

## ОПРЕДЕЛЕНИЯ, СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ

- 1) CSR (Client-Side Rendering) — отрисовка страницы на стороне клиента после загрузки минимального каркаса с сервера.
- 2) SSR (Server-Side Rendering) — рендеринг веб-страниц на стороне сервера перед отправкой клиенту.
- 3) SSG (Static Site Generation) — генерация статических HTML-страниц на этапе сборки приложения.
- 4) JWT (JSON Web Token) — формат токена для безопасной передачи информации между сторонами в виде JSON-объекта.
- 5) API (Application Programming Interface) — программный интерфейс, обеспечивающий взаимодействие между компонентами программного обеспечения.
- 6) UI (User Interface) — пользовательский интерфейс.
- 7) Socket.IO — библиотека для организации двусторонних WebSocket-соединений между клиентом и сервером с автоматическим фоллбеком.
- 8) CSRF (Cross-Site Request Forgery) — тип атаки, при которой злоумышленник вынуждает браузер пользователя выполнить нежелательный запрос от его имени; в веб-приложениях применяется защита с помощью специальных токенов.
- 9) XSS (Cross-Site Scripting) — уязвимость, позволяющая внедрять скрипты стороннего происхождения на страницы; одной из мер защиты является хранение JWT в HTTP-only cookie.
- 10) SSO (Single Sign-On) — единый вход, позволяющий пользователю аутентифицироваться один раз и использовать доступ ко множеству сервисов без повторной авторизации.
- 11) HTTP (Hypertext Transfer Protocol) — протокол передачи гипертекстовых данных между клиентом и сервером; используется для REST-запросов.

					ОПРЕДЕЛЕНИЯ, СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чатом и AI-анализом кода лабораторных работ	Лит.	Лист	Листов
Руковод.	Мельников А.Б.						4	94
Консул.						БГТУ им. В.Г. Шухова, ПВ-212		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							

- 12) REST (Representational State Transfer) — архитектурный стиль взаимодействия с API через HTTP-методы (GET, POST и др.).
- 13) JSON (JavaScript Object Notation) — текстовый формат обмена данными, широко используемый в REST-API и для передачи полезной нагрузки токенов.
- 14) HTML (HyperText Markup Language) — язык разметки веб-страниц, создающий структуру документа.
- 15) CSS (Cascading Style Sheets) — язык каскадных таблиц стилей для описания внешнего вида HTML-элементов.
- 16) SCSS (Sassy CSS) — расширение CSS с поддержкой переменных, вложенности и других возможностей препроцессора.
- 17) CRUD (Create, Read, Update, Delete) — базовые операции над данными, которые выполняются при создании, чтении, обновлении и удалении записей.
- 18) CLI (Command Line Interface) — интерфейс командной строки, используемый для генерации проекта и выполнения скриптов (например, Angular CLI или Next.js CLI).
- 19) MVP (Minimum Viable Product) — минимально жизнеспособный продукт, концепция, описывающая начальную версию продукта с базовым функционалом.
- 20) Hooks (хуки) — в контексте библиотеки React это специальные функции, позволяющие функциональным компонентам использовать состояние, жизненный цикл и другие возможности React.

					ОПРЕДЕЛЕНИЯ, СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		5

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	9
1 ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ .....	11
1.1 Введение в предметную область .....	11
1.2 Анализ существующих образовательных платформ .....	12
1.2.1 Система управления обучением Moodle.....	12
1.2.2 Платформа дистанционного обучения Google Classroom .....	12
1.2.3 Платформа для онлайн-обучения Microsoft Teams .....	12
1.2.4 Платформы для анализа текста программы .....	13
1.3 Проблемы существующих решений .....	13
1.4 Потребности образовательной среды.....	13
1.4.1 Доступность материалов и заданий .....	13
1.4.2 Автоматизация проверок и оценки .....	14
1.4.3 Удобная система заданий и общения.....	14
1.4.4 Интеграция всех процессов в одну систему .....	14
1.4.5 Вывод.....	14
1.5 Технологии разработки клиентской части приложения .....	15
1.5.1 Язык программирования TypeScript .....	15
1.5.2 Библиотека React .....	16
1.5.3 Веб-фреймворк Angular.....	17
1.5.4 Веб-фреймворк Vue.js .....	17
1.5.5 Сравнительный анализ фреймворков .....	18
1.6 Требования к функциональности приложения .....	24
1.7 Диаграмма вариантов использования .....	25
1.8 Коммуникация и взаимодействие .....	25
1.9 Безопасность данных .....	26
1.9.1 Авторизация с использованием библиотеки Auth.js .....	26
1.9.2 Роль библиотека Auth.js .....	26
1.9.3 Меры защиты.....	27

					СОДЕРЖАНИЕ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чаттами и AI-анализом кода лабораторных работ	Лит.	Лист	Листов
Руковод.	Мельников А.Б.						6	94
Консул.						БГТУ им. В.Г. Шухова, ПВ-212		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							

1.10 Система создания заданий .....	27
1.10.1 Общее описание модуля .....	27
1.10.2 Функциональные возможности.....	28
1.10.3 Автоматический анализ текста программы.....	28
1.11 Анализ текста программ на основе ИИ.....	29
1.11.1 Общее описание модуля DeepSeek.....	29
1.11.2 Функциональные возможности модуля DeepSeek.....	29
1.11.3 Принцип работы модуля DeepSeek.....	30
1.12 Тестирование с использованием библиотеки Jest .....	30
1.13 Выводы.....	31
2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА.....	32
2.1 Архитектура клиентской части системы .....	32
2.1.1 Общая структура архитектуры .....	32
2.1.2 Архитектурная диаграмма .....	32
2.1.3 Слои архитектуры Feature-Sliced Design .....	33
2.1.4 Концепция срезов .....	33
2.1.5 Горизонтальное деление на сегменты.....	34
2.2 Проектирование интерфейсных подсистем и экранов.....	35
2.2.1 Выделение ключевых интерфейсных подсистем .....	35
2.2.2 Страницы и их структура .....	38
2.2.3 Компоненты и принципы их структурирования .....	41
2.2.4 Распределение логики по слоям архитектуры .....	41
2.2.5 Пользовательские сценарии .....	42
2.3 Проектирование взаимодействия с сервером .....	44
2.3.1 Аутентификация и управление токенами.....	44
2.3.2 Унифицированная функция отправки запросов.....	44
2.3.3 Взаимодействие через протокол WebSocket .....	45
2.4 Интеграция модуля DeepSeek в архитектуру проекта .....	46
2.4.1 Контейнеризация модуля DeepSeek .....	46
2.4.2 Преимущества контейнеризированного подхода .....	46
2.4.3 Взаимодействие клиентской части с модулем DeepSeek.....	47
2.4.4 Вывод.....	47

2.5	Обеспечение безопасности клиентской части .....	48
2.5.1	Разграничения доступа .....	48
2.5.2	Роль и защита при работе с форматируемым текстом .....	49
2.5.3	Вывод .....	49
2.6	Покрытие бизнес-логики юнит-тестами .....	50
3	ПРОГРАММНАЯ РЕАЛИЗАЦИЯ .....	51
3.1	Архитектура по Feature-Sliced Design .....	51
3.1.1	Слой <i>shared</i> .....	51
3.1.2	Слой <i>entities</i> .....	52
3.1.3	Слои <i>features, widgets, pages</i> .....	52
3.1.4	Пример виджета RegistrationUniversity .....	53
3.1.5	Соглашения по именованию и структуре .....	53
3.2	Собственная UI-библиотека и генерация форм .....	54
3.2.1	Общая идея и мотивация .....	54
3.2.2	Компонент <i>FormBuilder</i> .....	54
3.2.3	Типы элементов схемы и их поведение .....	56
3.2.4	Преимущества и выводы .....	58
3.2.5	Работа с токеном JWT .....	58
3.3	Модуль «Регистрация и вход» .....	62
3.3.1	Модуль формирования и отправки приглашений .....	64
3.3.2	Последовательная диаграмма работы модуля Classrooms .....	66
3.3.3	Последовательная диаграмма работы модуля Chats .....	67
3.4	Тестирование .....	68
3.4.1	Покрытие текста программы в приложении .....	69
3.4.2	Покрытие текста программы в отдельной библиотеке .....	69
3.4.3	Методы тестирования .....	70
	ЗАКЛЮЧЕНИЕ .....	71
	СПИСОК ЛИТЕРАТУРЫ .....	74
	ПРИЛОЖЕНИЕ А .....	76
	ПРИЛОЖЕНИЕ Б .....	81
	ПРИЛОЖЕНИЕ В .....	86
	ПРИЛОЖЕНИЕ Г .....	89

## ВВЕДЕНИЕ

Развитие цифровых технологий в сфере образования значительно меняет способы взаимодействия между преподавателями и студентами, предоставляя новые возможности для обучения и обмена информацией. В условиях дистанционного и смешанного обучения особенно важной становится необходимость создания приложения, которое бы объединяло образовательные инструменты в едином пространстве. Приложения, решающие задачи взаимодействия, позволяют сократить барьеры между преподавателями и студентами, улучшить коммуникацию и повысить качество образования. Цифровая среда должна обеспечивать не только размещение учебных материалов и заданий, но и средства для общения, автоматической оценки и анализа решений с использованием современных технологий, включая искусственный интеллект.

**Актуальность** темы заключается в потребности создания интегрированного образовательного веб-приложения, которое объединяет функции чатов, проведения занятий и автоматического анализа решений, используя возможности ИИ. Сейчас отсутствует единое решение, которое бы эффективно сочетало в себе эти ключевые аспекты: общения через чаты, создание заданий и автоматическую проверку решений с помощью ИИ. Современные системы, как правило, фрагментированы — отдельные модули для чатов, другие для размещения заданий, третьи для автоматической проверки текста программ, что значительно усложняет организацию учебного процесса и снижает его эффективность. Разработка интегрированного решения, которое объединит эти элементы, позволяет улучшить качество образовательного процесса, повысив продуктивность студентов и преподавателей, а также упростив взаимодействие и автоматизировав многие рутинные задачи.

**Целью** данной работы является улучшение и упрощение опыта работы преподавателей и процесса обучения студентов путем разработки клиентской части интегрированного образовательного веб-приложения, обеспечивающего эффективное взаимодействие, автоматизированную проверку решений и удоб-

					ВВЕДЕНИЕ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чатами и AI-анализом кода лабораторных работ	Лит.	Лист	Листов
Руковод.	Мельников А.Б.						9	94
Консул.						БГТУ им. В.Г. Шухова, ПВ-212		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							

ные средства коммуникации.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) Проанализировать предметную область и существующие системы, выявив их сильные и слабые стороны;
- 2) Определить архитектурные и технологические решения, подходящие для реализации клиентской части веб-приложения;
- 3) Спроектировать пользовательский интерфейс, обеспечивающий интуитивное и удобное взаимодействие для преподавателей и студентов;
- 4) Разработать компоненты для управления учебными структурами (институт, кафедра, группа), заданиями и чатами;
- 5) Интегрировать средства для автоматизированной проверки решений студентов с применением ИИ;
- 6) Реализовать тестирование бизнес-логики веб-приложения для обеспечения её корректности и эффективности.

Структура пояснительной записки включает следующие разделы:

- 1) В первом разделе рассматриваются особенности предметной области, проводится анализ существующих решений и обоснование выбора технологий и методов проектирования;
- 2) Во втором разделе описывается архитектура клиентской части веб-приложения, структура пользовательского интерфейса, проектирование компонентов и их взаимодействие;
- 3) В третьем разделе приводится описание реализации: структура приложения, используемые технологии, описание экранов и взаимодействий, примеры реализации различных компонентов веб-приложения;
- 4) В заключении приводятся выводы по выполненной работе, оценивается эффективность разработанного интерфейса и функционала, а также определяются направления для дальнейшего развития и улучшения системы.

					ВВЕДЕНИЕ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		10

# 1 ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Введение в предметную область

Современные образовательные процессы переживают значительные изменения под воздействием цифровых технологий. В условиях быстрого роста объёмов информации и перехода на дистанционное и смешанное обучение возникает потребность в создании платформ, которые объединяют различные образовательные инструменты в единую систему. Проблемы, с которыми сталкиваются преподаватели и студенты, включают фрагментацию существующих решений: чаты для общения, отдельные системы для размещения и проверки заданий, а также инструменты для анализа решений студентов.

Существующие платформы не всегда обеспечивают интеграцию всех этих функций в одном приложении, что приводит к необходимости использования множества разных сервисов для выполнения учебных задач. В рамках образовательных процессов это усложняет взаимодействие между преподавателями и студентами, увеличивает время на организацию обучения и снижает его эффективность.

Одной из важнейших задач является создание платформы, которая объединяет все эти компоненты в одном месте, обеспечивая удобный интерфейс для студентов и преподавателей. Такая система должна включать:

- 1) Возможность создания и размещения учебных заданий;
- 2) Автоматическое тестирование решений студентов с использованием искусственного интеллекта для проверки правильности текста программы;
- 3) Модуль чата для общения студентов с преподавателями и внутри групп;
- 4) Централизованный доступ к учебным материалам.

Интеграция всех этих функций в одну платформу позволит значительно упростить организацию учебного процесса, улучшить взаимодействие между преподавателями и студентами, а также повысить качество обучения за счёт автоматизации рутинных задач.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чатами и AI-анализом кода лабораторных работ	Лит.	Лист	Листов
Руковод.	Мельников А.Б.						11	94
Консул.						БГТУ им. В.Г. Шухова, ПВ-212		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							



## 1.2 Анализ существующих образовательных платформ

Современные образовательные платформы, такие как Moodle [1], Google Classroom [2], Microsoft Teams для образования [3], а также специализированные решения, предназначенные для работы с программированием, предлагают различные функциональные возможности для взаимодействия преподавателей и студентов. Однако каждая из этих платформ имеет свои ограничения и не всегда покрывает все потребности в рамках единой системы.

### 1.2.1 Система управления обучением Moodle

Moodle является одной из самых популярных образовательных платформ, используемых во многих учебных заведениях [1]. Она предоставляет инструменты для размещения учебных материалов, организации тестов и заданий, а также ведения онлайн-курсов. Однако, несмотря на свои возможности, Moodle не предоставляет встроенных решений для автоматической проверки текста программы студентов, а также не включает в себя продвинутые механизмы общения в реальном времени, что делает её менее эффективной для динамичного взаимодействия в процессе обучения.

### 1.2.2 Платформа дистанционного обучения Google Classroom

Google Classroom предлагает простоту в использовании и позволяет интегрировать различные Google сервисы [2]. Платформа позволяет преподавателям создавать задания, прикреплять материалы и отслеживать выполнение студентами. Однако Google Classroom не предоставляет функциональности для автоматического анализа решений, особенно в контексте программирования. Это требует интеграции с внешними инструментами, что усложняет использование системы в образовательных учреждениях.

### 1.2.3 Платформа для онлайн-обучения Microsoft Teams

Microsoft Teams, в отличие от Moodle и Google Classroom, активно используется для организации видеоконференций и групповых чатов [3]. Он позволяет преподавателям и студентам взаимодействовать в реальном времени, а также интегрирует различные сервисы Microsoft 365. Однако, как и в случае с другими платформами, Microsoft Teams не предоставляет функционала для интегрированного анализа текста программы студентов с использованием ис-

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		12

кусственного интеллекта, что ограничивает его возможности в обучении программированию.

#### **1.2.4 Платформы для анализа текста программы**

Существуют специализированные платформы, такие как CodeSignal [4], Codility [5] и LeetCode [6], которые позволяют преподавателям и работодателям тестировать навыки программирования студентов. Эти системы используют алгоритмы для автоматической проверки решений, однако они ограничены в функционале взаимодействия с преподавателями и студентами, а также не обеспечивают централизованный доступ к учебным материалам и заданиям.

#### **1.3 Проблемы существующих решений**

Основной проблемой существующих образовательных платформ является фрагментация функционала. На данный момент нет единой платформы, которая бы эффективно объединяла создание и проверку заданий, общение преподавателей и студентов, а также использовала бы технологии ИИ для автоматизированного анализа решений студентов. Это затрудняет образовательный процесс и снижает его эффективность, особенно в условиях быстро меняющихся требований дистанционного обучения.

Таким образом, для улучшения образовательного процесса существует необходимость в разработке единой интегрированной платформы, которая бы сочетала в себе все эти компоненты и обеспечивала бы максимально удобное взаимодействие для всех участников учебного процесса.

#### **1.4 Потребности образовательной среды**

Современные образовательные процессы предъявляют высокие требования к функциональности учебных платформ. Для эффективного взаимодействия между преподавателями и студентами необходимо создавать приложения, которые обеспечивают организационную и техническую поддержку всех ключевых элементов образовательного процесса.

##### **1.4.1 Доступность материалов и заданий**

Материалы и задания должны быть доступны студентам в любое время. Приложение должно обеспечивать размещение учебных ресурсов в различных

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		13

форматах (текст, видео, презентации) и упрощать их поиск и использование. Это позволяет студентам готовиться к занятиям и выполнять задания без привязки ко времени, а преподавателям — быстро обновлять и дополнять учебные модули.

#### **1.4.2 Автоматизация проверок и оценки**

Автоматизированная проверка заданий существенно ускоряет процесс получения обратной связи. Использование искусственного интеллекта для анализа текста программы позволяет выявлять ошибки, давать подсказки и оценивать работы без участия преподавателя. Это освобождает ресурсы для индивидуальной поддержки студентов и более сложной экспертной оценки.

#### **1.4.3 Удобная система заданий и общения**

Приложение должно включать удобную систему создания и отслеживания заданий. Важно, чтобы преподаватели могли формулировать задания, прикреплять к ним материалы и получать результаты выполнения. Неотъемлемой частью также является возможность общения между участниками процесса — как в групповых, так и личных чатах, для обмена мнениями и получения поддержки.

#### **1.4.4 Интеграция всех процессов в одну систему**

Отдельные решения для чатов, размещения заданий и анализа текста программы создают фрагментированную среду. Необходима единая платформа, объединяющая все эти компоненты. Это упрощает взаимодействие, повышает удобство и эффективность обучения, а также снижает затраты на сопровождение и обучение работе с системой.

#### **1.4.5 Вывод**

Таким образом, при проектировании образовательной платформы следует учитывать потребности в постоянном доступе к материалам, автоматической проверке решений, поддержке взаимодействия и целостности функционала в рамках одного интерфейса.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		14

## 1.5 Технологии разработки клиентской части приложения

Для реализации клиентской части платформы выбраны современные инструменты, обеспечивающие модульность, производительность, типизацию и масштабируемость интерфейса.

### 1.5.1 Язык программирования TypeScript

JavaScript является одним из самых популярных языков программирования для веб-разработки. Он широко используется для создания динамичных веб-страниц и приложений, поскольку позволяет работать с элементами DOM-дерева (представление HTML-документа в виде дерева тегов), асинхронно загружать данные и обеспечивать интерактивность пользовательских интерфейсов. Однако JavaScript имеет важный недостаток — отсутствие статической типизации. Это означает, что переменные и функции не привязываются к определённым типам данных, что может привести к ошибкам на этапе выполнения, которые трудно обнаружить в процессе разработки. Особенно это может быть проблемой в крупных приложениях, где сложно отслеживать все возможные типы данных и их изменения.

Для устранения этих проблем был разработан язык TypeScript, являющийся надмножеством JavaScript. Язык программирования TypeScript добавляет в JavaScript статическую типизацию, что позволяет разработчикам явно указывать типы данных для переменных и функций. Это значительно снижает вероятность ошибок и улучшает поддержку приложения в будущем. Благодаря строгой типизации язык программирования TypeScript помогает предотвращать баги, связанные с динамическими типами в языке программирования JavaScript, и улучшает автозаполнение в редакторах текста программы. Язык программирования TypeScript распространяется как библиотека, которую можно интегрировать в проекты на языке JavaScript, обеспечивая совместимость с существующим приложением и позволяя постепенно внедрять типизацию без необходимости переписывать весь проект. Это особенно важно в крупных и масштабируемых приложениях, где несколько разработчиков работают с общими компонентами, и типизация помогает поддерживать консистентность текста программы на протяжении всего проекта. Для получения большей информации смотрите документацию [7].

