

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Бизнес-проект «Серверная часть
информационной платформы трудоустройства
и стажировки студентов в ИТ-компаниях»**

Студент

(подпись, дата)

Годов Дмитрий Алексеевич
(ФИО полностью)

Руководитель

(подпись, дата)

д.т.н., профессор Козлов А.Ю.
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.
(фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«26» декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Годов Дмитрий Алексеевич гр. 22ВП1 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Бизнес-проект «Серверная часть информационной платформы трудоустройства и стажировки студентов в ИТ-компаниях»

Тема утверждена приказом ПГУ №1583/о от "30" декабря 2025 г.

3. Руководитель работы д.т.н., профессор кафедры «МО и ПЭВМ» Козлов А.Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: серверная часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях для объединения студентов, работодателей и преподавателей в едином цифровом пространстве

2) Среда выполнения: операционная система Windows 10

Языковые средства разработки: C#, SQL

Инструментальные средства: Visual Studio 2022, pgAdmin 4, Visual Paradigm

Технологии разработки: REST API, ООП

3) Основные функции приложения:

а) регистрация и авторизация пользователей;

б) просмотр и редактирование пользовательского профиля;

в) создание, редактирование и удаление вакансий работодателем;

г) просмотр, поиск и фильтрация вакансий студентом;

д) создание, редактирование, и удаление резюме студентом;

е) формирование рекомендованных вакансий для студента на основе резюме;

ж) формирование рекомендованных резюме для работодателя на основе вакансий;

з) создание откликов на вакансии;

и) просмотр и изменение статусов откликов работодателем;

- к) просмотр студентом истории своих откликов;
- л) управление учебными группами преподавателем;
- м) обмен сообщениями между студентами и работодателями в режиме реального времени;
- н) сопровождение производственных практик;
- о) выход пользователя из системы.

5. Перечень подлежащих разработке вопросов

- 1) Анализ предметной области и требования к системе;
- 2) Проектирование программных средств;
- 3) Реализация программных средств;
- 4) Тестирование программных средств;
- 5) Оформление пояснительной записки;

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____ 26.12.2025 г.
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025 г.
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 180 листов, 36 рисунков, 104 таблицы, 21 использованный источник и 5 приложений.

ИНФОРМАЦИОННАЯ ПЛАТФОРМА, WEB-ПРИЛОЖЕНИЕ, СЕРВЕРНАЯ ЧАСТЬ, REST API, C#, ASP.NET, SIGNALR, ENTITY FRAMEWORK, POSTGRESQL, БРОКЕР СООБЩЕНИЙ, APACHE KAFKA, DOCKER, GIT.

Объект исследования – серверная часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.

Цель выпускной квалификационной работы – разработать серверную часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях, объединяющую студентов, работодателей в едином цифровом пространстве.

Разработка серверной части приложения проводилась на языке программирования C# с использованием фреймворка ASP.NET в среде программирования Visual Studio 2022. В качестве СУБД использовалась PostgreSQL.

Среда исполнения – Windows 10.

В ходе выполнения выпускной квалификационной работы была разработана серверная часть информационной платформы трудоустройства и стажировки студентов в ИТ-компаниях. Все требования к разработке приложения соблюдены.

ПГУ 09.03.04 – 08ВР221.06 ПЗ

					ПГУ 09.03.04 – 08ВР221.06 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата							
Разраб.		Годов Д. А.			Бизнес-проект «Серверная часть информационной платформы трудоустройства и стажировки студентов в ИТ-компаниях»	Лит.		Лист		Листов	
Провер.		Козлов А. Ю.						4		180	
Реценз.						Группа 22ВП1					
Н. Контр.		Такташкин Д. В.									
Утверд.		Козлов А. Ю.									

Содержание

Введение.....	9
1 Стартап часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.....	11
1.1 Описание существующей проблемы на рынке	11
1.2 Описание решения проблемы и продуктового решения	12
1.3 Объём, структура и развитие рынка	13
1.4 Анализ конкурентов.....	14
1.5 Бизнес-модель проекта	16
1.6 Команда стартап-проекта	17
1.7 Описание финансовой модели платформы и необходимости привлечения инвестиций.....	19
1.8 Описание развития бизнес-проекта	20
1.9 Описание рисков	21
2 Анализ предметной области и постановка задачи на разработку информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.....	23
2.1 Анализ предметной области	23
2.2 Анализ программных аналогов платформ трудоустройства.....	25
2.2.1 Анализ общих платформ поиска работы.....	25
2.2.2 Анализ специализированных платформ поиска работы в ИТ-сфере...	25
2.2.3 Анализ университетских информационных систем.....	26
2.2.4 Результаты анализа	26
2.3 Постановка задачи на разработку информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях	28
2.4 Анализ требований к разрабатываемой информационной платформе трудоустройства и стажировок студентов в ИТ-компаниях	30
2.4.1 Анализ функциональных требований.....	30

2.4.2 Анализ нефункциональных требований.....	40
2.5 Выбор инструментальных средств.....	42
2.6 Планирование проекта и оценка бюджета	44
3 Проектирование серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях	47
3.1 Проектирование архитектуры серверной части	47
3.2 Проектирование базы данных	49
3.2.1 Разработка концептуальной модели базы данных	49
3.2.2 Разработка логической модели базы данных.....	52
3.3 Проектирование серверной части	54
3.3.1 Проектирование модуля «Авторизация и пользователи»	55
3.3.2 Проектирование модуля «Резюме и навыки»	57
3.3.3 Проектирование модуля «Вакансии и отклики»	60
3.3.4 Проектирование модуля «Встроенная коммуникация».....	63
3.3.5 Проектирование модуля «Учебный процесс и университеты»	66
3.3.6 Проектирование модуля «Производственные практики»	69
3.3.7 Проектирование сервиса рассылки уведомлений	72
3.4 Проектирование алгоритмов.....	73
3.4.1 Проектирование алгоритма вступления в учебную группу	73
3.4.2 Проектирование алгоритма трудоустройства.....	75
3.4.3 Проектирование алгоритма прохождения практики.....	77
4 Реализация серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.....	80
4.1 Развертывание базы данных	80
4.1.1 Разработка физической модели базы данных	80
4.1.2 Развертывания базы данных в среде pgAdmin 4.....	82
4.2 Реализация взаимодействия с базой данных.....	84
4.2.1 Реализация запросов к базе данных	84
4.2.2 Наполнение базы данных тестовыми данными.....	85
4.3 Реализация инфраструктуры и политик безопасности	88

4.3.1 Аутентификация и управление доступом	88
4.3.2 Реализация файловой подсистемы	90
4.3.3 Интеграция брокера сообщений.....	92
4.4 Реализация бизнес-логики.....	94
4.4.1 Реализация встроенных чатов	94
4.4.2 Реализация алгоритма рекомендаций вакансий и резюме	96
4.4.3 Реализация сопровождения жизненного цикла откликов	98
4.4.4 Реализация сопровождения жизненного цикла производственных практик	101
4.5 Реализация сервиса рассылки уведомлений	104
4.6 Контейнеризация и развертывание информационной платформы.....	107
5 Тестирование серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях	109
5.1 Ручное функциональное тестирование методом «белого ящика» для процесса трудоустройства.....	109
5.1.1 Покрытие операторов	111
5.1.2 Покрытие условий	111
5.1.3 Покрытие маршрутов	112
5.2 Модульное тестирование процесса трудоустройства	113
5.3 Оценка качества кода.....	116
5.3.1 Размер и сложность кода.....	116
5.3.2 Метрика Холстеда.....	117
Заключение	119
Список использованных источников	121
Приложение А (рекомендуемое) Спецификация отношений логической схемы базы данных	123
Приложение Б (рекомендуемое) Скрипты (исходные коды) для создания объектов базы данных.....	136

Приложение В (рекомендуемое) Спецификация API серверного приложения информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.....	146
Приложение Г (обязательное) Код серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.....	166
Приложение Д (рекомендуемое) Результаты тестирования серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.....	177

Введение

В условиях стремительного развития ИТ-отрасли и цифровой экономики вопрос трудоустройства и прохождения стажировок студентами становится всё более актуальным. Рынок труда предъявляет к начинающим специалистам высокие требования. Одних теоретических знаний уже недостаточно – работодатели ожидают наличия практических навыков и реального опыта участия в проектах. Также требуется понимание командной разработки и умение быстро адаптироваться к рабочим процессам. При этом многие студенты сталкиваются с трудностями при поиске подходящих вакансий и стажировок. Особенно остро эта проблема ощущается при отсутствии опыта работы и профессиональных связей.

Популярные платформы для поиска работы ориентированы на широкий круг соискателей и работодателей. В результате студентам бывает сложно найти предложения, соответствующие их уровню подготовки, образовательному направлению и текущим профессиональным навыкам. Их резюме попросту теряются на фоне резюме более опытных кандидатов. Кроме того, взаимодействие между университетами, студентами и потенциальными работодателями чаще всего организовано разрозненно. Информация о стажировках может передаваться через отдельные объявления, почтовые рассылки, социальные сети или личные контакты преподавателей. Такой подход снижает эффективность коммуникации и создает дополнительную нагрузку на преподавателей.

Целью выпускной квалификационной работы является разработка серверной части информационной платформы, ориентированной на университетскую среду. Платформа объединяет студентов, преподавателей и работодателей в едином цифровом пространстве. Студенты могут размещать резюме, просматривать вакансии, откликаться на них, общаться с работодателями и записываться на практики. Работодатели получают возможность размещать вакансии, просматривать резюме студентов и искать

подходящих кандидатов. Преподаватели могут создавать учебные группы и отслеживать активность студентов. Руководители практик получают доступ к информации о компаниях, в которых студенты проходят практику, а также к связанной с практикой отчётности.

Особенностью разрабатываемой платформы является её ориентация на взаимодействие между университетом, студентами и ИТ-компаниями. Благодаря этому система может использоваться не только для поиска вакансий и стажировок, но и для сопровождения образовательного процесса, связанного с практической подготовкой студентов.

Архитектура платформы реализуется в виде многоуровневого монолитного приложения с разделением логики представления, бизнес-логики и слоя доступа к данным [1]. Уровень представления реализован в виде отдельного веб-приложения, через которое пользователи взаимодействуют с системой. Такой подход упрощает разработку, тестирование и дальнейшее сопровождение системы, а также позволяет при необходимости развивать пользовательский интерфейс независимо от внутренней логики приложения. База данных разворачивается отдельно от основного приложения, что делает конфигурацию более гибкой и позволяет при необходимости масштабировать инфраструктуру.

1 Стартап часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

1.1 Описание существующей проблемы на рынке

Современный ИТ-рынок труда предъявляет высокие требования к молодым специалистам. От кандидатов ожидают не только сухой теории. Работодателям нужны практические навыки и реальный опыт командной разработки. На старте карьеры большинство студентов неизбежно испытывает трудности с поиском первой работы ввиду отсутствия опыта и профессиональных связей.

На традиционных сервисах поиска работы, ориентированных на широкий круг соискателей, профили студентов нередко теряются на фоне резюме более опытных кандидатов. В итоге начинающим специалистам становится крайне сложно найти позиции под их уровень подготовки.

Основным барьером решения данной проблемы является то, что высшие учебные заведения крайне неэффективно задействуются в процессе трудоустройства своих студентов. Информация о доступных вакансиях и стажировках распространяется через объявления, личные контакты преподавателей, почтовые рассылки или публикации в социальных сетях.

В совокупности это порождает сразу несколько проблем:

- для студентов отсутствует единый источник получения информации по релевантным предложениям ИТ-компаний;
- компании, со своей стороны, лишены удобного канала для привлечения молодых специалистов;
- преподавателям приходится тратить личное время на размещение информации и подбор кандидатов.

Таким образом, на рынке наблюдается дефицит специализированных решений, которые могли бы интегрировать университеты прямо в процесс трудоустройства. Отсутствие единого пространства для взаимодействия

студентов, преподавателей и работодателей усложняет и замедляет процесс профессионального становления студентов ИТ-направлений.

1.2 Описание решения проблемы и продуктового решения

В качестве решения выявленной проблемы предлагается разработать информационную платформу для трудоустройства и стажировок студентов в ИТ-компаниях. Главная ценность продукта заключается в интеграции университетской среды напрямую в процесс трудоустройства. Это позволит автоматизировать взаимодействие пользователей.

Для студентов платформа создает комфортную стартовую площадку. Здесь представлены соискатели сопоставимого уровня квалификации. Пользователи могут публиковать резюме и откликаться на вакансии. Более того, платформа позволяет напрямую общаться с работодателями через встроенные чаты, а также централизованно проходить производственные практики.

Бизнес получает удобный инструментарий для найма молодых специалистов непосредственно из университетской среды. Работодатель может не просто размещать вакансии, но и искать кандидатов с помощью удобной системы фильтрации и приглашать их на собеседования. Что немаловажно, компании могут организовывать производственные практики, получая отчеты и решения прямо внутри платформы.

Образовательным учреждениям платформа предоставляет инструменты автоматизации и контроля. Преподаватели формируют на платформе учебные группы. Помимо этого, система снимает с преподавателей нагрузку по рассылке вакансий и предоставляет удобные инструменты для отслеживания активности студентов, проверки отчётности по производственным практикам и выставления по ним итоговых оценок.

Ключевые особенности платформы:

- система рекомендаций: алгоритм сопоставления резюме и вакансий на основе ключевых параметров;

- встроенная коммуникация: платформа предоставляет чаты реального времени для обсуждения вакансий;
- сквозное сопровождение производственных практик: полный перевод жизненного цикла прохождения практики в цифровую среду – от подачи заявки на предприятие до загрузки отчётности и получения оценки;
- защита от сторонних пользователей: доступ к полному функционалу соискателя открывается исключительно после подтверждения преподавателем принадлежности студента к учебной группе.

Исходя из этого, можно утверждать: разрабатываемая информационная платформа не просто очередная доска объявлений, а комплексный продукт, который позволяет эффективно организовать процесс трудоустройства студентов.

1.3 Объём, структура и развитие рынка

Рынок разрабатываемой информационной платформы находится на пересечении двух крупных и активно развивающихся ИТ-сегментов: HRTech (технологии управления персоналом) и EdTech (образовательные технологии). Главным драйвером развития этого рынка является стабильно высокий дефицит квалифицированных ИТ-кадров. Рынок специалистов уровня Middle и Senior перегрет. Поэтому ИТ-компании всё чаще переходят к стратегии «выращивания» кадров со студенческой скамьи. Это многократно повышает актуальность цифровых систем для поиска и найма молодых талантов.

Структуру целевого рынка можно разделить на три ключевых сегмента, которые полностью соответствуют ролевой модели разрабатываемой платформы:

- сегмент B2C охватывает студентов, обучающихся по направлениям, связанным с информационными технологиями;
- сегмент B2B – это предприятия, которым требуются младшие специалисты и стажеры;

– сегмент B2G/B2B – университеты, заинтересованные в цифровизации учебного процесса, автоматизации контроля производственных практик и повышении процента успешного трудоустройства своих выпускников.

Оценивать объем рынка целесообразнее всего по классической стартап-модели (TAM, SAM, SOM):

- общий объем рынка (TAM) включает в себя весь российский рынок онлайн-рекрутмента и найма начинающих специалистов;
- доступный объем рынка (SAM): нишевый сегмент рынка найма, сфокусированный исключительно на сфере информационных технологий и прямом взаимодействии с университетами;
- реально достижимый объем рынка (SOM): ИТ-компании, образовательные учреждения и студенты конкретного региона на начальном этапе запуска проекта, с потенциалом последующего масштабирования на другие регионы.

С точки зрения развития рынка наблюдается явный тренд на узкую специализацию. Универсальные платформы поиска работы больше не справляются со специфическими запросами академической среды и ИТ-отрасли. Будущее рынка найма начинающих специалистов лежит в создании закрытых, верифицированных экосистем.

Вектор развития смещается к тому, чтобы взаимодействие работодателя и кандидата начиналось задолго до получения диплома – на этапе распределения учебных и производственных практик. Разрабатываемая платформа полностью соответствует этому вектору. Продукт занимает свободную нишу специализированного связующего звена между вузами и ИТ-бизнесом.

1.4 Анализ конкурентов

Для проведения конкурентного анализа существующие на рынке решения целесообразно разделить на три основные категории. Это массовые платформы поиска работы, профильные ИТ-сервисы и внутренние образовательные

системы. Каждая из групп решает лишь часть задач рассматриваемой предметной области.

К общим платформам относятся такие лидеры масс-маркета, как HeadHunter, SuperJob и портал «Работа России». Данные продукты обладают обширной базой работодателей, однако они ориентированы на массовый сегмент соискателей и совершенно не учитывают специфику университетской среды. В них нет роли преподавателя. Соответственно, отсутствуют инструменты для администрирования учебных групп и функционал для контроля прохождения производственных практик. Более того, на таких площадках профили студентов теряются, так как начинающим специалистам сложно конкурировать за рабочие места с опытными кандидатами.

Типичным представителем сегмента профильных ИТ-сервисов является «Хабр Карьера». Конкуренты из этой группы частично решают проблему релевантности. Они предлагают вакансии исключительно в сфере информационных технологий. Однако, профильные платформы фокусируются строго на классическом процессе найма. Они полностью игнорируют академическую составляющую старта карьеры и не предоставляют образовательным учреждениям инструментов для организации стажировок или отслеживания активности обучающихся.

В качестве примера внутренней образовательной системы выступает модуль «ВУЗ + Работодатель» на базе электронной информационно-образовательной среды ПГУ. Несомненное преимущество конкурентов из этого сегмента – изначальная интеграции в структуру университета. Однако стоит отметить, что подобные модули обладают серьезными ограничениями во взаимодействии с реальным бизнесом. Коммуникация с ИТ-компаниями в них, как правило, осуществляется через личные контакты за пределами платформы из-за отсутствия встроенных чатов. К тому же, весь функционал часто сводится лишь к помощи в подборе места стажировки. Полноценное цифровое сопровождение практики и контроль загружаемой отчетности здесь не обеспечивается.

Проведенный анализ показывает отсутствие прямых конкурентов, комплексно закрывающих потребности студентов, вузов и работодателей. Разрабатываемая платформа отстраивается от аналогов благодаря гибридной бизнес-модели. Она объединяет сильные стороны классических сервисов трудоустройства с инструментами образовательных систем. Возникает формирует уникальное закрытое цифровое пространство, функционал которого на данный момент недоступен ни одной из категорий текущих конкурентов.

1.5 Бизнес-модель проекта

Информационная платформа реализуется в формате веб-сервиса. Такой подход избавляет конечных пользователей от необходимости устанавливать дополнительное программное обеспечение, гарантируя доступ к системе с любого устройства через интернет-браузер.

Финансовая модель платформы базируется на B2B-монетизации. Для образовательных учреждений и студентов использование сервиса является полностью бесплатным, что обеспечивает быстрый органический рост базы соискателей. Основным же источником дохода выступает продажа подписок работодателям.

Тарифная сетка для бизнеса дифференцируется в зависимости от масштаба организации:

- базовый тариф – предназначен для небольших ИТ-команд и включает лимитированное количество публикуемых вакансий и доступов к контактам студентов;
- расширенный тариф – ориентирован на средний и крупный бизнес, предоставляя безлимитное размещение вакансий, приоритетную выдачу в поиске и доступ к расширенной статистике.

Дополнительным вектором монетизации выступает внедрение индивидуальных B2B-решений. Сюда входит платная интеграция платформы с внутренними кадровыми системами заказчиков и разработка

специализированных аналитических модулей под конкретные бизнес-процессы компании.

В соответствии с заложенной ролевой моделью аудитория делится на три сегмента:

- студенты ИТ-направлений и начинающие специалисты;
- ИТ-компании, HR-специалисты и рекрутеры;
- высшие учебные заведения, кураторы учебных групп и руководители практик.

На старте основной упор делается на стратегическое партнерство с университетами. Это позволяет одномоментно сформировать базу соискателей. Чтобы привлечь работодателей используются B2B-продажи через профильные ИТ-сообщества, контекстную рекламу и предоставление бесплатного пробного доступа к системе.

Ключевое преимущество бизнес-модели заключается в узкой специализации на ИТ-отрасли и прямом взаимодействии с академической средой. Заложенная архитектура и уже реализованный базовый функционал, включающий систему рекомендаций, встроенные чаты и модуль сопровождения практик, обеспечивают высокую масштабируемость. В дальнейшем проект планируется выходить на новые регионы, подключать новые образовательные учреждения и расширять спектр платных B2B-услуг. Подводя итог, проект формирует экономически устойчивую модель со стабильным потоком рекуррентных платежей.

1.6 Команда стартап-проекта

Реализация стартап-проекта осуществляется командой из двух специалистов, компетенции которых полностью покрывают потребности полного цикла разработки программного продукта. Идея создания информационной платформы возникла из личного опыта и анализа рынка HR-технологий, показавшего острую нехватку профильных сервисов для начинающих ИТ-специалистов. Оба участника команды заканчивают обучение

в Пензенском государственном университете по направлению подготовки «Программная инженерия». За время учебы они приобрели устойчивый опыт совместной работы над информационными системами и веб-сервисами.

Годов Дмитрий Алексеевич выступает в роли backend-разработчика и системного архитектора проекта. В зону его ответственности входит создание серверной части платформы: проектирование программной архитектуры, написание бизнес-логики, реализация REST API, а также настройка инфраструктуры и обеспечение информационной безопасности. Технологический стек специалиста включает язык программирования C# с использованием ASP.NET, СУБД PostgreSQL, инструменты Entity Framework, а также средства контейнеризации Docker и реализацию механизмов JWT-аутентификации.

Поршнева Павел Александрович занимает позицию frontend-разработчика и UI/UX-дизайнера. Он отвечает за визуальную составляющую продукта, разработку клиентского веб-приложения и обеспечение комфортного пользовательского опыта. В его задачи входит проектирование адаптивного дизайна, верстка личных кабинетов для соискателей и компаний, страниц авторизации, а также бесшовная интеграция клиентской части с разработанным серверным API. В своей работе специалист опирается на современные фронтенд-технологии и библиотеки, включая React, Vite, Axios, React Router и Tailwind CSS.

Подобное распределение обязанностей обеспечивает комплексный подход к разработке и позволяет эффективно управлять проектом. Четкое разграничение зон ответственности на серверную логику, клиентские интерфейсы и настройку инфраструктуры гарантирует высокую скорость создания цифрового продукта. Имеющиеся компетенции команды полностью соответствуют заявленным требованиям и позволяют в перспективе уверенно масштабировать систему, расширяя ее функциональные возможности.

1.7 Описание финансовой модели платформы и необходимости привлечения инвестиций

Финансовая модель информационной платформы базируется на предоставлении программного обеспечения как услуги. Основным источником монетизации выступают регулярные подписочные платежи от ИТ-компаний, получающих доступ к функционалу поиска и найма студентов.

Тарифная сетка для работодателей включает три уровня:

- базовый тариф – для небольших команд с лимитированным числом вакансий;
- стандартный тариф – для средних ИТ-компаний с расширенными лимитами;
- расширенный тариф – для крупного бизнеса с приоритетным размещением и персональной поддержкой.

Ежемесячная стоимость подписки колеблется от 3 000 до 15 000 рублей в зависимости от выбранного плана. При подключении хотя бы 20 компаний на стандартный тариф базовая ежемесячная прогнозируемая выручка составит 200 000 рублей. Дополнительным источником прибыли выступает кастомизация платформы и её интеграция с внутренними HR-системами конкретных заказчиков.

Ежемесячные расходы проекта включают:

- оплату облачной инфраструктуры и SSL-сертификатов (20000 – 40000 рублей);
- маркетинговое продвижение платформы (40000 – 60000 рублей);
- затраты на техническое сопровождение системы.

При совокупных ежемесячных расходах около 100000 рублей точка безубыточности проекта достигается при привлечении 10–15 компаний на стандартный тарифный план.

Прямо сейчас проект не требует колоссальных вливаний в backend-разработку, так как ключевые интеллектуальные модули уже спроектированы и

реализованы. Однако для форсированного вывода продукта на рынок требуется привлечение инвестиций в размере от 500000 до 700000 рублей.

Привлеченные средства планируется направить на следующие цели:

- полномасштабная маркетинговая кампания для привлечения пользователей;
- масштабирование серверной инфраструктуры для работы с высокими нагрузками;
- формирование отдела продаж и технической поддержки.

Таким образом, предложенная финансовая модель является экономически устойчивой. Низкий порог входа для работодателей и использование подписочной системы гарантируют стабильный рекуррентный доход. Это обеспечит проекту высокий потенциал на рынке цифровых HR-сервисов.

1.8 Описание развития бизнес-проекта

Стратегия развития информационной платформы строится на поэтапном масштабировании продукта и расширении его функциональных возможностей. Дорожная карта проекта разделена на три ключевых этапа.

Пилотное внедрение целесообразно провести на базе профильных факультетов Пензенского государственного университета (ПГУ). На этом этапе приоритетом выступает сбор обратной связи от первых пользователей, а также адаптация бизнес-процессов под реальные нужды вуза.

После успешного завершения пилотного тестирования и подтверждения жизнеспособности бизнес-модели планируется переход к активной коммерциализации. Главная цель этапа – продажа платных подписок B2B-клиентам и подключение к платформе других образовательных учреждений. В техническом плане здесь предполагается оптимизация развернутой базы данных под возросшие нагрузки, а также разработка адаптивной мобильной версии платформы для повышения вовлеченности соискателей.

Долгосрочная стратегия нацелена на выход платформы за пределы изначального региона. На данном этапе планируется реализация модулей для

глубокой B2B/B2G-интеграции. Платформа будет напрямую связываться с внутренними образовательными порталами других вузов и корпоративными HR-системами крупных предприятий. Кроме того, функционал системы будет дополнен инструментами продвинутой аналитики, позволяющими вузам отслеживать карьерные траектории своих выпускников.

Предложенный план развития позволяет проекту плавно перейти от стадии технической разработки к полноценному коммерческому продукту. Методичное наращивание базы пользователей обеспечит стабильный рост финансовых показателей.

1.9 Описание рисков

Реализация любого стартап-проекта сопряжена с рядом неопределенностей. Для обеспечения устойчивого развития информационной платформы был проведен анализ потенциальных рисков и разработаны стратегии их минимизации.

Рыночные и коммерческие риски преимущественно связаны с классической для многосторонних платформ проблемой привлечения аудитории. Работодателям невыгодно приобретать подписку при недостаточном количестве соискателей, а студенты не смогут использовать систему без участия вуза из-за требования обязательного подтверждения куратором. Для минимизации этого риска бизнес-модель предусматривает полностью бесплатный доступ для образовательных учреждений и студентов. Первоначальный фокус на пилотном внедрении в рамках одного университета позволит сформировать достаточную базу кандидатов до старта активных продаж.

Технические угрозы обусловлены вероятностью сбоев в работе серверной части при пиковых нагрузках, например, во время массовой подачи заявок на практики студентами. Данная проблема решается непосредственно на архитектурном уровне. Использование технологии контейнеризации позволяет быстро разворачивать и масштабировать систему. Кроме того, разделение

архитектуры на основное монолитное приложение и независимый сервис уведомлений через брокер сообщений защищает центральный сервер от перегрузок побочными задачами, а изолированное развертывание базы данных повышает общую отказоустойчивость.

Риски информационной безопасности связаны с возможной утечкой персональных данных пользователей, таких как резюме или переписки во встроенных чатах. Это может повлечь за собой серьезные репутационные и финансовые издержки. В качестве превентивных мер в платформу внедрены надежные механизмы аутентификации на базе JWT-токенов и строгие политики ролевого доступа. Эти решения гарантируют, что взаимодействие с данными доступно исключительно авторизованным и верифицированным участникам системы в рамках их прав.

Финансовые риски связаны с вероятностью возникновения кассовых разрывов или недостижения точки безубыточности из-за низкой конверсии работодателей в платящих клиентов. Для снижения финансовой напряженности применяется гибкая тарифная сетка, которая существенно уменьшает порог входа для небольших ИТ-команд. Дополнительная стабильность бюджета обеспечивается за счет диверсификации доходов, включающей оказание ИТ-компаниям платных услуг по интеграции платформы с их внутренними корпоративными системами.

Таким образом, системный подход к управлению рисками, заложенный как в бизнес-стратегию, так и в программную архитектуру серверной части, позволяет свести к минимуму вероятность возникновения критических ситуаций и гарантирует стабильную работу продукта.

2 Анализ предметной области и постановка задачи на разработку информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

2.1 Анализ предметной области

Прохождение практик и стажировок является важным этапом профессиональной подготовки студентов ИТ-направлений. Для студентов это возможность применить полученные в университете знания и навыки и приобрести первый практический опыт. Для работодателей практики и стажировки позволяют познакомиться с начинающими специалистами и оценить их уровень подготовки в процессе выполнения реальных рабочих задач, а также привлечь наиболее перспективных кандидатов в свою компанию.

В рамках данной работы разрабатывается серверная части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях. Она предназначена для объединения студентов, работодателей и преподавателей в едином цифровом пространстве. Ориентация разрабатываемой платформы на университетскую среду позволяет упростить студентам поиск вакансий, стажировок и мест прохождения практики, снизить нагрузку на преподавателей при сопровождении студентов и предоставить работодателям удобный способ поиска начинающих специалистов.

В рассматриваемой предметной области можно выделить три основных роли: студент, работодатель и преподаватель. Каждая из этих ролей обладает собственными правами и набором функций. Общим функционалом для каждой роли является управление личным кабинетом.

Полный доступ к функционалу платформы студенты получают после вступления в свою учебную группу. Для этого куратору необходимо зарегистрироваться на платформе, заполнить сведения о своей учебной группе и пригласить туда студентов. После вступления в учебную группу студентам становятся доступны следующие возможности:

- управление своими резюме;

- поиск и просмотр вакансий;
- создание откликов;
- запись на практики и их прохождение;
- общение с работодателями.

Основной функционал работодателей:

- управление своими вакансиями;
- поиск и просмотр резюме;
- приглашение кандидатов на собеседования;
- размещение предложений по практике, набор студентов и получение

отчётности по её прохождению.

В рамках системы преподаватель может выполнять функции куратора учебной группы, руководителя практики либо совмещать обе эти функции. Как куратор он создаёт учебные группы, приглашает в них студентов и отслеживает их активность на платформе. Как руководитель практики преподаватель получает доступ к группам, закреплённым за ним для сопровождения практики, просматривает информацию о компаниях, в которых студенты проходят практику, контролирует предоставление отчётности и выставляет оценки. Такой подход позволяет использовать одну роль преподавателя для разных сценариев взаимодействия со студентами без введения дополнительных ролей в системе.

Основной функционал преподавателей:

- создание учебных групп;
- просмотр активности студентов своих групп;
- просмотр отчетности студентов на практике, выставление оценок;
- просмотр информации о компаниях, в которых студенты проходят

практику.

Таким образом разрабатываемая платформа ориентирована на организацию взаимодействия между тремя группами пользователей: студентами, работодателями и преподавателями.

2.2 Анализ программных аналогов платформ трудоустройства

Рассматриваемые платформы для поиска работы можно разделить на три категории:

- общие платформы поиска работы;
- специализированные платформы поиска работы в ИТ-сфере;
- университетские информационные системы.

Ниже приведен разбор решений, относящихся к каждой из категорий, и их сильных и слабых сторон.

2.2.1 Анализ общих платформ поиска работы

К данной категории относятся такие сервисы, как HeadHunter, SuperJob и портал «Работа России». Они предоставляют весь необходимый функционал для размещения резюме и поиска работы. Преимуществом этих сервисов является большая база вакансий и работодателей с рейтингами и отзывами. Однако данные платформы ориентированы на широкий круг пользователей и не учитывают специфику университетской среды. В них отсутствует роль преподавателя, а также не предусмотрены инструменты для работы с учебными группами и контроля практик. Кроме того, резюме студентов могут теряться среди резюме более опытных кандидатов.

