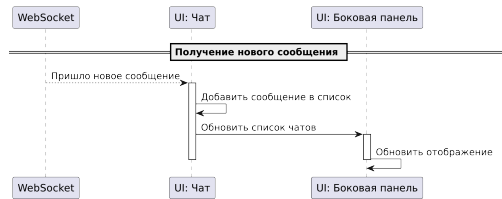


Открытие и отображение списка чатов

# Диаграммы отправки и получения сообщений пользователем



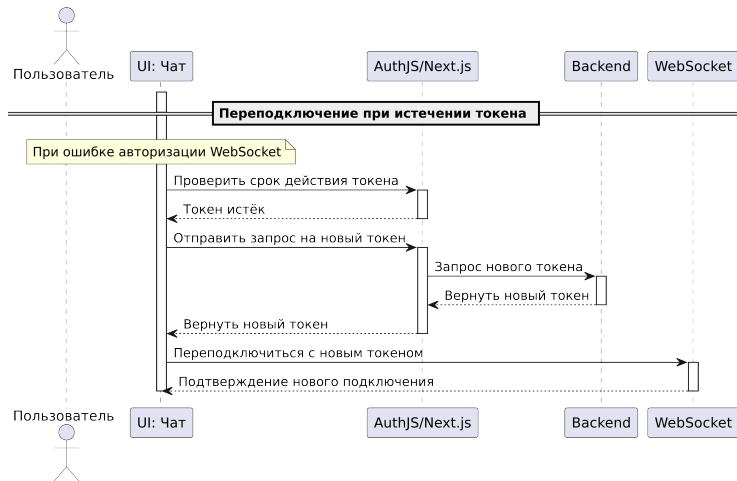
Отправка сообщения пользователем



Получение сообщения получателем

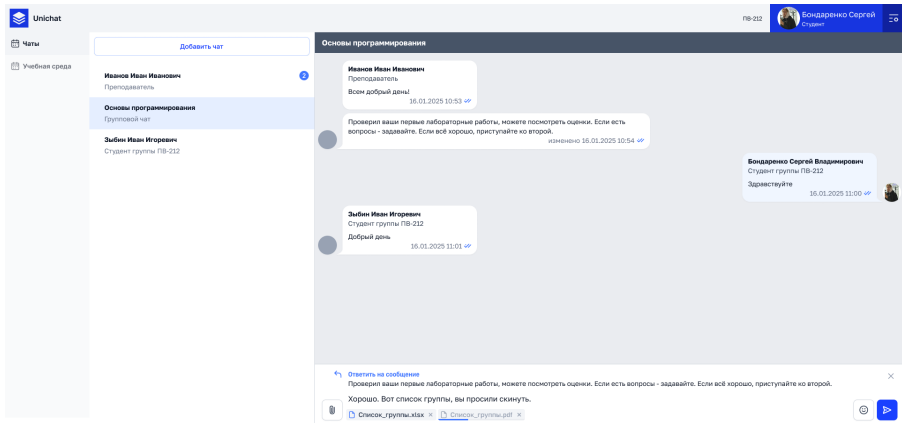
# Диаграмма взаимодействия клиентской части с WebSocket

При истечении срока действия access-токена реализован механизм его автоматического обновления и повторного переподключения WebSocket-соединения.



# Интерфейс чата

Пользовательский интерфейс чата предоставляет возможность обмена сообщениями в режиме реального времени.



- Модульное тестирование (unit testing): для каждой функции и компонента написаны независимые тесты, покрывающие: граничные и некорректные входные данные (undefined, пустые массивы), типичные сценарии и пограничные случаи.
- TDD–подход (Test–Driven Development): реализация функций по циклу «*test* → *fail* → написать минимальный код → *test* → *pass* → рефакторинг».
- Покрытие ветвлений (branch coverage): каждый сценарий условных операторов (*if/else*, тернарные выражения, *switch*) проверяется отдельными тестами.

## Метрики тестирования

### Покрытие кода приложения.

File	%Stmts	%Branch	%Funcs	%Lines
All files	95.65	77.77	100.00	100.00
features/Chats/lib	100.00	100.00	100.00	100.00
mergeMessages.ts	100.00	100.00	100.00	100.00
shared/ui/TextEditor/lib	90.47	68.42	100.00	100.00
processTextToTiptap.ts	94.11	72.22	100.00	100.00
processTiptapToText.ts	75.00	0.00	100.00	100.00

### Покрытие кода библиотеки компонентов.

File	%Stmts	%Branch	%Funcs	%Lines
All files	52.89	48.76	20.68	52.65
lib/dict/getDeepValue.ts	86.36	73.33	100.00	85.71
lib/dict/setDeepValue.ts	100.00	100.00	100.00	100.00
ui/DateTimePicker/lib/changeInterval.ts	100.00	96.29	100.00	100.00



## Заключение

В ходе выполнения выпускной квалификационной работы:

- 1 Проведен анализ существующих образовательных решений.
- 2 Определена архитектура Web-приложения и технологический стек.
- 3 Разработан пользовательский интерфейс.
- 4 Реализованы модули для управления учебными структурами, заданиями и системой общения.
- 5 Интегрирован модуль автоматической проверки решений с использованием ИИ.
- 6 Проведено тестирование бизнес-логики Web-приложения.

Результатом работы стало Web-приложение с системой управления заданиями, поддерживающей автоматическую проверку решений студентов с помощью ИИ, а также чатами для общения в реальном времени через WebSocket.