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		15

### 1.5.2 Библиотека React

React — библиотека для построения пользовательских интерфейсов, разработанная Facebook (см. документацию [8]). Она широко используется в веб-разработке благодаря своей простоте, гибкости и высокой производительности. Библиотека React обеспечивает декларативный стиль программирования, при котором разработчик описывает, как должен выглядеть интерфейс при заданном состоянии, а библиотека самостоятельно обновляет DOM-дерево при изменениях. Это упрощает разработку сложных и динамичных интерфейсов.

Можно выделить следующие ключевые особенности

- 1) Приложение разбивается на переиспользуемые и изолированные компоненты, что отражает компонентный подход;
- 2) JSX-синтаксис объединяет синтаксисы языка JavaScript и HTML-разметки, упрощая написание пользовательского интерфейса;
- 3) Использование виртуального дерева компонентов позволяет эффективно обновлять только изменённые элементы страницы;
- 4) Применение функций подключения обеспечивает современное управление состоянием и побочными эффектами.

Преимущества для образовательных платформ:

- 1) Быстрая разработка за счёт декларативности и компонентного подхода;
- 2) Большое сообщество и развитая экосистема;
- 3) Поддержка SSR и SSG рендерингов при использовании фреймворка Next.js;
- 4) Простая интеграция с библиотеками и сторонними сервисами.

Ограничения:

- 1) Отсутствие встроенной архитектуры требует выбора и настройки дополнительных инструментов;
- 2) Более низкий порог входа может привести к «разнообразию» архитектурных подходов в команде;
- 3) Отсутствует маршрутизация и SSR рендеринга.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		16

### 1.5.3 Веб-фреймворк Angular

Angular — полнофункциональный фреймворк для создания больших приложений [9]. В отличие от библиотеки React, он требует строгого следования архитектурным правилам, что особенно ценно при командной разработке.

Одна из ключевых возможностей фреймворка Angular — автоматическое обновление данных в обоих направлениях между моделью и представлением. Для управления зависимостями между компонентами фреймворк использует специальные сервисы [10]. Разработчики могут быстро создавать элементы приложения через командную строку благодаря встроенным инструментам. Полная поддержка языка программирования TypeScript обеспечивает строгую типизацию на всех уровнях приложения.

При использовании фреймворка Angular обеспечивает чёткую структуру проекта, что упрощает совместную работу. Встроенные механизмы проверки данных в формах ускоряют разработку. Готовые решения для маршрутизации и HTTP-запросов позволяют сразу перейти к реализации бизнес-логики.

Однако фреймворк Angular имеет высокий порог входа, поэтому новичкам придется нелегко. Другим ограничением является размер итогового приложения: даже минимальная сборка имеет размер около 500 КБ, что влияет на скорость загрузки. Фреймворк также имеет жёсткие требования к структуре приложения, оставляя мало свободы для отклонений от официальных рекомендаций.

### 1.5.4 Веб-фреймворк Vue.js

Vue.js — гибкий фреймворк, объединяющий лучшие идеи библиотек React и Angular [11]. Его архитектура идеально подходит как для быстрого прототипирования, так и для проектов средней сложности.

Основу Vue.js составляет система автоматического обновления интерфейса при изменении данных. Каждый компонент реализуется в одном файле, где объединены HTML-шаблон, логика на языке программирования JavaScript и стили на языке разметки CSS. Для создания анимаций предусмотрены специальные встроенные директивы.

Фреймворк Vue.js выгоден в проектах благодаря понятной документации с интерактивными примерами. Низкий порог вхождения позволяет новичкам

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		17

быстро освоиться.

К ограничениям Vue.js относится меньший размер сообщества по сравнению с библиотеками React и Angular, что может замедлить поиск решений сложных задач. Без использования фреймворка Nuxt.js возможности серверного рендеринга существенно ограничены. Также наблюдается дефицит готовых библиотек компонентов интерфейса по сравнению с другими фреймворками.

### 1.5.5 Сравнительный анализ фреймворков

На основе сравнения характеристик трёх популярных JavaScript-фреймворков (см. таблицу 1.1) можно сделать следующие выводы: библиотека React с фреймворком Next.js демонстрирует высокую производительность и даёт гибкую архитектуру, а встроенная поддержка SSR/SSG рендера и SEO-оптимизация делают этот стек идеальным выбором для масштабируемых образовательных приложений. Фреймворк Angular, с другой стороны, благодаря строгой типизации и продуманной структуре, подходит для крупных систем, где важны единообразие и безопасность приложения. Фреймворк Vue.js сочетает в себе простоту освоения и возможность расширения экосистемы, что отлично для быстрого запуска MVP-продуктов (продуктов с минимальным необходимым функционалом) и небольших команд.

Таблица 1.1 – Сравнение характеристик фреймворков

Параметр	React и Next.js	Angular	Vue.js
Кривая обучения	Средняя	Высокая	Средняя
Сообщество	Крупное	Крупное	Растущее
Производительность	Высокая	Средняя	Высокая
Гибкость архитектуры	Высокая	Минимальная	Средняя
Поддержка SSR	Встроенная	Встроенная	Nuxt.js
Поддержка TypeScript	Да	Да	Да
Готовая маршрутизация	Да	Да	Да

### Фреймворк Next.js

Фреймворк содержит в себе серверный рендеринг и «гидратацию» клиентского приложения, обеспечивая быструю и плавную работу приложений. Благодаря гидратации браузер получает только необходимый на текущий момент минимальный объём текста программы языка программирования

JavaScript, а по мере навигации и взаимодействия фреймворк автоматически подгружает дополнительные скрипты, расширяя уже отрендеренный контент. Это снижает время первого отображения страницы и ускоряет переходы между разделами, что особенно важно для крупных проектов с большими бандлами.

В документации [12] выделяют несколько режимов серверного рендеринга:

- 1) Классический серверный рендеринг SSR предполагает, что HTML-страница полностью генерируется на сервере при каждом запросе и отправляется клиенту. Пользователь сразу видит готовый контент без ожидания выполнения JavaScript-логики.
- 2) Режим SSG (Static Site Generation) использует статическую генерацию страниц на этапе билда проекта. Все HTML-файлы подготавливаются заранее и хранятся на сервере, что позволяет отдавать их мгновенно и без лишних вычислений [13].
- 3) Подход ISR (Incremental Static Regeneration) представляет собой расширение SSG рендеринга. Он позволяет автоматически пересобирать отдельные страницы через заданные интервалы времени, сохраняя преимущества статической отдачи и обеспечивая актуальность контента [13].

Такой многообразный набор инструментов делает фреймворк Next.js крайне гибким: разработчики могут подобрать оптимальный способ рендеринга для каждой страницы — от полностью статических лендингов до динамических разделов с актуальными данными. В результате ваше приложение получает высокую производительность, хорошую индексацию поисковиками и комфортный пользовательский опыт без избыточной передачи данных.

## Библиотека Socket.IO

Библиотека Socket.IO с открытым исходным кодом предназначена для реализации двустороннего взаимодействия между клиентом и сервером в режиме реального времени. В соответствии с документацией [14] особенностью данной технологии является использование собственного настраиваемого поверх WebSocket протокола с возможностью автоматического переключения на альтернативные методы связи при отсутствии поддержки WebSocket протокола.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		19



Преимущества использования библиотеки Socket.IO:

- 1) Обеспечивает гибкость за счёт полного контроля над архитектурой соединения, включая маршрутизацию сообщений, обработку событий, систему комнат и пространств имён;
- 2) Позволяет достигать масштабируемости благодаря поддержке кластеризации и горизонтального масштабирования с использованием адаптеров из библиотеки Redis;
- 3) Обеспечивает совместимость с средой Node.js, благодаря чему легко интегрируется в стек на его основе и упрощает реализацию единой инфраструктуры;
- 4) Достигает высокой производительности за счёт низкой задержки при передаче сообщений через постоянное соединение между клиентом и сервером;
- 5) Поддерживает интеграцию с языком программирования Python посредством библиотеки *python-socketio*, которая предоставляет аналогичные возможности для реализации чатов и двустороннего общения между клиентом и сервером.

Из недостатков библиотеки можно выделить следующее:

- 1) Необходимость разработки и поддержки собственной серверной инфраструктуры;
- 2) Повышенная сложность при масштабировании без использования внешних инструментов;
- 3) Отсутствие встроенной панели мониторинга или аналитики соединений.

### **Сервис обмена событиями в реальном времени Pusher**

Pusher — это облачная платформа, предоставляющая инструменты для реализации push-уведомлений и двусторонней передачи данных в реальном времени [15]. В отличие от библиотеки Socket.IO, платформа Pusher представляет собой менеджер, абстрагирующий низкоуровневые детали инфраструктуры.

Преимущества использования платформы Pusher:

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		20

- 1) Интеграция осуществляется с минимальными усилиями благодаря SDK-инструментам и готовым клиентским библиотекам, которые позволяют быстро настроить соединение и передавать события;
- 2) Масштабируемость достигается на уровне платформы и не требует участия разработчика;
- 3) Надёжность обеспечивается за счёт устойчивой облачной инфраструктуры с балансировкой нагрузки;
- 4) Аналитика и мониторинг доступны через панель управления, где отображаются данные о соединениях, событиях и каналах.

#### Ограничения:

- 1) Бесплатный тариф ограничен по числу соединений и событий, что делает использование невыгодным при росте нагрузки;
- 2) Использование зависит от стороннего сервиса, что создаёт потенциальные риски, связанные с отказоустойчивостью внешнего провайдера;
- 3) Возможности изменения ограничены, так как структура событий и поведение зависят от особенностей платформы.

### Двунаправленный обмен сообщениями в языке программирования JavaScript

WebSocket — это сетевой протокол, предназначенный для организации постоянного двунаправленного соединения между клиентом и сервером, что позволяет обмениваться сообщениями в реальном времени. Язык программирования JavaScript предоставляет встроенный класс *WebSocket* для организации двустороннего обмена данными между клиентом и сервером [16]. Этот класс реализует базовую функциональность протокола WebSocket, предоставляя разработчикам простой способ обмена данными в реальном времени без необходимости использовать сторонние библиотеки. Однако, несмотря на свою доступность, использование класса *WebSocket* требует значительных усилий для реализации различных важных аспектов взаимодействия.

К примеру, при использовании класса *WebSocket* разработчик должен самостоятельно реализовать:

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		21

- 1) Переподключение сокета при его падении;
- 2) Буферизацию сообщений в случае разрыва соединения;
- 3) Обработку таймаутов и ошибок;
- 4) Поддержку различных сред Node.js и браузера;
- 5) Масштабирование на сервере.

Таким образом, несмотря на его доступность и гибкость, использование класса *WebSocket* требует написания значительного объёма дополнительной логики. Это делает его менее удобным для быстрого внедрения в проект, особенно в случае масштабируемых приложений.

С учётом всех этих факторов было принято решение отказаться от использования низкоуровневого класса *WebSocket* в пользу более высокоуровневых решений, предоставляемых библиотекой *Socket.IO* или платформой *Pusher*, которые предоставляют необходимую функциональность и обрабатывают множество нюансов «из коробки», позволяя сосредоточиться на бизнес-логике приложения.

### Сравнение и выбор технологии для чатов

При сравнении библиотеки *Socket.IO* и платформы *Pusher* необходимо учитывать как технические, так и организационные аспекты. В таблице 1.2 представлено краткое сопоставление ключевых параметров:

Таблица 1.2 – Сравнение технологий для реализации чатов

Критерий	Socket.IO	Pusher
Тип решения	Open-source библиотека	Облачный менеджер
Контроль над архитектурой	Полный	Ограниченный
Поддержка	Через Redis и кластеризацию	Встроенная на уровне платформы
Простота настройки	Средняя (требует сервера)	Высокая
Затраты на	Бесплатно	Платная модель
Интеграция с Node.js	Нативная	Через SDK
Надёжность	Высокая	Высокая
Кастомизация	Да	Нет

С учётом специфики проекта — ограниченного бюджета, необходимости полной кастомизации и тесной интеграции с Node.js-сервером — наилучшим выбором является использование библиотеки Socket.IO. Данная библиотека предоставляет все необходимые механизмы для реализации масштабируемой и надёжной системы общения, при этом позволяя оптимизировать производительность без привлечения сторонних сервисов.

Более того, благодаря открытому исходному тексту библиотеки Socket.IO не ограничивает разработчика в выборе архитектурных решений, а также обеспечивает возможность расширения функционала в будущем. В условиях ограниченных ресурсов образовательной платформы такой подход оказывается наиболее целесообразным.

### **Библиотека аутентификации Auth.js**

Auth.js — это библиотека для реализации аутентификации и авторизации в веб-приложениях. Она является официальным решением, рекомендуемым и поддерживаемым фреймворком Next.js, что гарантирует хорошую интеграцию и поддержку всех необходимых функций. Библиотека позволяет легко подключать сторонние провайдеры аутентификации, такие как Google, Facebook и другие (см. [17]), а также реализовывать собственную аутентификацию с использованием базы данных. Библиотека Auth.js обеспечивает надёжную защиту пользовательских данных, управление сессиями, работу с токенами и предоставляет удобный интерфейс для быстрой настройки. Это решение упрощает реализацию всех ключевых механизмов безопасности, освобождая разработчиков от необходимости погружаться в тонкости реализации.

### **Библиотека управления состоянием приложения Redux**

Redux — это библиотека для управления состоянием в приложениях, основанных на библиотеке React [18]. Она используется для централизованного хранения состояния приложения, что облегчает обмен данными между компонентами и упрощает их взаимодействие. Библиотека Redux помогает избежать проблемы передачи данных через большое количество вложенных компонентов в больших приложениях, когда передача данных через множество вложенных компонентов становится сложной. Хотя библиотека Redux часто используется в более сложных приложениях, в данном проекте его роль за-

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		23

ключается в том, чтобы сделать взаимодействие между компонентами более организованным и простым.

## 1.6 Требования к функциональности приложения

Разрабатываемое приложение представляет собой образовательную платформу, ориентированную на университетскую среду. Основная цель — предоставить единое пространство для организации учебного процесса, взаимодействия между преподавателями и студентами, а также управления учебными структурами.

Каждый университет может зарегистрироваться на платформе и получить доступ к собственной административной панели. Через неё администраторы создают внутреннюю структуру, включающую институты, кафедры и учебные группы. Эти сущности служат основой для распределения доступа, назначения преподавателей и приглашения студентов.

Преподаватели, закреплённые за определёнными кафедрами, получают доступ ко всем учебным группам соответствующего подразделения. Через личную панель преподавателя реализован следующий функционал:

- 1) Создание групповых чатов для любой группы своей кафедры;
- 2) Размещение учебных материалов — как в групповых чатах, так и в личных сообщениях;
- 3) Формирование и отправка заданий для студентов;
- 4) Просмотр и анализ результатов выполнения заданий;
- 5) Предоставление обратной связи студентам.

Студенты, присоединённые к учебным группам, имеют доступ к следующему функционалу:

- 1) Групповым чатам своей учебной группы;
- 2) Личной переписке с преподавателями;
- 3) Материалам, отправленным преподавателями;
- 4) Заданиям, опубликованным в рамках их группы;
- 5) Форме отправки решений и получению обратной связи.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		24

## 1.7 Диаграмма вариантов использования

На рисунке А.1 представлена диаграмма вариантов использования, демонстрирующая взаимодействие пользователей с системой. Каждый пользователь (актёр) обладает определённым набором действий, доступных в рамках его роли.

Преподаватели в системе могут создавать и управлять учебными группами и заданиями. Они имеют возможность взаимодействовать со студентами через чаты, а также использовать инструменты искусственного интеллекта для анализа текста программы в лабораторных работах.

Студенты могут участвовать в групповых и индивидуальных чатах, выполнять назначенные задания и просматривать результаты своей работы.

Администраторы системы управляют структурными подразделениями (институтами и кафедрами), а также работают с учетными записями пользователей (студентов и преподавателей). В их обязанности входит создание приглашений для новых участников системы.

## 1.8 Коммуникация и взаимодействие

Ключевым элементом платформы является система обмена сообщениями, включающая следующие компоненты:

- 1) Групповые чаты — позволяют участникам учебной группы общаться в режиме реального времени, передавать файлы (поддерживаются форматы .jpg, .png, .mp4, .pdf, .zip и другие) и привязываются к конкретным группам;
- 2) Личная переписка — поддерживает обмен сообщениями один на один (между студентом и преподавателем либо между студентами), с возможностью передачи тех же форматов файлов, что и в групповых чатах;
- 3) Отображение истории сообщений — пользователи могут просматривать ранее отправленные и полученные сообщения;
- 4) Индикаторы прочтения — система отображает статус прочтения сообщений собеседником;
- 5) Уведомления о новых сообщениях — пользователи получают оповещения о входящих сообщениях в реальном времени.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		25

Таким образом, платформа предоставляет функциональность, охватывающую весь цикл учебной коммуникации.

## 1.9 Безопасность данных

В данном разделе приводится обзор ключевых принципов и практик, обеспечивающих конфиденциальность, целостность и доступность данных в приложении.

### 1.9.1 Авторизация с использованием библиотеки Auth.js

Механизм авторизации в приложении реализован на основе JWT токена и библиотеки Auth.js, которая обеспечивает безопасную и гибкую аутентификацию пользователей на клиентской стороне. Процесс состоит из следующих этапов:

- 1) Пользователь выполняет вход с помощью логина и пароля или через одного из OAuth-провайдеров (например, Google). Библиотека Auth.js инициирует процесс аутентификации и при успешной проверке получает токен JWT.
- 2) Полученный токен содержит минимально необходимые полезные данные, включая идентификатор пользователя, его роль и срок действия. Токен сохраняется в данные cookie с флагами *HTTP-only*, *Secure* и *SameSite=Strict*, что защищает от атак типа XSS и CSRF.
- 3) При каждом обращении к защищённым ресурсам токен автоматически прикрепляется к запросу. Если токен оказался недействительным или истёк, библиотека Auth.js может автоматически выполнить его обновление с использованием токена обновления, если он присутствует.

### 1.9.2 Роль библиотека Auth.js

Библиотека Auth.js упрощает реализацию безопасной авторизации за счёт следующих возможностей:

- 1) Обрабатывает все основные сценарии авторизации, включая вход, выход из системы и обновление токена;

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		26

- 2) Управляет хранением токенов и обеспечивает их безопасную передачу между клиентом и сервером;
- 3) Генерирует CSRF-токены, предотвращая атаки с подделкой межсайтовых запросов;
- 4) Поддерживает стратегию единого входа, обеспечивая единообразный доступ к различным системам.

### 1.9.3 Меры защиты

Для повышения безопасности механизма авторизации реализованы дополнительные меры:

- 1) Токен доступа действует ограниченное время (например, 5 минут), а токен обновления — 30 дней;
- 2) Библиотека Auth.js валидирует параметры входа.

Таким образом, связка JWT токена и библиотеки Auth.js позволяет реализовать надёжный и гибкий механизм авторизации на клиентской стороне с минимальной утечкой чувствительных данных.

## 1.10 Система создания заданий

В данном разделе описаны основные функциональные возможности анализа текста программы на основе искусственного интеллекта для преподавателей и студентов.

### 1.10.1 Общее описание модуля

Разработанная система позволяет преподавателям создавать виртуальные классы, выдавать задания и автоматически анализировать решения студентов с использованием инструментов искусственного интеллекта. Это аналог образовательной платформы (например, Google Classroom), ориентированный на технические дисциплины с программированием.

Основные функции:

- 1) Создание виртуального класса преподавателем;
- 2) Назначение заданий с параметрами оценки;

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		27



- 3) Загрузка решений студентами;
- 4) Получение отчётов об автоматическом анализе текста программы на основе искусственного интеллекта;
- 5) Просмотр статистики и аналитики преподавателем.

### **1.10.2 Функциональные возможности**

Для преподавателей представлены следующие функциональные возможности:

- 1) Создание и управление классами;
- 2) Назначение заданий;
- 3) Просмотр отчётов искусственного интеллекта по каждому студенту;
- 4) Сводная статистика по группе.