2.2.2 Анализ специализированных платформ поиска работы в ИТ-сфере

Платформы из данной категории, например «Хабр Карьера» ориентированы на поиск работы именно в ИТ-сфере. Это делает поиск более релевантным для студентов, обучающихся по направлениям, связанным с информационными технологиями. Однако платформы данной категории, как и общие платформы поиска работы, не предусматривают роли преподавателя, работы с учебными группами и сопровождения практик.

2.2.3 Анализ университетских информационных систем

Отдельную категорию составляет внутренние университетские системы. Они используются для организации учебного процесса: хранения информации о студентах, расписании, успеваемости, учебных группах и документах. Их главным преимуществом является непосредственная интеграция со структурой университета.

В качестве примера рассмотрим модуль «ВУЗ + Работодатель» на базе электронной информационно-образовательной среды (ЭИОС) ПГУ. Система позволяет студентам создавать резюме, откликаться на вакансии и подбирать компании для стажировок и прохождения практики. Несмотря на связь с университетской средой, модуль имеет ограничения. Взаимодействие с работодателями в нём организовано в основном через контактные данные компании, например электронную почту или телефон, без встроенных чатов внутри платформы. Кроме того, система помогает студентам находить вакансии, стажировки и места практики, но не предназначена для полноценного сопровождения производственной практики. В ней отсутствуют инструменты для просмотра активности студентов и отслеживания прохождения практики в конкретных компаниях.

2.2.4 Результаты анализа

В результате анализа программных аналогов были определены критерии оценивания, а также преимущества и недостатки представителей различных категорий платформ трудоустройства. Сравнение выполнялось как по базовому функционалу, необходимому для комфортного использования платформы студентами и работодателями, так и по специализированным возможностям, связанным с университетской средой. Сравнение разрабатываемой информационной платформы с представителями платформ рассмотренных категорий приведено в таблице 1.

Таблица 1 – Сравнение разрабатываемой платформы с программными аналогами

Критерий	Общие платформы поиска работы	Специализированные платформы поиска работы в ИТ-сфере	Университетские системы	Разрабатываемая платформа
Размещение и управление резюме	Да	Да	Да	Да
Поиск и просмотр вакансий	Да	Да	Да	Да
Встроенная коммуникация между студентами и компаниями	Да	Да	Нет	Да
Ориентация на ИТ-сферу	Частично	Да	Частично	Да
Участие преподавателей	Нет	Нет	Да	Да
Работа с учебными группами	Нет	Нет	Да	Да
Просмотр активности студентов	Нет	Нет	Нет	Да
Подбор места практики	Нет	Нет	Да	Да
Запись на практику внутри платформы	Нет	Нет	Нет	Да
Просмотр информации о местах прохождения практик студентами	Нет	Нет	Нет	Да
Контроль отчётности по практике	Нет	Нет	Частично	Да

По результатам сравнения можно сделать вывод, что существующие программные аналоги решают только часть задач рассматриваемой предметной области. Общие и специализированные платформы поиска работы предоставляют необходимые инструменты для размещения резюме и вакансий, однако не учитывают особенности университетской среды. Университетские

системы, напротив, связаны со структурой образовательной организации, но не всегда обеспечивают полноценное взаимодействие студентов с работодателями и сопровождение производственных практик. Разрабатываемая платформа объединяет возможности поиска вакансий, управления резюме, взаимодействия с работодателями, работы с учебными группами и контроля прохождения практики, что отличает её от рассмотренных аналогов.

2.3 Постановка задачи на разработку информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

На основе анализа предметной области и программных аналогов была выявлена необходимость разработки информационной платформы, предназначенной для организации взаимодействия студентов, работодателей и преподавателей в едином цифровом пространстве.

Целью работы является разработка серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях. В рамках системы предусматривается три основные роли: студенты, работодатели и преподаватели. Ниже описаны функциональные требования для каждой из ролей.

Базовый функционал, доступный всем пользователям:

- регистрация и авторизация в системе;
- просмотр и изменение данных личного кабинета.

Функционал студентов:

- размещение, редактирование, удаление и просмотр своих резюме;
- поиск и просмотр вакансий;
- просмотр профилей компаний;
- история просмотров вакансий;
- статистика просмотров резюме;
- сохранение избранных вакансий;
- создание откликов на вакансии;
- общение с работодателями во встроенных чатах;

- просмотр своих откликов;
- просмотр предложений по практике;
- запись на практику;
- загрузка отчётов и выполненных заданий по практике.

Функционал работодателей:

- размещение, редактирование, удаление и просмотр своих вакансий;
- поиск и просмотр резюме;
- просмотр профилей студентов;
- история просмотров резюме;
- статистика просмотров вакансий;
- сохранение избранных резюме;
- приглашение студентов к рассмотрению вакансии;
- общение со студентами во встроенных чатах;
- просмотр своих откликов;
- размещение предложений на практику;
- зачисление студентов на практику;
- получение отчётов и выполненных заданий по практике.

Преподаватель в системе может выступать куратором учебной группы, руководителем практики либо совмещать обе функции. В зависимости от назначения ему доступны функции сопровождения собственной учебной группы и контроля прохождения практики закреплёнными группами.

Функционал преподавателей:

- создание учебных групп;
- просмотр профилей и отслеживание активности студентов своих групп;
- просмотр информации о закреплённых для прохождения практики группах;
- просмотр информации о местах прохождения практик;
- просмотр отчётности практики;

– выставление оценок за практику.

Нефункциональные требования предполагают, что система должна выдерживать нагрузку от одновременной работы нескольких пользователей, обеспечивать разграничение доступа по ролям, целостность данных и возможность дальнейшего расширения функциональности. Хранение данных должно выполняться в отдельно разворачиваемой базе данных, а простота развертывания и сопровождения должна обеспечиваться за счёт использования контейнеризации.

Отдельным требованием к процессу разработки является выполнение проектирования программного обеспечения с подготовкой диаграмм компонентов, диаграмм вариантов использования и других артефактов, описывающих структуру и взаимодействие частей системы.

Программное обеспечение должно быть отлажено и протестировано и функционировать в среде, развёрнутой с использованием Docker и Docker Compose [2], что обеспечивает единообразие конфигураций, переносимость между машинами и упрощённое управление жизненным циклом сервисов.

Сформулированные требования являются исходными данными для дальнейшего проектирования и разработки информационной платформы.

2.4 Анализ требований к разрабатываемой информационной платформе трудоустройства и стажировок студентов в ИТ-компаниях

2.4.1 Анализ функциональных требований

В соответствии с определенными при постановке задачи функциональными требованиями были разработаны диаграммы вариантов использования [3] для каждой из ролей. Они позволяют наглядно продемонстрировать основные действия пользователей и их связь с информационной платформой. На рисунке 1 представлена диаграмма вариантов использования для роли студента.

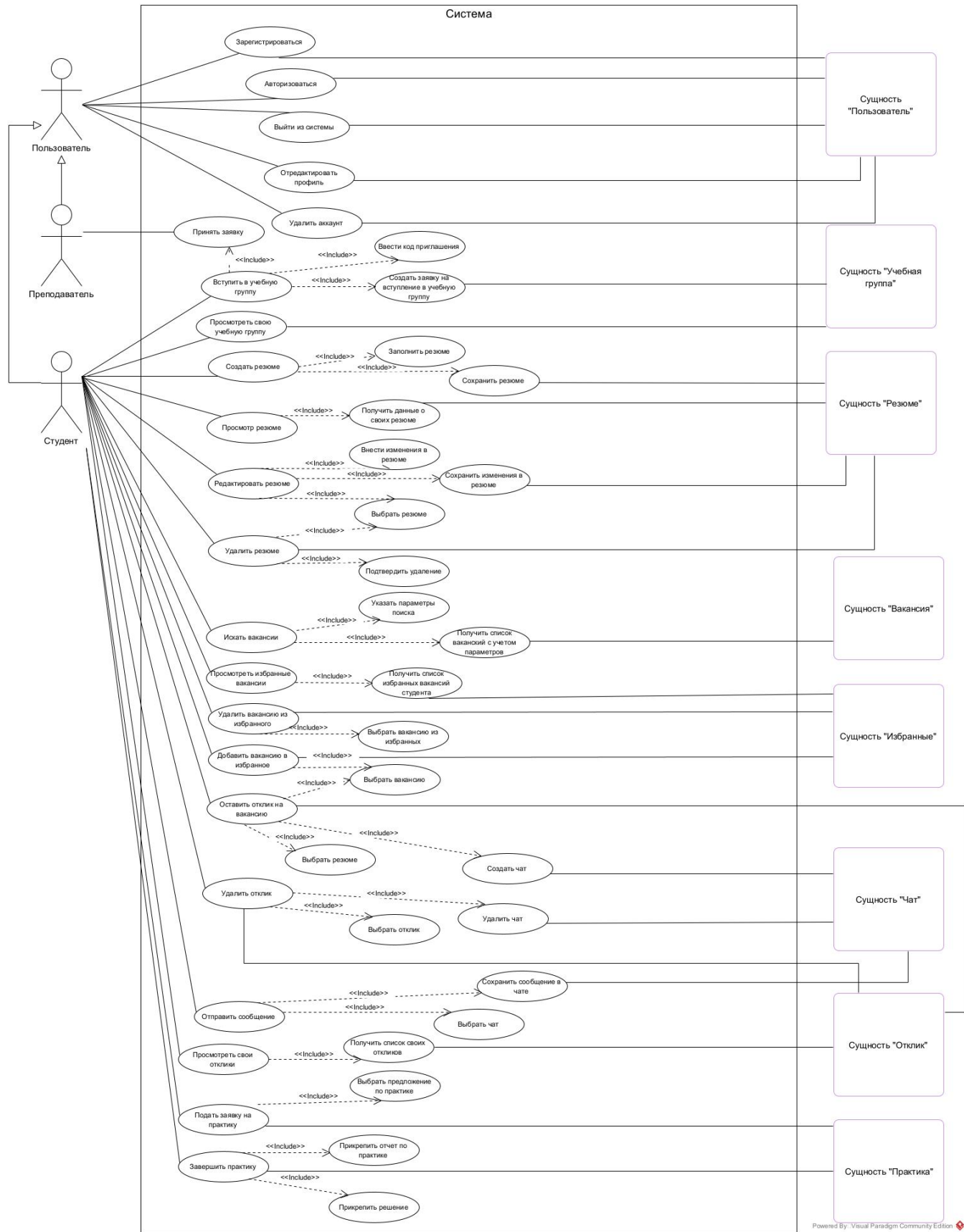


Рисунок 1 – Диаграмма вариантов использования для роли студента

Ниже представлены спецификации для наиболее важных вариантов использования. Спецификация варианта использования «Вступить в учебную группу» представлена в таблице 2.

Таблица 2 – Спецификация варианта использования «Вступить в учебную группу»

Поле	Содержание
Наименование	Вступить в учебную группу
Актёры	Студент, преподаватель
Цель	Привязать студента к учебной группе
Краткое описание	Студент вводит код приглашения учебной группы. Система проверяет код и создаёт заявку на вступление. Преподаватель рассматривает заявку и принимает или отклоняет её. При принятии заявки система привязывает студента к учебной группе и открывает ему доступ к платформе.
Предусловия	Студент авторизован. Учебная группа создана преподавателем. Для группы сформирован код приглашения. Студент ещё не привязан к учебной группе.
Основной поток	<ol style="list-style-type: none"> 1. Студент вводит код приглашения. 2. Система проверяет существование учебной группы с введенным кодом приглашения. 3. Система проверяет, что студент ещё не состоит в учебной группе. 4. Система проверяет, что у студента нет заявки на вступление в другую группу. 5. Система создаёт заявку на вступление в учебную группу. 6. Преподаватель открывает заявку и принимает её. 7. Система привязывает студента к учебной группе и открывает доступ ко всему функционалу студента.
Альтернативные потоки	<p>A1: Код приглашения не найден – система выводит сообщение об ошибке, заявка не создаётся.</p> <p>A2: Студент уже состоит в учебной группе – система выводит сообщение об ошибке, заявка не создаётся.</p> <p>A3: У студента уже есть активная заявка – система выводит информацию о текущей заявке, новая заявка не создаётся.</p>

Продолжение таблицы 2

Поле	Содержание
Альтернативные потоки	А4: Преподаватель отклоняет заявку – система удаляет обработанную заявку и уведомляет студента об отказе. Студент может подать новую заявку по другому коду приглашения.
Постусловия	Студент привязан к группе. Студенту доступен весь функционал платформы.

Спецификация варианта использования «Создать резюме» представлена в таблице 3.

Таблица 3 – Спецификация варианта использования «Создать резюме»

Поле	Содержание
Наименование	Создать резюме
Актёры	Студент
Цель	Добавить новое резюме в профиль.
Краткое описание	Студент заполняет все поля резюме и сохраняет его – система записывает созданное резюме в базу данных
Предусловия	Права доступа студента; студент привязан к учебной группе.
Основной поток	1. Студент заполняет все необходимые поля. 2. Студент указывает ключевые навыки, которыми он владеет. 3. Система проверяет корректность введенных данных. 4. Студент указывает сведения об опыте работы (если имеется). 5. Система сохраняет резюме в базе данных.
Альтернативные потоки	А1: Не заполнены обязательные поля или данные не проходят валидацию – система выводит сообщения об ошибках, вакансия не создаётся.
Постусловия	Резюме создано и доступно для просмотра на платформе.

Спецификация варианта использования «Создать резюме» представлена в таблице 4.

Таблица 4 – Спецификация варианта использования «Оставить отклик на вакансию»

Поле	Содержание
Наименование	Оставить отклик на вакансию
Актёры	Студент

Продолжение таблицы 4

Поле	Содержание
Цель	Отправить работодателю отклик на выбранную вакансию.
Краткое описание	Студент выбирает вакансию и отправляет резюме.
Предусловия	Права доступа студента; студент привязан к учебной группе.
Основной поток	<ol style="list-style-type: none"> 1. Студент выбирает вакансию. 2. Студент нажимает кнопку для создания отклика. 3. Студент выбирает резюме для отклика и отправляет его. 4. Система проверяет принадлежность резюме студенту. 5. Система проверяет, что выбранная вакансия существует и является активной. 6. Система проверяет отсутствие активного отклика от этого студента на эту же вакансию. 7. Система создаёт или находит существующий чат между студентом и компанией по данной вакансии и прикрепляет его к отклику. 8. Система создаёт отклик со статусом «Ожидание ответа».
Альтернативные потоки	<p>A1: Вакансия не найдена или неактивна – система выводит сообщение об ошибке, отклик не создаётся.</p> <p>A2: Выбранное резюме не принадлежит студенту или не найдено – система выводит сообщение об ошибке, отклик не создаётся.</p> <p>A3: У студента уже есть активный отклик на эту вакансию – система выводит сообщение об ошибке, повторный отклик не создаётся.</p>
Постусловия	Отклик создан и доступен для просмотра студенту и работодателю

Следующей ролью разрабатываемой платформы является работодатель. Данная роль выступает инициатором процесса трудоустройства. Работодатели обеспечивают студентов актуальными вакансиями и предложениями по производственной практике. На рисунке 2 представлена диаграмма вариантов использования для роли работодателя.

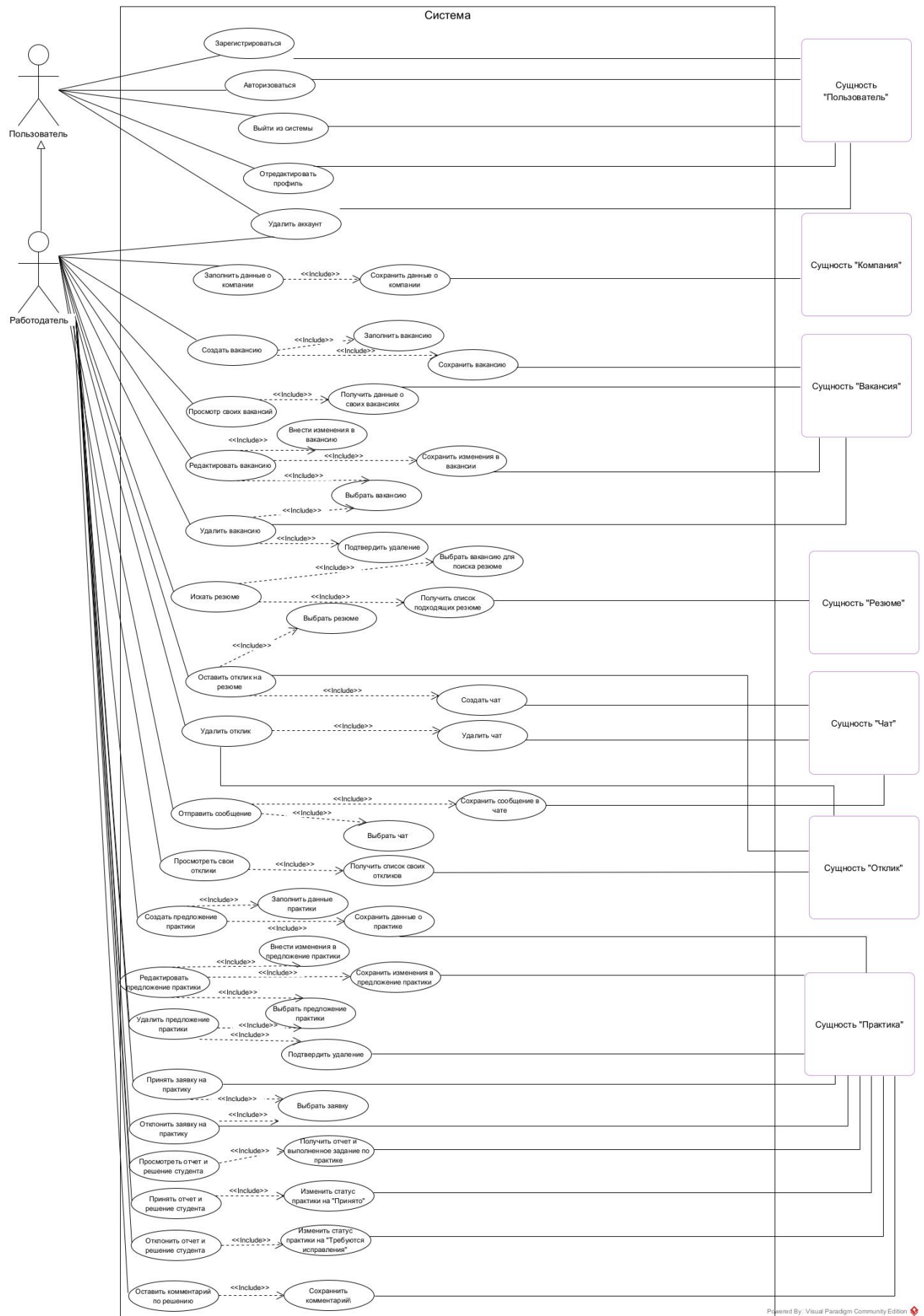


Рисунок 2 – Диаграмма вариантов использования для роли работодателя

Ниже представлены спецификации для наиболее важных вариантов использования.

Спецификация варианта использования «Создать вакансию» представлена в таблице 5.

Таблица 5 – Спецификация варианта использования «Создать вакансию»

Поле	Содержание
Наименование	Создать вакансию
Актёры	Работодатель
Цель	Разместить новую вакансию на платформе.
Краткое описание	Работодатель заполняет все поля вакансии и сохраняет её – система записывает созданную вакансию в базу данных.
Предусловия	Права доступа работодателя.
Основной поток	1. Работодатель заполняет все необходимые поля. 2. Работодатель указывает требуемые для вакансии навыки. 3. Система проверяет корректность введенных данных. 4. Система сохраняет вакансию в базе данных.
Альтернативные потоки	A1: Не заполнены обязательные поля или данные не проходят валидацию – система выводит сообщения об ошибках, вакансия не создаётся.
Постусловия	Вакансия создана и доступна для просмотра на платформе.

Спецификация варианта использования «Искать резюме» представлена в таблице 6.

Таблица 6 – Спецификация варианта использования «Искать резюме»

Поле	Содержание
Наименование	Искать резюме
Актёры	Работодатель
Цель	Найти подходящие резюме для приглашения
Краткое описание	Работодатель указывает фильтры поиска. Система отображает подходящие резюме.
Предусловия	Права доступа работодателя.

Продолжение таблицы 6

Поле	Содержание
Основной поток	1.Работодатель указывает параметры поиска. 2. Работодатель отправляет запрос на поиск. 3.. Система фильтрует резюме в соответствии с переданными параметрами и возвращает полученный список резюме. 4. Система отображает список резюме.
Альтернативные потоки	А1: По заданным параметрам не найдено ни одного резюме – система отображает пустой список и сообщение об отсутствии результатов.
Постусловия	Работодатель получил список резюме в соответствии с указанными параметрами поиска.

Заключительной ключевой ролью пользователей в рамках предметной области разрабатываемой информационной платформы является преподаватель. Он выступает связующим звеном между образовательным процессом и этапом интеграции студентов в профессиональную среду. Его главная задача заключается в отслеживании процесса трудоустройства обучающихся, проверке отчетности по производственным практикам, а также в итоговом выставлении оценок.

Стоит упомянуть, что преподаватель в системе может выступать как куратором учебной группы, так и руководителем практики, либо совмещать обе функции. Выступая в роли куратора, преподаватель получает доступ к функциям сопровождения собственных учебных групп. Помимо это ему доступен просмотр активности на платформе студентов своих групп. В свою очередь, в роли руководителя практики преподавателю открываются функции контроля прохождения практики: от скачивания и проверки отчетов до выставления итоговой оценки. Для визуализации всех доступных преподавателю функциональных возможностей была спроектирована диаграмма вариантов использования (рисунок 3).

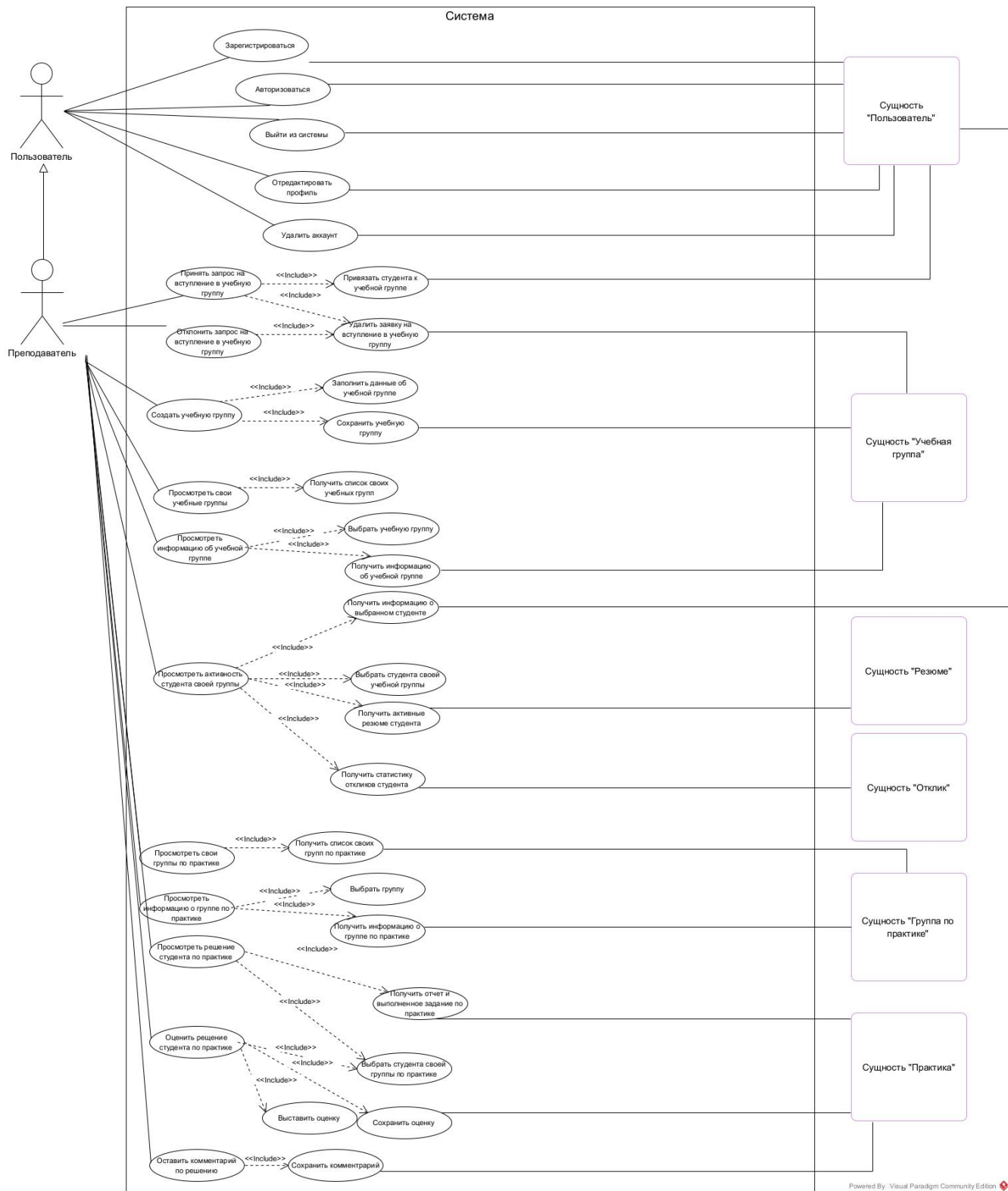


Рисунок 3 – Диаграмма вариантов использования для роли преподавателя

Ниже представлены спецификации для наиболее важных вариантов использования.

Спецификация варианта использования «Создать учебную группу» представлена в таблице 7.

Таблица 7 – Спецификация варианта использования «Создать учебную группу»

Поле	Содержание
Наименование	Создать учебную группу
Актёры	Преподаватель
Цель	Создать новую учебную группу под образовательную программу.
Краткое описание	Преподаватель указывает данные об учебной группе – система сохраняет группу и формирует для неё код приглашения.
Предусловия	Права доступа преподавателя.
Основной поток	1. Преподаватель указывает образовательную программу и код поступления. 2. Система формирует название группы на основании предоставленных сведений. 3. Система генерирует уникальный код приглашения для группы. 4. Система сохраняет данные об учебной группе в базу данных. 5. Система отображает преподавателю код приглашения группы.
Альтернативные потоки	A1: Не заполнены обязательные поля или данные не проходят валидацию – система выводит сообщения об ошибках, учебная группа не создаётся.
Постусловия	Учебная группа создана с уникальным кодом приглашения.

Спецификация варианта использования «Оценить решение студента по практике» представлена в таблице 8.

Таблица 8 – Спецификация варианта использования «Оценить решение студента по практике»

Поле	Содержание
Наименование	Оценить решение студента по практике
Актёры	Преподаватель
Цель	Выставить оценку решению студента и завершить практику.
Краткое описание	Преподаватель выставляет оценку – система сохраняет оценку и помечает практику пройденной.

Продолжение таблицы 8

Поле	Содержание
Предусловия	Права доступа преподавателя; за преподавателем закреплена группа по практике.
Основной поток	1. Преподаватель выбирает студента, просматривает его решение и выставляет оценку. 2. Система проверяет, что практика принята работодателем. 3. Система сохраняет оценку по практике в БД.
Альтернативные потоки	A1: Указана не корректная оценка – система выводит сообщение об ошибке, оценка не выставляется. A2: Практика ещё не принята работодателем – система выводит сообщение об ошибке, оценка не выставляется.
Постусловия	Учебная группа создана с уникальным кодом приглашения.

По итогам анализа функциональных требований были определены основные варианты использования системы для каждой из ролей. Полученные диаграммы вариантов использования служат основой для дальнейшего проектирования платформы трудоустройства и стажировок в ИТ-компаниях.

2.4.2 Анализ нефункциональных требований

Разрабатываемая серверная часть должна соответствовать ряду нефункциональных требований, обеспечивающих стабильную, безопасную и удобную эксплуатацию платформы. Нефункциональные требования – это общие характеристики системы, определяющие как она будет функционировать. К нефункциональным требованиям относятся.

1. Производительность: система должна обеспечивать приемлемое время отклика при выполнении основных операций. При обычной нагрузке пользователи не должны испытывать заметных задержек. Для повышения производительности должны использоваться оптимизированные запросы к базе данных, индексы, а также пагинация при получении списков.

2. Надежность и целостность данных: система должна обеспечивать корректное хранение и обработку данных пользователей. Для этого необходимо использовать ограничения целостности на уровне базы данных, проверку входных данных и обработку ошибок на стороне сервера. Для обеспечения согласованности при выполнении нескольких связанных операций следует использовать транзакции [4].

3. Безопасность: доступ к функционалу платформы должен предоставляться только авторизованным пользователям. Для аутентификации и авторизации в системе будет использоваться JWT (JSON Web Token) [5]. Зашифрованный токен передаётся клиентом на сервер с каждым запросом. На стороне сервера выполняется проверка токена с последующим извлечением из него идентификатора и роли пользователя. Это позволяет однозначно определять какой именно пользователь выполняет запрос и какими правами он обладает. Каждый пользователь должен иметь доступ только к тем данным и операциям, которые соответствуют его роли.

4. Расширяемость и сопровождаемость: архитектура системы должна предусматривать расширение функционала без существенного изменения текущего кода. При этом сам код должен иметь понятную структуру, позволяющую упростить дальнейшее сопровождение, исправление ошибок и добавление новых функций. Для этого необходимо использовать разделение кода на слои, а внутри слоев – на компоненты с четко определенными обязанностями. Это позволяет снизить связанность между компонентами, упростить тестирование и дальнейшее развитие платформы.

5. Удобство развертывания: система должна быть простой в развертывании. Для этого используется контейнеризация с помощью Docker. Такой подход позволяет запускать серверную часть в единой конфигурации, что снижает зависимость от настроек конкретной машины и упрощает перенос системы.

Таким образом анализ нефункциональных требований позволил определить качественные характеристики разрабатываемой платформы. Данные

требования будут учитываться при проектировании и реализации серверной части.

2.5 Выбор инструментальных средств

При разработке серверной части информационной платформы трудоустройства и стажировок в ИТ-компаниях был выбран современный стек технологий, обеспечивающий надёжность, удобство разработки и простоту развертывания. Разработка программного обеспечения выполнялась на операционной системе Windows 10 с использованием интегрированной среды разработки Visual Studio 2022. Для контроля версий исходного кода использовалась система Git [6], позволяющая отслеживать изменения в проекте, фиксировать этапы разработки и при необходимости возвращаться к предыдущим версиям кода.

В качестве основного языка программирования был выбран C# [7]. Данный язык широко используется для разработки серверных приложений, обладает строгой типизацией и поддерживает объектно-ориентированный и асинхронный подходы к программированию. Для реализации серверной части используется фреймворк ASP.NET 8 [8]. Он предоставляет средства для создания и запуска веб-приложений, обработки HTTP-запросов, а также предоставляет встроенные механизмы маршрутизации, авторизации и аутентификации. Разрабатываемая платформа будет реализована в виде многоуровневого монолитного приложения, что позволяет разделить ответственность между компонентами и упростить дальнейшее развитие проекта. Выбор монолитной архитектуры обусловлен спецификой предметной области, функциональные модули которой имеют тесную связь. По сравнению с распределенными архитектурами монолит упрощает разработку, тестирование и отладку, а также избавляет от необходимости организации сложного межсервисного взаимодействия и дополнительных механизмов согласованности данных [1].

В качестве системы управления базами данных (СУБД) будет использоваться PostgreSQL [9]. Она является современной и надежной СУБД,

поддерживающей все необходимые механизмы для обеспечения целостности и согласованности данных, а также позволяет эффективно выполнять запросы.

Для документирования и тестирования API будет использоваться Swagger [10]. Swagger предоставляет интерактивную веб-страницу с описанием доступных API-методов, их параметров, моделей запросов и ответов. Это упрощает тестирование серверной части приложения, а также делает API более понятным для дальнейшей интеграции с клиентским веб-приложением.

Для развёртывания системы будет использоваться Docker Compose. Контейнеризация позволяет запускать серверную часть приложения, базу данных и дополнительные сервисы в единой конфигурации. Это снижает зависимость от настроек конкретной машины, упрощает перенос приложения между средами разработки и тестирования, а также делает процесс развёртывания более предсказуемым. При развёртывании системы будет выполняться миграция базы данных [11] с последующим заполнением таблиц тестовыми данными.

Отдельным компонентом системы является сервис рассылки уведомлений. Для обмена сообщениями между основным приложением и сервисом уведомлений используется брокер сообщений Apache Kafka [12]. Основное приложение отправляет в брокер данные о необходимости отправки уведомления, после чего сервис уведомлений получает эти данные и выполняет рассылку. Такой подход позволяет отделить процесс отправки уведомлений от основной бизнес-логики платформы.

Для визуального проектирования системы использовалась нотация UML 2.0, позволяющая наглядно отобразить взаимодействие пользователя с разрабатываемой системой и основные элементы будущей архитектуры.

Таким образом были определены все необходимые средства для разработки информационной платформы трудоустройства и стажировок в ИТ-компаниях, которые в полной мере удовлетворяют потребности разработки приложения.

2.6 Планирование проекта и оценка бюджета

Для планирования проекта использовался Microsoft Project 2007. Основные задачи проекта отображены на рисунке 4.

	Название задачи	Длительность	Начало	Окончание	Предшественники
1	Начало проекта	0 дней	Пн 13.04.26	Пн 13.04.26	
2	▢ Анализ предметной области и постановка задачи на разработку	5 дней?	Пн 13.04.26	Пт 17.04.26	
3	Анализ предметной области	1 день?	Пн 13.04.26	Пн 13.04.26	
4	Анализ программных аналогов	1 день?	Вт 14.04.26	Вт 14.04.26	3
5	Постановка задачи на разработку	1 день?	Пн 13.04.26	Пн 13.04.26	
6	Анализ функциональных требований	2 дней	Вт 14.04.26	Ср 15.04.26	5
7	Анализ нефункциональных требований	1 день?	Чт 16.04.26	Чт 16.04.26	6
8	Выбор инструментальных средств	1 день?	Пт 17.04.26	Пт 17.04.26	7
9	▢ Проектирование системы	7 дней?	Пн 20.04.26	Вт 28.04.26	2
10	Проектирование архитектуры	2 дней	Пн 20.04.26	Вт 21.04.26	
11	Проектирование базы данных	2 дней	Ср 22.04.26	Чт 23.04.26	10
12	Проектирование серверной части	2 дней	Пт 24.04.26	Пн 27.04.26	11
13	Проектирование средств контейнеризации	1 день?	Вт 28.04.26	Вт 28.04.26	12
14	▢ Реализация системы	21 дней	Ср 29.04.26	Ср 27.05.26	9
15	Реализация базы данных	7 дней	Ср 29.04.26	Чт 07.05.26	
16	Реализация серверной части	14 дней	Пт 08.05.26	Ср 27.05.26	15
17	▢ Тестирование	6 дней	Чт 28.05.26	Чт 04.06.26	14
18	Функциональное тестирование	2 дней	Чт 28.05.26	Пт 29.05.26	
19	Структурное тестирование	2 дней	Пн 01.06.26	Вт 02.06.26	18
20	Модульное тестирование	2 дней	Ср 03.06.26	Чт 04.06.26	19
21	▢ Завершение проекта	4 дней	Пт 05.06.26	Ср 10.06.26	17
22	Исправление выявленных ошибок	2 дней	Пт 05.06.26	Пн 08.06.26	
23	Составление документации	2 дней	Вт 09.06.26	Ср 10.06.26	22
24	Развертывание системы	2 дней	Вт 09.06.26	Ср 10.06.26	22
25	Конец проекта	0 дней	Ср 10.06.26	Ср 10.06.26	21

Рисунок 4 – Основные задачи проекта

На рисунке 5 представлены ресурсы, необходимые для реализации проекта. В качестве основных трудовых ресурсов выделены следующие специалисты: аналитик, проектировщик, программист и тестировщик.

Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на использ.	Начисление	Базовый календарь
Аналитик	Трудовой		А		100%	600,00 Р/ч	700,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
Проектировщик	Трудовой		П-к		100%	700,00 Р/ч	800,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
Программист	Трудовой		П-т		100%	800,00 Р/ч	1 000,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
Тестировщик	Трудовой		Т		100%	500,00 Р/ч	600,00 Р/ч	0,00 Р	Пропорциональное	Стандартный

Рисунок 5 – Лист ресурсов проекта

На следующем этапе для каждой задачи было выполнено распределение ресурсов с учетом специфики выполняемых работ. Результат распределения представлен на рисунке 6.

	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Пн 13.04.26	Пн 13.04.26		
2	Анализ предметной области и постановка задачи на разработку	5 дней?	Пн 13.04.26	Пт 17.04.26		
3	Анализ предметной области	1 день?	Пн 13.04.26	Пн 13.04.26		Аналитик
4	Анализ программных аналогов	1 день?	Вт 14.04.26	Вт 14.04.26	3	Аналитик
5	Постановка задачи на разработку	1 день?	Пн 13.04.26	Пн 13.04.26		Аналитик
6	Анализ функциональных требований	2 дней	Вт 14.04.26	Ср 15.04.26	5	Аналитик[70%];Проектировщик[30%]
7	Анализ нефункциональных требований	1 день?	Чт 16.04.26	Чт 16.04.26	6	Аналитик[70%];Проектировщик[30%]
8	Выбор инструментальных средств	1 день?	Пт 17.04.26	Пт 17.04.26	7	Проектировщик
9	Проектирование системы	7 дней?	Пн 20.04.26	Вт 28.04.26	2	
10	Проектирование архитектуры	2 дней	Пн 20.04.26	Вт 21.04.26		Проектировщик
11	Проектирование базы данных	2 дней	Ср 22.04.26	Чт 23.04.26	10	Проектировщик[60%];Программист[40%]
12	Проектирование серверной части	2 дней	Пт 24.04.26	Пн 27.04.26	11	Проектировщик[50%];Программист[50%]
13	Проектирование средств контейнеризации	1 день?	Вт 28.04.26	Вт 28.04.26	12	Проектировщик
14	Реализация системы	28 дней	Ср 29.04.26	Пт 05.06.26	9	
15	Реализация базы данных	7 дней	Ср 29.04.26	Чт 07.05.26		Программист
16	Реализация серверной части	21 дней	Пт 08.05.26	Пт 05.06.26	15	Программист
17	Тестирование	6 дней	Пн 08.06.26	Пн 15.06.26	14	
18	Функциональное тестирование	2 дней	Пн 08.06.26	Вт 09.06.26		Тестировщик
19	Структурное тестирование	2 дней	Ср 10.06.26	Чт 11.06.26	18	Тестировщик
20	Модульное тестирование	2 дней	Пт 12.06.26	Пн 15.06.26	19	Тестировщик
21	Завершение проекта	4,67 дней	Вт 16.06.26	Пн 22.06.26	17	
22	Исправление выявленных ошибок	2 дней	Вт 16.06.26	Ср 17.06.26		Программист
23	Составление документации	2,67 дней	Чт 18.06.26	Пн 22.06.26	22	Программист[50%];Аналитик[50%]
24	Развертывание системы	2 дней	Чт 18.06.26	Пт 19.06.26	22	Программист
25	Конец проекта	0 дней	Пн 22.06.26	Пн 22.06.26	21	

Рисунок 6 – Задачи проекта с распределенными ресурсами

Для наглядного отображения плана и графика работ по проекту средствами Microsoft Project была построена диаграмма Ганта (рисунок 7), на которой отображаются календарь, длительности задач и используемые ресурсы.

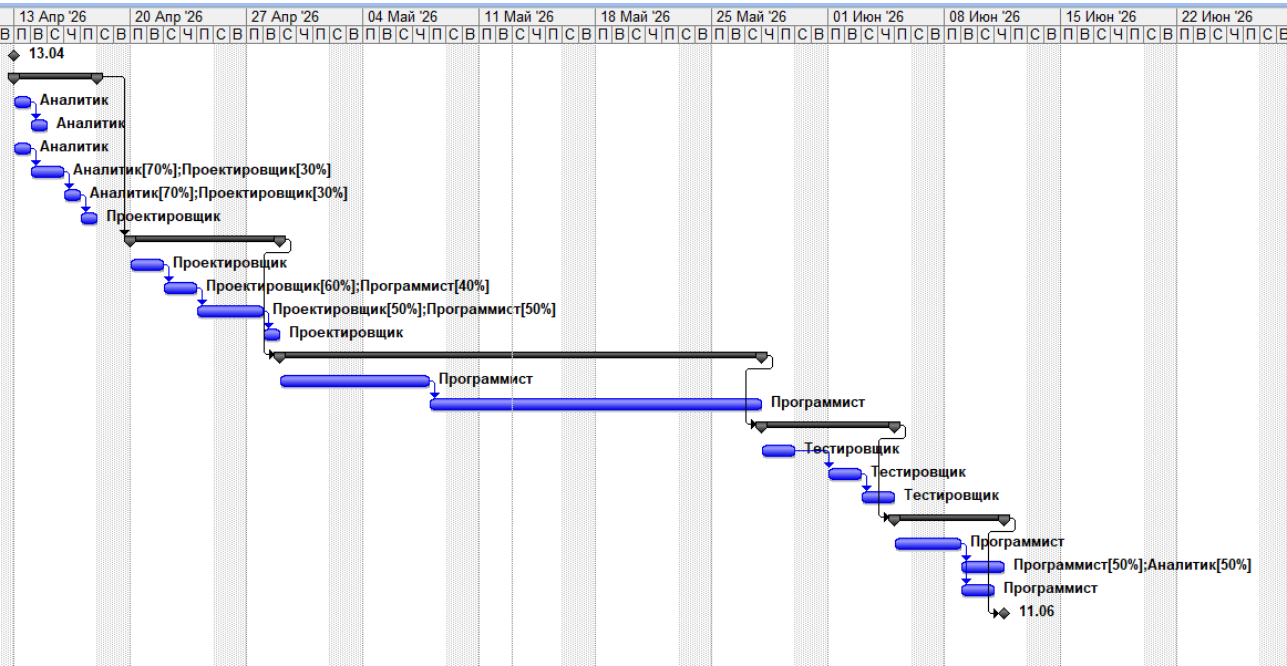


Рисунок 7 – Диаграмма Ганта для проекта

Также средствами Microsoft Project была сформирована статистика проекта (рисунок 8).

	Начало	Окончание	
Текущее	Пн 13.04.26	Чт 11.06.26	
Базовое	НД	НД	
Фактическое	НД	НД	
Отклонение	0д	0д	
	Длительность	Трудозатраты	Затраты
Текущие	43,67д?	376ч	316 293,33 Р
Базовые	0д?	0ч	0,00 Р
Фактические	0д	0ч	0,00 Р
Оставшиеся	43,67д?	376ч	316 293,33 Р
Процент завершения			
Длительность: 0%		Трудозатраты: 0%	
			Заккрыть

Рисунок 8 – Статистика проекта

Из представленной статистики видно, что длительность реализации проекта составляет практически 44 дня при денежных затратах в 316 тысяч рублей.

3 Проектирование серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

3.1 Проектирование архитектуры серверной части

Архитектуры разрабатываемой серверной части информационной платформы спроектирована в виде многоуровневого монолитного приложения с отдельно развертываемой базой данных. Выбор монолитной архитектуры обусловлен тесной функциональной связью между модулями системы. Помимо этого, монолитная архитектура упрощает разработку и тестирование, избавляет от необходимости организации сложного межсервисного взаимодействия и позволяет обойтись без использования дополнительных механизмов согласованности данных [1]. При этом для повышения гибкости системы вспомогательная функция рассылки уведомлений вынесена в отдельный сервис, взаимодействующий с основным приложением посредством брокера сообщений.

Многоуровневая монолитная архитектура предполагает разделение кода на отдельные уровни по функциональному признаку [1]. При разработке информационной платформы весь программный код разделяется на 4 слоя.

1. Слой API (Application Programming Interface): отвечает за первичную обработку и валидацию входящих запросов, а также за формирование ответов. Взаимодействие между слоями API и бизнес-логики организовано с использованием отдельных объектов передачи данных – DTO (Data Transfer Object). Это позволяет абстрагировать внутреннюю структуру доменных сущностей от внешнего интерфейса, обеспечить безопасность данных и снизить нагрузку на сеть, за счет передачи только требуемых данных.

2. Слой бизнес-логики: содержит реализацию бизнес-процессов и алгоритмов функционирования платформы.

3. Слой доступа к данным: обеспечивает взаимодействие приложения с базой данных.

4. Слой доменных сущностей: содержит описание ключевых объектов предметной области.

Такая структура позволяет минимизировать связанность между компонентами и при необходимости заменить реализацию одного из слоёв без влияния на остальные части системы.

В качестве архитектурного стиля взаимодействия с серверной частью используется REST API [13]. Данный стиль предполагает использование основных HTTP-методов для управления ресурсами. Важной особенностью стиля REST API является обмен сообщений без сохранения состояния сервером. Каждый запрос содержит в себе все необходимые для его выполнения данные.

Для реализации встроенных чатов между студентами и работодателями применяется протокол WebSocket [14], который позволяет организовать двунаправленное соединение между клиентами и сервером. Такой подход позволяет доставлять сообщения в режиме реального времени, без необходимости опроса сервера.

На рисунке 9 представлена диаграмма компонентов информационной платформы.

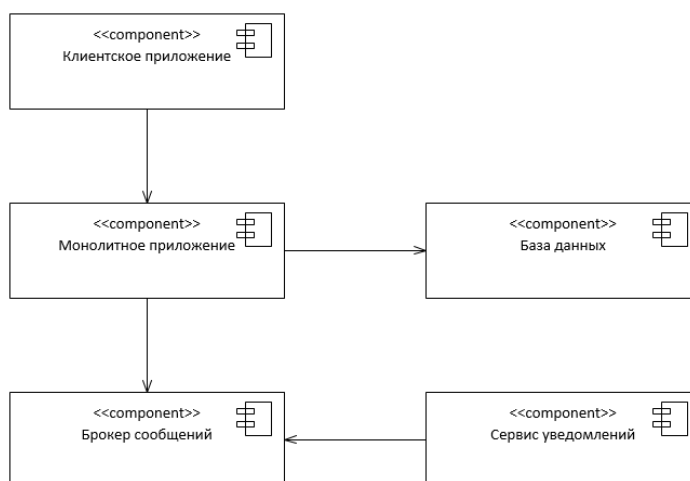


Рисунок 9 – Диаграмма компонентов информационной платформы

Рассмотрим основные компоненты диаграммы.

1. Клиентское приложение: веб-приложение, которое предоставляет пользователям удобный графический интерфейс для работы с платформой.

2. Монолитное приложение: серверное приложение цифровой платформы, занимающееся обработкой входящих запросов.

3. База данных: используется для хранения всех данных платформы.

4. Брокер сообщений: через него монолитное приложение асинхронно отправляет сведения для рассылки уведомлений.

5. Сервис уведомлений: занимается доставкой уведомлений пользователям. Получает информацию об уведомлениях от брокера.

Таким образом, спроектированная архитектура серверной части обеспечивает баланс между производительностью, сопровождаемостью и гибкостью развития, полностью соответствуя поставленным функциональным и нефункциональным требованиям.

3.2 Проектирование базы данных

3.2.1 Разработка концептуальной модели базы данных

Концептуальная модель позволяет отобразить данные и связи предметной области независимо от их представления в компьютере. В таблице 9 описаны спецификации сущностей концептуальной модели.

Таблица 9 – Спецификация сущностей концептуальной модели базы данных

№	Название	Назначение	Описание
1	Пользователь	Стержневая	Список всех зарегистрированных на платформе пользователей
2	Роль	Справочная	Список используемых на платформе ролей
3	Студент	Стержневая	Список всех зарегистрированных на платформе студентов
4	Работодатель	Стержневая	Список всех зарегистрированных на платформе работодателей
5	Преподаватель	Стержневая	Список всех зарегистрированных на платформе учителей
6	Компания	Стержневая	Список всех зарегистрированных на платформе компаний
7	Вакансия	Стержневая	Список размещенных на платформе вакансий
8	Резюме	Стержневая	Список размещенных на платформе резюме
9	Специализация	Справочная	Список используемых специализаций при создании резюме и вакансий
10	Навык	Справочная	Список используемых ключевых навыков при создании резюме и вакансий

Продолжение таблицы 9

№	Название	Назначение	Описание
11	Навык в резюме	Связующая	Используется для связи ключевых навыков с резюме
12	Навык в вакансии	Связующая	Используется для связи ключевых навыков с вакансиями
13	Опыт работы	Стержневая	Используется для сохранения информации о местах работы, указываемых в резюме
14	Отклик на вакансию	Стержневая	Список всех откликов на вакансии
15	Статус отклика	Справочная	Список допустимых статусов откликов на вакансии
16	Чат	Стержневая	Список чатов между работодателями и студентами для обсуждения вакансий
17	Сообщение	Стержневая	Список сообщений
18	Избранная вакансия	Стержневая	Используется для организации списка избранных вакансий у студентов
19	Просмотр резюме	Стержневая	Используется для сохранения данных о просмотре работодателем резюме
20	Просмотр вакансии	Стержневая	Используется для сохранения данных о просмотре студентом вакансии
21	Предложение по практике	Стержневая	Список размещенных на платформе предложений по практике
22	Материал по практике	Стержневая	Используется для сохранения материалов для практики
23	Университет	Справочная	Список всех добавленных на платформу университетов
24	Образовательная программа	Стержневая	Список предоставляемых университетами образовательных программ
25	Учебная группа	Стержневая	Список ученых групп университета
26	Запрос на вступление в учебную группу	Стержневая	Используется для сохранения заявок от студентов на вступление в учебные группы
27	Период практики	Стержневая	Список периодов практик для образовательной программы
28	Период практики в учебной группе	Связующая	Используется для связи периода практики с учебной группой
29	Заявка на практику	Стержневая	Используется для сохранения заявок от студентов на прохождение практики в компании
30	Практика студента	Стержневая	Список мест прохождения студентами практик
31	Отчетность по практике	Стержневая	Используется для сохранения отчетов и решений студентов по практике
32	Статус отчетности	Справочная	Список допустимых статусов отчетности
33	Комментарий к отчетности	Стержневая	Используется для сохранения комментариев по отчетности от преподавателей и работодателей

Для разрабатываемой информационной платформы была разработана концептуальная модель базы данных (рисунок 10).

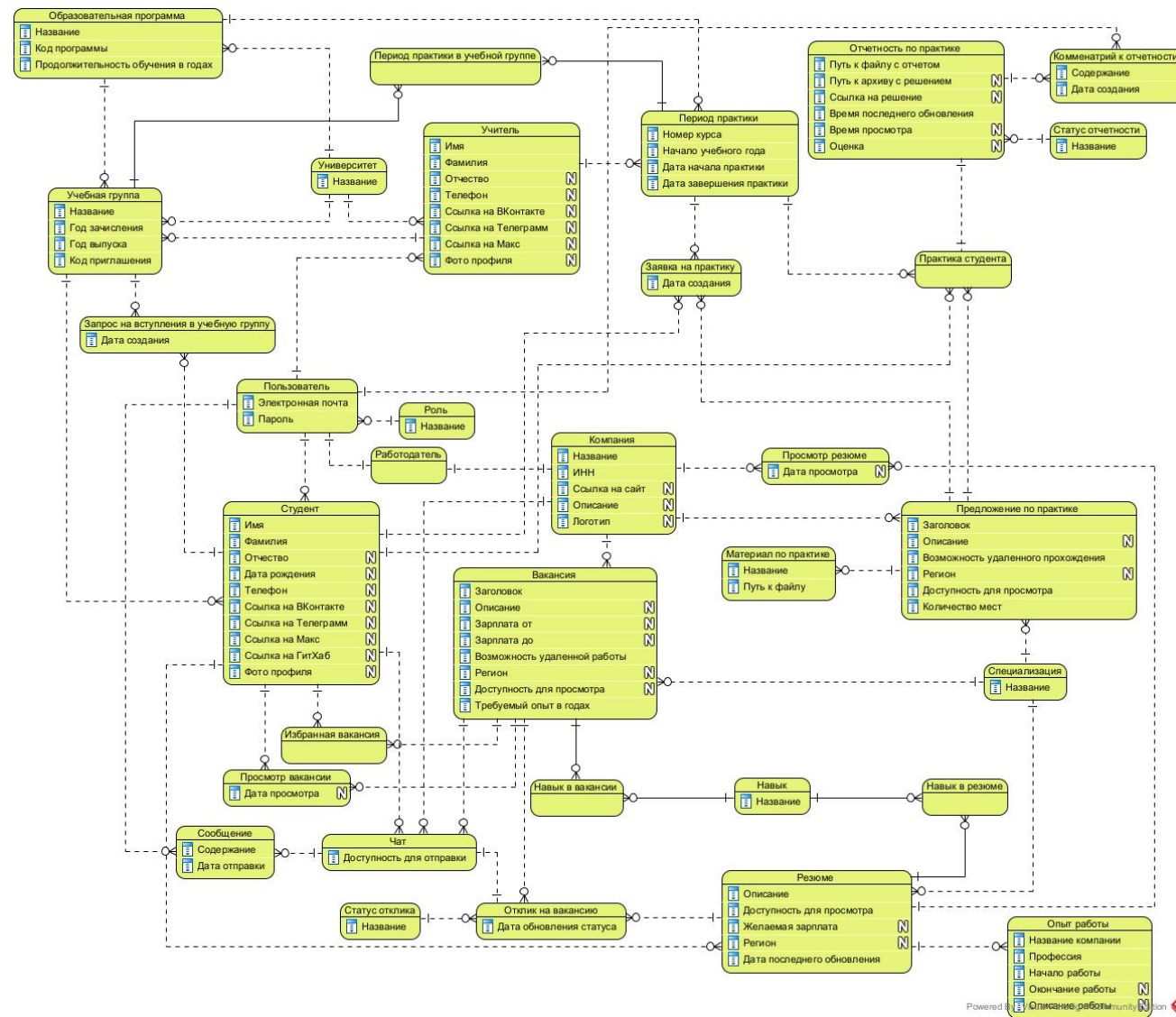


Рисунок 10 – Диаграмма концептуальной модели базы данных информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

Таким образом в ходе разработки концептуальной модели базы данных была построена диаграмма концептуальной модели базы данных информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях.

3.2.2 Разработка логической модели базы данных

В ходе логического моделирования базы данных была разработана диаграмма логической модели базы данных (рисунок 11). В качестве примера описания отношений логической модели данных на таблице 10 представлена спецификация атрибутов отношения «Студент». В приложении А представлены спецификации для всех отношений логической схемы базы данных.

Таблица 10 – Спецификация атрибутов отношения «Студент»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
ПользовательId	int4	Да	Идентификатор пользователя студента	PK, FK
Имя	text	Да	Имя студента	
Фамилия	text	Да	Фамилия студента	
Отчество	text	Нет	Отчество студента	
Дата рождения	date	Нет	Дата рождения студента	
Телефон	varchar(11)	Нет	Номер телефона студента	
Ссылка на ВКонтакте	text	Нет	Ссылка на профиль ВКонтакте	
Ссылка на Телеграм	text	Нет	Ссылка на профиль Telegram	
Ссылка на Макс	text	Нет	Ссылка на профиль Макс	
Ссылка на ГитХаб	text	Нет	Ссылка на аккаунт GitHub	
Фото профиля	text	Нет	Ссылка на фотографию профиля	
ГруппаId	int4	Нет	Идентификатор учебной группы	FK

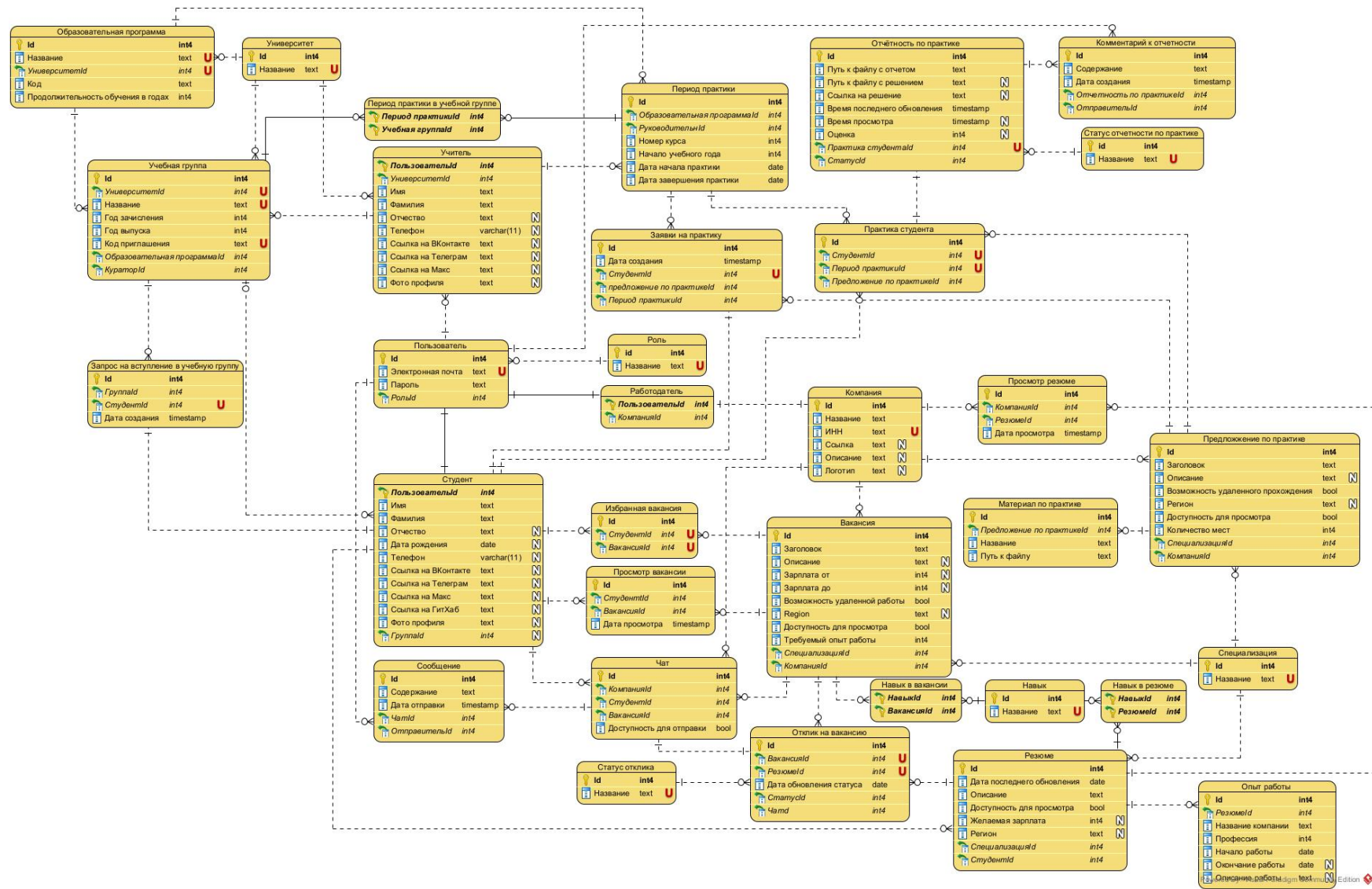


Рисунок 11 – Диаграмма логической модели базы данных информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

Таким образом в ходе логического моделирования базы данных была получена диаграмма логической модели базы данных информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях, которая впоследствии будет использоваться для проектирования и развертывания базы данных.

3.3 Проектирование серверной части

Ввиду масштабности предметной области информационной платформы проектирование серверной части целесообразно выполнять с декомпозицией на отдельные модули. Детальный анализ и описание будут приводиться изолированно для каждого из следующих функциональных блоков:

- авторизация и пользователи: охватывает регистрацию, авторизацию и управление профилями всех ролей;
- резюме и навыки: включает создание и управление резюме, заполнение ключевых навыков и добавление информации об опыте работы;
- вакансии и отклики: описывает размещение и управление вакансиями, а также сопровождение жизненного цикла откликов;
- встроенная коммуникация: организует чаты между студентами и работодателями для обсуждения вакансий;
- учебный процесс и университеты: описывает академическую структуру университета, включая образовательные программы и учебные группы;
- производственные практики: отвечает за организацию производственной практики.

Следует отметить, что данное разделение носит исключительно логический характер и используется только на этапе проектирования для повышения наглядности диаграмм классов и описания спецификаций API. На физическом уровне разрабатываемое приложение представляет собой единый многоуровневый монолит, как это было рассмотрено при проектировании архитектуры.

3.3.1 Проектирование модуля «Авторизация и пользователи»

Данный модуль является базовым компонентом разрабатываемой платформы, отвечающим за аутентификацию пользователей и разграничение их прав внутри системы. Помимо этого, модуль предоставляет функционал для просмотра и редактирования личных кабинетов студентов, работодателей и преподавателей.

Подробная структура модуля представлена на диаграмме классов (рисунок 12). Ниже (таблица 11-12) представлены спецификации некоторых методов API, предоставляемых контроллерами модуля. Полная спецификация всех методов API платформы представлена в приложении В.

Таблица 11 – Спецификация API контроллера AuthController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/auth/register/student	POST	Регистрация нового студента	-
/api/auth/register/employer	POST	Регистрация нового работодателя	-
/api/auth/register/teacher	POST	Регистрация нового преподавателя	-
/api/auth/login	POST	Аутентификация пользователя	-
/api/auth/refresh-token	POST	Обновление токена доступа	-

Таблица 12 – Спецификация API контроллера StudentProfilesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/{id}	GET	Поиск профиля студента по идентификатору	-

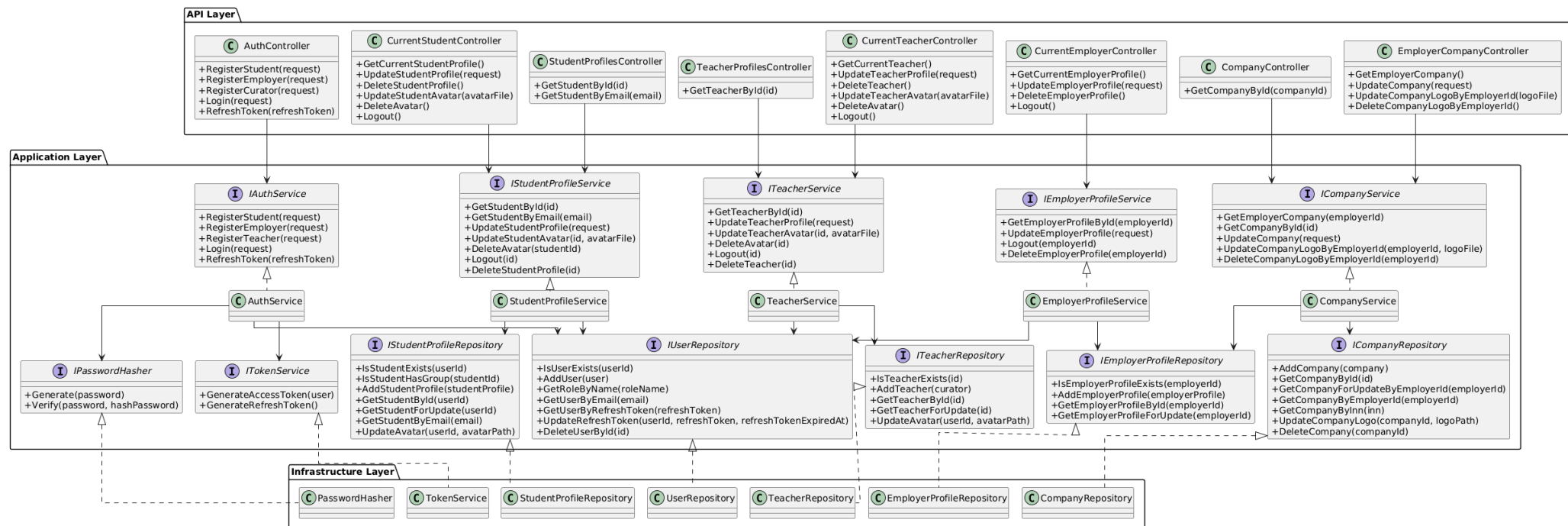


Рисунок 12 – Диаграмма классов модуля «Авторизация и пользователи»

Таким образом спроектированный модуль «Авторизация и пользователи» обеспечивает необходимый функционал для аутентификации и разграничения прав доступа внутри платформы.