У студентов возможностей меньше, а именно:

- 1) Просмотр активных заданий и крайних сроков;
- 2) Загрузка решения.

### **1.10.3 Автоматический анализ текста программы**

После загрузки решения студентом система автоматически выполняет его анализ с использованием инструментов искусственного интеллекта. Проверка охватывает корректность, читаемость, соответствие заданию и уровень оригинальности текста программы.

На основе заданных преподавателем критериев формируется интерактивный отчёт, включающий:

- 1) Комментарии и замечания по структуре и стилю текста программы;
- 2) Оценку соответствия решению поставленным требованиям.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		28

## 1.11 Анализ текста программ на основе ИИ

### 1.11.1 Общее описание модуля DeepSeek

DeepSeek — это облачная платформа для глубинного анализа текста программ, основанная на передовых архитектурах трансформеров и нейронных сетей [19]. Ее учебный модуль автоматизирует ручную проверку студенческих работ, минимизирует субъективность оценки и предоставляет преподавателям детальную, содержательную обратную связь.

- 1) Выявлять самые разнообразные синтаксические и логические ошибки, обнаруживать неточности в реализации алгоритмов;
- 2) Оценивать соответствие структурных блоков текста программы требованиям конкретного задания, обращая внимание на правильность использования функций и корректность их связи;
- 3) Анализировать степень оригинальности решения с помощью семантического сравнения embedding-представлений, что позволяет не только обнаружить плагиат, но и оценить творческий подход студента.

Доступ к вызовам модуля DeepSeek строго ограничен: напрямую вызвать проверку текста программы искусственным интеллектом нельзя — проверка доступна только при отправке решения студентом.

### 1.11.2 Функциональные возможности модуля DeepSeek

Общий функционал разделён на две части: для преподавателей и для студентов. Для преподавателей:

- 1) Автоматически сгенерированные отчёты, где каждая найденная проблема снабжена подробным описанием и примером исправления;
- 2) Механизм гибкой настройки критериев оценки — преподаватель может добавлять свои правила проверки в зависимости от характера задания.

Студент, в свою очередь, имеет следующие возможности:

- 1) Заявка на анализ через загрузку решения;
- 2) Получение готового отчёта от преподавателя без прямого доступа к сервису.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		29

### 1.11.3 Принцип работы модуля DeepSeek

- 1) Во время лексического и синтаксического разбора текст программы разбивается на токены, строится абстрактное синтаксическое дерево, на основе которого проводится первичный анализ;
- 2) Алгоритмы трансформеров сопоставляют логику решения с огромной базой примеров, выявляя нетривиальные отклонения от оптимальной структуры;
- 3) Рассчитываются показатели цикломатической сложности, глубины вложенности, соответствия стандартам написания текста программы и других параметров качества;
- 4) Сравнение векторов признаков загруженного решения с репозиторием эталонных и ранее проверенных работ для определения степени оригинальности;
- 5) На выходе генерируется детальный JSON-документ, содержащий список найденных замечаний, метрик и чётких рекомендаций по улучшению.

### 1.12 Тестирование с использованием библиотеки Jest

Jest — современная библиотека для автоматизированного тестирования приложений, разработанный и поддерживаемый компанией Facebook (см. документацию [20]). Он идеально подходит для оценки бизнес-логики компонентов благодаря следующим преимуществам:

- 1) Работает «из коробки» без дополнительной конфигурации, что позволяет быстро приступить к написанию тестов;
- 2) Параллельное выполнение тестов в изолированных средах сокращает время прогона и обеспечивает мгновенную обратную связь;
- 3) Встроенный сбор метрик покрытия текста программ помогает выявлять неохваченные участки и поддерживать высокий уровень качества;
- 4) Лёгкая подмена модулей и функций с помощью утилит упрощает эмуляцию сложных сценариев и зависимостей;
- 5) Возможность сохранять «снимки» выходных данных функций или компонентов и автоматически отслеживать их изменения с течением времени.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		30

Именно эти особенности делают библиотеку Jest лучшим выбором для тестирования бизнес-логики: он упрощает разработку и сопровождение тестов, обеспечивает понятную диагностику ошибок и легко интегрируется в клиентскую архитектуру без лишних накладных расходов.

### 1.13 Выводы

В результате проведённого анализа можно сделать вывод о наличии устойчивого запроса на интегрированное образовательное приложение, способное решать сразу несколько ключевых задач. Современные платформы зачастую фокусируются либо на предоставлении учебных материалов, либо на коммуникации, либо на автоматизации проверки знаний, при этом разрозненность этих функций создаёт неудобства для всех участников образовательного процесса.

Потребности преподавателей включают в себя удобное управление группами и заданиями, возможность оперативной обратной связи, загрузку и распространение материалов. Студентам, в свою очередь, важно иметь стабильный и понятный доступ к заданиям, личным сообщениям и учебным ресурсам, а также возможность взаимодействовать с преподавателями и одноклассниками в привычном цифровом формате.

Предлагаемое приложение должно закрыть этот разрыв, обеспечив единую среду, в которой объединены функции управления учебным процессом, общения, публикации и проверки заданий. Такой подход позволит повысить качество образовательного взаимодействия, сократить технические барьеры и обеспечить более высокую степень вовлечённости пользователей.

Таким образом, на основании проведённого анализа подтверждается необходимость разработки новой системы, в которой ключевые элементы образовательной среды будут интегрированы в одно приложение, удовлетворяющее современным требованиям пользователей.

					ОБЗОР И ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		31

## 2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА

### 2.1 Архитектура клиентской части системы

Клиентская часть разрабатываемой системы реализована в виде одностороннего приложения (SPA) с использованием фреймворка Next.js, поддерживающего гибкий рендеринг — как на стороне клиента (CSR), так и на стороне сервера (SSR). Такое решение позволяет обеспечить как высокую производительность и отзывчивость пользовательского интерфейса, так и оптимизацию индексации содержимого поисковыми системами за счёт серверного рендеринга.

#### 2.1.1 Общая структура архитектуры

Для организации структуры проекта был применён подход Feature-Sliced Design (FSD) [21] — современная парадигма проектирования клиентской части приложений, ориентированная на модульность, масштабируемость и соответствие предметной области. В отличие от традиционных архитектур, основанных на технических слоях (например, разделение на компоненты, страницы или сервисы), архитектура FSD предполагает смысловое разделение приложения на функциональные модули, которые отражают реальные пользовательские сценарии и бизнес-логику.

Каждый модуль в архитектуре FSD охватывает чётко определённую часть функциональности приложения и включает в себя всё необходимое для её реализации: пользовательский интерфейс, бизнес-логику, обмен данными с сервером и внутренние модели. Такой подход улучшает читаемость текста программы, облегчает его повторное использование и делает сопровождение и развитие проекта более удобным и предсказуемым в будущем.

#### 2.1.2 Архитектурная диаграмма

На диаграмме представлены основные уровни и элементы архитектуры приложения. Следует отметить, что каждый слой в данной структуре ориентирован на строгое разграничение ответственности. Компоненты нижних уров-

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чаттами и AI-анализом кода лабораторных работ	Лит.	Лист	Листов
Руковод.	Мельников А.Б.						32	94
Консул.						БГТУ им. В.Г. Шухова, ПВ-212		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							

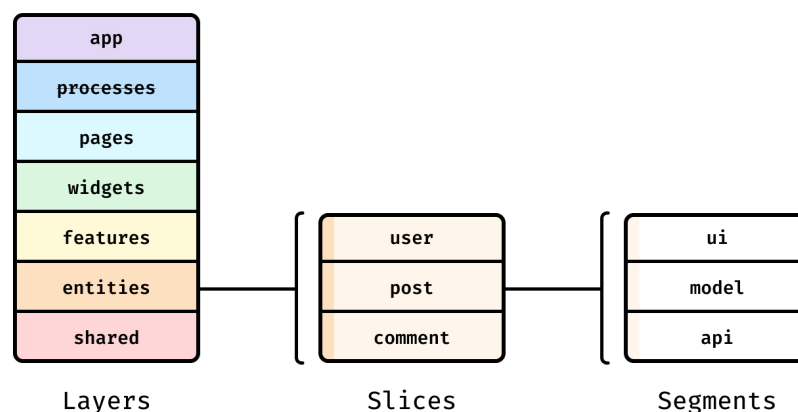


Рисунок 2.1 – Схема архитектуры клиентской части

ней не имеют информации о вышестоящих слоях, что позволяет реализовать принцип инверсии зависимостей и минимизировать связанность между модулями.

### 2.1.3 Слои архитектуры Feature-Sliced Design

В таблице 2.1 представлены слои архитектуры Feature-Sliced Design, сгруппированные по уровню абстракции и ответственности.

### 2.1.4 Концепция срезов

Ключевым элементом архитектурного подхода Feature-Sliced Design является понятие срезов (англ. *slices*). Под срезом понимается логически обособленный модуль, реализующий завершённую часть функциональности приложения. Каждый срез может содержать собственные модели данных, визуальные компоненты, бизнес-логику, а также механизмы взаимодействия с внешними источниками данных.

Ниже приведены примеры различных типов срезов и их ответственности в структуре проекта:

- 1) *features/login* — срез, реализующий сценарий авторизации пользователя;
- 2) *entities/task* — срез, содержащий всё, что связано с сущностью «задание»;
- 3) *widgets/ChatWindow* — срез, объединяющий функциональность и интерфейс чат-интерфейса;
- 4) *pages/home* — срез, реализующий главную страницу приложения.

Таблица 2.1 – Слои архитектуры Feature-Sliced Design

Слой	Описание и назначение
<i>app</i>	Точка входа в приложение: глобальные стили, маршрутизация, провайдеры состояния, интеграции с внешними сервисами.
<i>pages</i>	Страницы, связанные с маршрутизацией. Формируются из виджетов и не содержат бизнес-логики.
<i>widgets</i>	Крупные элементы интерфейса, отражающие пользовательские сценарии, например, чат, список заданий, панель управления.
<i>features</i>	Изолированные пользовательские функции, такие как авторизация, отправка сообщений или регистрация. Могут включать бизнес-логику.
<i>entities</i>	Базовые предметные сущности предметной области, включающие типы, схемы, API-функции и UI-представление.
<i>shared</i>	Универсальные компоненты, утилиты и типы, переиспользуемые во всём проекте.

### 2.1.5 Горизонтальное деление на сегменты

Каждый срез, независимо от своего уровня, может быть дополнительно разделён на сегменты (англ. *segments*) — логические подкатегории, структурирующие содержимое среза по назначению модуля. В отличие от слоёв, которые представляют вертикальную иерархию, сегменты формируют горизонтальное деление и обеспечивают внутреннюю организацию модулей.

Наиболее распространённые типы сегментов включают:

- 1) *ui* — визуальные компоненты и стили, определяющие отображение данных;
- 2) *model* — модели данных, хранилища состояния, типизация и бизнес-логика;
- 3) *api* — функции для работы с внешними сервисами, включая описание типов запросов и маппинг ответов;

- 4) *lib* — вспомогательные функции и библиотеки, используемые в пределах данного среза;
- 5) *config* — конфигурационные файлы и переключатели функциональности.

### **Преимущества выбранного подхода**

Применение архитектуры Feature-Sliced Design в контексте разрабатываемого клиентского приложения позволило достичь следующих результатов:

- 1) Чёткое разграничение обязанностей между модулями и слоями;
- 2) Улучшенная масштабируемость проекта без деградации структуры;
- 3) Повышенная модульность, обеспечивающая лёгкость в тестировании и повторном использовании текста программы;
- 4) Создание условий для быстрой и эффективной интеграции новых членов команды в разработку;
- 5) Архитектура, ориентированная на задачи и бизнес-логику, а не на технические детали.

В совокупности данные свойства делают архитектурное решение устойчивым к росту функциональности, улучшая поддержку и развитие системы в долгосрочной перспективе.

## **2.2 Проектирование интерфейсных подсистем и экранов**

Одной из ключевых задач при проектировании клиентской части является логическое и функциональное разделение интерфейса на подсистемы, каждая из которых реализует отдельный аспект пользовательского взаимодействия. Такое разделение позволяет обеспечить модульность, переиспользуемость компонентов и устойчивость к изменениям.

Проект разрабатывается с использованием архитектуры Feature-Sliced Design, что накладывает дополнительные требования к организации экранов и компонентов. Все подсистемы формируются из слоёв *entities*, *features*, *widgets* и собираются в слое *pages*, а общая инфраструктура размещается в слое *shared*.

### **2.2.1 Выделение ключевых интерфейсных подсистем**

Клиентская часть разработанной платформы организована в виде набора функционально обособленных интерфейсных подсистем, каждая из которых

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		35



отвечает за определённый аспект пользовательского взаимодействия и бизнес-логики. Такое разграничение позволяет повысить масштабируемость и сопровождаемость системы, а также упростить процесс тестирования и внедрения новых функций.

Подсистема авторизации и регистрации отвечает за обеспечение безопасного входа в систему, регистрацию новых пользователей и управление сессиями. Аутентификация реализована с применением библиотеки Auth.js и технологии JWT, что позволяет надёжно разграничивать доступ к различным разделам интерфейса в зависимости от роли пользователя.

Регистрация в системе представлена в виде трёх пользовательских сценариев, адаптированных под особенности образовательного процесса:

- 1) Первый сценарий реализован для новых организаций (институтов) и сопровождается созданием административной учётной записи. На этом этапе формируется корневая структура управления учреждением.
- 2) Второй и третий сценарии предназначены для регистрации преподавателей и студентов соответственно. Оба сценария доступны исключительно по индивидуальным приглашениям, что обеспечивает контроль над составом участников образовательного процесса и предотвращает несанкционированный доступ.

Подсистема тесно связана с механизмами контроля прав доступа и маршрутизации, определяя поведение интерфейса в зависимости от текущего статуса пользователя.

Подсистема управления университетом реализует административную логику, связанную с конфигурацией организационной структуры образовательного учреждения (структура компонентов показана на рисунке А.4).

Основными функциями данной подсистемы являются:

- 1) Создание и удаление структурных единиц — институтов, кафедр, учебных групп;
- 2) Управление персоналом: добавление и блокировка преподавателей и студентов;
- 3) Генерация приглашений для входа новых участников на платформу с конкретной ролью;

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		36

#### 4) Отображение данных по структуре учреждения.

Визуально подсистема представлена в виде панели управления с множеством таблиц, форм и интерактивных элементов, обеспечивающих быстрый доступ к ключевым административным операциям. Все действия защищены авторизацией и доступны только пользователям с соответствующими правами доступа.

Подсистема работы с заданиями и отправкой решений предназначена для организации учебной деятельности (структура компонентов представлена на рисунке А.5).

Основной интерфейс включает:

- 1) Панель создания и редактирования заданий с параметрами проверки;
- 2) Представление активных и завершённых заданий для студентов;
- 3) Историю отправок с отображением результатов и статуса проверки.

Задания связаны с группами. Система также предоставляет базовую аналитику по результатам выполнения.

Подсистема обмена сообщениями обеспечивает коммуникацию между участниками образовательного процесса (см. рисунок 2.2)

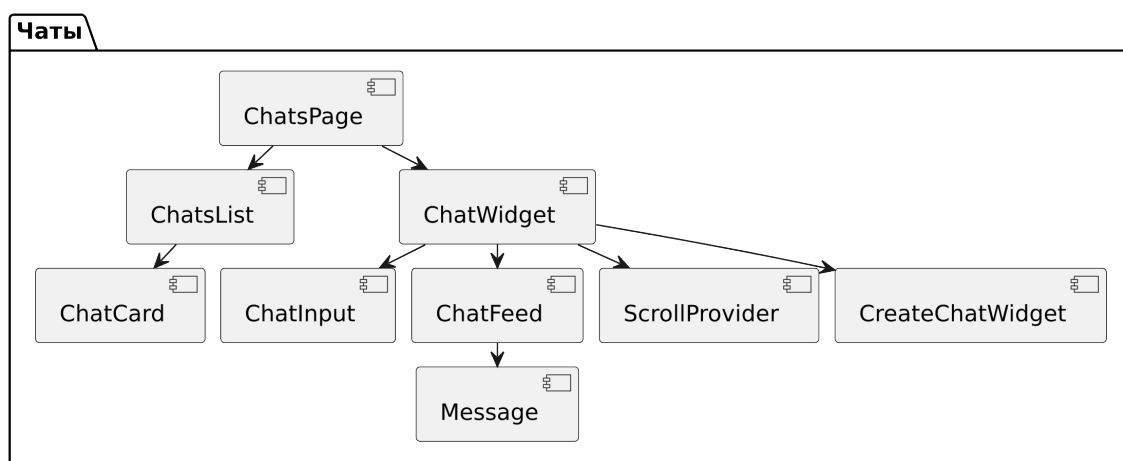


Рисунок 2.2 – Диаграмма компонентов системы чатов

Технически реализация основана на технологии WebSocket с использованием библиотеки *Socket.IO*, что обеспечивает мгновенную доставку сообщений и минимальную задержку при передаче данных.

Основной функционал включает:

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		37

- 1) Подключение к соответствующим «комнатам» (группам или диалогам);
- 2) Отправку и приём текстовых сообщений;
- 3) Отображение истории переписки;
- 4) Поддержку вложений и индикаторов прочтения.

Доступ к системе чатов осуществляется только после успешной авторизации, что исключает участие анонимных пользователей и обеспечивает безопасность переписки.

Подсистема анализа решений на основе искусственного интеллекта реализует автоматическую проверку студенческих заданий с использованием ИИ, что является одной из ключевых особенностей платформы. Подсистема предназначена для получения и визуализации результатов анализа на основе искусственного интеллекта, включая оценку корректности текста программ, проверку на соответствие заданию, а также вывод комментариев, рекомендаций и текстовых пояснений. Результаты анализа отображаются в виде отчёта с возможностью преподавателя оставить дополнительные замечания. Таким образом, снижается нагрузка на преподавателя и повышается объективность оценивания.

Каждая из указанных подсистем обладает чётко определёнными входными и выходными данными, а также взаимодействует с другими модулями системы. Например, подсистема работы с заданиями напрямую связана с анализом на основе искусственного интеллекта, а система чатов — с механизмами авторизации и маршрутизации. Такое проектирование обеспечивает гибкость, надёжность и чёткую масштабируемость клиентской архитектуры.

### 2.2.2 Страницы и их структура

Разработка интерфейсной части веб-приложения требует не только реализации функциональных компонентов, но и проектирования логически связанных экранов, отражающих ключевые сценарии взаимодействия пользователя с системой. В рамках платформы каждая страница представляет собой самостоятельный интерфейсный модуль, обслуживающий одну или несколько бизнес-задач, соответствующих определённой роли: студент, преподаватель, администратор.

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		38

Процесс формирования страниц реализован с применением маршрутизации, встроенной в фреймворк Next.js, что обеспечивает высокую производительность и поддержку серверного рендеринга. Страницы не только представляют визуальный уровень приложения, но и координируют работу между компонентами пользовательского интерфейса, бизнес-логикой и хранилищем состояния.

Архитектурно страницы собираются из обособленных функциональных элементов, разработанных согласно принципам архитектуры FSD: пользовательские действия реализуются в слое *features*, отображаемые сущности формируются на базе слоя *entities*, а объединение этих блоков происходит внутри слоя *widgets*. Такой подход позволяет повысить согласованность, переиспользуемость и модульность текст программ, а также снижает зависимость между различными частями интерфейса.

Далее описаны ключевые страницы, отражающие основную логику пользовательского взаимодействия.

Страница авторизации отвечает за вход пользователя в систему. Содержит форму для ввода учётных данных, а также реализует логику валидации, передачи данных на сервер, обработки ошибок и сохранения сессионного токена. После успешной авторизации пользователь перенаправляется на главную страницу, соответствующую его роли.