3.3.2 Проектирование модуля «Резюме и навыки»

Данный модуль предоставляет студентам удобный инструмент формирования своего профессионального портфолио, а работодателям – структурированную базу для поиска кандидатов. Каждое резюме включает в себя сразу несколько параметров, например регион проживания, желаемую заработную плату, специализацию и список профессиональных навыков. Это позволяет студентам детально настраивать свои резюме, а работодателем гибко фильтровать кандидатов под конкретные нужды. Важной особенностью является возможность создания студентами неограниченного количества независимых резюме. Данный функционал критически важен для начинающих специалистов, так как позволяет им одновременно рассматривать вакансии по нескольким направлениям. Кроме того, платформа позволяет детализировано указывать опыт работы, что обеспечивает отображение хронологии трудовой деятельности кандидата. В целях повышения интерактивности взаимодействия платформа автоматически фиксирует просмотры резюме работодателями, предоставляя студентам статистику для оценки актуальности их резюме.

Ниже (таблица 13-14) представлены спецификации некоторых методов API, предоставляемых контроллерами модуля. Полная спецификация всех методов API платформы представлена в приложении В

Таблица 13 – Спецификация API контроллера ResumesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/resumes/{resumeId}	GET	Получение информации по конкретному резюме студента	-

Продолжение таблицы 13

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/resumes	GET	Глобальный поиск и фильтрация резюме студентов	SalaryFrom – зарплата от, SalaryTo – зарплата до, Region – регион, SpecializationId – идентификатор специализации, AgeFrom – возраст от, AgeTo – возраст до, SkillIds – массив идентификаторов навыков, MinWorkExperienceYears – опыт от, MaxWorkExperienceYears – опыт до, ResumeUpdatedDateFilter – фильтр даты обновления, UpdatedFrom – дата обновления от, UpdatedTo – дата обновления до
/api/students/{studentId}/resumes	GET	Просмотр всех резюме конкретного студента	-

Таблица 14 – Спецификация API контроллера ResumeViewsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/resumes/{resumeId}/views	GET	Просмотр списка компаний, просматривавших резюме студента	-
/api/employers/me/views-history	GET	История просмотров резюме текущим работодателем	-

Подробная структура модуля представлена на диаграмме классов (рисунок 13).

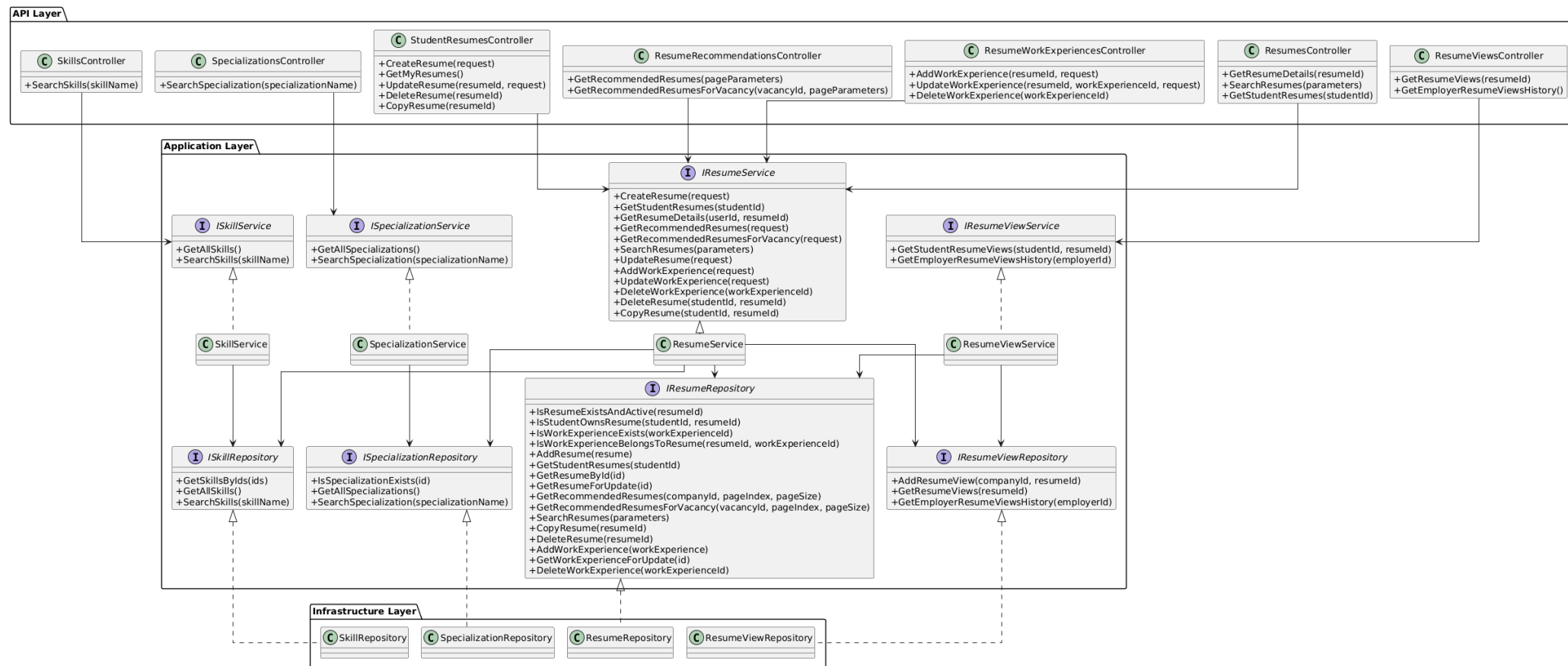


Рисунок 13 – Диаграмма классов модуля «Резюме и навыки»

Таким образом, спроектированный модуль «Резюме и навыки» обеспечивает студентам функционал для публикации и настройки резюме, а работодателем – возможности просмотра, поиска и фильтрации кандидатов.

3.3.3 Проектирование модуля «Вакансии и отклики»

Данный модуль предоставляет работодателям функционал для размещения, редактирования и управления вакансиями. В свою очередь студентам обеспечивается доступ к гибкому поиску и фильтрации открытых вакансий. Аналогично структуре резюме, каждая вакансия состоит из набора описывающих её параметров, таких как уровень заработной платы, регион, требуемый навыки или опыт работы. Сопоставление этих параметров между резюме и вакансиями позволяет формировать ленту рекомендаций как для студентов, так и для работодателей.

Помимо этого, платформа полностью сопровождает жизненный цикл откликов и поддерживает двусторонний процесс инициализации взаимодействия. Отклик может быть оставлен как студентом, так и работодателем в виде приглашения на собеседование. Для каждого отклика система фиксирует текущий статус, основываясь на действиях пользователей. В целях повышения удобства использования платформы студентам предоставляется функционал формирования списка избранных вакансий, для обеспечения быстрого доступа к интересующим позициям. По аналогии с резюме, система автоматически фиксирует факты просмотра вакансий. Это позволяет работодателям выявлять заинтересованных кандидатов, а студентам – формировать историю просмотров для того, чтобы не потерять ранее просмотренные вакансии.

Ниже (таблица 15-17) представлены спецификации некоторых методов API, предоставляемых контроллерами модуля. Полная спецификация всех спроектированных методов API информационной платформы представлена в приложении В.

Таблица 15 – Спецификация API контроллера EmployerVacanciesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/vacancies	POST	Создание вакансии текущим работодателем	-
/api/employers/me/vacancies	GET	Просмотр списка вакансий текущего работодателя	-
/api/employers/me/vacancies/{vacancyId}	PUT	Редактирование вакансии работодателя	-
/api/employers/me/vacancies/{vacancyId}	DELETE	Удаление вакансии работодателя	-

Таблица 16 – Спецификация API контроллера VacancyRecommendations

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/recommended-vacancies	GET	Получение ленты рекомендаций для студента	PageIndex – номер страницы, PageSize – размер страницы
/api/students/me/resumes/{resumeId}/recommended-vacancies	GET	Получение персональных рекомендаций вакансий под конкретное резюме	PageIndex – номер страницы, PageSize – размер страницы

Таблица 17 – Спецификация API контроллера VacancyViewsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/vacancies/{vacancyId}/views	GET	Получение списка студентов, просматривавших вакансию	-
/api/students/me/views-history	GET	Просмотр студентом собственной истории просмотров вакансий	-

Подробная структура модуля представлена на диаграмме классов (рисунок 14).

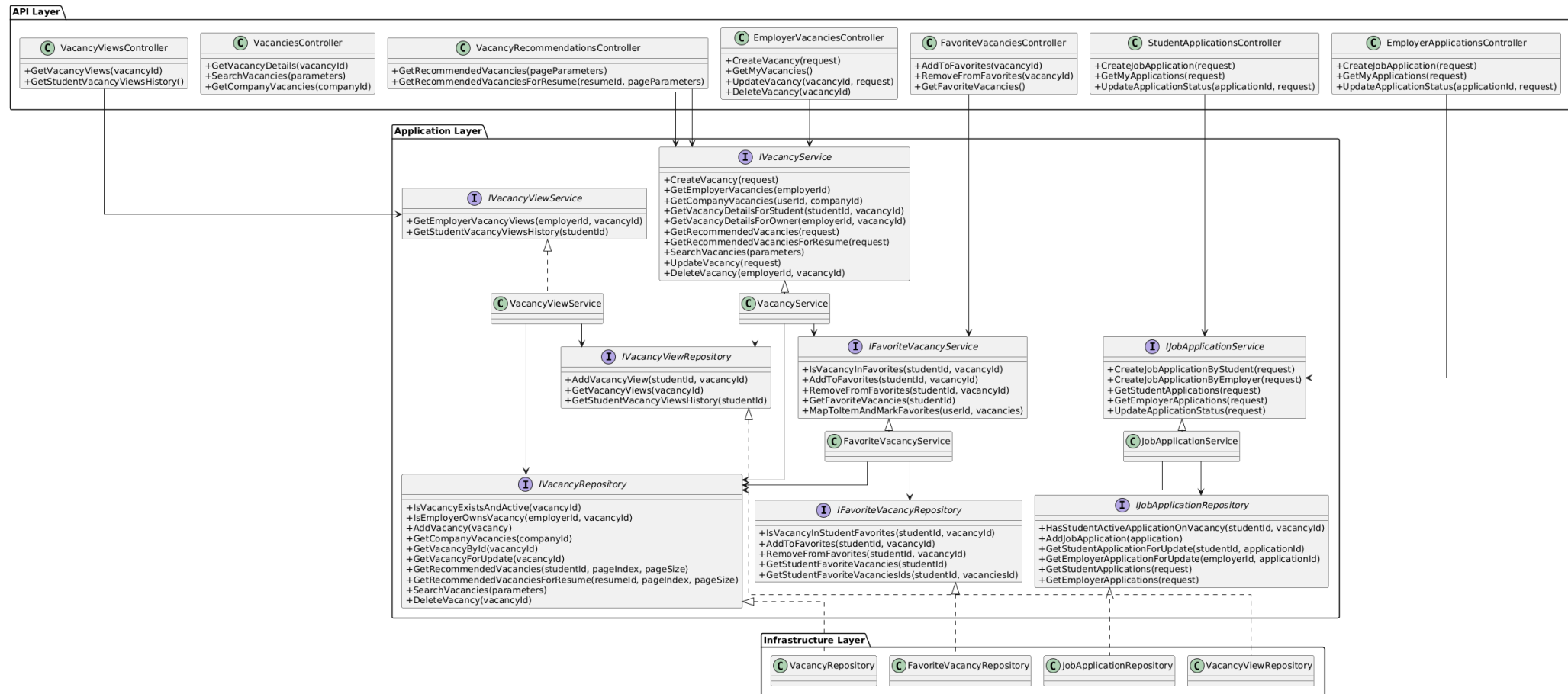


Рисунок 14 – Диаграмма классов модуля «Вакансии и отклики»

Таким образом, спроектированный модуль «Вакансии и отклики» обеспечивает работодателям функционал для публикации и настройки вакансий, а студентам – возможности просмотра, поиска и фильтрации интересных позиций. Помимо этого, модуль обеспечивает гибкое управление жизненным циклом откликов.

3.3.4 Проектирование модуля «Встроенная коммуникация»

Данный модуль интегрирует в платформу функционал обмена сообщениями. Основная цель – предоставление пользователям удобного встроенного инструмента для обсуждения деталей позиций.

Для технической реализации использовался гибридный подход, сочетающий в себе классические REST API запросы и протокол WebSocket для организации взаимодействия в реальном времени. REST API используется для загрузки списка чатов и первичного открытия диалога. После открытия конкретного чата клиентское приложение устанавливает WebSocket-соединение с сервером для обеспечения двунаправленной доставки сообщений в реальном времени без необходимости перезагрузки страницы.

Любой чат создаётся в контексте конкретной вакансии для фиксации предмета обсуждения. Чаты тесно связаны с откликами. Диалог может быть инициирован любой из сторон, например для уточнения требований или компетенций соискателя. В случае, если отклик отправляется позже начала диалога, система автоматически привязывает существующий чат к отклику. Если же на момент отклика диалога не было, то система создаёт новый чат и привязывает его к отклику.

Особое внимание при проектировании модуля было уделено безопасности. При каждом запросе будет выполняться строгая проверка наличия прав доступа у пользователя к запрашиваемому чату.

Для обеспечения надежности хранения данных все передаваемые сообщения будут фиксироваться в базе данных до их отправки. Это позволит исключить возможность потери данных в случае обрыва сетевого соединения.

Ниже (таблица 18-19) представлены спецификации некоторых методов API, предоставляемых контроллерами модуля. Полная спецификация всех спроектированных методов API информационной платформы представлена в приложении В.

Таблица 18 – Спецификация API контроллера StudentChatsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/chats	GET	Получение списка чатов студента	-
/api/students/me/chats	POST	Создание нового диалога с работодателем по поводу вакансии	-
/api/students/me/chats/{chatId}	GET	Получение списка сообщений внутри конкретного чата студента	-
/api/students/me/chats/{chatId}/read	POST	Отметка сообщений в чате как прочитанных студентом	-

Таблица 19 – Спецификация API контроллера EmployerChatsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/chats	GET	Получение списка чатов текущего работодателя	-
/api/employers/me/chats	POST	Инициация нового чата со студентом	-
/api/employers/me/chats/{chatId}	GET	Получение сообщений конкретного чата работодателя	-
/api/employers/me/chats/{chatId}/read	POST	Отметка сообщений чата как прочитанных	-

Подробная структура модуля представлена на диаграмме классов (рисунок 15).

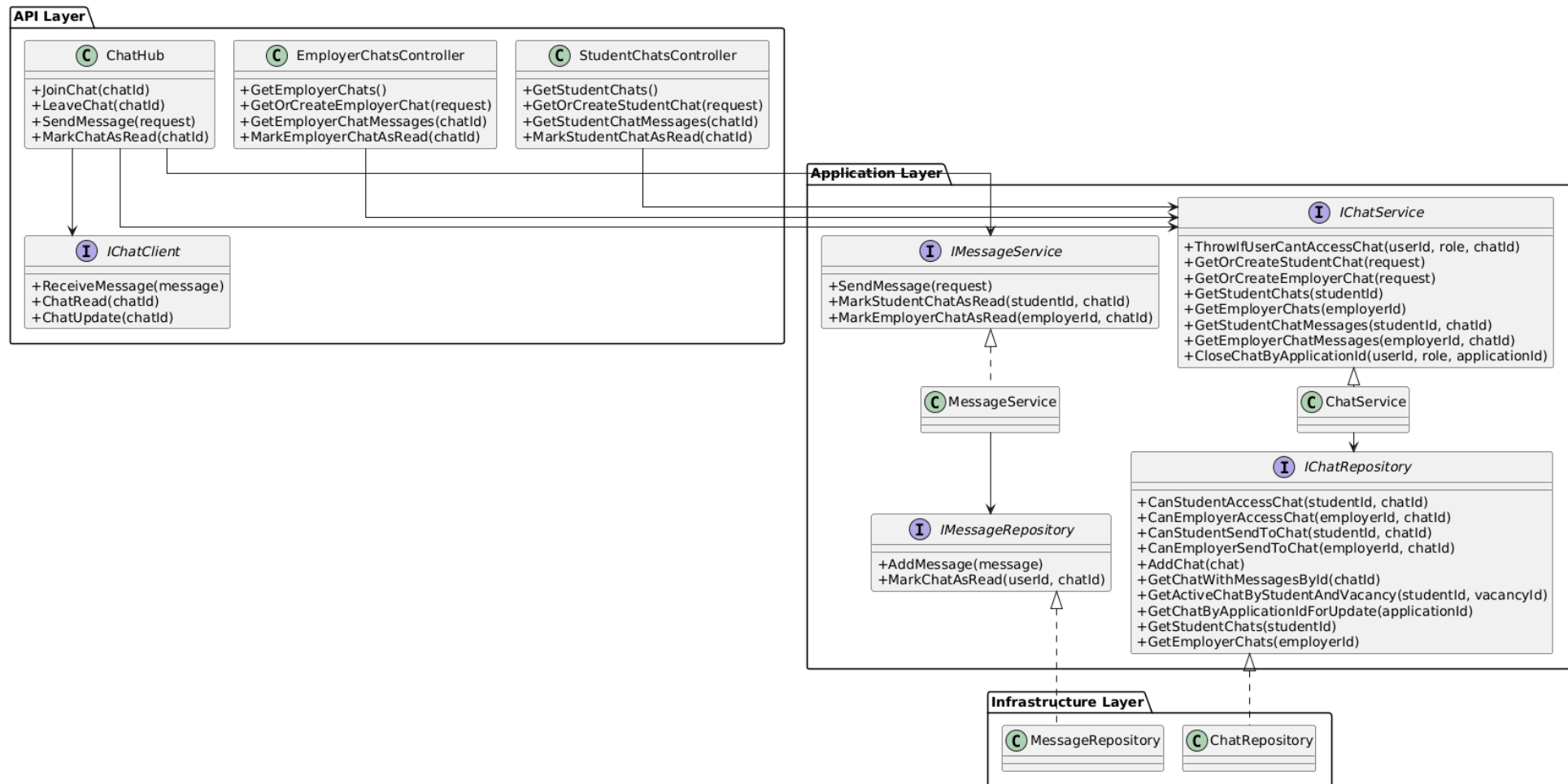


Рисунок 15 – Диаграмма классов модуля «Встроенная коммуникация»

Таким образом модуль «Встроенная коммуникация» организывает чаты между работодателями и студентами для обсуждения вакансий. Это избавляет от необходимости использования сторонних мессенджеров или каналов связи.

3.3.5 Проектирование модуля «Учебный процесс и университеты»

Данный модуль обеспечивает интеграцию академической структуры высших учебных заведений в процесс трудоустройства и прохождения практик. Он формирует цифровую модель учебной иерархии внутри платформы, связывая цепочку сущностей: «Университет – Образовательная программа – Учебная группа – Студент». Основная цель – предоставление преподавателям инструментов для отслеживания деятельности обучающихся внутри информационной платформы.

Модуль предоставляет преподавателям функционал для формирования учебных групп и отслеживания активности своих студентов. Для предотвращения несанкционированного доступа к группам, система генерирует уникальные приглашительные коды для каждой группы. По этим кодам студенты подают заявки на вступление в учебные группы. Кураторы, в свою очередь, получают уведомления о поступающих заявках и принимают решения об их подтверждении или отклонении. Важно отметить, что до момента зачисления в группу учетная запись студента имеет жесткие ограничения. Полный доступ к целевым возможностям платформы открывается соискателю исключительно после того, как преподаватель подтвердит его принадлежность к учебной группе. Данный подход исключает присутствие на платформе сторонних лиц, не имеющих отношения к образовательному процессу, и гарантирует компаниям достоверность статуса соискателей.

Кураторам доступна детальная статистика каждого студента своих групп, которая включает:

- просмотр профиля и опубликованных резюме;
- сводку по отправленным откликам и их текущим статусам.

Ниже (таблица 20-22) представлены спецификации некоторых методов API, предоставляемых контроллерами модуля. Полная спецификация всех спроектированных методов API информационной платформы представлена в приложении В.

Таблица 20 – Спецификация API контроллера CuratorGroupsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/curators/me/groups	POST	Создание новой учебной группы куратором	-
/api/curators/me/groups	GET	Получение списка учебных групп текущего куратора	-
/api/curators/me/groups/{groupId}/students	GET	Получение списка студентов конкретной учебной группы куратора	-
/api/curators/me/groups/{groupId}/invite-code	PUT	Обновление кода приглашения для вступления в группу	-

Таблица 21 – Спецификация API контроллера StudentGroupController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/group	GET	Получение информации об учебной группе студента	-

Таблица 22 – Спецификация API контроллера EducationalProgramsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/educational-programs	GET	Получение списка образовательных программ	name – название программы для фильтрации

Подробная структура модуля представлена на диаграмме классов (рисунок 16).

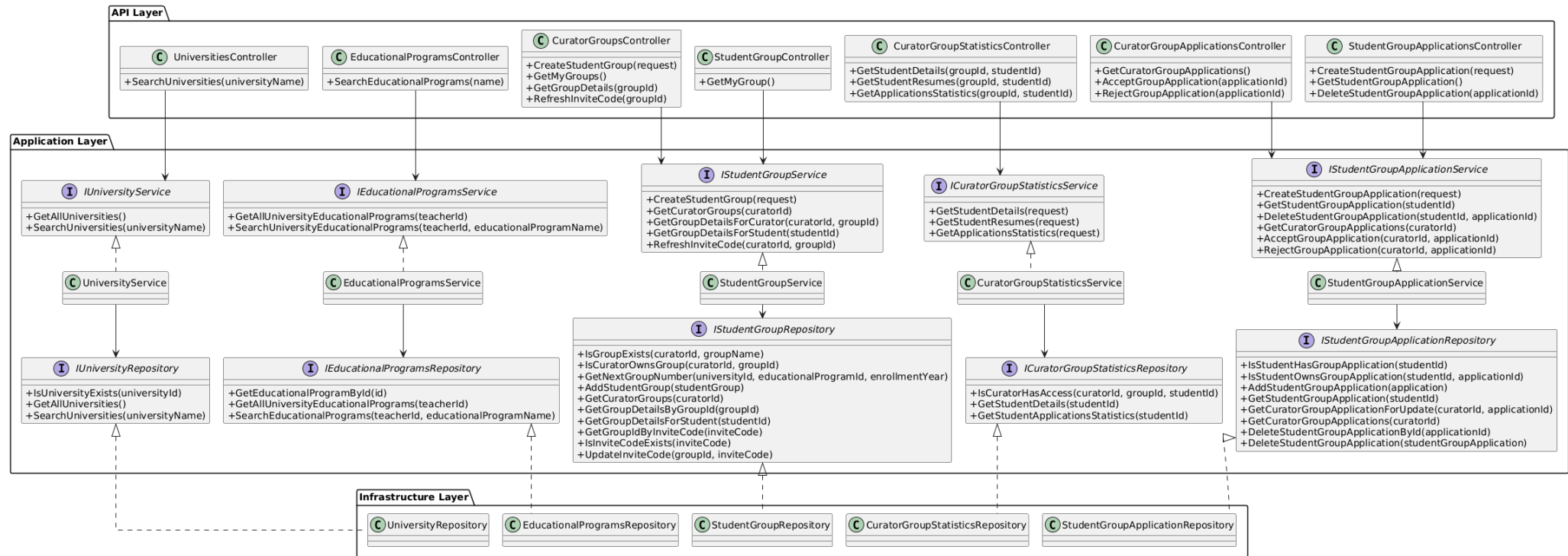


Рисунок 16 – Диаграмма классов модуля «Учебный процесс и университеты»

Таким образом данный модуль обеспечивает прозрачный контроль карьерного развития со стороны образовательной организации и защищает платформу от несанкционированного доступа.

3.3.6 Проектирование модуля «Производственные практики»

Данный модуль автоматизирует полный жизненный цикл сопровождения производственных практик. Его главной целью является предоставление студентам единой удобной площадки для записи на практики в предприятия и для сдачи отчетности, а преподавателям и работодателям – инструменты для просмотра, комментирования и оценивания отчетности студента.

Работодателям предоставляется инструмент для публикации предложений по практике. Помимо заполнения текстовых сведений о практике, работодатели имеют возможность прикрепить сопутствующие файлы и методические материалы.

Платформа позволяет студентам получать информацию о периодах практики, выполнять поиск и фильтрацию предложений по практикам, а также осуществлять запись на выбранный период практики в выбранные организации. Преподаватели же могут просматривать информацию об учебных группах, которые закреплены за ними для прохождения практики. Процесс зачисления на практику реализован через подачу электронных заявок, при этом при подаче заявки система проверяет количество свободных мест и наличие активной практики у студента. Работодатель, получив заявку, может принять студента к себе на практику или отказать ему. В случае принятия заявки студенту открывается доступ к просмотру загруженных работодателем материалов по практике для того, чтобы подробнее ознакомиться с заданием.

На этапе завершения практики студент загружает отчет по практике и архив с решением или ссылку на него. После загрузки, отчетность становится доступна для просмотра работодателю и руководителю практики. Каждый из них может оставлять комментарии по решению студента. Работодатель при проверке решения выбирает: принять решение или отправить его на доработку. После

финального утверждения практики со стороны предприятия студенту блокируется возможность изменять файлы, а преподаватель получает доступ к выставлению студенту итоговой оценки. Внесение оценки в базу данных, означает завершение практики студентом.

Ниже (таблица 23-24) представлены спецификации некоторых методов API, предоставляемых контроллерами модуля. Полная спецификация всех спроектированных методов API информационной платформы представлена в приложении В.

Таблица 23 – Спецификация API контроллера TeacherPracticePeriodsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/teachers/me/practice-periods/groups	GET	Просмотр групп по текущей практике	-
/api/teachers/me/practice-periods/{practicePeriodId}/groups/{groupId}	GET	Получение информации о периоде практики конкретной группы	-

Таблица 24 – Спецификация API контроллера EmployerPracticeMaterialsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/practice-offers/{practiceOfferId}/files	POST	Загрузка файлов к предложению о практике	-
/api/employers/me/practice-offers/{practiceOfferId}/files/{fileId}	DELETE	Удаление файла из материалов практики	-

Подробная структура модуля представлена на диаграмме классов (рисунок 17).

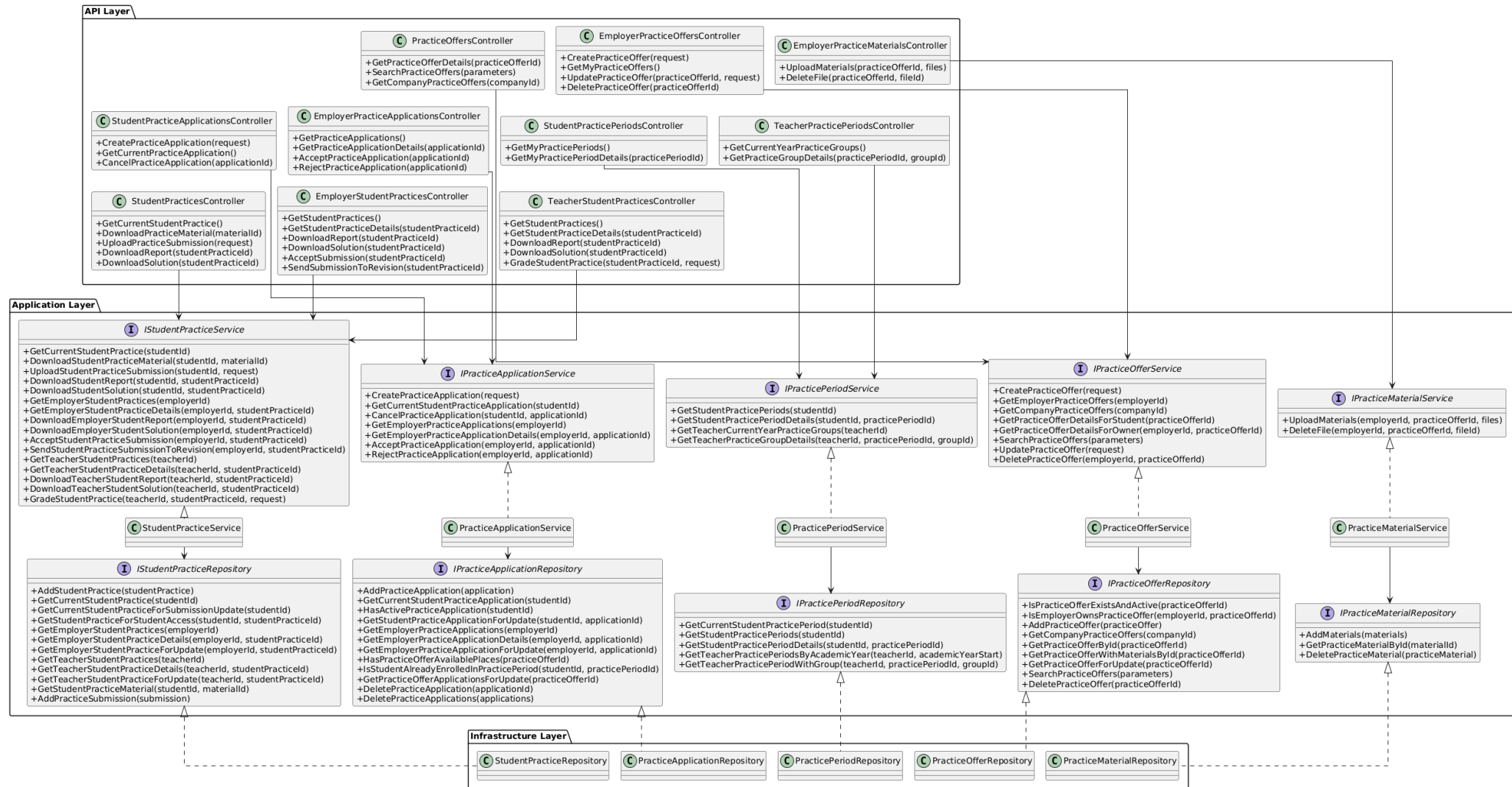


Рисунок 17 – Диаграмма классов модуля «Производственные практики»

Таким образом данный модуль предоставляет пользователям платформы единую площадку для сопровождения практик: от размещения предложений и подачи заявок на них до загрузки отчетности и выставления оценок преподавателями.