Страница заданий представляет собой ключевой интерфейс для организации и выполнения учебной деятельности. Интерфейс страницы включает:

- 1) Список классов и учебных групп, к которым привязан пользователь;
- 2) Перечень активных заданий в рамках каждой группы;
- 3) Доступ к подробному описанию заданий, срокам сдачи и параметрам оценивания;
- 4) Отправку решений и просмотр результатов, включая отчёты анализа на основе искусственного интеллекта.

Для преподавателя дополнительно предоставляется интерфейс управления заданиями, а также доступа к аналитике по группам и студентам.

Административная панель института является основным рабочим инструментом пользователя с ролью администратора. Интерфейс включает:

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		39

- 1) Управление иерархией образовательного учреждения (институты, кафедры, группы);
- 2) Назначение и блокировка пользователей (студентов и преподавателей);
- 3) Просмотр структуры учреждения в табличной форме;
- 4) Генерацию и отправку приглашений на регистрацию;
- 5) Журнал событий и контроль активности пользователей.

Все действия на данной странице требуют повышенного уровня доступа и сопровождаются системой уведомлений о результатах операций.

Страница чатов реализует коммуникационную составляющую платформы. Пользователь получает доступ к:

- 1) Перечню активных диалогов (личных и групповых);
- 2) Истории сообщений в рамках выбранного чата;
- 3) Форме для отправки сообщений и файлов;
- 4) Интерактивным элементам: индикаторы доставки, статус прочтения, поиск по переписке.

Для преподавателей также предусмотрена возможность создания новых групповых чатов для своих учебных групп.

Все функциональные страницы приложения, за исключением экранов регистрации и входа, используют единый шаблон компоновки, обеспечивающий целостность визуального восприятия и унификацию пользовательского опыта. Данный шаблон включает в себя общие элементы интерфейса — верхнюю панель навигации, боковое меню и основной контейнер для отображения содержимого, который динамически наполняется в зависимости от текущего маршрута.

Использование общего каркаса позволяет сохранить структурную согласованность между различными разделами системы, облегчает адаптацию пользователей к интерфейсу и упрощает внедрение изменений. Кроме того, архитектурное разделение логики и представления на уровне страниц способствует инкапсуляции ответственности, а также повышает читаемость и сопровождаемость текста программ. В рамках маршрутизации обеспечивается цен-

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		40

трализованное управление доступом, фильтрацией и визуализацией данных с учётом ролей пользователей.

Таким образом, структура страниц приложения отражает как технические требования архитектуры, так и практическую ориентацию на удобство и эффективность работы конечных пользователей.

### 2.2.3 Компоненты и принципы их структурирования

Компонентная модель проекта выстроена на основе принципов повторного использования, инкапсуляции и чёткого разделения ответственности между уровнями абстракции. Все компоненты, применяемые в рамках клиентского интерфейса, условно делятся на два основных класса: общие (универсальные) и специфические (бизнес-ориентированные).

- 1) Общие компоненты представляют собой переиспользуемые элементы пользовательского интерфейса, не зависящие от предметной области. К ним относятся кнопки, поля ввода, модальные окна, индикаторы загрузки, элементы навигации, уведомления и другие базовые визуальные элементы. Такие компоненты широко применяются на всех уровнях интерфейса и не содержат бизнес-логики.
- 2) Специфические компоненты, разрабатываемые в слоях *entities* и *widgets*, предназначены для реализации прикладной логики и отображения конкретных сущностей системы. Примерами являются компоненты отображения сообщений в чате, карточек заданий, панели управления преподавателя, таблиц пользователей и др. Они обладают внутренним состоянием и часто включают обращение к хранилищу.

Такое структурное разграничение существенно упрощает масштабирование проекта, облегчает поддержку и повторное использование элементов, а также способствует разделению труда между разработчиками.

### 2.2.4 Распределение логики по слоям архитектуры

Функциональная логика клиентской части системы строго распределяется по слоям архитектуры Feature-Sliced Design, что обеспечивает высокую модульность и инкапсуляцию поведения. Каждому слою соответствует свой уровень ответственности:

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		41

- 1) В слое *entities* сосредоточена модель предметной области: типизация, структура сущностей, атомарные компоненты отображения, такие как *Registration*, *Department*, *Group*. Данный слой реализует описание и базовое представление данных без привязки к конкретным действиям пользователя.
- 2) Слой *features* содержит реализацию отдельных действий, составляющих пользовательские сценарии: отправка сообщений, регистрация, загрузка задания, подтверждение действия и т.д. Эти модули инкапсулируют конкретные шаги взаимодействия пользователя с интерфейсом, часто включая локальное состояние. Функции из слоя *features* могут быть использованы многократно и комбинироваться для построения более сложных сценариев.
- 3) Слой *widgets* представляет собой реализацию полноценных пользовательских сценариев — законченных интерфейсных блоков, решающих определённую задачу. Примеры: интерфейс чата, панель с заданиями, административный модуль управления группами. Каждый виджет объединяет несколько фич и сущностей, обеспечивая завершённую и логически связанную единицу поведения.
- 4) Слой *pages* выполняет роль точки входа и финальной сборки пользовательских сценариев. Здесь происходит выбор и компоновка виджетов в зависимости от маршрута, роли пользователя и контекста сессии. Кроме того, на уровне страниц задаются глобальные обёртки, обеспечиваются ограничения доступа, инициализируются загрузки данных и подключаются необходимые провайдеры. Таким образом, слой *pages* является связующим слоем между навигацией и пользовательским опытом.

Такое строгое распределение обязанностей по слоям позволяет исключить дублирование логики, минимизировать связанность между модулями и обеспечить чёткую иерархию ответственности.

### 2.2.5 Пользовательские сценарии

Для повышения удобства и доступности платформы, особенно в условиях использования её разными категориями пользователей, были реализованы

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		42

следующие решения в области пользовательского опыта:

- 1) Централизованная навигация — через универсальный макет, включающий боковую и верхнюю панели, интерфейс остаётся единообразным и интуитивно понятным вне зависимости от текущего маршрута;
- 2) Toast-уведомления — реализация мгновенной обратной связи при выполнении действий: успешная отправка формы, ошибка сети, получение новых сообщений;
- 3) Обработка пустых состояний и ошибок — предусмотрены интерфейсы для ситуаций отсутствия данных, ошибок загрузки или недоступности сервера.

В результате, пользователь получает предсказуемый и непрерывный опыт взаимодействия с системой вне зависимости от своей роли и уровня подготовки.

### **Вывод**

Проектирование интерфейсной части приложения основывается на чётком структурном и функциональном разграничении компонентов, ориентированном на принципы модульности и масштабируемости. Использование архитектуры Feature-Sliced Design позволяет изолировать бизнес-логику, визуальные компоненты и маршрутизацию, что делает интерфейс легко расширяемым и сопровождаемым.

Реализованная организация интерфейса, объединяющая единый шаблон компоновки, повторно используемые компоненты и специфические бизнес-модули, способствует формированию целостного пользовательского опыта. Выбранные решения обеспечивают удобство и логичность навигации, а также высокую отзывчивость системы при взаимодействии с пользователем.

Интерфейсная часть проекта построена на модульной архитектуре, основанной на бизнес-функциях. Подсистемы выделены логически, а их реализация изолирована в независимые модули, что повышает удобство поддержки, расширения и переиспользования компонентов.

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		43



## 2.3 Проектирование взаимодействия с сервером

Клиентская часть приложения активно взаимодействует с сервером для получения и отправки данных, а также поддерживает постоянное соединение с помощью WebSocket в рамках подсистемы обмена сообщениями. При проектировании механизма взаимодействия были учтены требования безопасности, стабильности соединения, обработки ошибок, а также необходимость автоматического обновления сессионных данных пользователя.

### 2.3.1 Аутентификация и управление токенами

Для обеспечения защищённого доступа к функциональности платформы используется система авторизации с применением JSON Web Token (JWT). Управление сессией пользователя реализовано через библиотеку *Auth.js*, которая выполняет роль промежуточного слоя между клиентом и системой хранения токенов.

- 1) Пользователь выполняет вход в систему с помощью логина и пароля. Библиотека *Auth.js* инициирует процесс аутентификации и получает JWT токен при успешной проверке данных.
- 2) Полученный токен JWT содержит минимально необходимые данные — такие как идентификатор пользователя, его роль и срок действия. Этот токен сохраняется в cookie с флагами HTTP-only, Secure и SameSite=Strict, что предотвращает атаки типа XSS и CSRF.
- 3) При обращении клиента к защищённым ресурсам токен автоматически прикрепляется к HTTP-запросу. В случае, если токен недействителен или истёк, библиотека *Auth.js* выполняет его обновление с использованием токена обновления (refresh-токена), если он доступен.

### 2.3.2 Унифицированная функция отправки запросов

Для унификации сетевых запросов была реализована функция *sendRequest*, которая обеспечивает:

- 1) преобразование данных в JSON формат через функцию *JSON.stringify*;
- 2) добавление заголовков, включая *Authorization: Bearer*;

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		44

- 3) обработка ошибок и повторная попытка после обновления токена;
- 4) поддержка различных HTTP-методов.

Эта функция служит единой точкой для выполнения HTTP-запросов, упрощая их отправку и обработку.

### 2.3.3 Взаимодействие через протокол WebSocket

Для реализации обмена сообщениями в реальном времени используется библиотека *Socket.IO*, обеспечивающая:

- 1) автоматическое переподключение при обрыве соединения;
- 2) передачу структурированных событий с именами и аргументами;
- 3) интеграцию с промежуточным программным обеспечением для авторизации;
- 4) работу с пространствами имён и комнатами;
- 5) резервные транспорты при недоступности протокола WebSocket.

На стороне клиента реализован хук *useSocket*, который:

- 1) инициализирует соединение с сервером;
- 2) отправляет и получает события с типизированными данными;
- 3) подписывается и отписывается от каналов;
- 4) управляет жизненным циклом подключения и логирует события;
- 5) обрабатывает ошибки соединения.

При установлении WebSocket-соединения клиент передаёт токен доступа в параметрах, сервер проверяет его и активирует соединение. В случае истечения срока действия токена:

- 1) инициируется его обновление;
- 2) текущее соединение закрывается;
- 3) создаётся новое соединение с обновлённым токеном.

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		45

## Вывод

Реализованные механизмы автоматического обновления токенов, единая функция отправки запросов и продуманная интеграция WebSocket протокола через библиотеку *Socket.IO* обеспечивают безопасность, надёжность и масштабируемость взаимодействия клиентской части с сервером в режиме реального времени.

## 2.4 Интеграция модуля DeepSeek в архитектуру проекта

Для обеспечения эффективной работы ИИ-компонента в образовательной платформе реализована модульная архитектура с чётким разделением ответственности. Последующие подразделы детализируют ключевые аспекты интеграции: стратегию контейнеризации для изоляции сервиса, механизмы взаимодействия с клиентской частью и системные преимущества выбранного подхода. Основное внимание уделено сохранению прозрачности работы ИИ для конечных пользователей при обеспечении гибкости разработки и эксплуатации.

### 2.4.1 Контейнеризация модуля DeepSeek

Для обеспечения независимого жизненного цикла и лёгкой масштабируемости модуль DeepSeek развёртывается в виде изолированного Docker-контейнера. Такой подход позволяет:

- 1) Быстро запускать и останавливать сервис без влияния на основное приложение;
- 2) Поддерживать разные версии модулей DeepSeek параллельно, экспериментируя с обновлениями моделей;
- 3) Мигрировать между хостами и облачными средами с минимальными изменениями конфигурации.

### 2.4.2 Преимущества контейнеризированного подхода

Контейнеризация модуля DeepSeek даёт следующие ключевые плюсы:

- 1) Анализ текста программы выполняется в отдельном окружении, не влияя на отзывчивость интерфейса;

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		46

- 2) При большом числе запросов можно запускать несколько экземпляров контейнера;
- 3) Обновление ИИ-компонента сводится к выпуску нового образа без правок в клиентской части;
- 4) Контейнеры можно запускать локально и в облаке с одинаковой конфигурацией.

### 2.4.3 Взаимодействие клиентской части с модулем DeepSeek

Клиентская часть приложения, реализованная на библиотеки React и фреймворке Next.js, отправляет HTTP-запросы к серверной части веб-приложения при прикреплении решения задания студентом, далее автоматический начинается проверка решения при помощи анализ на основе искусственного интеллекта. Результат проверки приходит пользователю при запросе на получения детальной информации по задаче, выполненной студентом.

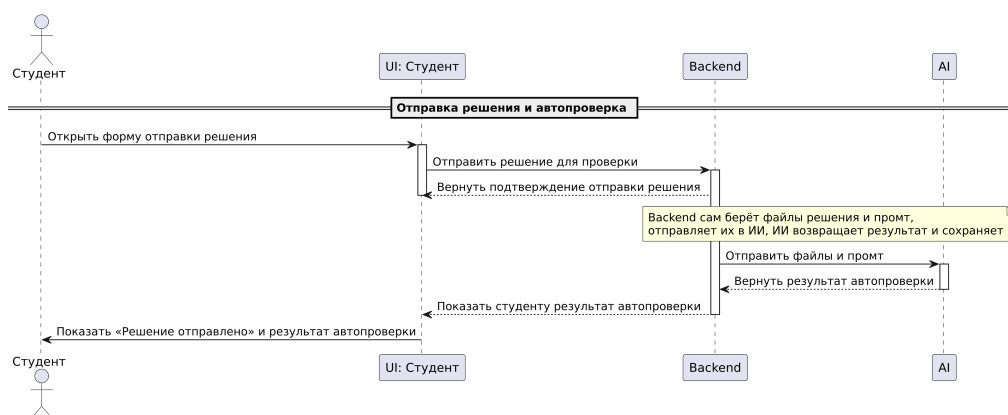


Рисунок 2.3 – Схема взаимодействия клиентской части с модулем DeepSeek

На рисунке 2.3 представлена схема взаимодействия клиентской части с модулем DeepSeek.

### 2.4.4 Вывод

Контейнеризация модуля DeepSeek обеспечивает полную независимость остальных компонентов приложения от ИИ-модуля, позволяя развёртывать и обновлять его без влияния на другие сервисы. Взаимодействие сервера создаёт своего рода «чёрный ящик» для клиента, что упрощает инкапсуляцию логики и даёт гибкость в распределении и масштабировании нагрузки на контейнер с ИИ.

## 2.5 Обеспечение безопасности клиентской части

Безопасность пользовательского взаимодействия является важнейшей составляющей архитектуры клиентской части платформы. В условиях, когда доступ к различным модулям приложения осуществляется на основе ролей, а взаимодействие с данными сопровождается отображением пользовательского контента, особое внимание уделяется как управлению доступом, так и защите от потенциальных атак, включая межсайтовое выполнение скриптов (XSS). В данной подсистеме реализован комплекс механизмов, направленных на защиту данных и поведения интерфейса со стороны клиента.

### 2.5.1 Разграничения доступа

Одним из ключевых компонентов обеспечения безопасности клиентской части является система контроля доступа на основе промежуточного слоя — промежуточное программное обеспечение [22]. В рамках архитектуры фреймворка Next.js, промежуточное программное обеспечение представляет собой функцию, исполняемую при каждом запросе к защищённым маршрутам. Она позволяет перехватывать обращения к страницам до их рендеринга и на этой стадии выполнять необходимые проверки: наличие токена, его валидность, а также права пользователя.

В контексте реализуемой платформы при обращении пользователя к любой защищённой странице клиентская логика через промежуточное программное обеспечение извлекает JWT-токен из cookies и дешифрует его содержимое, получая полезную нагрузку — уникальный идентификатор, срок действия сессии и роль в системе (*admin, teacher, student*).

На основе этой информации промежуточное программное обеспечение выполняет следующие действия:

- 1) Если пользователь не авторизован (отсутствует валидный токен) — происходит автоматический редирект на страницу входа;
- 2) Если пользователь авторизован, но не обладает достаточными правами — осуществляется перенаправление на главную страницу или отображается сообщение об отказе в доступе;
- 3) Если пользователь обладает необходимой ролью — доступ к ресурсу

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		48

предоставляется, и страница загружается с соответствующим контентом.

Таким образом, промежуточное программное обеспечение дает надёжную фильтрацию обращений к различным частям интерфейса, предотвращая несанкционированный доступ и соблюдая политику разграничения прав.

### 2.5.2 Роль и защита при работе с форматируемым текстом

Дополнительным вектором потенциальной угрозы в клиентских приложениях является отображение форматируемого текста, особенно если пользователь имеет возможность редактировать его содержимое. В таких случаях возрастает риск внедрения вредоносных скриптов, замаскированных под обычный HTML.

Для решения данной задачи в проекте используется библиотека *tiptap* — расширяемый редактор форматированного текста на основе редакторского фреймворка *ProseMirror*. Одним из ключевых преимуществ редактора *tiptap* является контроль над тем, какие HTML-теги и атрибуты допускаются к отображению. Таким образом, даже если пользователь попытается вставить опасный текст программы, редактор удалит такие элементы на этапе парсинга.

Технически это реализуется следующим образом:

- 1) При вводе содержимого редактор не сохраняет «сырые» HTML-строки, а формирует безопасное представление согласно заданным схемам;
- 2) При рендеринге текста из базы или состояния редактор отображает только те элементы, которые были описаны как допустимые;
- 3) Расширения (extensions), добавляемые к редактору *tiptap*, позволяют точно контролировать список разрешённых тегов и атрибутов.

Таким образом, даже при наличии активной формы редактирования форматируемого текста пользовательская среда остаётся защищённой от внедрения опасного контента.

### 2.5.3 Вывод

Комплекс реализованных решений позволяет эффективно защитить клиентскую часть приложения как от внешнего вмешательства, так и от ошибочного доступа пользователей. Промежуточное программное обеспечение производит проверку сессии и прав доступа до загрузки страниц, а редактор *tiptap*

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		49

гарантирует безопасность при работе с форматируемым текстом. Такое сочетание архитектурных и прикладных средств создаёт устойчивую и безопасную пользовательскую среду.

## 2.6 Покрытие бизнес-логики юнит-тестами

Наш подход к обеспечению надёжности клиентского приложения фокусируется на обязательном юнит-тестировании бизнес-логики при помощи библиотеки Jest. Тестирование UI-компонентов считается вторичным: написание и поддержка сравнений HTML-вывода часто оказывается более трудоёмким и хрупким, чем простая визуальная валидация. Визуальный осмотр интерфейса преподавателем или дизайнером даёт более быстрый и надёжный результат без лишних накладных расходов.

Основные принципы нашего подхода:

- 1) Юнит-тесты покрывают функции, отвечающие за валидацию данных, расчёт оценок и другие критичные механизмы, гарантируя корректность работы независимо от изменений UI;
- 2) Модульные тесты интерфейсов не используются: динамика верстки и частые мелкие правки приводят к избыточным провалам тестов и дополнительным усилиям на их поддержку;
- 3) Благодаря отказу от snapshot-тестирования HTML-структура текста программы остаётся гибкой, а команда освобождает время на развитие функциональности вместо постоянной правки тестов;
- 4) Автоматический запуск тестов бизнес-логики при каждом пуше позволяет мгновенно обнаруживать регрессии и поддерживать стабильность продукта;
- 5) Для окончательной валидации интерфейса используется ручной осмотр ключевых страниц после сборки, что даёт уверенность в корректности отображения без сложных технических средств.

Такой подход обеспечивает надёжность самой логики приложения и упрощает работу с UI: вместо громоздких автоматизированных тестов на вёрстку мы применяем человеческую экспертизу для финальной проверки внешнего вида и пользовательского опыта.

					ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОЕКТА	Лист
Изм.	Лист	№ докум.	Подп.	Дата		50

## 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

### 3.1 Архитектура по Feature-Sliced Design

Feature-Sliced Design (FSD) — это гибкий набор рекомендаций и подходов по логической организации клиентской части программы, основанных на выделении независимых функциональных слоёв и зон ответственности. В отличие от строгих стандартов, архитектура FSD предоставляет разработчикам свободу выбора конкретных решений, сохраняя при этом единый общий каркас структуры. Такая архитектура повышает читаемость, масштабируемость и тестируемость приложения, а также упрощает командную разработку и поддержку приложения.