3.3.7 Проектирование сервиса рассылки уведомлений

Данный компонент представляет собой отдельный сервис, функционирующий независимо от основного приложения. Главной задачей сервиса является информирование пользователей о ключевых событиях.

Для взаимодействия основного приложения с сервисом используется брокер сообщений Apache Kafka. В процессе работы монолитное приложение публикует информацию об уведомлениях в брокер сообщений. Сервер уведомлений непрерывно прослушивает целевой топик и при поступлении нового события незамедлительно отправляет его на почту. Отправка осуществляется по протоколу SMTP (Simple Mail Transfer Protocol) [15] через интеграцию со специализированным почтовым шлюзом. Подробная структура сервиса отображена на диаграмме классов (рисунок 18).

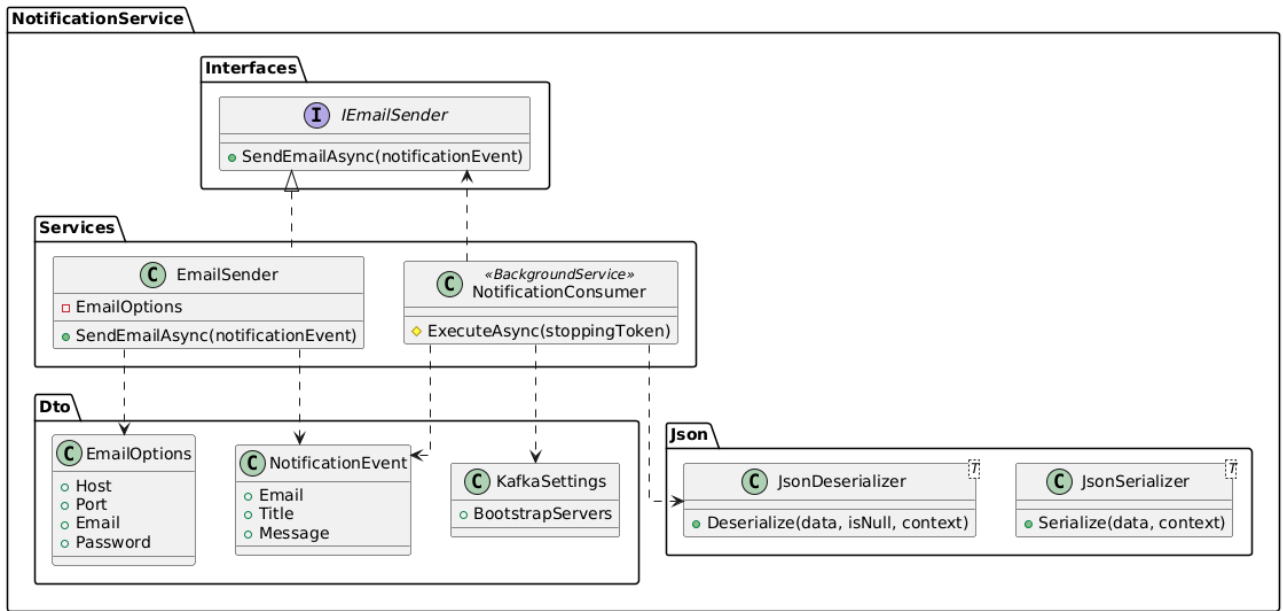


Рисунок 18 – Диаграмма классов сервиса рассылки уведомлений

Таким образом, спроектированный сервис рассылки уведомлений позволяет отправлять сообщения на электронную почту пользователей в асинхронном режиме с использованием брокера сообщений, что исключает задержки в работе основного серверного приложения.

3.4 Проектирование алгоритмов

В данном разделе будут рассмотрены три основных бизнес-процесса разрабатываемой информационной платформы. Для каждого алгоритма построена диаграмма деятельности [3] и приведено текстовое описание.

3.4.1 Проектирование алгоритма вступления в учебную группу

Алгоритм описывает последовательность действий для вступления студента в учебную группу. Он состоит из следующих шагов:

- преподаватель регистрируется на платформе и заполняет сведения о своей учебной группе;
- система автоматически генерирует для созданной группы уникальный код приглашения;
- преподаватель любым удобным способом передаёт код своим студентам;
- студент авторизуется на платформе и вводит код приглашения;
- система формирует заявку на вступление в группу и уведомляет преподавателя по электронной почте;
- при подтверждении заявки преподавателем система привязывает профиль студента к учебной группе и отправляет ему уведомление на электронную почту, после чего студенту открывается доступ к полному функционалу информационной платформы;
- в случае отклонения заявки студент получает соответствующее уведомление и может отправить новую заявку после уточнения данных.

На рисунке 19 представлена диаграмма деятельности данного алгоритма.

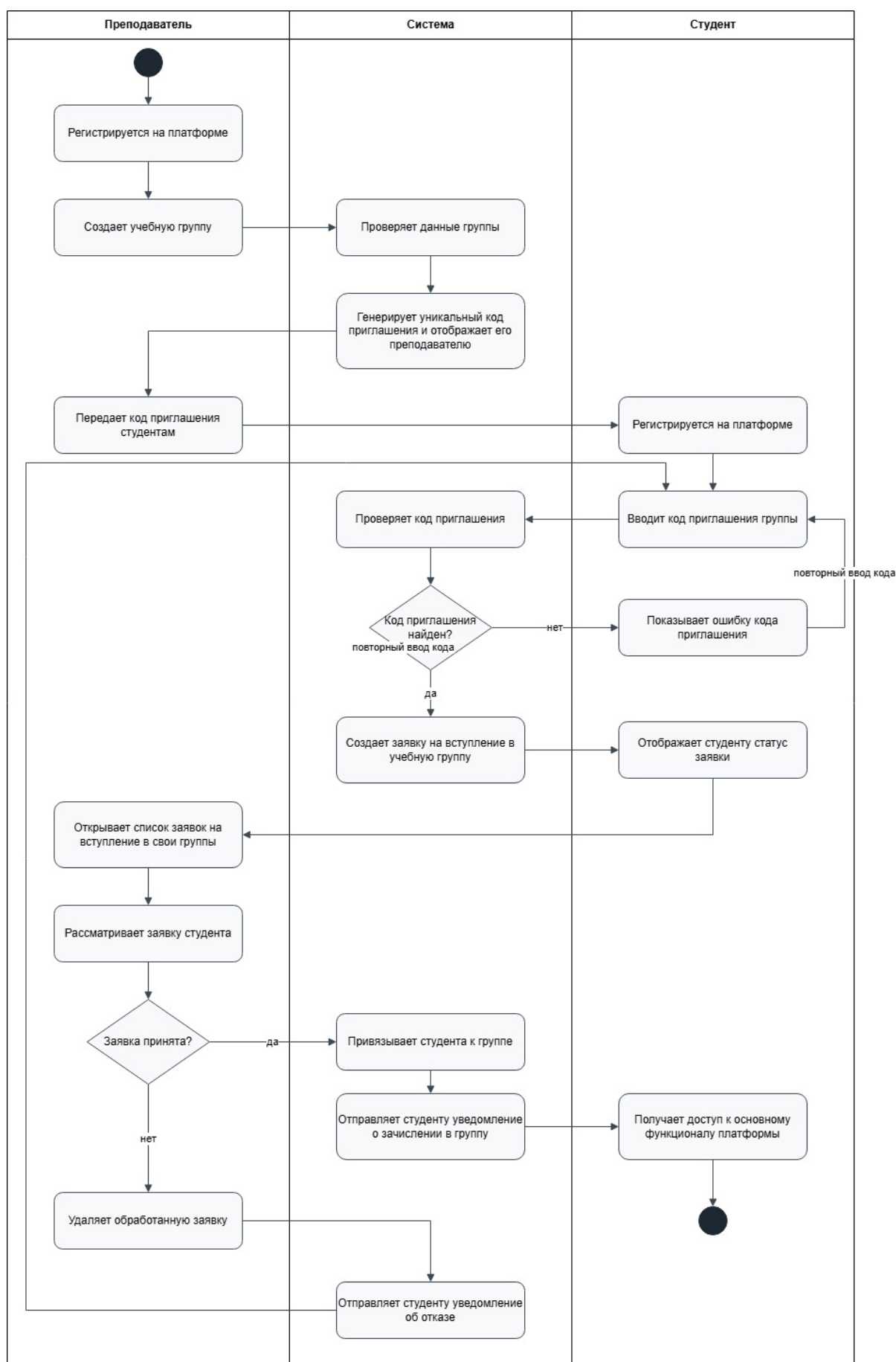


Рисунок 19 – Диаграмма деятельности алгоритма вступления в учебную группу

Спроектированный алгоритм обеспечивает корректную привязку студента к учебной группе. Использование кодов приглашений в связке с ручным подтверждением заявки преподавателем исключает возможность несанкционированного добавления сторонних пользователей.

3.4.2 Проектирование алгоритма трудоустройства

Данный алгоритм является центральным в информационной платформе. Он описывает полный жизненный цикл взаимодействия студента и представителя компании: от размещения резюме до трудоустройства. Алгоритм состоит из следующих шагов:

- студент создаёт резюме, которое система сохраняет в базе данных;
- студент осуществляет поиск вакансий;
- система формирует список вакансий и отображает студенту;
- студент просматривает вакансии, выбирает интересующую его вакансию и отправляет резюме;
- система регистрирует отклик на вакансию и создаёт для него чат (или привязывает уже существующий);
- работодатель получает отклик и рассматривает его;
- в случае отказа система фиксирует статус отклика «Отклонено» и закрывает связанный чат;
- в случае одобрения кандидата, система устанавливает статус отклика «Приглашен на собеседование»;
- по результатам собеседования работодатель либо отказывает студенту, либо приглашает его на работу;
- получив приглашение на работу, студент принимает решение согласиться или отказаться;
- система устанавливает статус отклика в зависимости от решения студента.

На рисунке 20 представлена диаграмма деятельности данного алгоритма.

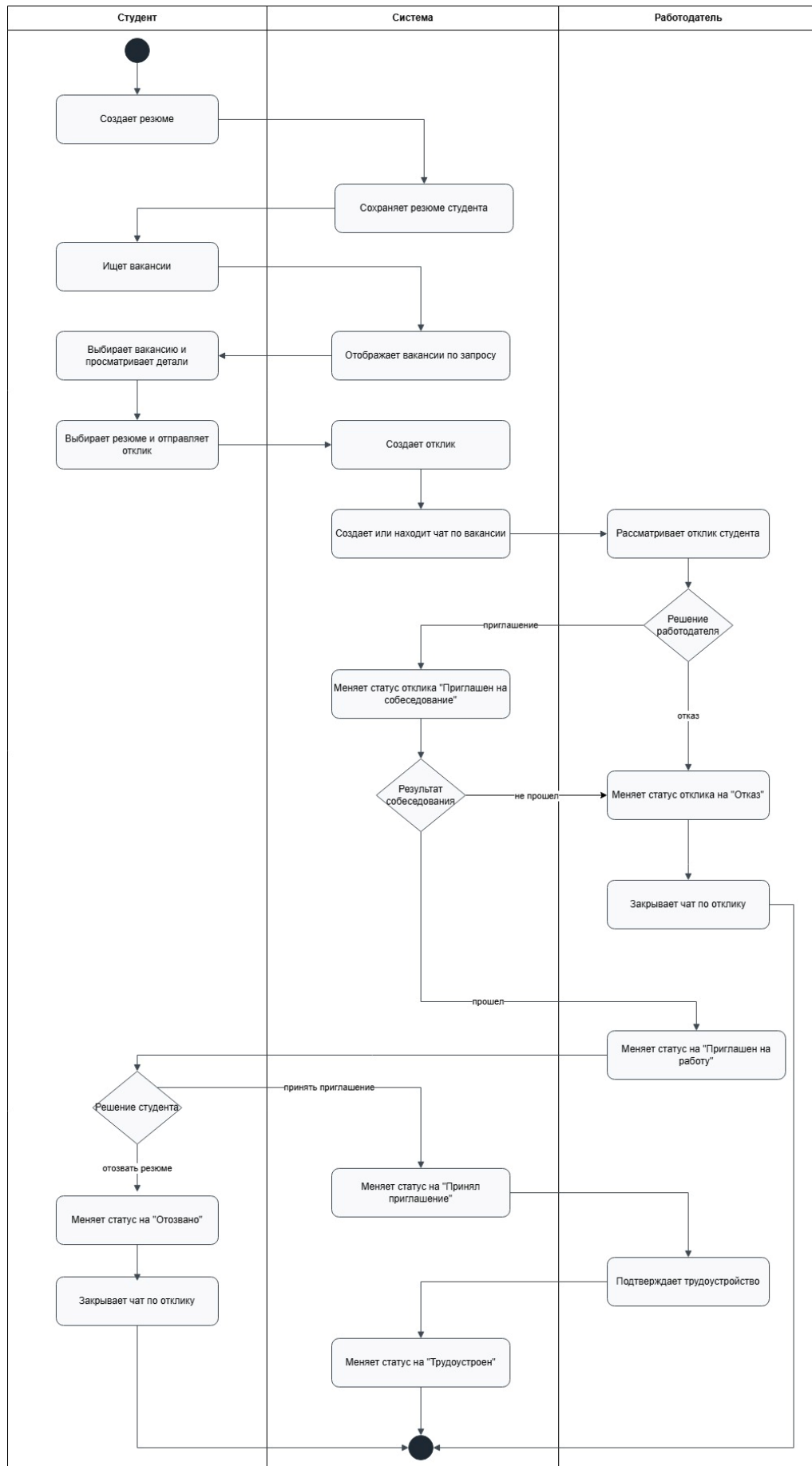


Рисунок 20 – Диаграмма деятельности алгоритма трудоустройства

Таким образом, спроектированный алгоритм трудоустройства покрывает все возможные сценарии развития событий при найме сотрудников.

3.4.3 Проектирование алгоритма прохождения практики

Данный алгоритм охватывает сразу все три роли платформы и описывает их взаимодействие при прохождении практики. Алгоритм состоит из следующих шагов:

- работодатель формирует предложение на прохождение практики и прикрепляет к нему сопроводительные файлы;
- студент осуществляет поиск предложений по практике;
- система формирует список предложений по практике и отображает студенту;
- студент просматривает предложения, выбирает интересующее его предложение и отправляет заявку;
- система создаёт заявку и уведомляет работодателя;
- работодатель получает заявку на прохождение практики и рассматривает её;
- в случае одобрения заявки система фиксирует факт прохождения студентом практики в этой компании, а студенту открывается возможность для скачивания материалов по практике;
- студент выполняет задание, формирует отчёт и загружает его на платформу;
- если работодатель выявляет недочеты в работе, он может отправить её на доработку;
- в случае принятия работы работодателем преподаватель фиксирует факт прохождения практики и выставляет итоговую оценку.

На рисунке 21 представлена диаграмма деятельности данного алгоритма.

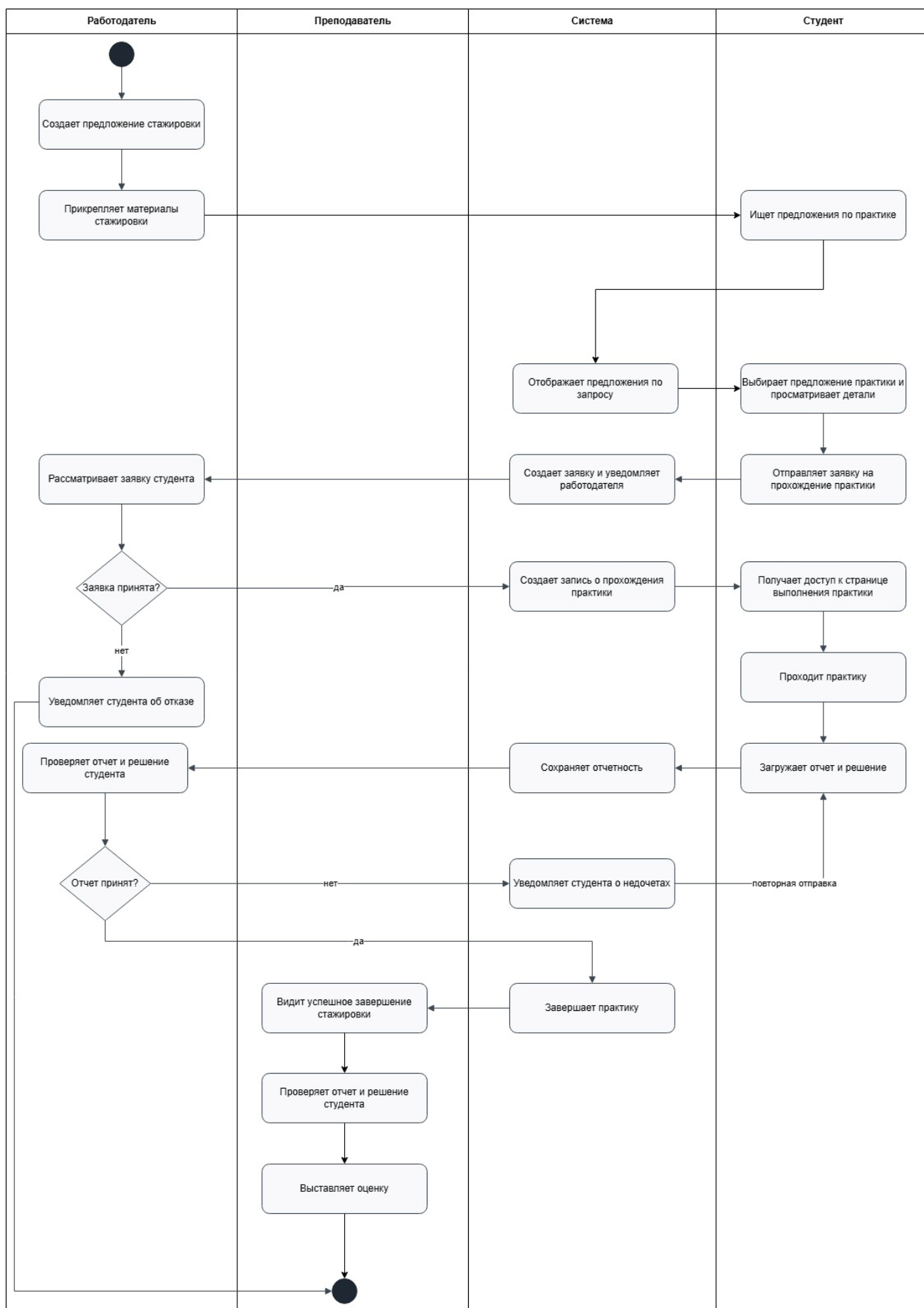


Рисунок 21 – Диаграмма деятельности алгоритма прохождения практики

Таким образом данный алгоритм обеспечивает комплексное сопровождение практики. Разделение процесса на четкие этапы позволяет автоматизировать передачу файлов, организовать удобное взаимодействие между студентом и компанией, а также предоставить преподавателю встроенный инструмент для оценивания результатов.

4 Реализация серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

4.1 Развертывание базы данных

4.1.1 Разработка физической модели базы данных

Физическая модель базы данных – это детализированная схема, показывающая как именно данные будут обрабатываться конкретной СУБД. На основании концептуальной и логической модели была разработана физическая модель базы данных информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях, представленная на рисунке 22. На основе полученной физической модели средствами Visual Paradigm был сгенерирован скрипт для создания объектов базы данных. Фрагмент скрипта представлен на листинге 1. Полный скрипт для создания объектов базы данных представлен в приложении Б.

Листинг 1 – Фрагмент SQL-скрипта для создания объектов базы данных

```
CREATE TABLE "Vacancies" (
    "Id" SERIAL NOT NULL,
    "Title" text NOT NULL,
    "Description" text,
    "SalaryFrom" integer NOT NULL,
    "SalaryTo" integer NOT NULL,
    "IsRemote" bool NOT NULL,
    "Region" text,
    "IsActive" bool NOT NULL,
    "MinWorkExperienceYears" integer NOT NULL,
    "SpecializationId" integer NOT NULL,
    "CompanyId" integer NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "PracticeMaterials" (
    "Id" SERIAL NOT NULL,
    "PracticeOfferId" integer NOT NULL,
    "FileName" text NOT NULL,
    "FilePath" text NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "Skills" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL UNIQUE,
    PRIMARY KEY ("Id"));
```

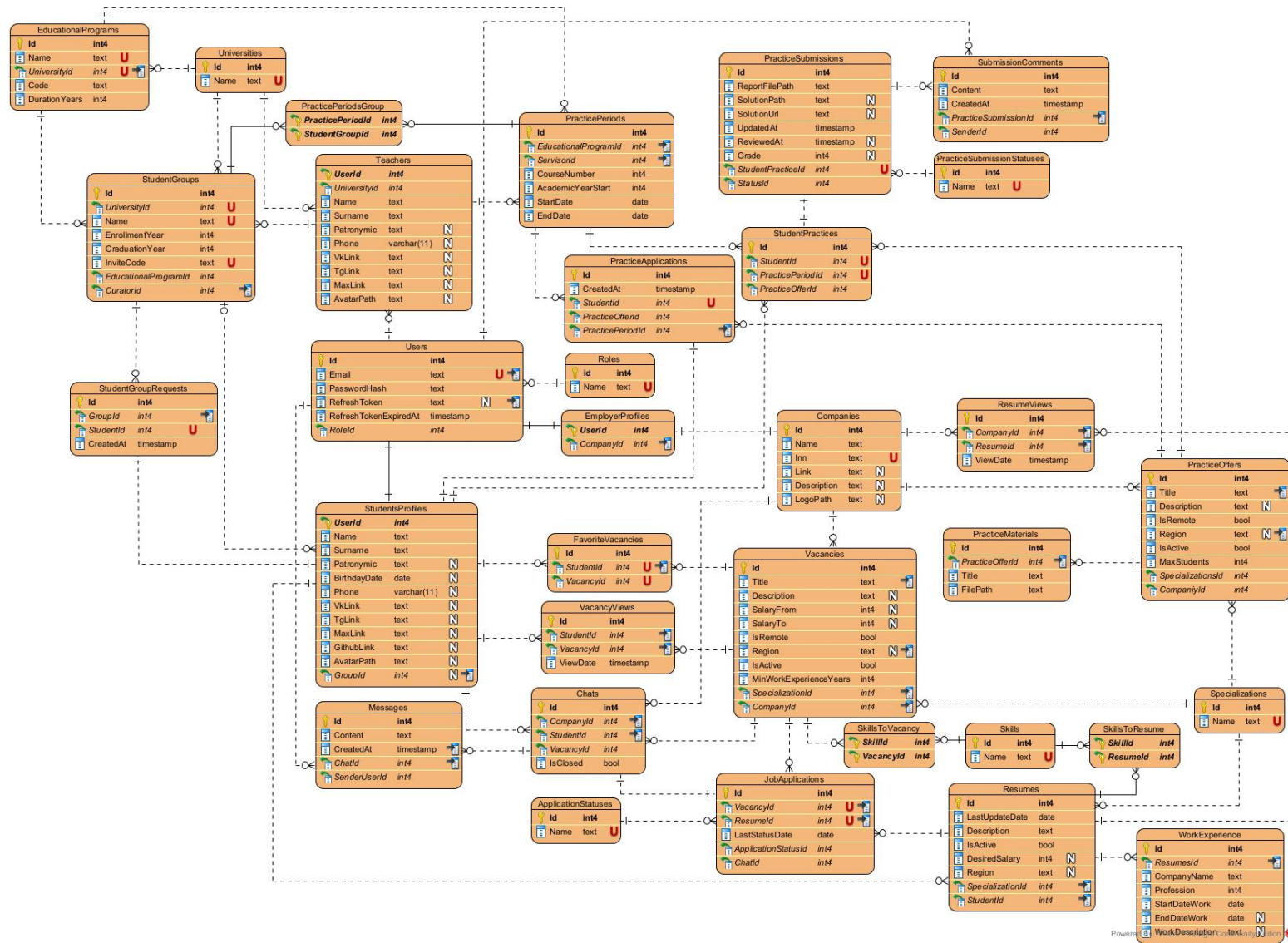


Рисунок 22 – Диаграмма физической модели базы данных информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

Таким образом в ходе физического моделирования базы данных была построена диаграмма физической модели базы данных информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях, а также средствами Visual Paradigm был сгенерирован скрипт для создания объектов базы данных.

4.1.2 Развертывания базы данных в среде pgAdmin 4

Для развертывания, администрирования и последующего сопровождения базы данных PostgreSQL используется специализированное программное обеспечение – pgAdmin 4. Данный инструмент предоставляет графический интерфейс для управления объектами СУБД, позволяя осуществлять визуальный контроль над структурой таблиц. В процессе развертывания базы данных средствами pgAdmin 4 выполняется подключение к целевому серверу и инициализация схемы данных с помощью сгенерированного средствами Visual Paradigm SQL-скрипта. Кроме того, встроенный в pgAdmin 4 инструмент анализа запросов позволяет выполнять мониторинг производительности СУБД.

Перед непосредственным выполнением SQL-скрипта в среде pgAdmin 4 была создана новая пустая база данных. В настройках создаваемой базы была явно указана кодировка UTF-8 для корректного сохранения и отображения многоязычных текстовых данных.

В дальнейшем для организации взаимодействия между серверным приложением и развернутой базой данных будет использоваться объектно-реляционное отображение на базе фреймворка Entity Framework Core [16]. Он позволяет автоматизировать трансляцию объектов C# в физические таблицы СУБД PostgreSQL, обеспечивая безопасную и эффективную работу с данными, без необходимости ручного написания SQL-запросов.

В результате развертывания базы данных в СУБД PostgreSQL была сгенерирована диаграмма полученной базы данных (рисунок 23).

Рисунок 23 – Диаграмма базы данных, созданная средствами СУБД PostgreSQL

Как видно из представленной диаграммы, построенная средствами СУБД схема базы данных полностью соответствует спроектированной ранее физической модели.

4.2 Реализация взаимодействия с базой данных

4.2.1 Реализация запросов к базе данных

Для организации программного взаимодействия между серверной частью и БД используется фреймворк Entity Framework Core (EF). Использование данного фреймворка позволяет абстрагироваться от написания SQL-запросов, перенося работу с данными на уровень классов C#.

Центральным компонентом взаимодействия является класс контекст базы данных. Он выполняет роль связующего звена между доменными сущностями приложения и таблицами БД. В нём для каждой сущности объявляются свойства обобщенного типа `DbSet<T>`, которые инкапсулируют коллекции объектов БД и позволяют применять к ним методы языка C#. Фрагмент реализации контекста базы данных представлен на листинге 2. Полный код контекста в открытом репозитории проекта [17].

Листинг 2 – Фрагмент класса контекста базы данных

```
public class
  InternshipPlatformContext(DbContextOptions<InternshipPlatformConte
xt> options) : DbContext(options)
{
    public DbSet<User> Users { get; set; }
    public DbSet<Role> Roles { get; set; }
    public DbSet<StudentProfile> StudentProfiles { get; set; }
    public DbSet<Company> Companies { get; set; }
    public DbSet<EmployerProfile> EmployerProfiles { get; set; }
    public DbSet<Specialization> Specializations { get; set; }
    public DbSet<Skill> Skills { get; set; }
    public DbSet<WorkExperience> WorkExperiences { get; set; }
    public DbSet<Resume> Resumes { get; set; }
    public DbSet<Vacancy> Vacancies { get; set; }
}
```

Непосредственная выборка данных из коллекций производится с использованием встроенного в C# языка интегрированных запросов LINQ (Language Integrated Query) [18]. При выполнении таких запросов фреймворк EF

осуществляет автоматическую трансляцию C#-кода в оптимизированный SQL-скрипт и передаёт его на исполнение серверу БД. Для предотвращения блокировок потоков ввода-вывода и обеспечения высокой отзывчивости серверного приложения все запросы к БД реализуются в асинхронном режиме. На листинге 3 представлен запрос для выборки всех групп куратора и подсчета количества обучающихся в них студентов.

Листинг 3 – Код запроса для выборки всех групп куратора и подсчета количества обучающихся студентов в каждой группе

```
public Task<List<StudentGroupResult>> GetCuratorGroups(int
curatorId)
{
    return context.StudentGroups
        .AsNoTracking()
        .Where(g => g.CuratorId == curatorId)
        .Select(g => new StudentGroupResult
        {
            StudentGroup = g,
            StudentsCount = g.StudentProfiles.Count()
        })
        .ToListAsync();
}
```

Как видно из кода, первоначально происходит фильтрация всех групп, у которых идентификатор куратора совпадает с требуемым. После этого с помощью вызова `Select()` указывается объект C#, в который следует преобразовать результат выборки. Это позволяет помимо самой группы запросить количество обучающихся в ней студентов. Окончательное формирование SQL-запроса и последующая его отправка на сервер БД осуществляется в момент вызова метода `ToListAsync()`.

4.2.2 Наполнение базы данных тестовыми данными

Для проведения тестирования разрабатываемой информационной платформы необходимо наличие набора начальных данных. В рамках данного проекта процесс инициализации данных разделен на два этапа: загрузка статических справочных данных и генерация динамических тестовых массивов.

К статической справочной информации относятся перечни ролей пользователей, университетов, профессиональных навыков, специализаций публикуемых резюме и вакансий, а также статусов откликов и практик. Эти данные интегрируются в БД с использованием стандартных SQL-инструкций вставки данных INSERT непосредственно в рамках применения миграций. Такой подход гарантирует, что минимально необходимый для запуска приложения набор данных будет развернут в СУБД автоматически на этапе создания объектов БД. На листинге 4 представлен SQL-скрипт для вставки в базу данных списка ролей информационной платформы.

Листинг 4 – SQL-скрипт для вставки в базу данных списка ролей

```
INSERT INTO "Roles" ("Name")
VALUES
    ('Student'),
    ('Employer'),
    ('Teacher')
ON CONFLICT ("Name") DO NOTHING;
```

Генерация тестовых массивов реалистичных данных производится для основных бизнес-сущностей платформы, например учетных записей, профилей всех ролей, сведений о компаниях, учебных групп, резюме и вакансий. Для этого применяется библиотека Bogus, которая предоставляет широкие возможности для создания генераторов псевдослучайных данных, функционирующих на основе заданных правил. На листинге 5 представлен фрагмент кода для генерации тестовых профилей студентов с использованием библиотеки Bogus.

Листинг 5 – Фрагмент кода для генерации тестовых профилей студентов с использованием библиотеки Bogus

```
private void GenerateStudents(List<User> users)
{
    var studentProfileFaker = new Faker<StudentProfile>("ru")
        .RuleFor(p => p.Name, f => f.Name.FirstName())
        .RuleFor(p => p.Surname, f => f.Name.LastName())
        .RuleFor(p => p.Patronymic, f => f.Random.Bool(0.7f) ?
f.Name.FirstName() + "ович" : null)
        .RuleFor(p => p.BirthdayDate, f =>
    {
        var birthDate = f.Date.BetweenDateOnly(
```

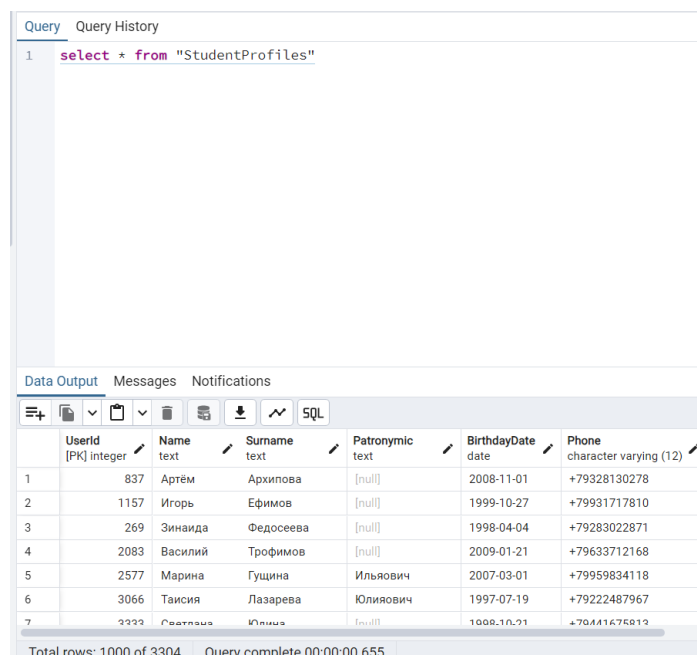
```

        DateOnly.FromDateTime(DateTime.Today.AddYears(-
30))),
        DateOnly.FromDateTime(DateTime.Today.AddYears(-
17))));

        return birthDate;
    })
    .RuleFor(p => p.Phone, f =>
f.Phone.PhoneNumber("+79#####"))
    .RuleFor(p => p.VkLink, f => f.Random.Bool(0.5f) ?
$"https://vk.com/{f.Internet.UserName()}" : null)
    .RuleFor(p => p.TgLink, f => f.Random.Bool(0.7f) ?
$"https://t.me/{f.Internet.UserName()}" : null)
    .RuleFor(p => p.MaxLink, f => f.Random.Bool(0.2f) ?
$"https://max.ru/{f.Internet.UserName()}" : null)
    .RuleFor(p => p.GithubLink, f => f.Random.Bool(0.6f) ?
$"https://github.com/{f.Internet.UserName()}" : null)
    .RuleFor(p => p.AvatarPath, _ => null);
var studentProfiles = students
.Select(user =>
{
    var profile = studentProfileFaker.Generate();
    profile.UserId = user.Id;
    return profile;
})
.ToList();
context.StudentProfiles.AddRange(studentProfiles);
context.SaveChanges();
}

```

Результат работы генератора тестовых профилей студентов представлен на рисунке 24.



	UserId [PK] integer	Name text	Surname text	Patronymic text	BirthdayDate date	Phone character varying (12)
1	837	Артём	Архипова	[null]	2008-11-01	+79328130278
2	1157	Игорь	Ефимов	[null]	1999-10-27	+79931717810
3	269	Зинаида	Федосеева	[null]	1998-04-04	+79283022871
4	2083	Василий	Трофимов	[null]	2009-01-21	+79633712168
5	2577	Марина	Гущина	Ильяович	2007-03-01	+79959834118
6	3066	Таисия	Лазарева	Юлияович	1997-07-19	+79222487967
7	2222	Святослав	Иванов	[null]	1999-10-21	+79001167612

Total rows: 1000 of 3304 Query complete 00:00:00.655

Рисунок 24 – Результат работы генератора тестовых профилей студентов

Процесс генерации выполняется при каждом запуске приложения, однако алгоритм предусматривает предварительную проверку таблиц на отсутствие записей, что предотвращает дублирование данных. Полный код генератора тестовых данных доступен в открытом репозитории проекта [17].

4.3 Реализация инфраструктуры и политик безопасности

4.3.1 Аутентификация и управление доступом

Для реализации аутентификации в системе выбран подход на основе JWT. После успешного входа система формирует подписанный токен, содержащий идентификатор пользователя и набор утверждений (claims), и возвращает его клиентскому приложению для последующего использования. Код формирования токена представлен на листинге 6. Клиент передаёт токен вместе с каждым запросом, а серверное приложение выполняет проверку подписи и срока действия токена, после чего принимается решение о предоставлении доступа.

Листинг 6 – Код формирования JWT-токена

```
private List<Claim> CreateClaims(User user)
{
    var claims = new List<Claim>
    {
        new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
        new Claim(ClaimTypes.Email, user.Email),
        new Claim(ClaimTypes.Role, user.Role.Name)
    };

    return claims;
}

public string GenerateAccessToken(User user)
{
    var claims = CreateClaims(user);

    var signingCredentials = new SigningCredentials(
        new
        SymmetricSecurityKey(Encoding.UTF8.GetBytes(TokenOptions.Key)),
        SecurityAlgorithms.HmacSha256);

    var token = new JwtSecurityToken(
        signingCredentials: signingCredentials,
        issuer: TokenOptions.Issuer,
        audience: TokenOptions.Audience,
```

```

        claims: claims,
        expires:
DateTime.UtcNow.AddHours(TokenOptions.ExpiresAfterHours));

return new JwtSecurityTokenHandler().WriteToken(token);
}

```

Для защиты программных интерфейсов используется атрибут [Authorize], который ограничивает доступ к методам контроллера пользователям, не аутентифицированным в системе. Для разграничения прав доступа ролевой модели для этого атрибута в качестве параметра указывается допускаемая роль, например: [Authorize(Roles = «Student»)].

Однако для реализации сложной бизнес-логики системы стандартной ролевой модели недостаточно. В частности, на этапе проектирования было предусмотрено правило: студент получает доступ к основному функционалу платформы исключительно после зачисления куратором в учебную группу. Для решения этой задачи в систему была добавлена политика для проверки принадлежности студента к учебной группе. Код этой политики представлен на листинге 7.

Листинг 7 – Политика для предоставления ограничения доступа к функционалу студентам без группы

```

public sealed class StudentMustHaveGroupHandler(
    IStudentProfileRepository studentProfileRepository)
    : AuthorizationHandler<StudentMustHaveGroupRequirement>
{
    protected override async Task HandleRequirementAsync(
        AuthorizationHandlerContext context,
        StudentMustHaveGroupRequirement requirement)
    {
        if (!context.User.Identity?.IsAuthenticated ?? true)
            return;

        if (!context.User.IsInRole(Roles.Student))
            return;

        if
(!int.TryParse(context.User.FindFirstValue(ClaimTypes.NameIdentifier), out int studentId))
            return;

        if (!await
studentProfileRepository.IsStudentHasGroup(studentId))

```

```

        return;

        context.Succeed(requirement);
    }
}

```

Ограничение доступа к контроллерам и методам применяется путём указания разработанной политики в параметрах атрибута: [Authorize(Policy = Policies.StudentMustHaveGroup)].

Таким образом внедрение JWT-токенов в связке с разработанной политикой безопасности позволяет организовать надежный и гибкий механизм контроля доступа.

4.3.2 Реализация файловой подсистемы

Для сопровождения производственных практик в рамках платформы разработана подсистема управления файлами. Данный функционал критически важен, поскольку позволяет представителям компаний загружать материалы для прохождения практик, а студентам – прикреплять отчёты и архивы с выполненными заданиями.

Архитектура хранения данных построена на разделении физического размещения файлов и их метаданных. При загрузке файла серверное приложение принимает его в виде потока. В целях безопасности и предотвращения конфликтов имён для каждого загружаемого файла генерируется уникальное имя. Физический файл сохраняется в выделенную на сервере директорию под сгенерированным именем, а в базу данных сохраняется запись, содержащая относительный путь к файлу и его исходное имя. Код метода загрузки файлов на сервер представлен на листинге 8. Полный код сервиса управления файлами доступен в приложении Г и в открытом репозитории проекта [17].

Листинг 8 – Код метода загрузки файлов на сервер

```

private async Task<SavedFileResult> SaveFile(Stream stream,
    string originalFileName,
    string folder,
    string extension)
{
    var originalSafeFileName = Path.GetFileName(originalFileName);

```

```

        var storageRoot =
            Path.GetFullPath(Path.Combine(env.ContentRootPath, FilesFolder));
        var targetFolder =
            Path.GetFullPath(Path.Combine(env.ContentRootPath, folder));

        if (!targetFolder.StartsWith(storageRoot,
            StringComparison.Ordinal))
            throw new InvalidOperationException("Некорректный путь к папке.");

        Directory.CreateDirectory(targetFolder);

        var storedFileName = $"{Guid.NewGuid():N}{extension}";
        var fullPath = Path.Combine(targetFolder, storedFileName);

        await using var fileStream = new FileStream(
            fullPath,
            FileMode.CreateNew,
            FileAccess.Write,
            FileShare.None);

        await stream.CopyToAsync(fileStream);

        var relativePath = Path.GetRelativePath(env.ContentRootPath,
            fullPath)
            .Replace(Path.DirectorySeparatorChar, '/');

        return new SavedFileResult
        {
            OriginalFileName = originalSafeFileName,
            RelativePath = relativePath
        };
    }

```

Данный метод возвращает объект, содержащий относительный путь до файла на сервере и его исходное имя для того, чтобы вызываемый сервис мог зафиксировать эти данные в БД.

При запросе на скачивание файла с сервера формируется файловый поток к требуемому файлу. При этом название файла при скачивании подставляется из БД, благодаря чему пользователь скачивает файл под его оригинальным названием.

Операция удаления файлов с сервера реализуется в два этапа: первоначально удаляется запись о метаданных из БД, после чего выполняется физическое удаление файла с накопителя сервера. Это предотвращает наличие в БД записей о несуществующих файлах и нерациональное использование

дискового пространства. В качестве примера на листинге 9 представлен метод для удаления материала по практике.

Листинг 9 – Код метода для удаления с сервера материала по практике

```
public async Task DeleteFile(int employerId, int practiceOfferId,
int fileId)
{
    await ThrowIfEmployerDoesNotOwnPracticeOffer(employerId,
practiceOfferId);

    var material = await
practiceMaterialRepository.GetPracticeMaterialById(fileId)
    ?? throw new PracticeMaterialNotFoundException();

    if (material.PracticeOfferId != practiceOfferId)
        throw new PracticeMaterialNotFoundException();

    await
practiceMaterialRepository.DeletePracticeMaterial(material);
    await unitOfWork.SaveChangesAsync();

    await
fileStorageService.DeleteFileIfExists(material.FilePath);
}
```

Таким образом разработанная подсистема управления файлами обеспечивает безопасное и надежное хранение пользовательских документов.

4.3.3 Интеграция брокера сообщений

Для организации взаимодействия с сервисом рассылки уведомлений в архитектуру серверного приложения интегрируется брокер сообщений Apache Kafka. Монолитное приложение формирует данные о требуемом уведомлении и асинхронно отправляет его брокеру. При таком подходе монолитное приложение не выполняет побочных операций, следовательно время отклика не увеличивается, а вся нагрузка по доставке уведомлений делегируется отдельному сервису.

Для отправки сообщений реализован отдельный класс, инкапсулирующий логику взаимодействия с брокером Kafka. При наступлении целевого события, монолитное приложение формирует объект уведомления, сериализует его в

формат JSON и асинхронно публикует в выделенный топик брокера. Код класса-издателя представлен на листинге 10.

Листинг 10 – Код класса для публикации сообщений в Apache Kafka

```
public class NotificationProducer(IOptions<KafkaSettings>
kafkaOptions)
{
    private const string Topic = "PlatformNotifications";

    public async Task SendNotificationAsync(NotificationEvent
notificationEvent)
    {
        var config = new ProducerConfig
        {
            BootstrapServers =
kafkaOptions.Value.BootstrapServers,
            AllowAutoCreateTopics = true,
            Acks = Acks.All
        };

        using (var producer = new ProducerBuilder<Null,
NotificationEvent>(config)
            .SetValueSerializer(new
JsonSerializer<NotificationEvent>())
            .Build())
        {
            await producer.ProduceAsync(topic: Topic,
                new Message<Null, NotificationEvent>
                {
                    Value = notificationEvent
                });

            producer.Flush();
        }
    }
}
```

Сервису уведомлений необходимо подписаться на тот же самый топик. В таком случае он сможет получать из него сообщения, выполнять десериализацию и осуществлять рассылку уведомлений на электронную почту.

Таким образом разработанный класс-издатель обеспечивает отправку событий в асинхронном режиме в брокер сообщений для последующей их обработки сервисом рассылки уведомлений.

4.4 Реализация бизнес-логики

4.4.1 Реализация встроенных чатов

Для предоставления пользователям удобного инструмента обсуждения деталей вакансий в платформу интегрирована подсистема встроенной коммуникации. Для обеспечения мгновенной доставки сообщений и обновления чатов в реальном времени используется протокол WebSocket. Он предоставляет двустороннее постоянное соединение между клиентом и сервером, что позволяет серверу инициировать отправку данных клиенту без перезагрузки страницы.

В качестве реализации данного протокола использовалась библиотека SignalR. Центром установления двустороннего соединения между пользователями является хаб. Он обрабатывает входящие события и рассылает всем подписчикам уведомления об обновлении. На листинге 11 представлен код метода для подключения к конкретному чату.

Листинг 11 – Код метода для подключения к чату

```
public async Task JoinChat(int chatId)
{
    try
    {
        await chatService.ThrowIfUserCantAccessChat(UserId, Role,
chatId);

        await Groups.AddToGroupAsync(Context.ConnectionId,
GetChatGroup(chatId));
    }
    catch (Exception ex)
    {
        throw new HubException(ex.Message);
    }
}
```

Как видно из кода, первоначально выполняется верификация прав доступа пользователя к запрашиваемому чату. При успешной проверке идентификатор клиентского соединения добавляется в группу подключений, соответствующую чату. Это позволяет рассылать уведомления всем участникам группы.

Процесс отправки включает в себя предварительное сохранение сообщения в базу данных с последующей его рассылкой всем участникам группы. Логика метода отправки сообщений представлена на листинге 12.

Листинг 12 – Код метода хаба для сохранения и отправки сообщений

```
public async Task SendMessage(SendMessageRequest request)
{
    if (string.IsNullOrEmpty(request.Content))
        throw new HubException("Пустое сообщение");

    request.SenderUserId = UserId;
    request.Role = Role;

    MessageResponse message;

    try
    {
        message = await messageService.SendMessage(request);
    }
    catch (Exception ex)
    {
        throw new HubException(ex.Message);
    }

    await
    Clients.Group(GetChatGroup(request.ChatId)).ReceiveMessage(message
    );

    await
    Clients.Group(GetChatGroup(request.ChatId)).ChatUpdate(request.ChatId);
}
```

Важно отметить, что протокол WebSocket не поддерживает модификацию пользовательских HTTP-заголовков (в частности заголовка Authorization), поэтому при обращении к хабу следует передавать JWT-токен в строке запроса (Query-параметр). Пример URL-адреса инициализации подключения к хабу: `http://localhost:5228/hubs/chat?access_token=<токен_авторизации>`.

Помимо этого, библиотека SignalR реализует собственный протокол поверх стандартного WebSocket-соединения. Каждое отправляемое сообщение представляет собой JSON-объект, который в обязательном порядке должен завершаться специальным символом-разделителем записей `\u001e`. Установление взаимодействия начинается с обязательной отправки клиентом

сообщения рукопожатия, после чего клиент может обращаться к методам хаба. На листинге 13 представлены примеры JSON-сообщений для установления рукопожатия и вызова методов хаба.

Листинг 13 – Примеры JSON-сообщений для взаимодействия с хабом

```
// обязательный запрос рукопожатия
{
  "protocol": "json",
  "version": 1
}\u001e

// подключение к чату
{
  "type": 1,
  "target": "JoinChat",
  "arguments": [
    101
  ]
}\u001e

// отправка сообщений
{
  "type": 1,
  "target": "SendMessage",
  "arguments": [
    {
      "chatId": 101,
      "content": "Здравствуйте! Меня интересует позиция стажера."
    }
  ]
}\u001e
```

Таким образом использование протокола WebSocket позволяет организовать встроенный в платформу канал связи реального времени, избавляя пользователей от необходимости использовать сторонние мессенджеры.

4.4.2 Реализация алгоритма рекомендаций вакансий и резюме

Для повышения эффективности процесса трудоустройства в платформу интегрирован алгоритм рекомендаций. Данный алгоритм осуществляет автоматическое сопоставление активных резюме студентов и открытых вакансий на основе заданных критериев.

Процесс поиска строится в два этапа: фильтрация и ранжирование. На этапе фильтрации формируется базовый запрос, который отсекает заведомо

неподходящие позиции. К строгим критериям относятся специализация, опыт работы и зарплатные ожидания.

После отсева нерелевантных записей выполняется ранжирование. Для каждой сущности динамически вычисляется метрика релевантности, состоящая из двух компонентов:

- оценка навыков: учитывает количество совпадающих навыков между резюме и вакансией;
- оценка региона: совпадение региона вакансии с местом проживания студента или доступность удаленного формата работы повышают приоритет.

На основании вычисленной метрики происходит сортировка записей, таким образом пользователь получает наиболее релевантные предложения в самом начале списка. Фрагмент кода, реализующий расчет метрики релевантности вакансий для резюме представлен на листинге 14. Полный код алгоритма доступен в открытом репозитории проекта [17]. Алгоритм подбора рекомендованных резюме для вакансий функционирует по аналогичному принципу.

Листинг 14 – Фрагмент алгоритма подбора рекомендованных вакансий под резюме

```
var rankedQuery = baseQuery
    .Select(v => new
    {
        VacancyId = v.Id,
        SkillsMatchCount = resumeSkillIds.Count == 0
            ? 0
            : v.Skills.Count(s =>
resumeSkillIds.Contains(s.Id)),
        RegionScore = hasRegion
            ? (v.Region == region ? 3 :
                v.IsRemote ? 2 : 1)
            : (v.IsRemote ? 3 : 1),
        v.Id
    })
    .Select(x => new
    {
        x.VacancyId,
        x.SkillsMatchCount,
        x.RegionScore,
```

```

        TotalScore = x.SkillsMatchCount * 10 +
x.RegionScore,
        x.Id
    });

```

Таким образом реализованный алгоритм сопоставления вакансий и резюме позволяет формировать для пользователей персонализированную ленту рекомендаций.

4.4.3 Реализация сопровождения жизненного цикла откликов

Одной из ключевых функций платформ трудоустройства является создание откликов на вакансии и управление ими. Поскольку именно через отклики осуществляется процесс найма сотрудников, корректная программная реализация данного функционала выступает одной из приоритетных задач. Взаимодействие студента с работодателем инициируется подачей электронной заявки, которая может быть сформирована как студентом, так и работодателем в виде прямого приглашения на собеседование.

Жизненный цикл отклика описывается в системе конечным набором состояний:

- ожидает рассмотрения;
- приглашен на собеседование;
- получено предложение на трудоустройство;
- студент принял предложение;
- студент официально трудоустроен;
- отказ со стороны работодателя;
- отзыв отклика по инициативе студента.

Смена статусов строго определяется ролевой моделью платформы. Например, работодатель может пригласить кандидата на собеседование или отклонить заявку, а студент – принять предложение или отозвать своё резюме. Для реализации смены статусов используется алгоритм сопоставления с образцом, который учитывает текущий и целевой статус заявки и роль пользователя, инициирующего изменение. Для каждого набора их этих трёх

значений определено правило перехода. Код метода, реализующего проверку возможности изменения статуса представлен на листинге 15.

Листинг 15 – Код метода, реализующего проверку возможности перевода пользователем отклика из одного состояния в другое

```
private bool IsValidStatusTransition(string role,
JobApplicationStatuses currentStatus, JobApplicationStatuses
newStatus)
{
    if (currentStatus == newStatus)
        return false;

    return (currentStatus, newStatus, role) switch
    {
        (JobApplicationStatuses.Pending,
JobApplicationStatuses.InterviewInvited, Roles.Employer) => true,
        (JobApplicationStatuses.Pending,
JobApplicationStatuses.Rejected, Roles.Employer) => true,
        (JobApplicationStatuses.Pending,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,

        (JobApplicationStatuses.InterviewInvited,
JobApplicationStatuses.OfferReceived, Roles.Employer) => true,
        (JobApplicationStatuses.InterviewInvited,
JobApplicationStatuses.Rejected, Roles.Employer) => true,
        (JobApplicationStatuses.InterviewInvited,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,

        (JobApplicationStatuses.OfferReceived,
JobApplicationStatuses.Accepted, Roles.Student) => true,
        (JobApplicationStatuses.OfferReceived,
JobApplicationStatuses.Rejected, Roles.Employer) => true,
        (JobApplicationStatuses.OfferReceived,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,

        (JobApplicationStatuses.Accepted,
JobApplicationStatuses.Employed, Roles.Employer) => true,
        (JobApplicationStatuses.Accepted,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,

        _ => false
    };
}
```

На этапе создания отклика платформа выполняет множество строгих проверок. Во-первых, необходимо убедиться в существовании резюме, а также в том, что оно является активным и принадлежит студенту. Во-вторых, необходимо проверить существование и активность вакансии. В-третьих,

необходимо убедиться, что студент не имеет активного отклика на эту вакансию (активным считается отклик, который не отклонен работодателем и не отозван студентом). Помимо этого, необходимо проверить существование чата между студентом и представителем компании по этой вакансии – если его нет, то создать новый. В качестве примера на листинге 16 приведён метод создания отклика студентом. Полный код класса сервиса, сопровождающего жизненный цикл откликов, доступен в приложении Г и в открытом репозитории проекта [17].

Листинг 16 – Метод создания отклика студентом

```
public async Task<int>
CreateJobApplicationByStudent(CreateJobApplicationRequest request)
{
    if (!await
resumeRepository.IsStudentOwnsResume(request.UserId,
request.ResumeId))
        throw new ResumeNotFoundException();

    if (!await
vacancyRepository.IsVacancyExistsAndActive(request.VacancyId))
        throw new VacancyNotFoundException();

    if (await
applicationRepository.HasStudentActiveApplicationOnVacancy(request
.ResumeId, request.VacancyId))
        throw new ActiveApplicationAlreadyExistsException();

    var chatId = await chatService.GetOrCreateStudentChat(new
StartStudentChatRequest
    {
        StudentId = request.UserId,
        VacancyId = request.VacancyId,
        Message = request.WelcomeMessage
    });

    var application = request.ToDomain(Roles.Student, chatId);
    await applicationRepository.AddJobApplication(application);

    await unitOfWork.SaveChangesAsync();

    return application.Id;
}
```

Чат привязывается к отклику и используется как канал для обсуждения деталей конкретной вакансии. При закрытии отклика по инициативе любой из сторон чат также закрывается для отправки сообщений.

Таким образом, реализованная логика сопровождения откликов обеспечивает строгий контроль за жизненным циклом заявок на трудоустройство. Все возможные варианты переходов статусов жестко зафиксированы в коде и базируются на ролевой модели платформы, что полностью исключает возможность несанкционированного или логически неверного изменения статусов.

4.4.4 Реализация сопровождения жизненного цикла производственных практик

Производственные практики выступают связующим звеном между всеми ролями платформы: студентами, работодателями и преподавателями. Жизненный цикл практики разделён на три этапа:

- запись на практику на предприятие;
- загрузка отчетности;
- оценивание отчетности.

Процесс подачи заявки на прохождение практики сопровождается каскадом строгих проверок, направленных на защиту базы данных от коллизий. На листинге 17 представлен код метода для подачи заявки на практику.

Листинг 17 – Код метода для подачи заявки на практику

```
public async Task<int>
CreatePracticeApplication(CreatePracticeApplicationRequest
request)
{
    if (await
practiceApplicationRepository.HasActivePracticeApplication(request
.StudentId))
        throw new
ActivePracticeApplicationAlreadyExistsException();

    var practicePeriod = await
practicePeriodRepository.GetCurrentStudentPracticePeriod(request.S
tudentId)
```

```

        ?? throw new
        CurrentSemesterPracticePeriodNotFoundExpection();

        if (IsApplicationPeriodClosed(practicePeriod))
            throw new
            PracticeApplicationPeriodAlreadyStartedExpection();

        if (!await
        practiceOfferRepository.IsPracticeOfferExistsAndActive(request.PracticeOfferId))
            throw new PracticeOfferNotFoundExpection();

        if (!await
        practiceApplicationRepository.HasPracticeOfferAvailablePlaces(request.PracticeOfferId))
            throw new PracticeOfferHasNoAvailablePlacesExpection();

        if (await
        practiceApplicationRepository.IsStudentAlreadyEnrolledInPracticePeriod(request.StudentId, practicePeriod.Id))
            throw new StudentAlreadyEnrolledInPracticeExpection();

        var application = request.ToDomain(practicePeriod);

        await
        practiceApplicationRepository.AddPracticeApplication(application);
        await unitOfWork.SaveChangesAsync();

        return application.Id;
    }

```

Как видно из представленного выше кода, алгоритм проверяет выполнение следующих условий перед сохранением заявки:

- студент не имеет других активных заявок на практику;
- за учебной группой студента закреплена практика в текущем семестре;
- период подачи заявок ещё не закрыт (то есть практика не началась);
- наличие свободных мест по выбранному предложению;
- студент ещё не зачислен на практику в другую организацию.

В случае одобрения заявки работодателем студент зачисляется на практику, и ему открывается доступ к скачиванию материалов и заданий, загруженных представителем компании.

После официального старта периода практики студенту предоставляется возможность загрузки своей работы. Для успешного прохождения практики

необходимо загрузить в систему отчет по практике, а также, если это предусмотрено заданием, архив с решением или внешнюю ссылку на него. Загруженные студентом файлы сохраняются на диске, а в базе данных создаётся соответствующая запись о загруженных материалах. При повторной загрузке система автоматически удаляет устаревшие файлы с накопителя в целях экономии дискового пространства.

Важной составляющей прохождения производственной практики является получение обратной связи. Для этого в платформу интегрирована подсистема комментирования: курирующий практику преподаватель или представитель организации могут оставлять текстовые комментарии к загруженному решению студента. Важно отметить, что комментарии относятся к решению, а не к практике, поэтому возможность отправки комментариев открывается только после публикации студентом работы.

Сам процесс прохождения практики включает в себя следующие статусы:

- работа загружена и ожидает проверки;
- требуются исправления (работодатель проверил работу и выявил ошибки);
- работа принята представителем компании;
- работа оценена (преподаватель выставил оценку).

Выставление оценки является завершающей стадией прохождения практики и открывается преподавателю исключительно после принятия отчета представителем компании. На листинге 18 представлен код метода для выставления оценки. Полный код классов сервисов, сопровождающих жизненный цикл производственной практики, доступен в открытом репозитории проекта [17].

Листинг 18 – Код метода для выставления оценки преподавателем

```
public async Task GradeStudentPractice(int teacherId, int
studentPracticeId, GradePracticeSubmissionRequest request)
{
    var practice = await studentPracticeRepository
```

```

        .GetTeacherStudentPracticeForUpdate(teacherId,
studentPracticeId)
        ?? throw new StudentPracticeNotFoundException();

    var submission = GetSubmissionOrThrow(practice);

    if (submission.StatusId !=
(int)PracticeSubmissionStatuses.CompanyAccepted
        && submission.StatusId !=
(int)PracticeSubmissionStatuses.Graded)
        throw new PracticeSubmissionCannotBeGradedException();

    submission.Grade = request.Grade;
    submission.StatusId = (int)PracticeSubmissionStatuses.Graded;
    submission.ReviewedAt = DateTime.UtcNow;

    await unitOfWork.SaveChangesAsync();
}

```

Таким образом, реализованная логика сопровождения производственной практики обеспечивает автоматизацию взаимодействия между студентами, компаниями и университетами. Строгая логика валидации на каждом этапе жизненного цикла исключает возможность возникновения коллизий в базе данных, а интегрированный функционал электронного документооборота и комментирования делает процесс прохождения практики прозрачным и контролируемым для всех участников платформы.

4.5 Реализация сервиса рассылки уведомлений

Отправка уведомлений является дополнительным функционалом, не относящимся к основной бизнес-логике предметной области платформы. Поэтому в дополнение к монолитному приложению был реализован независимый сервис рассылки уведомлений. Помимо этого, данное решение позволяет не нагружать основное серверное приложение побочными задачами, что положительно сказывается на времени отклика для пользователей. Реализованный сервис не имеет прямых связей с серверным приложением: для асинхронного обмена данными используется брокер сообщений Apache Kafka.

Сервис спроектирован в виде фонового процесса. При запуске он устанавливает постоянное соединение с брокером и оформляет подписку на целевой топик уведомлений. Фоновый процесс непрерывно прослушивает

очередь брокера: при получении очередного события, он формирует сообщение и отправляет его на электронную почту адресата. Код метода для чтения сообщений из брокера представлен на листинге 19. Полный код сервиса рассылки уведомлений доступен в приложении Г и в открытом репозитории проекта [17].

Листинг 19 – Код метода фонового сервиса для чтения сообщений из Apache Kafka

```
protected override Task ExecuteAsync(CancellationToken
stoppingToken)
{
    var config = new ConsumerConfig
    {
        GroupId = "InternshipPlatform",
        BootstrapServers = kafkaOptions.Value.BootstrapServers,
        AutoOffsetReset = AutoOffsetReset.Earliest,
    };

    Task.Run(async () =>
    {
        using (var consumer = new ConsumerBuilder<Ignore,
NotificationEvent>(config)
            .SetValueDeserializer(new
JsonDeserializer<NotificationEvent>())
            .Build())
        {
            consumer.Subscribe(Topic);

            while (!stoppingToken.IsCancellationRequested)
            {
                var consumerResult =
consumer.Consume(TimeSpan.FromSeconds(5));

                if (consumerResult == null)
                    continue;

                using var scope = scopeFactory.CreateScope();

                var emailSender = scope.ServiceProvider
                    .GetRequiredService<IEmailSender>();

                var emailMessage = consumerResult.Message.Value;

                await emailSender.SendEmailAsync(emailMessage);

                consumer.Commit(consumerResult);
            }
        }
    })
}
```

```

    }, stoppingToken);

    return Task.CompletedTask;
}

```

Непосредственная отправка писем на электронную почту по протоколу SMTP реализована с использованием современной библиотеки MailKit. Процесс формирования письма с последующей его отправкой на электронную почту продемонстрирован на листинге 20.

Листинг 20 – Код метода отправки сообщений на электронную почту

```

public async Task SendEmailAsync(NotificationEvent
notificationEvent)
{
    var emailMessage = new MimeMessage();
    emailMessage.From.Add(new MailboxAddress("InternshipPlatform",
EmailOptionsValue.Email));
    emailMessage.To.Add(new MailboxAddress("",
notificationEvent.Email));
    emailMessage.Subject = notificationEvent.Title;
    emailMessage.Body = new TextPart("plain") { Text =
notificationEvent.Message };

    using var client = new SmtplibClient();

    try
    {
        await client.ConnectAsync(
            EmailOptionsValue.Host,
            EmailOptionsValue.Port,
            SecureSocketOptions.SslOnConnect);
        await client.AuthenticateAsync(EmailOptionsValue.Email,
EmailOptionsValue.Password);

        await client.SendAsync(emailMessage);

        await client.DisconnectAsync(true);
    }
    catch (Exception ex)
    {
        logger.LogError(ex, ex.Message);
    }
}

```

Таким образом, реализованный сервис рассылки уведомлений позволяет организовать надежную асинхронную рассылку писем. Использование брокера сообщений Apache Kafka обеспечивает высокую отказоустойчивость платформы и гарантирует сохранность писем до момента их обработки.

4.6 Контейнеризация и развертывание информационной платформы

Для обеспечения единой конфигурации при развертывании информационной платформы использовалась технология контейнеризации Docker. Серверное приложение и сервис рассылки уведомлений изолированы в самостоятельные Docker-образы, внутри которых содержится скомпилированный код приложений, среда исполнения .NET и все необходимые зависимости.