#### 3.1.1 Слой *shared*

Слой *shared* служит хранилищем нижнего уровня для общих и переиспользуемых компонентов, утилит и ресурсов, не зависящих от конкретной бизнес-логики и состоит из следующих частей:

- 1) UI-компоненты общего назначения включают простые React-компоненты (кнопки, лоадеры, таблицы, модальные окна), не содержащие бизнес-логику и используемые в различных контекстах.
- 2) Провайдеры контекста, такие как *DragAndDropFilesProvider* и *EnterKeyHandlerProvider*, обеспечивают единообразную работу с событиями и состояниями по всему приложению.
- 3) Утилиты и хелперы включают функции для форматирования дат и чисел, генерации уникальных идентификаторов, работы с *localStorage* хранилищем и другие вспомогательные средства.
- 4) Кастомные хуки общего назначения (*useSearchParamsListener*, *useWindowSize*, *usePreviousValue* и др.) позволяют сократить дублирование текста программы и упрощают работу с состоянием.

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чатом и AI-анализом кода лабораторных работ	Лит.	Лист	Листов
Руковод.	Мельников А.Б.						51	94
Консул.						БГТУ им. В.Г. Шухова, ПВ-212		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							



- 5) SVG-иконки и графика подключаются через SVGR плагин, что обеспечивает единообразие отображения и управление атрибутами SVG.
- 6) Компоненты навигации и управления состоянием, включая *PaginationComponent*, *Breadcrumbs*, *Tabs* и другие, обеспечивают взаимодействие с URL-параметрами и организацию структуры интерфейса.

### 3.1.2 Слой *entities*

Слой *entities* отвечает за интеграцию с внешними сервисами.

- 1) *api/*: тонкий слой-абстракция над HTTP-клиентами (*fetch/axios*), где функции названы в соответствии с операциями в документации Swagger (например, *getUserProfile*, *createOrder*);
- 2) *types/*: TypeScript-интерфейсы и типы для запросов и ответов (например, *UserProfileResponseType*, *OrderCreateRequestBodyType*);
- 3) *models/*: статические данные для сущности;
- 4) *services/*: обёртки для работы с локальным кэшем (*IndexedDB*, *localStorage*) и реализации *retry*-логики и таймаутов.

### 3.1.3 Слои *features*, *widgets*, *pages*

Главные рабочие слои приложения, отвечающие за реализацию конкретной функциональности:

- 1) Слой *pages* отвечает за маршрутизацию и верхний уровень страниц, описывающий пути, *guards* для доступа, асинхронную загрузку данных и выбор виджетов.
- 2) Слой *widgets* содержит презентационные и «умные» компоненты по паттерну Smart/Presentational. Презентационный компонент (*Presentational Component*) отвечает исключительно за UI и принимает данные через пропсы, не взаимодействуя напрямую с API интерфейсом или глобальным состоянием. Умный хук (*Smart Hook*) инкапсулирует бизнес-логику и передаёт необходимые данные в презентационные компоненты.
- 3) Слой *features* реализует бизнес-логику и управление состоянием с помощью собственных хуков, редьюсеров и слайсов *Redux*. В рамках этого

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		52

слоя используются следующие структуры: файл *hooks.ts* содержит главный хук-фабрику (например, *useRegistrationUniversity*); файл *schema.ts* описывает схемы форм; файл *utils.ts* содержит вспомогательные функции и селекторы; файл *store.ts* подключает функциональность к библиотек Redux или Redux Toolkit.

### 3.1.4 Пример виджета RegistrationUniversity

В листинге 3.1 представлен презентационный компонент, отвечающий за отображение формы регистрации университета.

Листинг 3.1 – Функция RegistrationUniversityWidget

```

1 // Presentationкомпонент-
2 export function RegistrationUniversityWidget() {
3   const { formDataRef, isError, setIsError, onSubmit, isLoading } =
4     useRegistrationUniversity();
5
6   return (
7     <div>...</div>
8   );
9 }

```

В листинге 3.2 представлена функция, содержащий логику взаимодействия пользователя с формой регистрации университета и обработки отправки запроса на регистрацию.

Листинг 3.2 – Функция useRegistrationUniversity

```

1 // Smartкомпонент-: хукфабрика-
2 export function useRegistrationUniversity() {
3   const formDataRef = useRef<RegistrationData>();
4   const [isError, setIsError] = useState<string[]>([]);
5   const [isLoading, setIsLoading] = useState(false);
6   const router = useRouter();
7
8   const onSubmit = async () => {
9     ...
10  };
11  return { formDataRef, isError, setIsError, onSubmit, isLoading };
12 }

```

### 3.1.5 Соглашения по именованию и структуре

Для поддержания единого стиля и предсказуемости структуры проекта:

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
						53
Изм.	Лист	№ докум.	Подп.	Дата		

- 1) Имена функций и типов для взаимодействия с серверной частью совпадают;
- 2) Компоненты и хуки получают префиксы по зоне ответственности (*RegistrationForm*, *useFilesUpload* и т. д.);
- 3) В каждом каталоге *features/FeatureName* обязателен минимум файлов: *index.ts* и сегмент;
- 4) Названия типов и функций для взаимодействия с серверной частью должны содержать информацию о типе запроса и типе данных (тело запроса или возвращаемые данные).

## 3.2 Собственная UI-библиотека и генерация форм

### 3.2.1 Общая идея и мотивация

Для обеспечения единого стилистического и функционального каркаса клиентского приложения была разработана собственная библиотека компонентов и утилит, распространяемая через npm-пакет. В её составе присутствуют:

- 1) Набор готовых UI-компонентов (например, *Button*, *Tag*, *ScrollProvider*), не содержащих бизнес-логику;
- 2) Провайдеры глобальных состояний и контекстов (например, для управления скроллом или обработкой событий клавиатуры);
- 3) Унифицированный генератор форм *FormBuilder*, позволяющий описывать структуру и поведение сложных форм через декларативную схему.

Применение данной библиотеки ускоряет процессы разработки и упрощает поддержку интерфейса, так как все ключевые решения собраны в централизованном модуле с единым API интерфейс и консистентной документацией.

### 3.2.2 Компонент *FormBuilder*

Ключевым элементом библиотеки является компонент *FormBuilder*. Он реализует маршрутизацию данных и событий между декларативной схемой формы и её полями. Основные принципы работы:

- 1) Пользователь задаёт схему параметров формы, представляющую собой массив объектов с полем *type* и соответствующим набором параметров;

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		54

- 2) Компонент *FormBuilder* инициализирует внутреннее состояние формы и передаёт каждому полю текущие значения и функции обработки изменений;
- 3) При срабатывании события изменения значения поле уведомляет компонент *FormBuilder*, который обновляет общую модель данных и вызывает функцию обратного вызова *onChange*.

Схема формы описывается функцией, возвращающей массив объектов. Каждый объект содержит поле *type* — идентификатор типа элемента (например, *input\_field* или *array\_fields*), а также *props* — набор свойств, необходимых для рендеринга и обработки (таких как имя поля, текст метки и дополнительные параметры). Пример описания схемы формы приведён в листинге 3.3.

Листинг 3.3 – Пример описания схемы формы

```

1  export function inviteTeacherScheme(): FORM_BUILDER_SCHEMA {
2      return [
3          {
4              type: 'input_field',
5              props: {
6                  name: 'email',
7                  labelText: 'Email'
8              }
9          },
10         {
11             type: 'input_field',
12             props: {
13                 type: 'select',
14                 name: 'department_id',
15                 ownerInputComponent: <DepartmentSelectField />
16             }
17         }
18     ];
19 }

```

Пример использования компонента *FormBuilder* приведён в листинге 3.4.

Листинг 3.4 – Использование компонента *FormBuilder*

```

1  <FormBuilder schema={inviteTeacherScheme()}
2      onChange={onChangeFormData}/>

```

Компонент *FormBuilder* автоматически распределяет данные между полями и собирает итоговый объект формы, передавая его через функцию

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		55

*onChange*.

### 3.2.3 Типы элементов схемы и их поведение

Ниже приведены ключевые типы схем, поддерживаемые компонентом *FormBuilder*, и описание их функциональности.

Схема *INPUT\_FIELD\_SCHEMA* отвечает за отображение и управление единичным полем ввода. Она включает обязательное поле *name* (ключ в итоговом объекте данных), а также опциональные параметры: *labelText* (отображаемая метка поля), *hintText* (текст подсказки под полем), *type* (уточнение типа поля, например *select* или *datetime*) и *ownerInputComponent* (пользовательский компонент, принимающий параметры: *value*, *onChange*, *isError* и *onBlur*). Пример использования схемы представлен в листинге 3.5.

Листинг 3.5 – Пример схемы *INPUT\_FIELD\_SCHEMA*

```
1 const schema: INPUT_FIELD_SCHEMA = {
2   type: 'input_field',
3   props: {
4     name: 'username',
5     labelText: 'Имя пользователя',
6     hintText: 'Введите ваш логин'
7   }
8 };
```

Схема *ARRAY\_FIELDS\_SCHEMA* предназначена для формирования массива однотипных входных полей, где все вложенные элементы *input\_field* объединяются в массив. Обязательными параметрами схемы являются: *name* (имя массива в итоговой модели данных) и *children* (массив схем элементов, составляющих каждую отдельную запись). Пример использования данной схемы представлен в листинге 3.6.

Листинг 3.6 – Пример схемы *ARRAY\_FIELDS\_SCHEMA*

```
1 const schema: ARRAY_FIELDS_SCHEMA = [
2   {
3     type: 'array_fields',
4     props: {
5       name: 'subjects',
6       children: [
7         {
8           type: 'input_field',
9           props: {
```

```

10         name: 'subjectName',
11         labelText: 'Название предмета'
12     }
13 }
14 ]
15 }
16 }
17 ];

```

Схема *FORM\_WRAPPER\_SCHEMA* обеспечивает группировку полей без сброса индексов массивов, сохраняя вложенные элементы как объект. Обязательными параметрами выступают *name* (ключ в результирующем объекте) и *children* (схема вложенных элементов). Пример реализации приведён в листинге 3.7.

Листинг 3.7 – Пример схемы *FORM\_WRAPPER\_SCHEMA*

```

1  const schema: FORM_WRAPPER_SCHEMA = [{
2      type: 'form_wrapper',
3      props: {
4          name: 'teacherInfo',
5          children: [
6              {
7                  type: 'input_field',
8                  props: { name: 'firstName', labelText: 'Имя' }
9              },
10             {
11                 type: 'input_field',
12                 props: { name: 'lastName', labelText: 'Фамилия' }
13             }
14         ]
15     }
16 }];

```

Схема *BLOCK\_WRAPPER\_SCHEMA* позволяет визуально группировать элементы без изменения структуры данных: поля в нём обрабатываются как часть текущего массива или объекта. Пример схемы показан в листинге 3.8.

Листинг 3.8 – Пример схемы *BLOCK\_WRAPPER\_SCHEMA*

```

1  const schema: BLOCK_WRAPPER_SCHEMA = [
2      {
3          type: 'block_wrapper',
4          props: {
5              children: [
6                  {

```

```

7         type: 'input_field',
8         props: {
9             name: 'code',
10            labelText: 'Код'
11        }
12    }
13 ]
14 }
15 }
16 ];

```

Схема *REACT\_NODE\_SCHEMA* предназначена для вставки произвольного React-элемента в форму. Пример схемы показан в листинге 3.9.

Листинг 3.9 – Пример схемы *REACT\_NODE\_SCHEMA*

```

1 const schema: REACT_NODE_SCHEMA = [
2   {
3     type: 'react_node',
4     props: {
5       node: <CustomSeparator />
6     }
7   }
8 ];

```

### 3.2.4 Преимущества и выводы

Использование компонента *FormBuilder* существенно снижает сложность создания многоуровневых форм:

- 1) Единообразие описания различных полей;
- 2) Возможность единообразной валидации и управления ошибками;
- 3) Добавление новых полей сводится к регистрации нового блока схемы;
- 4) Повышение читаемости текста программы: структура формы полностью отражена в схеме без дублирования логики в компонентах.

### 3.2.5 Работа с токеном JWT

Для передачи и обновления токена JWT в сессии используется функция обратного вызова *jwt*, принимающий параметр *trigger*, позволяющий определить сценарий обработки. Логика работы примерно следующая: при первом входе пользователя (*trigger = signIn*) в токен записываются поля *access\_token*,

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		58

*refresh\_token* и прочие метаданные; при последующих запросах проверяется срок жизни *access\_token*, и при необходимости инициируется процесс его обновления (*trigger* = *update*). Если же токен ещё валиден, возвращается неизменная структура. Пример реализации приведён в листинге 3.10.

Листинг 3.10 – Функция обратного вызова *jwt*

```

1  async jwt({ token, trigger, user, session }): Promise<JWT> {
2    // Срабатывает при первичной аутентификации (signIn)
3    if (trigger === 'signIn') {
4      return { ...user, error: null };
5    }
6    // Если токена ещё нет например(, при восстановлении сессии из куки)
7    if (token = null) {
8      return { ...session?.user, error: 'another' } as JWT;
9    }
10   // При запросе обновления (trigger = 'update')
11   if (trigger === 'update') {
12     return await refreshingProcess(token);
13   }
14   // Во всех остальных случаях токен( валиден), возвращаем прежнее
15   ↪ состояние
16   return { ...token, error: null };
17 }

```

Ниже на рисунке 3.1 показана вся последовательная диаграмма, иллюстрирующая проверку срока жизни JWT токена на клиенте и, при необходимости, получение нового JWT токена по токену обновления. Эта схема помогает понять, как именно библиотека *Auth.js* взаимодействует с сервером для устойчивого хранения и своевременного обновления токенов без лишних повторных запросов.

На рисунке 3.1 можно выделить два основных этапа: аутентификация пользователя и процесс использования токена.

При входе пользователя в систему происходит следующая последовательность действий:

- 1) Пользователь вводит поля *username* и *password* в приложение;
- 2) Модуль *Auth.js* формирует объект с учётными данными и отправляет запрос на сервер (метод *login*);
- 3) Сервер возвращает токен доступа и токен обновления;



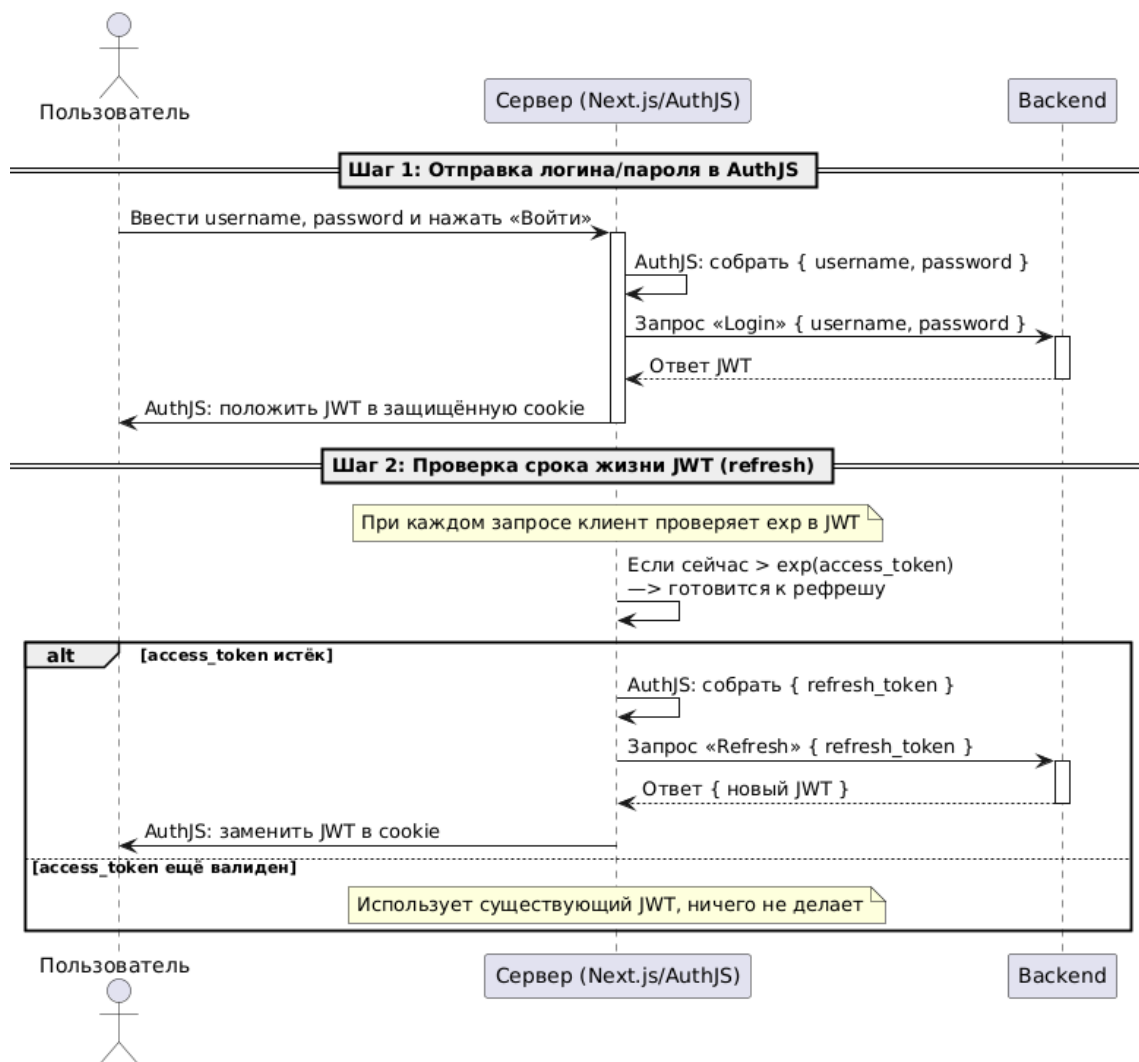


Рисунок 3.1 – Схема процесса проверки и обновления JWT токена через токен обновления

- 4) Модуль Auth.js сохраняет полученный JWT токен в защищённые cookie данные.

После успешной аутентификации при каждом запросе выполняется проверка срока жизни токена и, при необходимости, его обновление. Ниже приведены основные этапы этой проверки и процесса обновления:

- 1) При каждом запросе клиент проверяет поле *exp* (срок жизни) в токене доступа.
- 2) Если текущий момент времени превысил время жизни токена доступа, начинается подготовка к обновлению токена.
- 3) В случае истечения срока действия токена доступа модуль Auth.js отправ-

ляет токен обновления на сервер. Сервер возвращает новый токен доступа и новый токен обновления. После этого модуль Auth.js заменяет старый токен в cookie браузера на новый.

- 4) Если же токен доступа ещё валиден, модуль Auth.js просто использует существующий токен и не делает дополнительных запросов (промежуточные уведомления внизу диаграммы).

Таким образом, схема на рисунке 3.1 демонстрирует, что клиент всегда сначала пробует воспользоваться существующим токеном доступа, проверяя его валидность. Только если проверка не проходит, выполняется последовательность обновления, благодаря чему повышается отказоустойчивость и исключается ситуация «гонки» при параллельных запросах на обновление.

Важным дополнением к этой концепции является механизм предотвращения «гонки состояний» при одновременном запросе нескольких API-методов, обнаруживающих, что токен доступа просрочен. В таких случаях на клиенте сохраняется единственный промис обновления, который переиспользуется всеми последующими запросами до получения ответа от сервера. Пример реализации этого механизма приведён в листинге 3.11.

Листинг 3.11 – Механизм предотвращения гонки состояний при рефреше токена

```
1  export type RefreshPromiseStateType = Promise<JWT | null> | null;
2  let tokenPromiseState: RefreshPromiseStateType = null;
3  export const setTokenPromiseState = (
4    promise: RefreshPromiseStateType
5  ): void => {
6    tokenPromiseState = promise;
7  };
8  export const getTokenPromiseState = (): RefreshPromiseStateType => {
9    return tokenPromiseState;
10 };
11
12 let tokenState: JWT | null = null;
13 export const setTokenState = (newTokenState: JWT | null): void => {
14   tokenState = newTokenState;
15 };
16 export const getTokenState = (): JWT | null => {
17   return tokenState;
18 };
19
```

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		61

```
let timeoutState: NodeJS.Timeout | null = null;
```

На основе приведённой логики обеспечивается централизованная обработка авторизации и обновления токенов без дублирования текста программы в разных частях приложения. Кроме того, использование одной общей очереди запросов к серверу для обновления JWT токена предотвращает нежелательные состояния гонки и лишние обращения к серверу.

### 3.3 Модуль «Регистрация и вход»

В системе предусмотрены три сценария регистрации: для университета, студентов и преподавателей. В URL-параметрах передаются данные приглашения для последующей валидации и передачи на сервер.

Выбор формы регистрации осуществляется в зависимости от типа пользователя, переданного через параметры URL-строки. Пример логики выбора виджета приведён в листинге 3.12.

Листинг 3.12 – Выбор виджета регистрации по типу

```
1 const getForm = () => {
2   const type = getSearchParams(REGISTRATION_TYPE_PARAM_NAME) as
   ↳ RegistrationTypesType;
3   const inviteId = getInviteId();
4   switch (type) {
5     case 'teacher':
6       return <RegistrationTeacherWidget inviteId={inviteId} />;
7     case 'student':
8       return <RegistrationStudentWidget inviteId={inviteId} />;
9     case 'university':
10      default:
11        return <RegistrationUniversityWidget />;
12    }
13  };

```

На рисунке 3.2 показан интерфейс страницы регистрации университета. Пользователь должен ввести название учебного заведения, фамилию, имя и отчество контактного лица, почты администрации университета, а также задать и подтвердить пароль. После заполнения всех обязательных полей и нажатия кнопки «Зарегистрировать университет» инициируется отправка данных на сервер.

Листинг 3.13 демонстрирует структуру виджета регистрации преподава-

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		62

Рисунок 3.2 – Страница регистрации университета: поля для названия института, контактного лица (ФИО), почты администрации, пароля и подтверждения пароля.

теля. Здесь реализована обработка валидности приглашения и логика отображения состояния формы в зависимости от результата асинхронной проверки.

Листинг 3.13 – Виджет регистрации преподавателя

```

1  export function RegistrationTeacherWidget({ inviteId }:
    ⇨ RegistrationPropsType) {
2      const { initData, onSubmit, isError, setIsError, formDataRef } =
3          useRegistrationStudentAndTeacher<typeof registerTeacher>({
4              inviteId,
5              registrationRequest: registerTeacher
6          });
7
8      if (initData === undefined) {
9          return 'Loading';
10     }
11
12     if (initData === null) {
13         // Неверное приглашение или оно истекло
14         return 'Error';
    
```

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		63

```

15 }
16
17 // Дальнейшая логика отображения формы регистрации преподавателя
18 ...
19 }

```

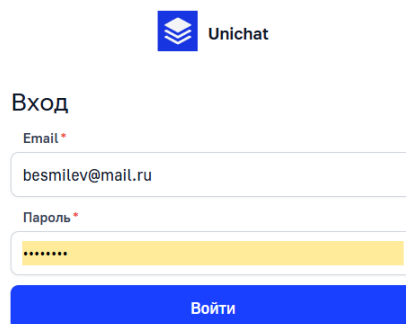


Рисунок 3.3 – Экран входа в систему: поля для ввода почты и пароля, а также кнопка «Войти».

На рисунке 3.3 представлен экран входа в систему. Для авторизации пользователь вводит почту и пароль, после чего нажимает кнопку «Войти». В случае успешной авторизации система перенаправляет его в личный кабинет; при некорректных учётных данных отображается сообщение об ошибке.

### 3.3.1 Модуль формирования и отправки приглашений

В административной панели университета реализован модуль, который позволяет формировать и отправлять приглашения как для преподавателей, так и для студентов. Этот модуль работает по следующему сценарию:

- 1) При инициации приглашения администратор вводит адрес электронной почты и идентификатор кафедры (для преподавателя) или адрес электронной почты и идентификатор группы (для студента), после чего нажимает кнопку «Сформировать ссылку».

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		64

- 2) Затем пользовательский интерфейс отправляет запрос на создание приглашения на сервер. Сервер проверяет права администратора, корректность введённых данных, генерирует уникальную ссылку приглашения и возвращает её обратно в интерфейс.
- 3) В ответ пользовательский интерфейс получает от сервера объект с полем *invite\_link* (ссылка приглашения), отображает администратору полученную ссылку, которую можно скопировать и отправить по электронной почте.

На рисунке 3.4 приведена последовательная диаграмма, иллюстрирующая полный процесс формирования и отправки приглашений: в верхней части — сценарий для преподавателя, в нижней части — сценарий для студента.

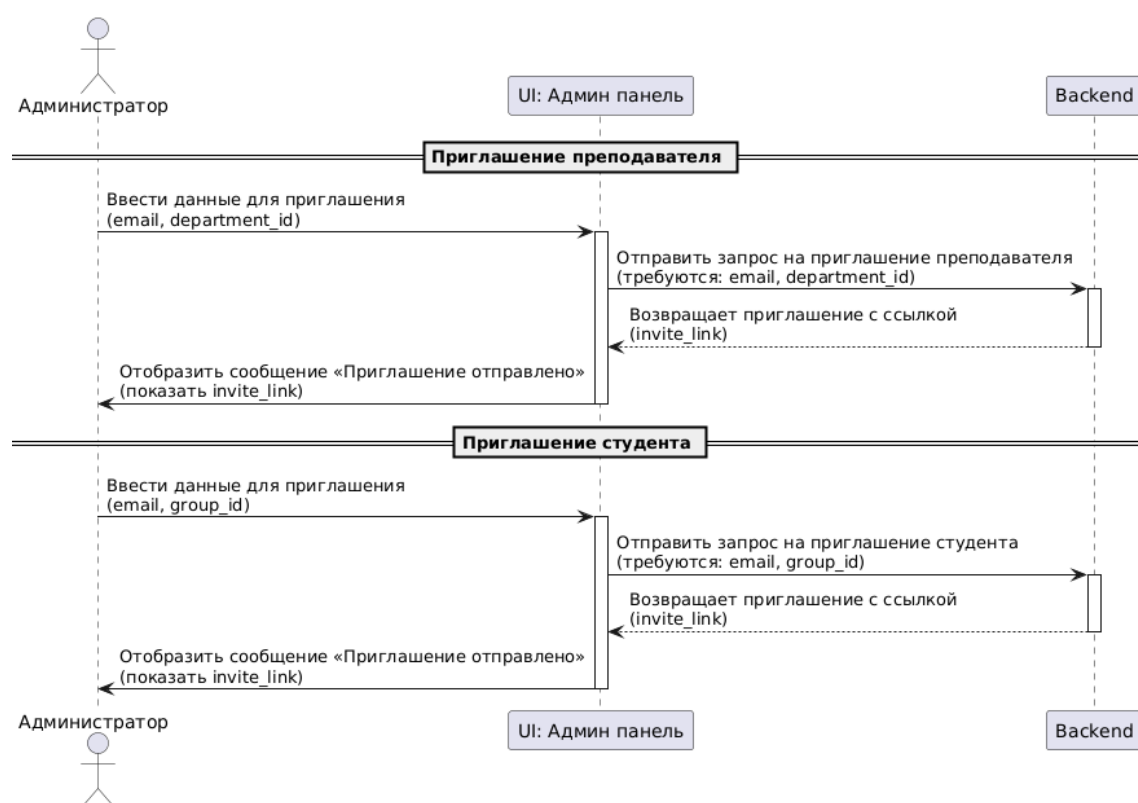


Рисунок 3.4 – Схема процесса формирования и отправки приглашений преподавателям и студентам

На рисунке 3.4 можно выделить следующие этапы:

- 1) При приглашении преподавателя администратор вводит электронную почту и идентификационный номер кафедры. Пользовательский интерфейс

отправляет запрос на сервер с указанными данными. Сервер проверяет данные, создаёт приглашение и возвращает уникальную ссылку. Пользовательский интерфейс отображает созданную ссылку.

- 2) При приглашении студента администратор вводит электронную почту и идентификационный номер группы. Пользовательский интерфейс отправляет запрос на сервер с указанными данными. Сервер проверяет данные, создаёт приглашение и возвращает уникальную ссылку. Пользовательский интерфейс отображает созданную ссылку.

Таким образом, схема на рисунке 3.4 демонстрирует алгоритм работы модуля: ввод данных в пользовательский интерфейс, отправка запроса на сервер, генерация и возврат уникальной ссылки, отображение ссылки администратору.

### 3.3.2 Последовательная диаграмма работы модуля Classrooms

Ниже приведена последовательная диаграмма, иллюстрирующая жизненный цикл взаимодействия преподавателя, студента и сервиса искусственного интеллекта при работе с виртуальными классами и заданиями. На рисунке А.3 показаны этапы: создание класса, создание задания с указанием промта (текстового запроса) для автопроверки, получение списка заданий студентом, отправка решения, автоматическая проверка решения и выставление оценки преподавателем.

Выделены следующие этапы:

- 1) При создании класса преподаватель открывает интерфейс создания класса, интерфейс отправляет данные класса на сервер и получает подтверждение.
- 2) При создании задания преподаватель указывает условие и текстовый запрос для проверки искусственным интеллектом, интерфейс отправляет данные на сервер, сервер подтверждает сохранение.
- 3) При получении списка заданий студент открывает интерфейс студента, интерфейс запрашивает список заданий по идентификатору класса, сервер возвращает список доступных заданий, интерфейс отображает их студенту.

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		66

- 4) При отправке решения студент выбирает задание и отправляет решение через интерфейс, отправляемые данные (файлы решения и метаданные, например, идентификаторы студента и задания) принимаются сервером, сервер подтверждает получение.
- 5) При автоматической проверке сервер получает решение и текстовый запрос, отправляет их во внешний сервис искусственного интеллекта, который анализирует текст программы (проверка корректности, стилистических нарушений и т.п.) и возвращает комментарии с оценкой. Сервер сохраняет результаты и передаёт их в интерфейсы преподавателя и студента, отображая им результат автопроверки (комментарии и предварительную оценку).
- 6) При выставлении финальной оценки преподаватель открывает интерфейс выставления оценки, отправляет оценку через интерфейс, сервер сохраняет оценку и подтверждает её сохранение. Интерфейс отображает подтверждение преподавателю и обновлённую оценку студенту.

Таким образом, последовательная диаграмма на рисунке А.3 демонстрирует цикл взаимодействий: от создания класса и задания преподавателем до получения студентом финальной оценки после автоматической проверки и ручной оценки преподавателем.

### 3.3.3 Последовательная диаграмма работы модуля Chats

Модуль «Chats» обеспечивает обмен сообщениями в реальном времени между пользователями (преподавателями и студентами) посредством WebSocket-соединения. На рисунке А.2 приведена последовательная диаграмма, демонстрирующая этапы работы чата: открытие приложения, выбор беседы, отправка и приём сообщений, обработку JWT токена и переподключение.

Выделены следующие этапы:

- 1) При открытии приложения пользователь переходит в раздел «Чаты», интерфейс пользователя загружает список бесед, получает токен доступа из cookie данным и устанавливает WebSocket-соединение с сервером, передавая токен;

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		67



- 2) При выборе чата пользователь выбирает беседу, интерфейс отправляет HTTP-запрос на сервер для получения сообщений чата и отображает их;
- 3) При отправке нового сообщения пользователь вводит текст сообщения, интерфейс отправляет его на сервер по WebSocket протоколу, сервер сохраняет сообщение и отправляет подтверждение обратно, интерфейс временно показывает отправленное сообщение со статусом «отправляется»;
- 4) При получении нового сообщения от другого участника сервер рассылает событие *new\_message* (новое сообщение), интерфейс и боковая панель обновляют сообщения и список бесед;
- 5) При редактировании сообщения интерфейс отправляет событие *edit\_message* (редактирование сообщения) на сервер, сервер обновляет сообщение и рассылает обновлённое сообщение всем участникам, интерфейс обновляет текст сообщения;
- 6) Когда пользователь просматривает сообщение, интерфейс отправляет событие *read\_message* (сообщение прочитано), сервер обновляет статус сообщения и рассылает обновления, интерфейс и боковая панель отражают изменения;
- 7) При истечении срока действия токена доступа интерфейс получает ошибку, запрашивает новый токен JWT через токен обновления, затем повторно подключается через WebSocket протокол и продолжает работу без потери данных.

Таким образом, последовательная диаграмма на рисунке А.2 отражает полный цикл работы модуля «Chats»: от открытия приложения и установления защищённого соединения до приёма, отправки, редактирования и пометки сообщений, а также автоматического обновления JWT токена и переподключения WebSocket-соединения.

### 3.4 Тестирование

В ходе разработки были протестированы модули как основного приложения, так и используемой при разработке библиотеки.

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
						68
Изм.	Лист	№ докум.	Подп.	Дата		

Таблица 3.1 – Покрытие текста программы клиентской части 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

### 3.4.1 Покрытие текста программы в приложении

В таблице 3.1 приведены четыре основные метрики покрытия:

- 1) %Stmts (Statements) — процент операторов текста программы, выполненных в ходе тестов;
- 2) %Branch (Branches) — процент ветвей условных операторов, затронутых тестами;
- 3) %Funcs (Functions) — процент функций, вызванных хотя бы одним тестом;
- 4) %Lines (Lines) — процент строк текста программы, исполненных тестами.

Для ключевых модулей (например, *mergeMessages.ts*) все метрики равны 100 %.

### 3.4.2 Покрытие текста программы в отдельной библиотеке

Таблица 3.2 – Покрытие текста программы 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

В таблице 3.2 показано, что основные утилитные функции библиотеки

(*getDeepValue*, *setDeepValue*, *changeInterval*) полностью или почти полностью покрыты.

### 3.4.3 Методы тестирования

В проекте использовались различные подходы к тестированию, направленные на проверку корректности функциональности, устойчивости к ошибкам и полноты покрытия. Ниже перечислены основные применённые методики:

- 1) Модульное тестирование (unit testing) применяется для каждой функции и компонента, при этом тесты охватывают граничные и некорректные входные данные (например, *undefined*, пустые массивы), типичные сценарии и пограничные случаи;
- 2) TDD-подход (Test-Driven Development) основывается на цикле разработки, где сначала пишется тест, затем фиксируется его неуспешное выполнение, после чего создаётся минимальная реализация и проводится повторное тестирование, завершающееся рефакторингом;
- 3) Покрытие ветвлений (branch coverage) достигается за счёт тестов, охватывающих все возможные пути выполнения условных операторов, включая конструкции *if/else*, тернарные выражения и операторы *switch*;
- 4) Round-trip-тесты проверяют корректность преобразований между текстом и HTML в функциях *processTextToTiptap* и *processTiptapToText*, особенно в сложных случаях с вложенными тегами и переносами строк;
- 5) Инструментация и сбор покрытия реализуются через запуск команды *npx jest --coverage*, которая автоматически добавляет счётчики в текст программы и формирует метрики по выражениям, ветвлениям, функциям и строкам.

					ПРОГРАММНАЯ РЕАЛИЗАЦИЯ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		70

## ЗАКЛЮЧЕНИЕ

В рамках проведённой дипломной работы было улучшено и упрощено взаимодействие преподавателей и студентов в образовательном процессе университета посредством разработки клиентской части интегрированного образовательного приложения. Основная цель заключалась в повышении качества опыта работы преподавателей и обучения студентов, обеспечивая единое пространство для коммуникаций, автоматизированной проверки заданий и управления учебными ресурсами.

Для достижения поставленной цели были выполнены следующие задачи:

- 1) Проанализирована предметная область и существующие образовательные системы, выявлены их сильные и слабые стороны;
- 2) Определены архитектурные и технологические решения (Next.js, React, TypeScript, Redux, Auth.js, Socket.IO) для реализации гибкой, надёжной и удобной клиентской части приложения;
- 3) Спроектирован пользовательский интерфейс, обеспечивающий интуитивное и удобное взаимодействие для преподавателей и студентов (навигация, адаптивная вёрстка, единые UI-компоненты);
- 4) Разработаны компоненты для управления учебными структурами (институты, кафедры, группы), заданиями и чатами, включая административную панель, что упростило работу преподавателей при ведении учебного процесса;
- 5) Интегрированы средства автоматизированной проверки решений студентов с применением ИИ в модуле виртуальных классов, что повысило качество и скорость проверки лабораторных работ;
- 6) Реализовано тестирование бизнес-логики клиентского приложения с использованием библиотеки Jest, что подтвердило корректность и стабильность основных функций.

					ЗАКЛЮЧЕНИЕ			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чатами и AI-анализом кода лабораторных работ	Лит.	Лист	Листов
Руковод.	Мельников А.Б.						71	94
Консул.						БГТУ им. В.Г. Шухова, ПВ-212		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							

Таким образом, все поставленные задачи выполнены, а основная цель достигнута: предложенные архитектурные решения обеспечили удобство работы преподавателей и улучшили опыт обучения студентов. Разработанный интерфейс продемонстрировал эффективность в упрощении коммуникаций, автоматизации рутинных задач и повышении качества образовательного процесса.

С практической точки зрения, данное клиентское приложение обладает следующими преимуществами:

- 1) Быстрая разработка за счёт переиспользования компонентов и генерации форм, что позволяет преподавателям оперативно адаптировать интерфейс под учебный процесс;
- 2) Надёжная безопасность благодаря механизму управления сессиями и авторизации;
- 3) Удобство сопровождения за счёт модульной структуры и чёткого разделения ответственности, что облегчает расширение и поддержку проекта;
- 4) Расширяемость и готовность к новым сценариям благодаря гибкой конфигурации интерфейса, что позволяет быстро внедрять новые образовательные инструменты;
- 5) Высокая устойчивость системы, подтверждённая результатами тестирования бизнес-логики, что гарантирует стабильность работы приложения для всех участников образовательного процесса.

Перспективными направлениями дальнейшего развития являются:

- 1) Расширение функционала анализа текста программ на основе искусственного интеллекта за счёт надстроек для автоматической проверки студенческих работ по заданным паттернам;
- 2) Добавление офлайн-режима работы с последующей синхронизацией изменений при восстановлении связи, что улучшит опыт студентов в условиях нестабильного интернета;
- 3) Разработка мобильных клиентских приложений для повышения доступности приложения и удобства работы преподавателей и студентов на разных устройствах;

					ЗАКЛЮЧЕНИЕ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		72

- 4) Внедрение системы аналитики пользовательской активности для оптимизации интерфейса, оценки эффективности учебного процесса и принятия обоснованных решений по улучшению образовательной среды.

Итоги работы подтверждают, что предложенные решения действительно упростили работу преподавателей, улучшили опыт обучения студентов и повысили качество образовательного процесса. Открываются новые возможности для дальнейших исследований и совершенствования системы.

					ЗАКЛЮЧЕНИЕ			
Изм.	Лист	№ докум.	Подп.	Дата	<p>Разработка front-end Web-приложения – учебной среды с чаттами и AI-анализом кода лабораторных работ</p>	Лит.	Лист	Листов
Разраб.	Бондаренко С.В.							
Руковод.	Мельников А.Б.						73	94
Консул.						<p>БГТУ им. В.Г. Шухова, ПВ-212</p>		
Н. контр.	Осипов О.В.							
Зав. Каф.	Поляков В.М.							