Оркестрация и одновременный запуск всех контейнеров осуществляется с помощью инструмента Docker Compose. Он позволяет описать архитектуру приложения, состоящего из нескольких компонентов, в одном конфигурационном файле и запускать всю систему одной командой. Docker Compose предоставляет синтаксис для определения строгой последовательности запуска компонентов, а также для настройки условий проверки их работоспособности. Помимо этого, для постоянного хранения пользовательских данных используются подключаемые тома (volumes). Фрагмент конфигурационного файла Docker Compose представлен на листинге 21. Полный конфигурационный файл доступен в открытом репозитории проекта [17].

Листинг 21 – Фрагмент конфигурационного файла Docker Compose

```
notificationsservice:
  image: ${DOCKER_REGISTRY-}notificationsservice
  build:
    context: .
    dockerfile: NotificationService/Dockerfile
  ports:
    - "5003:8080"
  environment:
    - ASPNETCORE_ENVIRONMENT=Development
    - ASPNETCORE_URLS=http://+:8080
    - Kafka__BootstrapServers=kafka:9092
  depends_on:
    internshipplatform.api:
      condition: service_started

postgres_internship_platform:
  image: postgres:15-alpine
  environment:
    - POSTGRES_DB=InternshipPlatform
```

```

- POSTGRES_USER=postgres
- POSTGRES_PASSWORD=admin
volumes:
- pgdata_internship_platform:/var/lib/postgresql/data
ports:
- "5433:5432"
healthcheck:
  test: ["CMD-SHELL", "pg_isready"]
  interval: 10s
  timeout: 5s
  retries: 5

```

Физическое расположение компонентов системы отражено на диаграмме развертывания (рисунок 25).

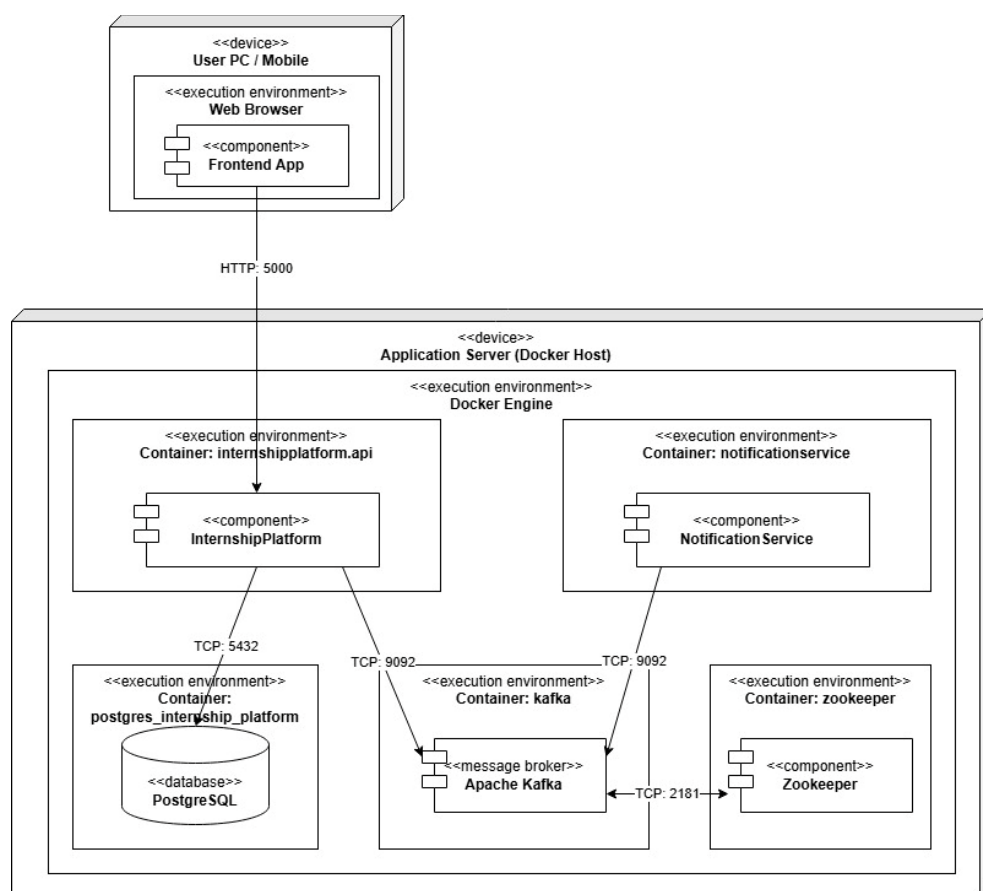


Рисунок 25 – Диаграмма развертывания информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

По результатам кодирования была разработана информационная платформу трудоустройства и стажировок студентов в ИТ-компаниях. Реализованное серверное приложение было запущено согласно диаграмме развертывания.

5 Тестирование серверной части информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях

Тестирование является важным этапом жизненного цикла разработки программного обеспечения. Его цель заключается в проверке соответствия реализованного функционала поставленным требованиям.

Разработанная информационная платформа включает в себя множество бизнес-процессов, но центральным выступает процесс трудоустройства. Он отвечает за сопровождение полного жизненного цикла отклика на вакансию. Поэтому тестирование будет рассмотрено на примере именно этого бизнес-процесса.

5.1 Ручное функциональное тестирование методом «белого ящика» для процесса трудоустройства

Тестирование по стратегии «белого ящика» предполагает проверку внутренней логики работы программы и её корректности написания с точки зрения работы компилятора языка, на котором она реализована [19]. Такой подход к тестированию позволяет проверить внутреннюю структуру программы. Исходя из этой стратегии, тестировщик формирует тестовые наборы данных путём анализа логики работы программы. В рамках данного исследования тестирование методом «белого ящика» включает в себя следующие критерии:

- покрытие операторов;
- покрытие условий;
- покрытие маршрутов.

Для корректного проектирования тестовых сценариев требуется построение управляющего графа тестируемого алгоритма. В качестве него была выбрана ранее спроектированная диаграмма деятельности алгоритма трудоустройства. Для обеспечения удобства отслеживания путей и наглядного обращения к конкретным блокам при описании тестов, каждому оператору присвоена буква латинского алфавита (рисунок 26).

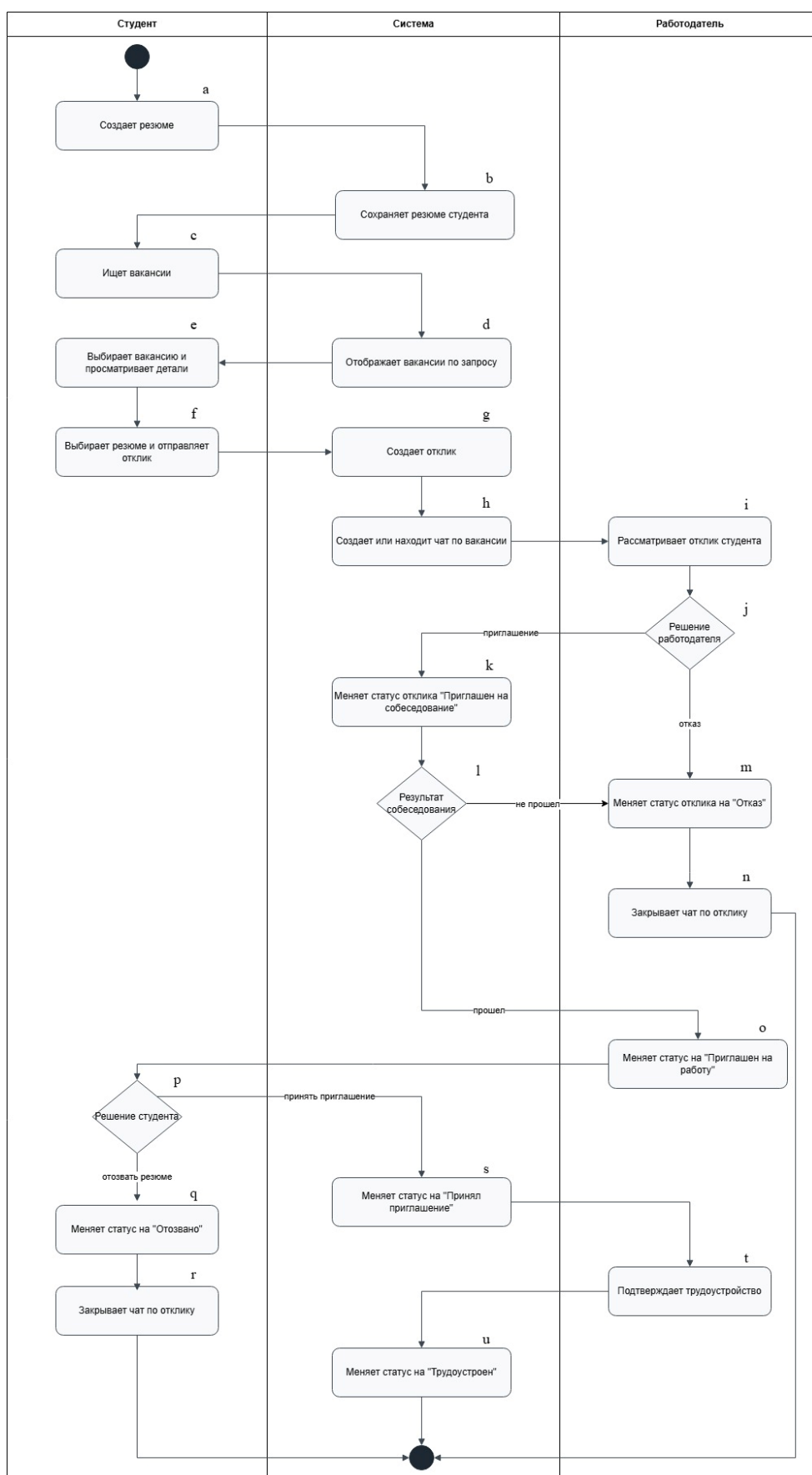


Рисунок 26 – Размеченная диаграмма деятельности процесса трудоустройства

На основе представленной размеченной схемы будут последовательно разработаны тесты по всем заявленным критериям.

5.1.1 Покрытие операторов

Критерий покрытия операторов требует разработки такого набора тестовых сценариев, при выполнении которых каждый оператор программы будет выполнен хотя бы один раз. Разработанные для алгоритма трудоустройства сценарии и результаты их тестирования представлены в таблице 25.

Таблица 25 – Результат тестирования методом покрытия операторов

№	Операторы	Входные данные	Выходные данные	Фактический результат
1	a, b, c, d, e, f, g, h, i, j, k, l, m, n	Работодатель отклоняет отклик после собеседования.	Отклик перешёл в статус «Отклонено» и чат по отклику закрылся.	Отклик перешёл в статус «Отклонено» (рисунок Д.1 – Д.2). Чат по отклику закрылся для отправки (рисунок Д.3).
2	a, b, c, d, e, f, g, h, i, j, k, l, o, p, q, r	Студент отзывает резюме после приглашения на работу.	Отклик перешёл в статус «Отозвано».	Отклик перешёл в статус «Отозвано» (рисунок Д.4 – Д.5).
3	a, b, c, d, e, f, g, h, i, j, k, l, o, p, s, t, u	Студент принял приглашение на работу. Работодатель подтвердил трудоустройство.	Отклик перешёл в статус «Трудоустроен»	Отклик перешёл в статус «Трудоустроен» (рисунок Д.6 – Д.8).

Как видно из представленной таблицы тестирования методом покрытия операторов, спроектированные сценарии позволяют успешно выполнить каждый из операторов алгоритма.

5.1.2 Покрытие условий

Следующим критерием тестирования методом «белого ящика» является покрытие условий. Он требует составить такое число тестов, чтобы все возможные исходы условий в алгоритме выполнялись хотя бы раз.

В рассматриваемом алгоритме узлами принятия решений являются:

- узел j: ветви «приглашение» и «отказ»;

- узел l: ветви «прошел» и «не прошел»;
- узел p: ветви «принять приглашение» и «отозвать резюме».

Комбинируя ветви узлов возможно составить четыре сценария. Они рассмотрены в таблице 26.

Таблица 26 – Результат тестирования методом покрытия условий

№	Условия	Входные данные	Выходные данные	Фактический результат
1	j – отказ	Работодатель отклоняет отклик при рассмотрении.	Отклик перешёл в статус «Отклонено» и чат по отклику закрылся.	Отклик перешёл в статус «Отклонено» (рисунок Д.1 – Д.2). Чат по отклику закрылся для отправки (рисунок Д.3).
2	j – приглашение; l – не прошел	Работодатель отклоняет отклик после собеседования.	Отклик перешёл в статус «Отклонено» и чат по отклику закрылся.	Отклик перешёл в статус «Отклонено» (рисунок Д.1 – Д.2). Чат по отклику закрылся для отправки (рисунок Д.3).
3	j – приглашение; l – прошел; p – отозвать резюме	Студент отзывает резюме после приглашения на работу.	Отклик перешёл в статус «Отозвано».	Отклик перешёл в статус «Отозвано» (рисунок Д.4 – Д.5).
4	j – приглашение; l – прошел; p – принять приглашение	Студент принял приглашение на работу. Работодатель подтвердил трудоустройство.	Отклик перешёл в статус «Трудоустроен».	Отклик перешёл в статус «Трудоустроен» (рисунок Д.6 – Д.8).

Успешное выполнение всех четырёх представленных сценариев показывает, что механизм условных переходов в алгоритме реализован корректно.

5.1.3 Покрытие маршрутов

Завершающим критерием тестирования методом «белого ящика» является покрытие маршрутов. Он требует подобрать такие тестовые сценарии, при котором будут выполнены все возможные пути выполнения алгоритма от начального узла до конечного.

В таблице 27 представлены результаты тестирования алгоритма с использованием всех возможных маршрутов.

Таблица 27 – Результат тестирования методом покрытия маршрутов

№	Маршруты	Входные данные	Выходные данные	Фактический результат
1	a, b, c, d, e, f, g, h, i, j(отказ), m, n	Работодатель отклоняет отклик при рассмотрении.	Отклик перешёл в статус «Отклонено» и чат по отклику закрылся.	Отклик перешёл в статус «Отклонено» (рисунок Д.1 – Д.2). Чат по отклику закрылся для отправки (рисунок Д.3).
2	a, b, c, d, e, f, g, h, i, j(приглашение), k, l(не прошел), m, n	Работодатель отклоняет отклик после собеседования.	Отклик перешёл в статус «Отклонено» и чат по отклику закрылся.	Отклик перешёл в статус «Отклонено» (рисунок Д.1 – Д.2). Чат по отклику закрылся для отправки (рисунок Д.3).
3	a, b, c, d, e, f, g, h, i, j(приглашение), k, l(прошел), o, p(отозвать резюме), q, r	Студент отзывает резюме после приглашения на работу.	Отклик перешёл в статус «Отозвано».	Отклик перешёл в статус «Отозвано» (рисунок Д.4 – Д.5).
4	a, b, c, d, e, f, g, h, i, j(отказ), k, l(прошел), o, p(принять приглашение), s, t, u	Студент принял приглашение на работу. Работодатель подтвердил трудоустройство.	Отклик перешёл в статус «Трудоустроен».	Отклик перешёл в статус «Трудоустроен» (рисунок Д.6 – Д.8).

Таким образом проведение ручного функционального тестирования методом «белого ящика» подтвердило логическую целостность алгоритма трудоустройства.

5.2 Модульное тестирование процесса трудоустройства

Для автоматизированной проверки выполнения алгоритма трудоустройства было проведено модульное тестирование сервисного слоя. В качестве инструмента тестирования использовался современный фреймворк xUnit [20], который является стандартом для платформы .NET.

Разработанные тесты разделены на две категории. Первая категория – тестирование жизненного цикла отклика. Тестовые методы повторяют определенные в пункте 4.1.3 маршруты выполнения алгоритма. Ниже приведено описание данных тестов:

– StudentEmployed_Scenario – проверяет маршрут успешного трудоустройства;

- `ImmediateRejection_Scenario` – проверяет маршрут отказа со стороны работодателя после рассмотрения отклика;
- `RejectionAfterInterview_Scenario` – проверяет маршрут отказа со стороны работодателя после проведения собеседования;
- `WithdrawalAfterOffer_Scenario` – проверяет маршрут отзыва резюме студентом после получения предложения о работе.

Вторая категория тестов направлена на негативное тестирование и проверку ограничений бизнес-логики. В рамках данной категории реализованы следующие проверки:

- `UpdateApplicationStatus_InvalidRole_ThrowsInvalidJobApplicationStatusException` – верифицирует работу ролевой модели (например, гарантирует, что студент не может пригласить сам себя на собеседование);
- `UpdateApplicationStatus_InvalidTransition_ThrowsInvalidJobApplicationStatusException` – проверяет невозможность нелогичных переходов (например, пресекает попытку работодателя трудоустроить студента сразу после создания отклика);
- `CreateJobApplicationByStudent_ResumeNotFound_ThrowsResumeNotFoundException` – проверяет, что студент не может отправить отклик с неактивным, чужим или несуществующим резюме;
- `CreateJobApplicationByStudent_VacancyNotFound_ThrowsVacancyNotFoundException` – проверяет, что невозможно создать отклик на несуществующую или неактивную вакансию;
- `CreateJobApplicationByStudent_ActiveApplicationExists_ThrowsActiveApplicationAlreadyExistsException` – гарантирует, что студент не сможет создать дублирующий отклик.

В качестве примера использования фреймворка `xUnit` рассмотрим фрагмент теста, проверяющего сценарий отказа сразу после рассмотрения отклика (листинг 22).

Листинг 22 – Фрагмент кода теста, проверяющего сценарий отказа сразу после рассмотрения отклика

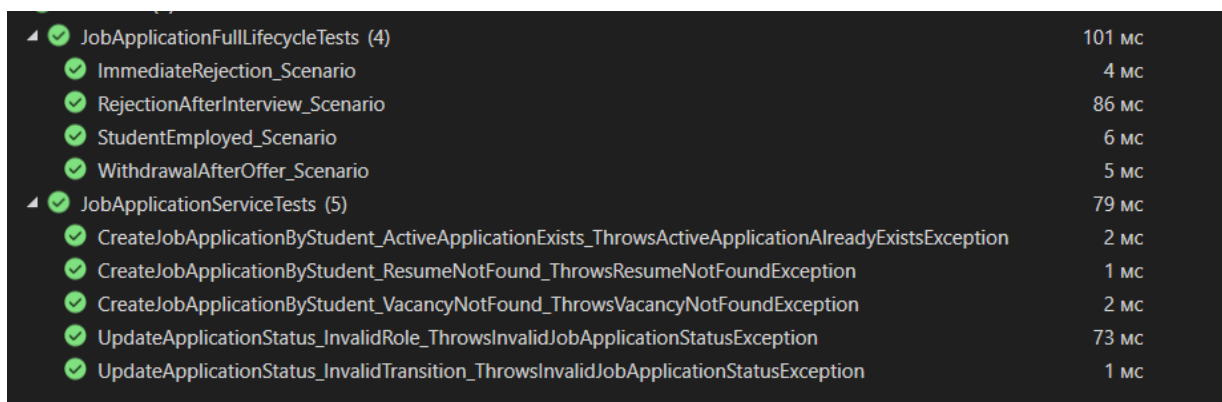
```
// Шаг 1: Создание отклика
var createRequest = new CreateJobApplicationRequest { UserId =
studentId, ResumeId = resumeId, VacancyId = vacancyId };
int createdAppId = await
_service.CreateJobApplicationByStudent(createRequest);

// Шаг 2: Работодатель сразу отказывает
await _service.UpdateApplicationStatus(new
UpdateApplicationStatusRequest
{
    UserId = employerId,
    ApplicationId = createdAppId,
    Role = Roles.Employer,
    ApplicationStatus = JobApplicationStatuses.Rejected
});

// Финальный шаг: проверка статуса
Assert.Equal((int)JobApplicationStatuses.Rejected,
storedApplication!.ApplicationStatusId);
_chatServiceMock.Verify(x =>
x.CloseChatByApplicationId(employerId, Roles.Employer,
createdAppId), Times.Once);
```

Представленный фрагмент наглядно демонстрирует, как по шагам симулируется жизненный цикл отклика. В конце происходит проверка соответствия статуса отклика состоянию «Отклонено», а также проверяется вызов метода закрытия чата, привязанного к данному отклику.

Результат выполнения всех тестов представлен на рисунке 27.



JobApplicationFullLifecycleTests (4)	101 мс
ImmediateRejection_Scenario	4 мс
RejectionAfterInterview_Scenario	86 мс
StudentEmployed_Scenario	6 мс
WithdrawalAfterOffer_Scenario	5 мс
JobApplicationServiceTests (5)	79 мс
CreateJobApplicationByStudent_ActiveApplicationExists_ThrowsActiveApplicationAlreadyExistsException	2 мс
CreateJobApplicationByStudent_ResumeNotFound_ThrowsResumeNotFoundException	1 мс
CreateJobApplicationByStudent_VacancyNotFound_ThrowsVacancyNotFoundException	2 мс
UpdateApplicationStatus_InvalidRole_ThrowsInvalidJobApplicationStatusException	73 мс
UpdateApplicationStatus_InvalidTransition_ThrowsInvalidJobApplicationStatusException	1 мс

Рисунок 27 – Результат выполнения модульных тестов процесса трудоустройства

Успешное прохождение всех тестов подтверждает высокую надежность реализованного алгоритма трудоустройства. Логика управления статусами строго соответствует заложенным на этапе проектирования правилам, а все данные надежно защищены от некорректных или несанкционированных действий пользователя.

5.3 Оценка качества кода

5.3.1 Размер и сложность кода

Для автоматизированного сбора метрик кода использовалась утилита scc, которая подсчитывает общее количество строк, количество пустых строк и комментариев, а также вычисляет цикломатическую сложность. Результат работы утилиты представлен на рисунке 28.

Language	Files	Lines	Blanks	Comments	Code	Complexity
C#	447	17,524	3,380	20	14,124	813
SQL	8	1,087	50	46	991	0
JSON	7	166	0	0	166	0
MSBuild	7	222	30	0	192	0
YAML	3	167	14	0	153	0
Dockerfile	2	66	10	10	46	2
Docker ignore	1	31	0	0	31	0
Markdown	1	1	0	0	1	0
Total	476	19,264	3,484	76	15,704	815
Estimated Cost to Develop (organic) \$486,862						
Estimated Schedule Effort (organic) 10.46 months						
Estimated People Required(organic) 4.13						
Processed 725437 bytes, 0.725 megabytes (SI)						

Рисунок 28 – Результат оценки объёма и сложности кода

На основе полученных аналитических данных можно сделать вывод о масштабе платформы. Общий объём C#-кода составил 14124 строки на 447 файлов, что свидетельствует о глубокой декомпозиции проекта и разделении классов по зонам ответственности. Помимо этого, низкая цикломатическая сложность проекта доказывает отсутствия глубокой вложенности в бизнес-

правилах. Суммарный показатель сложности для C#-кода составляет всего 813 единиц, что при 447 файлах даёт в среднем 2 единицы на файл.

5.3.2 Метрика Холстеда

Для более глубокого анализа качества кода [21] была проведена оценка сложности метода `CreateJobApplicationByStudent` класса `JobApplicationService`, который используется для создания отклика студентом. Код данного метода представлен на листинге 23.

Листинг 23 – Код метода создания отклика студентом

```
public async Task<int>
CreateJobApplicationByStudent(CreateJobApplicationRequest request)
{
    if (!await
resumeRepository.IsStudentOwnsResume(request.UserId,
request.ResumeId))
        throw new ResumeNotFoundException();

    var vacancy = await
vacancyRepository.GetVacancyById(request.VacancyId);

    if (vacancy is null || !vacancy.IsActive)
        throw new VacancyNotFoundException();

    if (await
applicationRepository.HasStudentActiveApplicationOnVacancy(request
.ResumeId, request.VacancyId))
        throw new ActiveApplicationAlreadyExistsException();

    var chatId = await chatService.GetOrCreateStudentChat(new
StartStudentChatRequest
    {
        StudentId = request.UserId,
        VacancyId = request.VacancyId,
        Message = request.WelcomeMessage
    });

    var application = request.ToDomain(Roles.Student, chatId);
    await applicationRepository.AddJobApplication(application);

    await unitOfWork.SaveChangesAsync();

    var employerEmail = await
employerProfileRepository.GetEmployerEmailByCompanyId(vacancy.Comp
anyId);
```

```

        if (!string.IsNullOrEmpty(employerEmail))
            await
jobApplicationNotifier.NotifyEmployerAboutStatusChangedAsync(employerEmail, vacancy.Title, JobApplicationStatuses.Pending);

        return application.Id;
    }

```

Для расчёта метрики были определены:

- количество уникальных операторов ($n1 = 19$);
- общее количество операторов ($N1 = 41$);
- количество уникальных операндов ($n2 = 29$);
- общее количество операндов ($N2 = 48$).

На основе полученных метрик были вычислены показатели метрики Холстеда (таблица 28).

Таблица 28 – Результаты вычислений метрики Холстеда

Значение	Результат
Длина реализации программы	89
Словарь программы	48
Длина программы	221,6
Объём программы	497,06
Потенциальный объём программы	31,61
Теоретический уровень программы	0,06
Сложность программы	15,72
Работа программирования	7813,78

Анализ полученных метрик показывает, что фактическая длина реализации значительно меньше расчётной длины, что свидетельствует о ёмкости кода и отсутствии избыточности. Показатель сложности программы имеет оптимальное значение для одного из ключевых методов бизнес-логики, а невысокое значение работы программирования указывает на то, что метод легко читается и поддерживается.

Таким образом расчёт показателей метрики Холстеда позволил аналитически оценить сложность и объём функции, а также уровень программирования.

Заключение

В рамках выполнения выпускной квалификационной работы была разработана серверная часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях. Данная платформа позволяет объединить студентов, работодателей и преподавателей в едином цифровом пространстве. Ориентация на взаимодействие с университетом позволяет использовать платформу не только для поиска вакансий, но и для сопровождения образовательного процесса, связанного с практической подготовкой студентов.

На основе анализа предметной области и существующих программных аналогов были сформированы задачи на разработку с подробным описанием функциональных и нефункциональных требований. Помимо этого, для проекта были выбран и обоснован современный технологический стек, а также была выполнена оценка проекта с приближенным планированием бюджета.

На этапе проектирования была разработана архитектура серверной части информационной платформы, а также определена схема базы данных. Проектирование серверной части включило в себя построение диаграмм классов и определение спецификаций для API-методов. Были построены концептуальная, логическая и физическая модели базы данных с последующим её развертыванием. В дополнение к основному монолитному приложению был спроектирован сервис рассылки уведомлений.

Процесс реализации серверной части информационной платформы охватил реализацию взаимодействия с базой данных, включая наполнение базы данных тестовыми данными с использованием генераторов псевдослучайных данных, функционирующих на основе заданных правил. Вместе с этим были запрограммированы алгоритмы бизнес-логики, внедрены механизмы безопасности и подготовлена инфраструктура системы.

С целью проверки корректности работы реализованного приложения было проведено комплексное тестирование центрального алгоритма платформы – алгоритма трудоустройства. Его надежность была проверена с помощью ручного

функционального тестирования методом «белого ящика». Для автоматизированной проверки бизнес-правил был разработан набор модульных тестов. Дополнительно была проведена оценка качества кода, включая анализ его размера и сложности и расчёт показателей метрики Холстеда.

Полученная в результате разработки серверная часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях полностью удовлетворяет определенным на этапе анализа функциональным и нефункциональным требованиям. В ходе выполнения выпускной квалификационной работы все поставленные на разработку задачи были выполнены в полном объёме.

Список использованных источников

1. Ричардс М. Фундаментальный подход к программной архитектуре : паттерны, свойства, проверенные методы / М. Ричардс, Н. Форд. – СПб. : Питер, 2023. – 448 с.
2. Руководство по Docker Compose для начинающих // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/companies/ruvds/articles/450312/> (дата обращения 02.05.2026).
3. Арлоу, Д. UML 2 и Унифицированный процесс : практический объектно-ориентированный анализ и проектирование / Д. Арлоу, И. Нейштадт. – СПб. : Символ-Плюс, 2007. – 624 с.
4. Что такое транзакция // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/537594/> (дата обращения 02.05.2026).
5. Подробно про JWT // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/842056/> (дата обращения 02.05.2026).
6. Что такое Git и для чего он нужен программистам // Яндекс Практикум [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/chto-takoe-git-i-dlya-chego-nuzhen/> (дата обращения 03.05.2026).
7. Язык программирования C# и платформа .NET // Metanit.com – Сайт о программировании [Электронный ресурс]. URL: <https://metanit.com/sharp/> (дата обращения 03.05.2026).
8. Платформа ASP.NET // Metanit.com – Сайт о программировании [Электронный ресурс]. URL: <https://metanit.com/sharp/mvc.php> (дата обращения 03.05.2026).
9. СУБД PostgreSQL // Яндекс Практикум [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/chto-takoe-subd-postgresql/> (дата обращения 03.05.2026).
10. Что такое Swagger // Яндекс Практикум [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/chto-takoe-swagger/> (дата обращения 03.05.2026).

11. Как работает миграция между базами данных в реальном мире // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/758868/> (дата обращения 02.05.2026).

12. Kafka для начинающих: основы // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/957824/> (дата обращения 02.05.2026).

13. REST API: что это такое и как работает // Яндекс Практикум [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/chto-takoe-rest-api-i-kak-rabotaet/> (дата обращения 23.05.2026).

14. WebSocket: разбираем как работает // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/sandbox/171066/> (дата обращения 23.05.2026).

15. Что такое SMTP-протокол и как он отправляет электронные письма // Skillbox Media | Код [Электронный ресурс]. URL: <https://skillbox.ru/media/code/smtp-protokol/> (дата обращения 24.05.2026).

16. Руководство по Entity Framework Core // Metanit.com – Сайт о программировании [Электронный ресурс]. URL: <https://metanit.com/sharp/entityframeworkcore/> (дата обращения 25.05.2026).

17. InternshipPlatform – серверная часть информационной платформы трудоустройства и стажировок студентов в ИТ-компаниях // GitHub [Электронный ресурс]. URL: <https://github.com/AsdWELL/InternshipPlatform>

18. C# и .NET | LINQ // Metanit.com – Сайт о программировании [Электронный ресурс]. URL: <https://metanit.com/sharp/tutorial/15.1.php> (дата обращения 26.05.2026).

19. Майерс, Г. Искусство тестирования программ / Г. Майерс. – М. : Финансы и статистика, 1982. – 176 с.

20. Введение в написание тестов и знакомство с xUnit // Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/357648/> (дата обращения 30.05.2026).

21. Семахин, А. М. Методы верификации и оценки качества программного обеспечения : учебное пособие / А. М. Семахин. – Курган : Изд-во КГУ, 2018. – 150 с.