## СПИСОК ЛИТЕРАТУРЫ

1. Moodle Docs: Main Page. — Accessed: 2025-06-10. [https://docs.moodle.org/4/en/Main\\_page](https://docs.moodle.org/4/en/Main_page).
2. Google Classroom Documentation. — Accessed: 2025-06-10. <https://edu.google.com/intl/ru/products/classroom/>.
3. Microsoft Teams for Education Documentation. — Accessed: 2025-06-10. <https://learn.microsoft.com/education>.
4. CodeSignal. — Accessed: 2025-06-10. <https://en.wikipedia.org/wiki/CodeSignal>.
5. Codility. — Accessed: 2025-06-10. <https://en.wikipedia.org/wiki/Codility>.
6. LeetCode. — Accessed: 2025-06-10. <https://en.wikipedia.org/wiki/LeetCode>.
7. *TypeScript*. TypeScript Handbook. — Accessed: 2025-06-10. <https://www.typescriptlang.org/docs/handbook/intro.html>.
8. *React*. Getting Started with React. — Accessed: 2025-06-10. <https://reactjs.org/docs/getting-started.html>.
9. *Angular*. What is Angular? — Accessed: 2025-06-10. <https://angular.io/guide/what-is-angular>.
10. *Angular*. Dependency Injection. — Accessed: 2025-06-10. <https://angular.io/guide/dependency-injection>.
11. *Vue.js*. Introduction to Vue.js. — Accessed: 2025-06-10. <https://vuejs.org/v2/guide/>.
12. Next.js Server-Side Rendering. — Accessed: 2025-06-10. <https://nextjs.org/docs/basic-features/pages#server-side-rendering>.
13. Next.js Static Generation and ISR. — Accessed: 2025-06-10. <https://nextjs.org/docs/basic-features/data-fetching/get-static-props>.
14. Socket.IO Documentation. — Accessed: 2025-06-10. <https://socket.io/docs/v4/>.
15. Pusher Documentation. — Accessed: 2025-06-10. <https://pusher.com/docs>.
16. *Docs M. W.* WebSocket API. — Accessed: 2025-06-10. <https://developer.mozilla.org/docs/Web/API/WebSocket>.
17. NextAuth.js Documentation. — Accessed: 2025-06-10. <https://next-auth.js.org/getting-started/introduction>.

					СПИСОК ЛИТЕРАТУРЫ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		74

18. Redux – Getting Started. — Accessed: 2025-06-10. <https://redux.js.org/introduction/getting-started>.
19. DeepSeek Documentation: Module Overview. — Accessed: 2025-06-10. <https://deepseek.ai/docs/overview>.
20. Jest Documentation. — Accessed: 2025-06-10. <https://jestjs.io/docs/getting-started>.
21. Feature-Sliced Design Documentation. — Accessed: 2025-06-10. <https://feature-sliced.design/docs/introduction>.
22. Middleware. — Accessed: 2025-06-10. <https://nextjs.org/docs/advanced-features/middleware>.

					СПИСОК ЛИТЕРАТУРЫ	Лист
Изм.	Лист	№ докум.	Подп.	Дата		75



## ПРИЛОЖЕНИЕ А

### Диаграммы пользовательских сценариев и системных компонентов

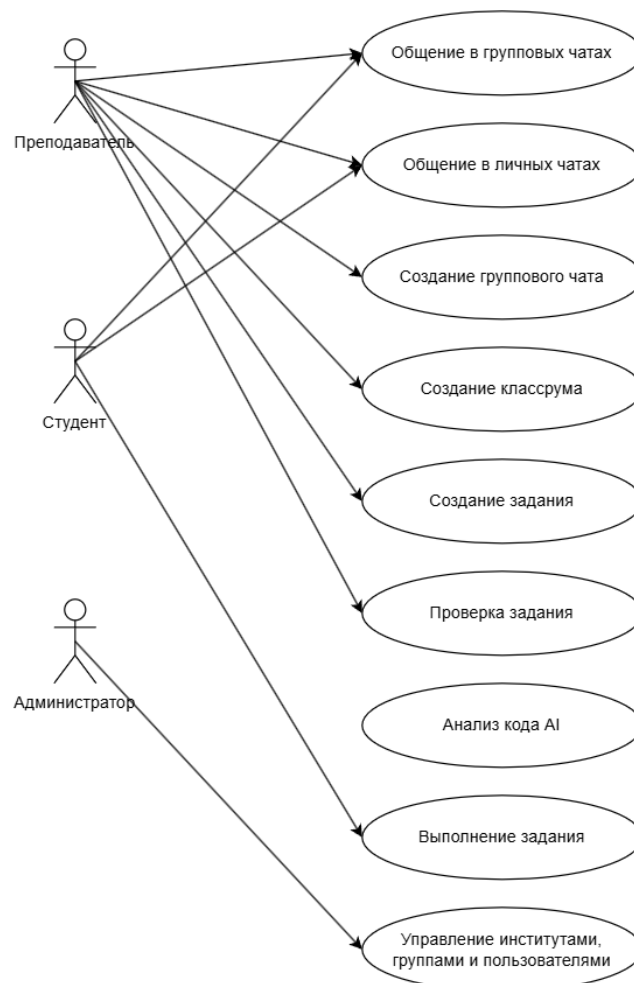


Рисунок А.1 – Диаграмма вариантов использования системы для различных ролей пользователей.

					ПРИЛОЖЕНИЕ А						
Изм.	Лист	№ докум.	Подп.	Дата	Разработка front-end Web-приложения – учебной среды с чатами и AI-анализом кода лабораторных работ			Лит.	Лист	Листов	
Разраб.	Бондаренко С.В.										
Руковод.	Мельников А.Б.									76	94
Консул.								БГТУ им. В.Г. Шухова, ПВ-212			
Н. контр.	Осипов О.В.										
Зав. Каф.	Поляков В.М.										

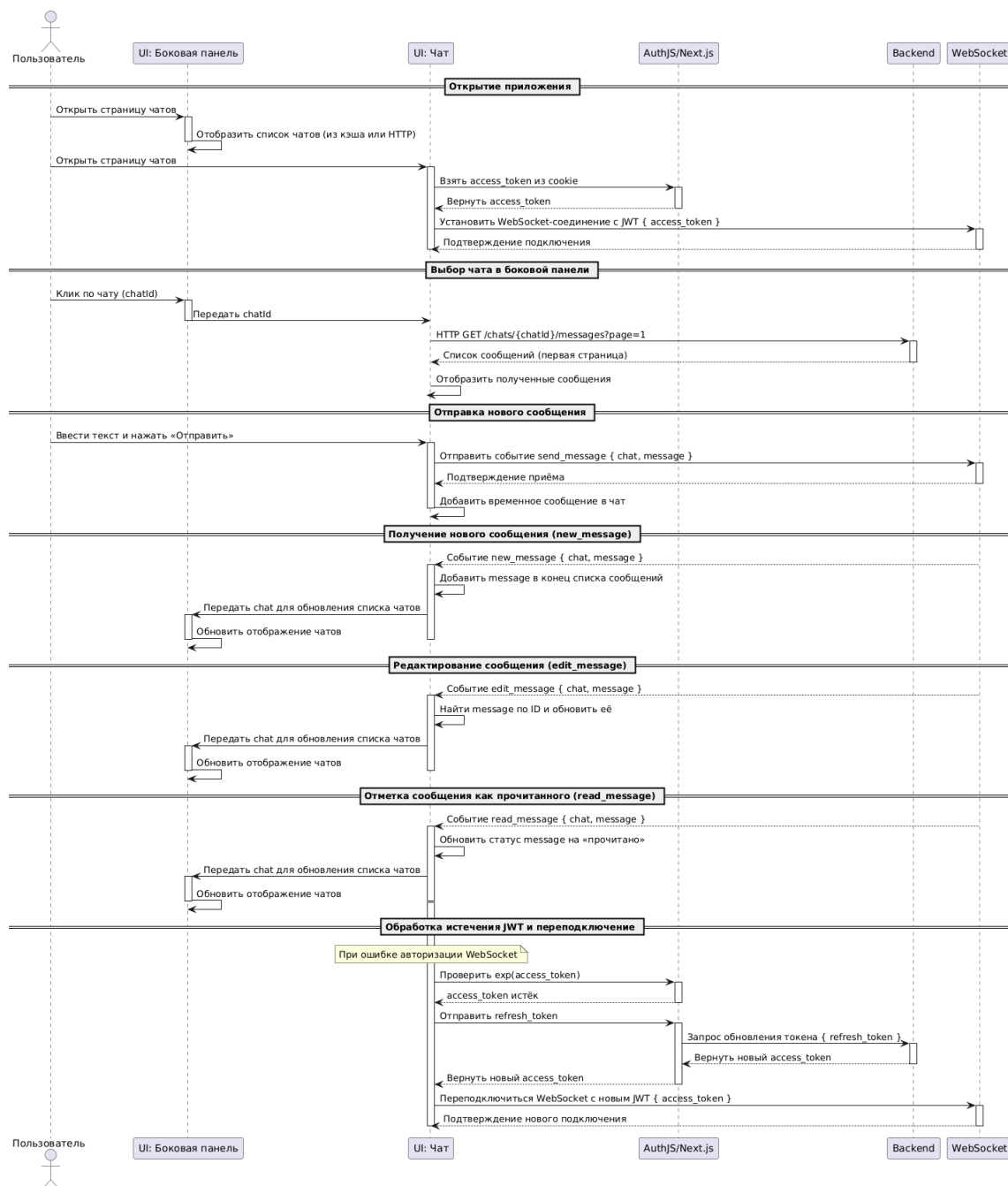


Рисунок А.2 – Схема взаимодействия клиента (UI: Боковая панель и UI: Чат), AuthJS (Next.js), сервера и WebSocket протокола при работе модуля «Chats».

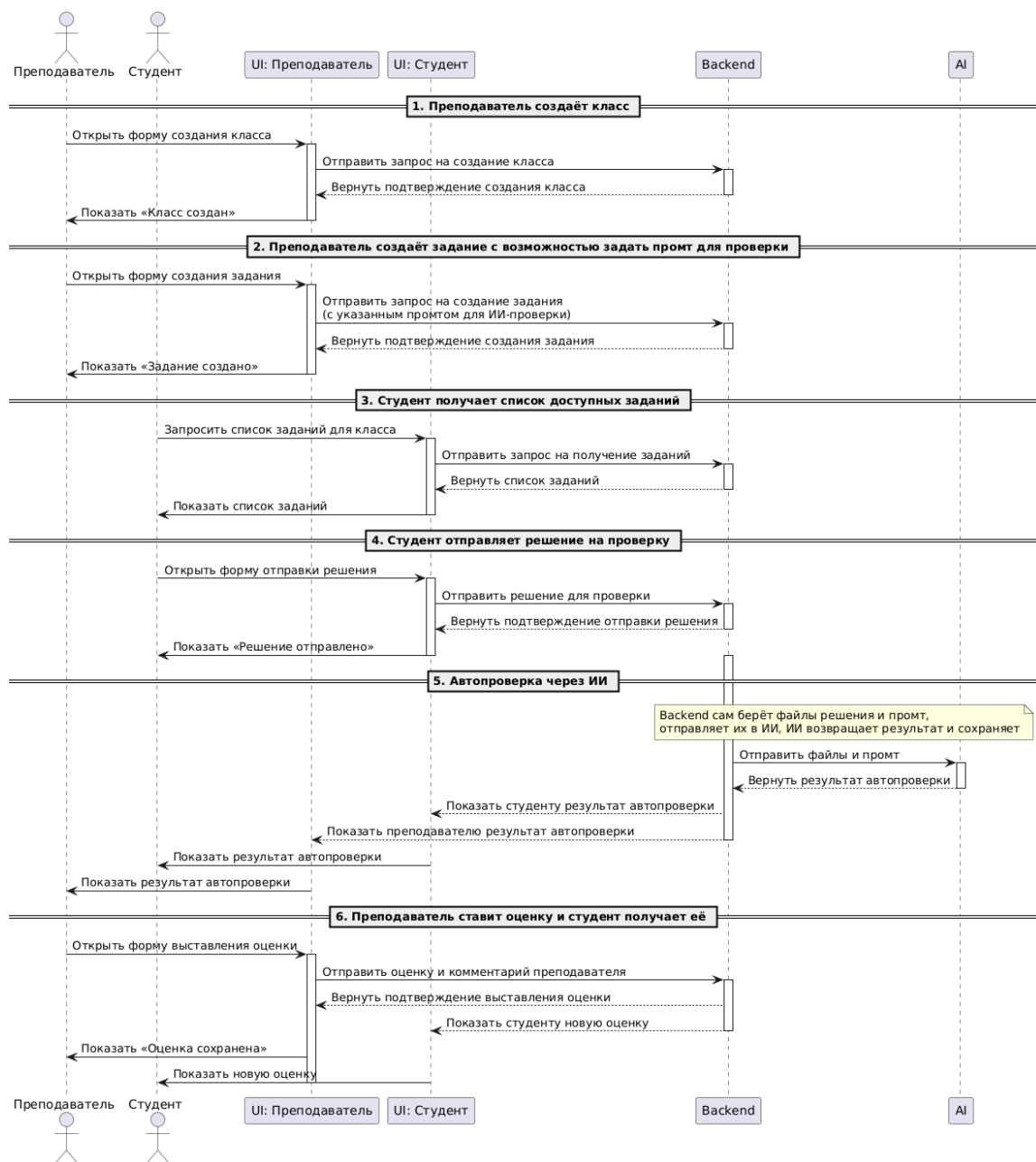


Рисунок А.3 – Схема взаимодействия преподавателя, студента, и искусственного интеллекта при работе с виртуальными классами

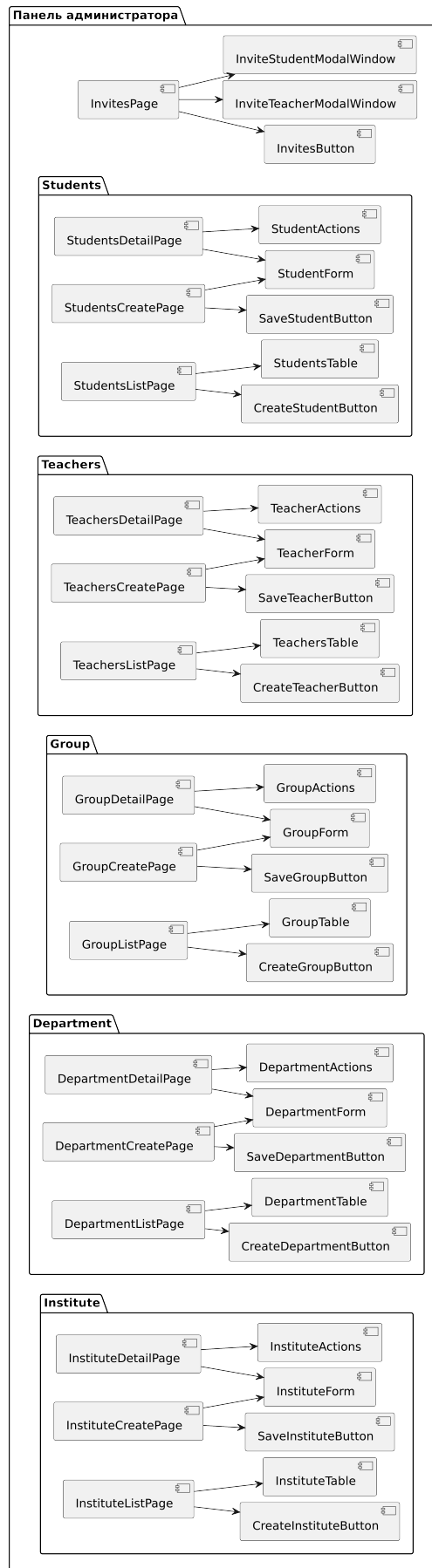


Рисунок А.4 – Диаграмма компонентов административной панели

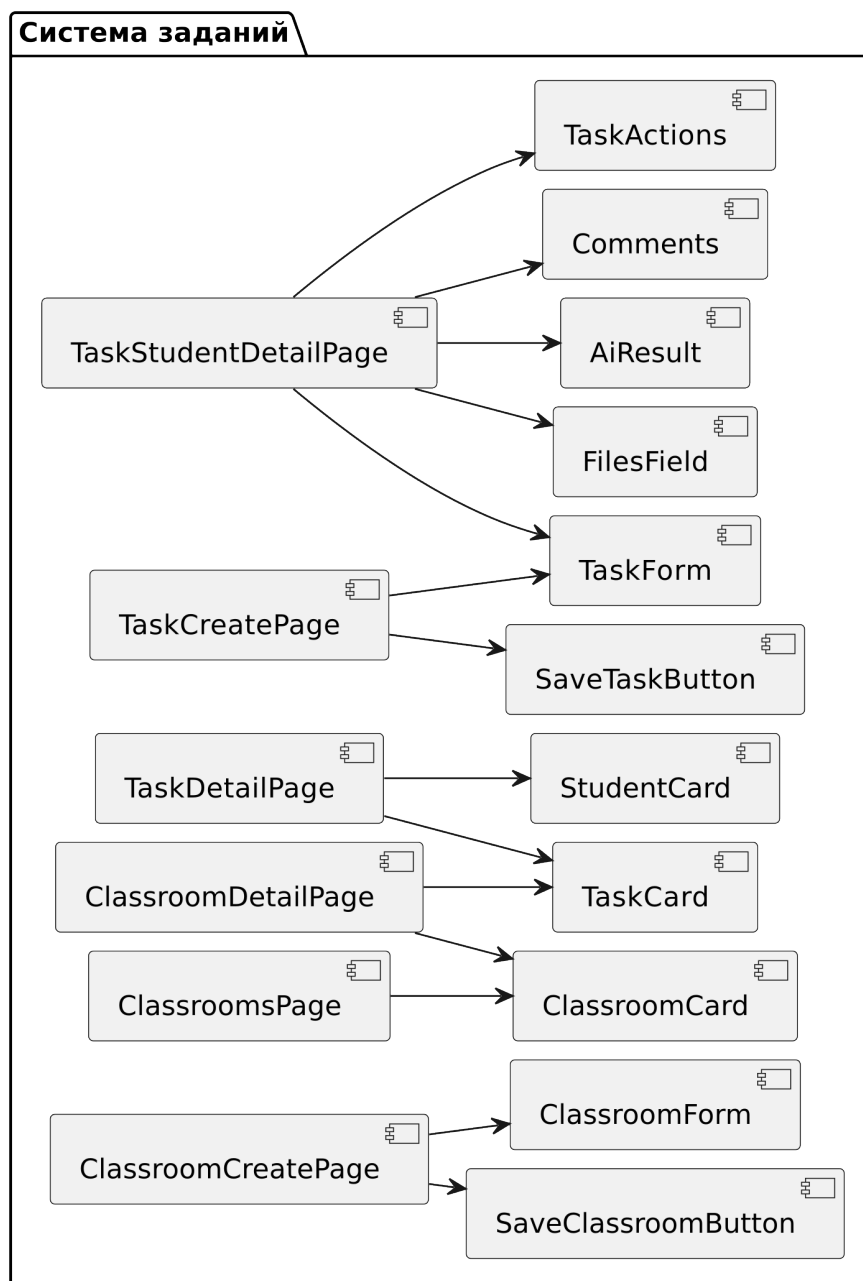


Рисунок А.5 – Диаграмма компонентов системы заданий

## ПРИЛОЖЕНИЕ Б

### Фрагмент текста программы модуля административной панели

```

1  import {useEffect, useRef, useState} from "react";
2  import {UseIsErrorFieldIsErrorType} from "indicator-ui";
3  import {signIn} from "next-auth/react";
4  import {
5      getInviteInfo,
6      InviteStudentRequestBodyType,
7      InviteTeacherRequestBodyType,
8      LoginRequestType,
9      registerStudent,
10     registerTeacher,
11     RegistrationQueryType,
12     RegistrationStudentRequestBodyType,
13     RegistrationTeacherRequestBodyType
14 } from "@/entities/Auth";
15 import {UseRegistrationPropsType} from "../types";
16
17 export function useRegistrationStudentAndTeacher<T extends typeof
    ↪ registerTeacher | typeof registerStudent>({
18
19     ↪ inviteId,
20
21     ↪ registrationRequest,
22
23     ↪ }: UseRegistrationPropsType<T>) {
24     type FormDataType = Parameters<T>[1]
25     type InviteBodyType = T extends typeof registerTeacher ?
    ↪ InviteTeacherRequestBodyType : InviteStudentRequestBodyType
26     const formDataRef = useRef<Omit<FormDataType, 'invite_id'> |
    ↪ undefined>(undefined)
27     const [isError, setIsError] = useState<UseIsErrorFieldIsErrorType>
    ↪ ([])
28     const [initData, setInitData] = useState<InviteBodyType | null |
    ↪ undefined>({email: 'besmylev@gmail.com'} as туктany)
29
30     useEffect(() => {
31         const getInitData = async () => {
32             let response
33             if (inviteId) {

```

					ПРИЛОЖЕНИЕ Б			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чаттами и AI-анализом кода лабораторных работ		Лит.	Лист
Руковод.	Мельников А.Б.							Листов
Консул.								81
Н. контр.	Осипов О.В.						БГТУ им. В.Г. Шухова, ПВ-212	
Зав. Каф.	Поляков В.М.							

```

31         response = await getInviteInfo({invite_id: inviteId}) as
↪ InviteBodyType
32         } else {
33             response = null
34         }
35         setInitData(response)
36     }
37     getInitData()
38 }, [inviteId]);
39
40 const onSubmit = async () => {
41     const formData = formDataRef.current
42     if (formData && inviteId) {
43         // Пришлось заткнуть ts так как ему мозгов не хватает
↪ сопоставить тип T и функции,
44         // а также изза- отсутствия полноценной типизации именно
↪ function
45         const response = await registrationRequest({invite_id:
↪ inviteId}, {
46             ...formData,
47             } as unknown as (RegistrationQueryType &
↪ RegistrationTeacherRequestBodyType &
↪ RegistrationStudentRequestBodyType))
48
49         if (response) {
50             const loginData: LoginRequestType = {username: formData.
↪ email, password: formData.password}
51             const loginResponse = await signIn("credentials", {
52                 ...loginData,
53                 redirect: false,
54             })
55             if (loginResponse.error) {
56                 // Обработка ошибки аутентификации
57             } else {
58                 // Успешная аутентификации
59             }
60         }
61     }
62 }
63
64 return {
65     initData,
66     onSubmit,
67     isError,
68     setIsError,
69     formDataRef,
70 }

```

					ПРИЛОЖЕНИЕ Б	Лист
						82
Изм.	Лист	№ докум.	Подп.	Дата		

```

71 }
72
73 export function CreateStudentsInvites({onClose}: { onClose?: () => void
  ↪ }) {
74   const {onSend, inviteUrl, onCopy, onChangeFormData} =
  ↪ useCreateStudentInvites()
75
76   return (
77     <div className={InviteStyle.invite}>
78       <MicroButton size={'28'}
79         color={'light'}
80         icon={<XCcloseSVG/>}
81         additionStyles={InviteStyle.closeButton}
82         onClick={onClose}/>
83       <h3 className={InviteStyle.header}>Пригласить студентов</h3>
  ↪ >
84       <div className={InviteStyle.form}>
85         <FormBuilder onChange={onChangeFormData} schema={
  ↪ inviteStudentsScheme({})}/>
86       </div>
87       <Button size={'large'}
88         hierarchy={'primary'}
89         width={'fill'}
90         onClick={onSend}
91         text={'Сформировать ссылку'}/>
92       <InputField labelText={'Ссылка'}
93         value={inviteUrl}
94         help={<Button size={'small'}
95           onClick={onCopy}
96           hierarchy={'secondary-color'}
97           iconLeft={<Copy06SVG/>}/>}/>
98     </div>
99   )
100 }
101
102 export function CreateTeacherInvites({onClose}: { onClose?: () => void
  ↪ }) {
103   const {onSend, onChangeFormData, inviteUrl, onCopy} =
  ↪ useCreateTeacherInvites()
104
105   return (
106     <div className={InviteStyle.invite}>
107       <MicroButton size={'28'}
108         color={'light'}
109         icon={<XCcloseSVG/>}
110         additionStyles={InviteStyle.closeButton}
111         onClick={onClose}/>

```

					ПРИЛОЖЕНИЕ Б	Лист
						83
Изм.	Лист	№ докум.	Подп.	Дата		



```

112      <h3 className={InviteStyle.header}>Пригласить преподавателя<
↪ /h3>
113      <div className={InviteStyle.form}>
114        <FormBuilder onChange={onChangeFormData} schema={
↪ inviteTeacherScheme()}/>
115      </div>
116      <Button size={'large'}
117        hierarchy={'primary'}
118        width={'fill'}
119        onClick={onSend}
120        text={'Сформировать ссылку'}/>
121      <InputField labelText={'Ссылка'}
122        value={inviteUrl}
123        help={<Button size={'small'}
124          hierarchy={'secondary-color'}
125          onClick={onCopy}
126          iconLeft={<Copy06SVG/>}/>}/>
127    </div>
128  )
129 }
130
131 export function AdminInvitesPage() {
132   const [isActiveModalWindow, setIsActiveModalWindow] = useState<
↪ teacher' | 'students' | undefined>(undefined)
133
134   const isShow = () => {
135     return isActiveModalWindow === undefined
136   }
137
138   const setIsShow = (isShow: boolean) => {
139     if (!isShow) {
140       setIsActiveModalWindow(undefined)
141     }
142   }
143
144   const getModalWindow = () => {
145     switch (isActiveModalWindow) {
146       case 'teacher':
147         return <CreateTeacherInvites onClose={() =>
↪ setIsActiveModalWindow(undefined)}/>
148       case 'students':
149         return <CreateStudentsInvites onClose={() =>
↪ setIsActiveModalWindow(undefined)}/>
150     }
151   }
152
153   return (

```

					ПРИЛОЖЕНИЕ Б	Лист
						84
Изм.	Лист	№ докум.	Подп.	Дата		

```

154         <div className={AdminDetailPageStyle.AdminDetailPage}>
155             <div className={clsx(AdminDetailPageStyle.content,
↪ AdminDetailPageStyle.fill, AdminDetailPageStyle.offWrapper)}>
156                 <h1 className={AdminDetailPageStyle.header}>Создание
↪ приглашений</h1>
157                 <ActionField title={'Пригласить преподавателя'}
158                     subtitle={'Сформировать ссылку, по которой
↪ преподаватель может пройти регистрацию'}
159                     onClick={() => setIsActiveModalWindow('
↪ teacher')}>/>
160                 <ActionField title={'Пригласить студентов'}
161                     subtitle={'Сформировать ссылку, по которой
↪ студенты могут пройти регистрацию'}
162                     onClick={() => setIsActiveModalWindow('
↪ students')}>/>
163             </div>
164             <BackgroundModalWindowWrapper isShow={isShow()}
165                 setIsShow={setIsShow}
166                 className={AdminDetailPageStyle
↪ .modalWindowWrapper}>
167                 {getModalWindow()}
168             </BackgroundModalWindowWrapper>
169
170         </div>
171     )
172 }

```

					ПРИЛОЖЕНИЕ Б	Лист
						85
Изм.	Лист	№ докум.	Подп.	Дата		

## ПРИЛОЖЕНИЕ В

### Фрагмент текста программы модуля системы заданий

```

1  'use client'
2
3  import {Button, FormBuilder} from "indicator-ui";
4  import Link from "next/link";
5  import {ArrowNarrowLeftSVG} from "@shared/assets";
6  import {ClassroomPostType} from "@entities/Classroom";
7  import {ROUTES_CONFIG} from "@features/Routing";
8  import {classroomScheme} from "@features/Classrooms";
9  import {ClassroomCreatePageStyle} from '../styles'
10 import {useClassroomCreate} from "../hooks";
11
12 export function ClassroomCreatePage() {
13     const {onChangeFormData, onSubmit} = useClassroomCreate()
14
15     return (
16         <div className={ClassroomCreatePageStyle.ClassroomCreatePage}>
17             <Button hierarchy={'link-color'}
18                 text={'Назад'}
19                 iconLeft={<ArrowNarrowLeftSVG/>}
20                 customComponent={<Link href={ROUTES_CONFIG.
21 ↵ CLASSROOMS_PAGE}/>}/>
22             <div className={ClassroomCreatePageStyle.content}>
23                 <FormBuilder<ClassroomPostType> schema={classroomScheme
24 ↵ ()} onChange={onChangeFormData}/>
25                 <Button size={'large'} width={'fill'} text={'Создать
26 ↵ класс'} onClick={onSubmit}/>
27             </div>
28         </div>
29     )
30 }
31
32 export function ClassroomDetailPage({id}: { id: number }) {
33     const {list, classroom} = useClassroomDetail(id)
34
35     if (list === undefined || classroom === undefined) {
36         return 'Loading'
37     }
38 }

```

					ПРИЛОЖЕНИЕ В							
Изм.	Лист	№ докум.	Подп.	Дата	Разработка front-end Web-приложения – учебной среды с чаттами и AI-анализом кода лабораторных работ			Лит.	Лист	Листов		
Разраб.	Бондаренко С.В.									86	94	
Руковод.	Мельников А.Б.							БГТУ им. В.Г. Шухова, ПВ-212				
Консул.												
Н. контр.	Осипов О.В.											
Зав. Каф.	Поляков В.М.											

```

36     if (list === null || classroom === null) {
37         return 'Error'
38     }
39
40     return (
41         <div className={ClassroomDetailPageStyle.ClassroomDetailPage}>
42             <ClassroomCard name={classroom.name}
43                 description={classroom.description}
44                 groupName={classroom.group.name}
45                 width={'fill'}/>
46             <Button size={'large'}
47                 hierarchy={'secondary-color'}
48                 text={'Добавить задание'}
49                 width={'fill'}/>
50             <div className={ClassroomDetailPageStyle.list}>
51                 {list.data.map((item, idx) => <TaskCard deadline={item.
↵ deadline}
52                                     name={item.name}
53                                     href={
54 ↵ ROUTES_CONFIG.TASKS_DETAIL_SLUG_PAGE + item.id}
55                                     key={idx}/>)}
56             </div>
57             <PaginationComponent totalCount={list.total_count}/>
58         </div>
59     )
60 }
61
62 export function ClassroomsPage() {
63     const {list} = useClassrooms()
64
65     if (list === undefined) {
66         return 'Loading...'
67     }
68
69     if (list === null) {
70         return 'Error'
71     }
72
73     return (
74         <div className={ClassroomsPageStyle.ClassroomsPage}>
75             <div className={ClassroomsPageStyle.list}>
76                 {list.data.map((item, idx) => <ClassroomCard name={item.
↵ name}
77                                     description
78                                     = {item.description}
79                                     groupName={
↵ item.group.name}

```

					ПРИЛОЖЕНИЕ В	Лист
						87
Изм.	Лист	№ докум.	Подп.	Дата		

```

78                                     avatar={item
↪ .avatar}
79                                     href={
↪ ROUTES_CONFIG.CLASSROOMS_DETAIL_SLUG_PAGE + item.id}
80                                     key={idx}/>
↪ )}
81     </div>
82     <PaginationComponent totalCount={list.total_count}/>
83     <Button size={'large'}
84         width={'fill'}
85         hierarchy={'secondary-color'}
86         text={'Создать класс'}
87         customComponent={<Link href={ROUTES_CONFIG.
↪ CLASSROOMS_CREATE_PAGE}/>}/>
88     </div>
89 )
90 }

```

					ПРИЛОЖЕНИЕ В	Лист
Изм.	Лист	№ докум.	Подп.	Дата		88

## ПРИЛОЖЕНИЕ Г

### Фрагмент текста программы модуля обмена сообщениями

```

1 import React, {forwardRef, useImperativeHandle, useRef} from "react";
2 import {v4 as uuidv4} from "uuid";
3 import {MessageInput, MessageInputRefType} from "@shared/ui";
4 import {ChatSocketEmitType, GetMessageByIdResponseType} from "@entities/
   ↳ Message";
5 import {MessageInputStyle} from "../..styles";
6 import {MessageInputPropsType, MessageInputRefType} from "../..types";
7
8 type SendMessageType = Parameters<ChatSocketEmitType['send_message']>[
   ↳ number];
9 type EditMessageType = Parameters<ChatSocketEmitType['edit_message']>[
   ↳ number];
10 type ReplayMessageDataType = GetMessageByIdResponseType['id']
11
12 type OnSendMessageType<ReplayMessageDataType> = Parameters<typeof
   ↳ MessageInput<ReplayMessageDataType>>[number]['onSend']
13 type OnEditMessageType<ReplayMessageDataType> = Parameters<typeof
   ↳ MessageInput<ReplayMessageDataType>>[number]['onEdit']
14 export const MessageInput = forwardRef<MessageInputRefType,
   ↳ MessageInputPropsType>((props, ref) => {
15     const {
16         curChat,
17         sendMessage,
18         editMessage,
19     } = props
20     const inputServicesRef = useRef<MessageInputRefType<
   ↳ ReplayMessageDataType>>(null);
21
22     const onEditMessage: OnEditMessageType<ReplayMessageDataType> =
   ↳ async ({message}) => {
23         if ((message.text || message.attachment?.length) && message.data)
   ↳ {
24             const newMessage: EditMessageType = {
25                 id: message.data,
26                 text: message.text || '',
27                 attachments: message.attachment || [],
28             }
29             editMessage(newMessage)

```

					ПРИЛОЖЕНИЕ Г			
Изм.	Лист	№ докум.	Подп.	Дата				
Разраб.	Бондаренко С.В.				Разработка front-end Web-приложения – учебной среды с чатом и AI-анализом кода лабораторных работ		Лит.	Лист
Руковод.	Мельников А.Б.							89
Консул.								94
Н. контр.	Осипов О.В.						БГТУ им. В.Г. Шухова, ПВ-212	
Зав. Каф.	Поляков В.М.							

```

30         return true
31     }
32     return false
33 }
34 const onSendMessage: OnSendMessageType<ReplayMessageDataType> =
↪ async ({message, replayMessage}) => {
35     if ((message.text || message.attachment?.length)) {
36         // Генерируем uuid, потому что будем показывать сообщение
↪ сразу на фронте,
37         // не дожидаясь ответа от бэка по socket
38         const uuid = uuidv4();
39         const newMessage: SendMessageType = {
40             chat_id: curChat.id,
41             text: message.text || '',
42             attachments: message.attachment || [],
43             reply_id: replayMessage?.data || null,
44             local_id: uuid,
45         }
46         sendMessage(newMessage)
47         return true
48     }
49     return false
50 }
51
52 const onEdit: MessageInputRefType['onEdit'] = (message) => {
53     inputServicesRef.current?.editMessage({
54         text: message.text,
55         attachment: message.attachments,
56         data: message.id,
57     })
58 }
59
60 const onReplay: MessageInputRefType['onReplay'] = (message) => {
61     inputServicesRef.current?.replayMessage({
62         data: message.id,
63         text: message.text,
64         attachment: message.attachments,
65     })
66 }
67
68 useImperativeHandle(ref, () => {
69     return {
70         onEdit,
71         onReplay,
72     }
73 }, []);
74

```

					ПРИЛОЖЕНИЕ Г	Лист
Изм.	Лист	№ докум.	Подп.	Дата		90

```

75     return (
76         <div className={MessageInputStyle.MessageInput}>
77             <MessageInput<ReplayMessageDataType> onSend={onSendMessage}
78                 onEdit={onEditMessage}
79                 ref={inputServicesRef}/>
80         </div>
81     )
82 })
83
84
85 export const MessageFeed = forwardRef<MessageFeedRefType,
86     ↳ MessageFeedPropsType>((props, ref) => {
87     const {
88         curChat,
89         onReplayMessage,
90         onReadMessage,
91         messages,
92         onScrollTop,
93         onScrollBottom,
94         jumpToLastMessage,
95         jumpToMessage,
96         onEditMessage
97     } = props;
98
99     const {
100         userId,
101         SCROLL_ACCURACY,
102         listRef,
103         scrollToMessage,
104         scrollToBottom,
105         showScrollButton,
106         onScroll,
107         addFeedItemService,
108         isFirstInGroup,
109         isLastInGroup,
110         saveScrollPosition,
111         revertScrollPosition,
112         blinkMessage,
113
114         onChooseMessage,
115         onReply,
116         onEdit,
117         chatActionsServiceRef,
118     } = useMessageFeed(props)
119
120     useImperativeHandle(ref, () => {

```

					ПРИЛОЖЕНИЕ Г	Лист
						91
Изм.	Лист	№ докум.	Подп.	Дата		



```

120         return {
121             scrollToBottom,
122             scrollToMessage,
123             saveScrollPosition,
124             revertScrollPosition,
125             blinkMessage,
126         }
127     }, []);
128
129     if (!userId) {
130         return <LoaderPage/>
131     }
132
133     return (
134         <div className={MessageFeedStyle.MessageFeed}>
135             <ChatActionsWindow ref={chatActionsServiceRef} onEdit={
136 ↪ onEdit} onReplay={onReply}/>
137             <ScrollProvider className={MessageFeedStyle.feed}
138                 onScrollBottom={onScrollBottom}
139                 onScrollTop={onScrollTop}
140                 accuracy={SCROLL_ACCURACY}
141                 onScroll={onScroll}
142                 ref={listRef}>
143                 {messages.map((item, idx) => {
144                     const id = getMessageId(item);
145                     return <MessageFeedItem item={item}
146                         curChat={curChat}
147                         onReply={() =>
148 ↪ onReplayMessage(item)}
149                         onEdit={() =>
150 ↪ onEditMessage(item)}
151                         readMessage={() =>
152 ↪ onReadMessage(item)}
153                         jumpToMessage={
154 ↪ jumpToMessage}
155                         firstInGroup={
156 ↪ isFirstInGroup(idx)}
157                         lastInGroup={
158 ↪ isLastInGroup(idx)}
159                         onChooseMessage={() =>
160 ↪ onChooseMessage(item, {
161                             x: event.x,
162                             y: event.y,
163                         })}
164                         key={id}

```

					ПРИЛОЖЕНИЕ Г	Лист
Изм.	Лист	№ докум.	Подп.	Дата		92

```

157                                     ref={ (node) =>
↪ addFeedItemService(id, node)}/>
158                                     }
159                                 })
160                             </ScrollProvider>
161                             <div className={MessageFeedStyle.scrollButton}>
162                                 {showScrollButton && <ChatScrollButton counter={curChat.
↪ unread_messages || undefined}
163                                     onClick={
↪ jumpToLastMessage}/>}
164                             </div>
165                         </div>
166                     )
167 })
168
169 export const ChatWidget = forwardRef<ChatWidgetRefType,
↪ ChatWidgetPropsType>((props, ref) => {
170     const {
171         curChat,
172     } = props;
173
174     const {
175         messages,
176         sendMessage,
177         jumpToMessage,
178         addNewMessage,
179         changeMessage,
180         feedServicesRef,
181         jumpToLastMessage,
182         editMessage,
183         inputServicesRef,
184         onReplayMessage,
185         onReadMessage,
186         onEditMessage,
187         onScrollTop,
188         onScrollBottom,
189         testSkip,
190     } = usesChatWidget(props);
191
192     useImperativeHandle(ref, () => {
193         return {
194             addNewMessage,
195             changeMessage,
196         }
197     }, []);
198
199     if (messages === undefined) {

```

					ПРИЛОЖЕНИЕ Г	Лист
						93
Изм.	Лист	№ докум.	Подп.	Дата		

```

200     return <LoaderPage/>
201 }
202
203 if (messages === null) {
204     return 'Error'
205 }
206
207 return (
208     <div className={sChatWidgetStyle.sChatWidget}>
209         <div className={sChatWidgetStyle.testSkip}>{testSkip}</div>
210         <div className={sChatWidgetStyle.messageList}>
211             <MessageFeed messages={messages}
212                 curChat={curChat}
213                 jumpToMessage={jumpToMessage}
214                 jumpToLastMessage={jumpToLastMessage}
215                 onReadMessage={onReadMessage}
216                 onEditMessage={onEditMessage}
217                 onScrollTop={onScrollTop}
218                 onScrollBottom={onScrollBottom}
219                 onReplayMessage={onReplayMessage}
220                 ref={feedServicesRef}/>
221         </div>
222         <MessageInput curChat={curChat}
223             editMessage={editMessage}
224             sendMessage={sendMessage}
225             ref={inputServicesRef}/>
226     </div>
227 )
228 })

```

					ПРИЛОЖЕНИЕ Г	Лист
						94
Изм.	Лист	№ докум.	Подп.	Дата		