Приложение А
(рекомендуемое)
Спецификация отношений логической схемы базы данных

Таблица А.1 – Спецификация атрибутов отношения «Образовательная программа»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор образовательной программы	РК
Название	text	Да	Название образовательной программы	
УниверситетId	int4	Да	Идентификатор университета	FK
Код	text	Да	Код специальности/направления	
Продолжительность обучения в годах	int4	Да	Срок обучения в годах	

Таблица А.2 – Спецификация атрибутов отношения «Университет»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор университета	РК
Название	text	Да	Название университета	

Таблица А.3 – Спецификация атрибутов отношения «Учебная группа»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор учебной группы	РК
УниверситетId	int4	Да	Идентификатор университета	FK
Название	text	Да	Название группы	
Год зачисления	int4	Да	Год начала обучения группы	
Год выпуска	int4	Да	Год окончания обучения группы	
Код приглашения	text	Да	Код для вступления студентов в группу	
Образовательная программаId	int4	Да	Идентификатор образовательной программы	FK
КураторId	int4	Да	Идентификатор куратора (преподавателя)	FK

Таблица А.4 – Спецификация атрибутов отношения «Запрос на вступление в учебную группу»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор запроса	РК
ГруппаId	int4	Да	Идентификатор учебной группы	FK
СтудентId	int4	Да	Идентификатор студента	FK
Дата создания	timestamp	Да	Дата и время создания запроса	

Таблица А.5 – Спецификация атрибутов отношения «Период практики в учебной группе»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
ПериодПрактикиId	int4	Да	Идентификатор периода практики	РК, FK
УчебнаяГруппаId	int4	Да	Идентификатор учебной группы	РК, FK

Таблица А.6 – Спецификация атрибутов отношения «Учитель»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
ПользовательId	int4	Да	Идентификатор пользователя преподавателя	РК, FK
УниверситетId	int4	Да	Идентификатор университета	FK
Имя	text	Да	Имя преподавателя	
Фамилия	text	Да	Фамилия преподавателя	
Отчество	text	Нет	Отчество преподавателя	
Телефон	varchar(11)	Нет	Контактный номер телефона	
Ссылка на ВКонтакте	text	Нет	Ссылка на профиль ВКонтакте	
Ссылка на Телеграм	text	Нет	Ссылка на аккаунт Telegram	
Ссылка на Макс	text	Нет	Ссылка на профиль в Макс	

Продолжение таблицы А.6

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Фото профиля	text	Нет	Относительный путь на сервере до фото профиля	

Таблица А.7 – Спецификация атрибутов отношения «Пользователь»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор пользователя	РК
Электронная почта	text	Да	Адрес электронной почты пользователя	
Пароль	text	Да	Зашифрованный пароль	

Таблица А.8 – Спецификация атрибутов отношения «Роль»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор роли	РК
Название	text	Да	Наименование роли в системе	

Таблица А.9 – Спецификация атрибутов отношения «Работодатель»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
ПользовательId	int4	Да	Идентификатор пользователя работодателя	РК, FK
КомпанияId	int4	Да	Идентификатор компании	FK

Таблица А.10 – Спецификация атрибутов отношения «Компания»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор компании	РК
Название	text	Да	Название организации	
ИНН	text	Да	Идентификатор налогоплательщика	
Ссылка	text	Нет	Ссылка на сайт компании	
Описание	text	Нет	Описание деятельности компании	

Продолжение таблицы А.10

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Логотип	text	Нет	Относительный путь на сервере до фото логотипа	

Таблица А.11 – Спецификация атрибутов отношения «Студент»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
ПользовательId	int4	Да	Идентификатор пользователя студента	РК, FK
Имя	text	Да	Имя студента	
Фамилия	text	Да	Фамилия студента	
Отчество	text	Нет	Отчество студента	
Дата рождения	date	Нет	Дата рождения студента	
Телефон	varchar(11)	Нет	Номер телефона студента	
Ссылка на ВКонтакте	text	Нет	Ссылка на профиль ВКонтакте	
Ссылка на Телеграм	text	Нет	Ссылка на профиль Telegram	
Ссылка на Макс	text	Нет	Ссылка на профиль Макс	
Ссылка на ГитХаб	text	Нет	Ссылка на аккаунт GitHub	
Фото профиля	text	Нет	Ссылка на фотографию профиля	
ГруппаId	int4	Нет	Идентификатор учебной группы	FK

Таблица А.12 – Спецификация атрибутов отношения «Период практики»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор периода практики	РК
ОбразовательнаяПрограммаId	int4	Да	Идентификатор образовательной программы	FK
РуководительId	int4	Да	Идентификатор руководителя от кафедры (преподавателя)	FK

Продолжение таблицы А.12

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Номер курса	int4	Да	Номер курса обучения	
Начало учебного года	int4	Да	Календарный год начала обучения	
Дата начала практики	date	Да	Дата старта практики	
Дата завершения практики	date	Да	Дата окончания практики	

Таблица А.13 – Спецификация атрибутов отношения «Заявка на практику»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор заявки	РК
Дата создания	timestamp	Да	Дата отправки заявки	
СтудентId	int4	Да	Идентификатор студента	FK
Предложение по практикеId	int4	Да	Идентификатор предложения практики	FK
Период практикиId	int4	Да	Идентификатор периода практики	FK

Таблица А.14 – Спецификация атрибутов отношения «Практика студента»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор практики	РК
СтудентId	int4	Да	Идентификатор студента	FK
Период практикиId	int4	Да	Идентификатор периода практики	FK
Предложение по практикеId	int4	Да	Идентификатор предложения практики	FK

Таблица А.15 – Спецификация атрибутов отношения «Отчетность по практике»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор записи отчетности	РК

Продолжение таблицы А.15

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Путь к файлу с отчетом	text	Да	Относительный путь на сервере до файла с отчетом студента	
Путь к файлу с решением	text	Нет	Относительный путь на сервере до файла с решением студента	
Ссылка на решение	text	Нет	Внешняя ссылка на репозиторий с решением	
Время последнего обновления	timestamp	Да	Дата и время последнего изменения ответа	
Время просмотра	timestamp	Нет	Дата и время проверки ответа	
Оценка	int4	Нет	Выставленный балл за практику	
ПрактикаСтудентаId	int4	Да	Идентификатор привязанной практики студента	FK
СтатусId	int4	Да	Идентификатор текущего статуса	FK

Таблица А.16 – Спецификация атрибутов отношения «Комментарий к отчетности»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор комментария	PK
Содержание	text	Да	Текст комментария	
Дата создания	timestamp	Да	Дата и время написания комментария	
Отчетность по практикеId	int4	Да	Идентификатор записи отчетности	FK
ОтправительId	int4	Да	Идентификатор отправителя	

Таблица А.17 – Спецификация атрибутов отношения «Статус отчетности по практике»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор статуса отчета	РК
Название	text	Да	Наименование статуса проверки	

Таблица А.18 – Спецификация атрибутов отношения «Избранная вакансия»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор избранной вакансии	РК
СтудентId	int4	Да	Идентификатор студента	FK
ВакансииId	int4	Да	Идентификатор вакансии	FK

Таблица А.19 – Спецификация атрибутов отношения «Просмотр вакансии»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор просмотра вакансии	РК
СтудентId	int4	Да	Идентификатор студента	РК, FK
ВакансииId	int4	Да	Идентификатор вакансии	РК, FK
Дата просмотра	timestamp	Да	Дата и время просмотра вакансии	

Таблица А.20 – Спецификация атрибутов отношения «Просмотр резюме»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор просмотра резюме	РК
КомпанияId	int4	Да	Идентификатор компании	FK
РезюмеId	int4	Да	Идентификатор резюме	FK
Дата просмотра	timestamp	Да	Дата и время просмотра резюме	

Таблица А.21 – Спецификация атрибутов отношения «Предложение по практике»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор предложения	PK
Заголовок	text	Да	Заголовок предложения практики	
Описание	text	Нет	Подробное описание условий практики	
Возможность удаленного прохождения	bool	Да	Флаг дистанционного формата	
Регион	text	Нет	Географический регион компании	
Доступность для просмотра	bool	Да	Флаг видимости предложения на платформе	
Количество мест	int4	Да	Лимит доступных мест для студентов	
СпециализацияId	int4	Да	Идентификатор направления специализации	FK
КомпанияId	int4	Да	Идентификатор компании-организатора	FK

Таблица А.22 – Спецификация атрибутов отношения «Материал по практике»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор учебного материала	PK
Предложение по практикеId	int4	Да	Идентификатор предложения практики	FK
Название	text	Да	Название методического материала	
Путь к файлу	text	Да	Относительный путь на сервере до файла с материалом	

Таблица А.23 – Спецификация атрибутов отношения «Вакансия»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор вакансии	РК
Заголовок	text	Да	Название вакантной должности	
Описание	text	Нет	Детальные обязанности и требования	
Зарплата от	int4	Нет	Минимальный порог оплаты труда	
Зарплата до	int4	Нет	Максимальный порог оплаты труда	
Возможность удаленной работы	bool	Да	Флаг возможности работать удаленно	
Region	text	Нет	Территориальное расположение рабочего места	
Доступность для просмотра	bool	Да	Флаг видимости вакансии на платформе	
Требуемый опыт работы	text	Да	Необходимый стаж кандидата	
СпециализацияId	int4	Да	Идентификатор специализации вакансии	FK
КомпанияId	int4	Да	Идентификатор компании работодателя	FK

Таблица А.24 – Спецификация атрибутов отношения «Навыки в вакансии»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
ВакансияId	int4	Да	Идентификатор вакансии	РК, FK
НавыкId	int4	Да	Идентификатор ключевого навыка	РК, FK

Таблица А.25 – Спецификация атрибутов отношения «Навыки»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор профессионального навыка	РК

Продолжение таблицы А.25

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Название	text	Да	Наименование технологии или компетенции	

Таблица А.26 – Спецификация атрибутов отношения «Навыки в резюме»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
НавыкId	int4	Да	Идентификатор навыка	РК, FK
РезюмеId	int4	Да	Идентификатор резюме	РК, FK

Таблица А.27 – Спецификация атрибутов отношения «Специализация»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор специализации	РК
Название	text	Да	Наименование профессионального направления	

Таблица А.28 – Спецификация атрибутов отношения «Резюме»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор резюме	РК
Дата последнего обновления	date	Да	Дата последнего обновления резюме кандидатом	
Описание	text	Нет	Сведения о себе и личные достижения	
Доступность для просмотра	bool	Да	Флаг видимости резюме на платформе	
Желаемая зарплата	int4	Нет	Целевой уровень оплаты труда	
Регион	text	Нет	Место проживания соискателя	
СпециализацияId	int4	Да	Идентификатор желаемой специализации	FK
СтудентId	int4	Да	Идентификатор студента-автора резюме	FK

Таблица А.29 – Спецификация атрибутов отношения «Опыт работы»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор записи об опыте работы	РК
РезюмеId	int4	Да	Идентификатор резюме	FK
Название компании	text	Да	Наименование места работы	
Профессия	text	Да	Занимаемая должность	
Описание работы	text	Нет	Перечень выполняемых трудовых функций	
Начало работы	date	Да	Дата трудоустройства	
Окончание работы	date	Нет	Дата увольнения	

Таблица А.30 – Спецификация атрибутов отношения «Отклик на вакансию»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор отклика	РК
ВакансияId	int4	Да	Идентификатор вакансии	FK
РезюмеId	int4	Да	Идентификатор резюме соискателя	FK
Дата обновления статуса	date	Да	Дата изменения состояния отклика	
СтатусId	int4	Да	Идентификатор текущего статуса отклика	FK
ЧатId	int4	Да	Идентификатор связанного чата для обсуждения вакансии	FK

Таблица А.31 – Спецификация атрибутов отношения «Статус отклика»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор статуса отклика	РК
Название	text	Да	Наименование стадии	

Таблица А.32 – Спецификация атрибутов отношения «Чат»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор чата	РК
КомпанияId	int4	Да	Идентификатор компании участника	FK
СтудентId	int4	Да	Идентификатор студента участника	FK
Доступность для отправки	bool	Да	Флаг, допускающий отправки сообщений в чат	

Таблица А.33 – Спецификация атрибутов отношения «Сообщение»

Имя атрибута	Тип данных	Обязательный	Описание	Роль
Id	int4	Да	Идентификатор сообщения	РК
Содержание	text	Да	Текст сообщения	
Дата отправки	timestamp	Да	Дата и время отправки сообщения	
ЧатId	int4	Да	Идентификатор чата	FK
ОтправительId	int4	Да	Идентификатор пользователя-отправителя	

Приложение Б
(рекомендуемое)

Скрипты (исходные коды) для создания объектов базы данных

Листинг Б.1 – SQL-скрипт для создания объектов базы данных информационной платформы трудоустройства и стажировок в ИТ-компаниях

```

CREATE TABLE "Users" (
    "Id" SERIAL NOT NULL,
    "Email" text UNIQUE NOT NULL,
    "PasswordHash" text NOT NULL,
    "RefreshToken" text,
    "RefreshTokenExpiredAt" timestamp NOT NULL,
    "RoleId" integer NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "Roles" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL UNIQUE,
    PRIMARY KEY ("Id"));
CREATE TABLE "StudentProfiles" (
    "UserId" integer NOT NULL,
    "Name" text NOT NULL,
    "Surname" text NOT NULL,
    "Patronymic" text,
    "BirthdayDate" date,
    "Phone" varchar(12),
    "VkLink" text,
    "MaxLink" text,
    "TgLink" text,
    "GithubLink" text,
    "AvatarPath" text,
    "GroupId" integer,
    PRIMARY KEY ("UserId"));
CREATE TABLE "EmployerProfiles" (
    "UserId" integer NOT NULL,
    "CompanyId" integer NOT NULL,
    PRIMARY KEY ("UserId"));
CREATE TABLE "Companies" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL,
    "Inn" text NOT NULL UNIQUE,
    "Link" text,
    "Description" text,
    "LogoPath" text,
    PRIMARY KEY ("Id"));
CREATE TABLE "Vacancies" (
    "Id" SERIAL NOT NULL,
    "Title" text NOT NULL,
    "Description" text,
    "SalaryFrom" integer NOT NULL,
    "SalaryTo" integer NOT NULL,
    "IsRemote" bool NOT NULL,
    "Region" text,
    "IsActive" bool NOT NULL,
    "MinWorkExperienceYears" integer NOT NULL,
    "SpecializationId" integer NOT NULL,

```

```

        "CompanyId"          integer NOT NULL,
        PRIMARY KEY ("Id"));
CREATE TABLE "PracticeMaterials" (
    "Id" SERIAL NOT NULL,
    "PracticeOfferId" integer NOT NULL,
    "FileName" text NOT NULL,
    "FilePath" text NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "PracticeOffers" (
    "Id" SERIAL NOT NULL,
    "Title" text NOT NULL,
    "Description" text,
    "IsRemote" bool NOT NULL,
    "Region" text,
    "IsActive" bool NOT NULL,
    "MaxStudents" integer NOT NULL,
    "SpecializationId" integer NOT NULL,
    "CompanyId" integer NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "FavoriteVacancies" (
    "Id" SERIAL NOT NULL,
    "StudentId" integer NOT NULL,
    "VacancyId" integer NOT NULL,
    CONSTRAINT "unique_favorites"
        UNIQUE ("StudentId", "VacancyId"),
    PRIMARY KEY ("Id"));
CREATE TABLE "JobApplications" (
    "Id" SERIAL NOT NULL,
    "VacancyId" integer NOT NULL,
    "ResumeId" integer NOT NULL,
    "LastStatusDate" date NOT NULL,
    "ApplicationStatusId" integer NOT NULL,
    "ChatId" integer NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "ApplicationStatuses" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL UNIQUE,
    PRIMARY KEY ("Id"));
CREATE TABLE "Skills" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL UNIQUE,
    PRIMARY KEY ("Id"));
CREATE TABLE "Specializations" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL UNIQUE,
    PRIMARY KEY ("Id"));
CREATE TABLE "Resumes" (
    "Id" SERIAL NOT NULL,
    "LastUpdateDate" date NOT NULL,
    "Description" text,
    "DesiredSalary" integer NOT NULL,
    "Region" text,
    "IsActive" bool NOT NULL,

```

```

        "SpecializationId" integer NOT NULL,
        "StudentId"         integer NOT NULL,
        PRIMARY KEY ("Id"));
CREATE TABLE "WorkExperiences" (
    "Id" SERIAL NOT NULL,
    "ResumeId" integer NOT NULL,
    "CompanyName" text NOT NULL,
    "Profession" text NOT NULL,
    "StartDateWork" date NOT NULL,
    "EndDateWork" date,
    "WorkDescription" text,
    PRIMARY KEY ("Id"));
CREATE TABLE "SkillsToResume" (
    "SkillId" integer NOT NULL,
    "ResumeId" integer NOT NULL,
    PRIMARY KEY ("ResumeId",
        "SkillId"));
CREATE TABLE "SkillsToVacancy" (
    "SkillId" integer NOT NULL,
    "VacancyId" integer NOT NULL,
    PRIMARY KEY ("VacancyId",
        "SkillId"));
CREATE TABLE "Chats" (
    "Id" SERIAL NOT NULL,
    "CompanyId" integer NOT NULL,
    "StudentId" integer NOT NULL,
    "VacancyId" integer NOT NULL,
    "IsClosed" bool NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "Messages" (
    "Id" SERIAL NOT NULL,
    "Content" text NOT NULL,
    "CreatedAt" timestamp NOT NULL,
    "IsRead" bool NOT NULL,
    "ChatId" integer NOT NULL,
    "SenderUserId" integer NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "ResumeViews" (
    "Id" SERIAL NOT NULL,
    "ResumeId" integer NOT NULL,
    "CompanyId" integer NOT NULL,
    "ViewDate" timestamp NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "VacancyViews" (
    "Id" SERIAL NOT NULL,
    "VacancyId" integer NOT NULL,
    "StudentId" integer NOT NULL,
    "ViewDate" timestamp NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "Universities" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL UNIQUE,
    PRIMARY KEY ("Id"));

```

```

CREATE TABLE "EducationalPrograms" (
  "Id" SERIAL NOT NULL,
  "Name" text NOT NULL,
  "UniversityId" integer NOT NULL,
  "SpecializationCode" text NOT NULL,
  "GroupCode" text NOT NULL,
  "DurationYears" integer NOT NULL,
  CONSTRAINT "unique_university_program"
    UNIQUE ("Name", "UniversityId"),
  PRIMARY KEY ("Id"));
CREATE TABLE "Teachers" (
  "UserId" integer NOT NULL,
  "UniversityId" integer NOT NULL,
  "Name" text NOT NULL,
  "Surname" text NOT NULL,
  "Patronymic" text,
  "Phone" varchar(12),
  "VkLink" text,
  "MaxLink" text,
  "TgLink" text,
  "AvatarPath" text,
  PRIMARY KEY ("UserId"));
CREATE TABLE "StudentGroups" (
  "Id" SERIAL NOT NULL,
  "Name" text NOT NULL,
  "UniversityId" integer NOT NULL,
  "EducationalProgramId" integer NOT NULL,
  "EnrollmentYear" integer NOT NULL,
  "GraduationYear" integer NOT NULL,
  "InviteCode" text NOT NULL UNIQUE,
  "CuratorId" integer NOT NULL,
  CONSTRAINT "unique_university_group"
    UNIQUE ("Name", "UniversityId"),
  PRIMARY KEY ("Id"));
CREATE TABLE "StudentGroupApplications" (
  "Id" SERIAL NOT NULL,
  "GroupId" integer NOT NULL,
  "StudentId" integer NOT NULL UNIQUE,
  "CreatedAt" timestamp NOT NULL,
  PRIMARY KEY ("Id"));
CREATE TABLE "PracticePeriods" (
  "Id" SERIAL NOT NULL,
  "SupervisorId" integer NOT NULL,
  "EducationalProgramId" integer NOT NULL,
  "CourseNumber" integer NOT NULL,
  "AcademicYearStart" integer NOT NULL,
  "StartDate" date NOT NULL,
  "EndDate" date NOT NULL,
  PRIMARY KEY ("Id"));
CREATE TABLE "PracticePeriodsGroup" (
  "StudentGroupId" integer NOT NULL,
  "PracticePeriodId" integer NOT NULL,
  PRIMARY KEY ("StudentGroupId",

```

```

    "PracticePeriodId"));
CREATE TABLE "PracticeApplications" (
    "Id" SERIAL NOT NULL,
    "StudentId" integer NOT NULL UNIQUE,
    "PracticeOfferId" integer NOT NULL,
    "PracticePeriodId" integer NOT NULL,
    "CreatedAt" timestamp NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "StudentPractices" (
    "Id" SERIAL NOT NULL,
    "StudentId" integer NOT NULL,
    "PracticeOfferId" integer NOT NULL,
    "PracticePeriodId" integer NOT NULL,
    CONSTRAINT "unique_practice_per_period"
        UNIQUE ("StudentId", "PracticePeriodId"),
    PRIMARY KEY ("Id"));
CREATE TABLE "PracticeSubmissionStatuses" (
    "Id" SERIAL NOT NULL,
    "Name" text NOT NULL UNIQUE,
    PRIMARY KEY ("Id"));
CREATE TABLE "PracticeSubmissions" (
    "Id" SERIAL NOT NULL,
    "ReportFilePath" text,
    "ReportFileName" text,
    "SolutionFileName" text,
    "SolutionPath" text,
    "SolutionUrl" text,
    "UpdatedAt" timestamp NOT NULL,
    "ReviewedAt" timestamp,
    "Grade" integer,
    "StudentPracticeId" integer NOT NULL,
    "StatusId" integer NOT NULL,
    PRIMARY KEY ("Id"));
CREATE TABLE "SubmissionComments" (
    "Id" SERIAL NOT NULL,
    "Content" text NOT NULL,
    "CreatedAt" timestamp NOT NULL,
    "PracticeSubmissionId" integer NOT NULL,
    "SenderId" integer NOT NULL,
    PRIMARY KEY ("Id"));
CREATE UNIQUE INDEX "Users1"
    ON "Users" ("Email");
CREATE INDEX "Users2"
    ON "Users" ("RefreshToken");
CREATE INDEX "StudentProfiles1"
    ON "StudentProfiles" ("GroupId");
CREATE INDEX "EmployerProfiles1"
    ON "EmployerProfiles" ("CompanyId");
CREATE INDEX "Vacancies1"
    ON "Vacancies" ("CompanyId");
CREATE INDEX "Vacancies2"
    ON "Vacancies" ("SpecializationId");
CREATE INDEX "Vacancies3"

```

```

    ON "Vacancies" ("Title");
CREATE INDEX "Vacancies4"
    ON "Vacancies" ("Region");
CREATE INDEX "PracticeOffers1"
    ON "PracticeOffers" ("CompanyId");
CREATE INDEX "PracticeOffers2"
    ON "PracticeOffers" ("SpecializationId");
CREATE INDEX "PracticeOffers3"
    ON "PracticeOffers" ("Title");
CREATE INDEX "PracticeOffers4"
    ON "PracticeOffers" ("Region");
CREATE INDEX "PracticeMaterials1"
    ON "PracticeMaterials" ("PracticeOfferId");
CREATE INDEX "Applications1"
    ON "JobApplications" ("ResumeId");
CREATE INDEX "Applications2"
    ON "JobApplications" ("VacancyId");
CREATE INDEX "Resumes1"
    ON "Resumes" ("StudentId");
CREATE INDEX "Resumes2"
    ON "Resumes" ("SpecializationId");
CREATE INDEX "Messages1"
    ON "Messages" ("ChatId", "CreatedAt");
CREATE INDEX "WorkExperiences1"
    ON "WorkExperiences" ("ResumeId");
CREATE INDEX "Chats1"
    ON "Chats" ("CompanyId");
CREATE INDEX "Chats2"
    ON "Chats" ("StudentId");
CREATE INDEX "FavoriteVacancies1"
    ON "FavoriteVacancies" ("StudentId");
CREATE INDEX "ResumeViews1"
    ON "ResumeViews" ("ResumeId");
CREATE INDEX "ResumeViews2"
    ON "ResumeViews" ("CompanyId");
CREATE INDEX "VacancyViews1"
    ON "VacancyViews" ("VacancyId");
CREATE INDEX "VacancyViews2"
    ON "VacancyViews" ("StudentId");
CREATE INDEX "StudentGroups1"
    ON "StudentGroups" ("CuratorId");
CREATE INDEX "StudentGroupRequests1"
    ON "StudentGroupApplications" ("GroupId");
CREATE INDEX "EducationalPrograms1"
    ON "EducationalPrograms" ("UniversityId");
CREATE INDEX "PractisePeriods1"
    ON "PracticePeriods" ("EducationalProgramId");
CREATE INDEX "PractisePeriods2"
    ON "PracticePeriods" ("SupervisorId");
CREATE INDEX "PracticeApplications1"
    ON "PracticeApplications" ("PracticePeriodId");
CREATE INDEX "StudentPractices1"
    ON "StudentPractices" ("PracticePeriodId");

```

```

CREATE INDEX "SubmissionComments1"
ON "SubmissionComments" ("PracticeSubmissionId");
ALTER TABLE "Users" ADD CONSTRAINT "FKUsers379039" FOREIGN KEY
("RoleId") REFERENCES "Roles" ("Id") ON DELETE Restrict;
ALTER TABLE "StudentProfiles" ADD CONSTRAINT "FKStudentsPr711691"
FOREIGN KEY ("UserId") REFERENCES "Users" ("Id") ON UPDATE Cascade
ON DELETE Cascade;
ALTER TABLE "StudentProfiles" ADD CONSTRAINT "FKStudentsPr711692"
FOREIGN KEY ("GroupId") REFERENCES "StudentGroups" ("Id");
ALTER TABLE "EmployerProfiles" ADD CONSTRAINT "FKEmployerPr861889"
FOREIGN KEY ("UserId") REFERENCES "Users" ("Id") ON UPDATE Cascade
ON DELETE Cascade;
ALTER TABLE "EmployerProfiles" ADD CONSTRAINT "FKEmployerPr684251"
FOREIGN KEY ("CompanyId") REFERENCES "Companies" ("Id") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "Vacancies" ADD CONSTRAINT "FKVacancies430548" FOREIGN
KEY ("SpecializationId") REFERENCES "Specializations" ("Id");
ALTER TABLE "SkillsToResume" ADD CONSTRAINT "FKSkillsToRe373177"
FOREIGN KEY ("SkillId") REFERENCES "Skills" ("Id");
ALTER TABLE "SkillsToResume" ADD CONSTRAINT "FKSkillsToRe581449"
FOREIGN KEY ("ResumeId") REFERENCES "Resumes" ("Id") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "SkillsToVacancy" ADD CONSTRAINT "FKSkillsToVa373177"
FOREIGN KEY ("SkillId") REFERENCES "Skills" ("Id");
ALTER TABLE "SkillsToVacancy" ADD CONSTRAINT "FKSkillsToVa581449"
FOREIGN KEY ("VacancyId") REFERENCES "Vacancies" ("Id") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "Resumes" ADD CONSTRAINT "FKResumes64954" FOREIGN KEY
("SpecializationId") REFERENCES "Specializations" ("Id");
ALTER TABLE "Vacancies" ADD CONSTRAINT "FKVacancies845540" FOREIGN
KEY ("CompanyId") REFERENCES "Companies" ("Id") ON UPDATE Cascade
ON DELETE Cascade;
ALTER TABLE "PracticeOffers" ADD CONSTRAINT
"FKPracticeOffers845540" FOREIGN KEY ("CompanyId") REFERENCES
"Companies" ("Id") ON UPDATE Cascade ON DELETE Cascade;
ALTER TABLE "PracticeOffers" ADD CONSTRAINT
"FKPracticeOffers430548" FOREIGN KEY ("SpecializationId")
REFERENCES "Specializations" ("Id");
ALTER TABLE "PracticeMaterials" ADD CONSTRAINT
"FKPracticeMaterials430548" FOREIGN KEY ("PracticeOfferId")
REFERENCES "PracticeOffers" ("Id");
ALTER TABLE "JobApplications" ADD CONSTRAINT
"FKApplications397797" FOREIGN KEY ("VacancyId") REFERENCES
"Vacancies" ("Id") ON UPDATE Cascade ON DELETE Cascade;
ALTER TABLE "JobApplications" ADD CONSTRAINT "FKApplications15717"
FOREIGN KEY ("ResumeId") REFERENCES "Resumes" ("Id") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "JobApplications" ADD CONSTRAINT
"FKApplications321884" FOREIGN KEY ("ApplicationStatusId")
REFERENCES "ApplicationStatuses" ("Id");
ALTER TABLE "JobApplications" ADD CONSTRAINT
"FKApplications321885" FOREIGN KEY ("ChatId") REFERENCES "Chats"
("Id") ON UPDATE Cascade ON DELETE Cascade;

```

```

ALTER TABLE "Chats" ADD CONSTRAINT "FKChats573641" FOREIGN KEY
("CompanyId") REFERENCES "Companies" ("Id") ON UPDATE Cascade ON
DELETE Cascade;
ALTER TABLE "Chats" ADD CONSTRAINT "FKChats573642" FOREIGN KEY
("VacancyId") REFERENCES "Vacancies" ("Id") ON UPDATE Cascade ON
DELETE Cascade;
ALTER TABLE "Chats" ADD CONSTRAINT "FKChats573643" FOREIGN KEY
("StudentId") REFERENCES "StudentProfiles" ("UserId") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "Messages" ADD CONSTRAINT "FKMessages297040" FOREIGN
KEY ("ChatId") REFERENCES "Chats" ("Id") ON UPDATE Cascade ON
DELETE Cascade;
ALTER TABLE "Resumes" ADD CONSTRAINT "FKResumes739148" FOREIGN KEY
("StudentId") REFERENCES "StudentProfiles" ("UserId") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "Messages" ADD CONSTRAINT "FKMessages187268" FOREIGN
KEY ("SenderUserId") REFERENCES "Users" ("Id") ON UPDATE Cascade
ON DELETE Cascade;
ALTER TABLE "WorkExperiences" ADD CONSTRAINT
"FKWorkExperiences12341" FOREIGN KEY ("ResumeId") REFERENCES
"Resumes" ("Id") ON UPDATE Cascade ON DELETE Cascade;
ALTER TABLE "FavoriteVacancies" ADD CONSTRAINT
"FKFavoriteVacancies573642" FOREIGN KEY ("VacancyId") REFERENCES
"Vacancies" ("Id") ON UPDATE Cascade ON DELETE Cascade;
ALTER TABLE "FavoriteVacancies" ADD CONSTRAINT
"FKFavoriteVacancies573643" FOREIGN KEY ("StudentId") REFERENCES
"StudentProfiles" ("UserId") ON UPDATE Cascade ON DELETE Cascade;
ALTER TABLE "ResumeViews" ADD CONSTRAINT "FKResumeViews573641"
FOREIGN KEY ("CompanyId") REFERENCES "Companies" ("Id") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "ResumeViews" ADD CONSTRAINT "FKResumeViews15717"
FOREIGN KEY ("ResumeId") REFERENCES "Resumes" ("Id") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "VacancyViews" ADD CONSTRAINT "FKVacancyViews573643"
FOREIGN KEY ("StudentId") REFERENCES "StudentProfiles" ("UserId")
ON UPDATE Cascade ON DELETE Cascade;
ALTER TABLE "VacancyViews" ADD CONSTRAINT "FKVacancyViews573642"
FOREIGN KEY ("VacancyId") REFERENCES "Vacancies" ("Id") ON UPDATE
Cascade ON DELETE Cascade;
ALTER TABLE "Teachers" ADD CONSTRAINT "FKTeachersPr711691" FOREIGN
KEY ("UserId") REFERENCES "Users" ("Id") ON UPDATE Cascade ON
DELETE Cascade;
ALTER TABLE "Teachers" ADD CONSTRAINT "FKTeachersPr711692" FOREIGN
KEY ("UniversityId") REFERENCES "Universities" ("Id");
ALTER TABLE "EducationalPrograms" ADD CONSTRAINT
"FKEducationalProgramsPr711692" FOREIGN KEY ("UniversityId")
REFERENCES "Universities" ("Id");
ALTER TABLE "StudentGroups" ADD CONSTRAINT
"FKStudentGroupsPr711690" FOREIGN KEY ("EducationalProgramId")
REFERENCES "EducationalPrograms" ("Id");
ALTER TABLE "StudentGroups" ADD CONSTRAINT
"FKStudentGroupsPr711691" FOREIGN KEY ("UniversityId") REFERENCES
"Universities" ("Id");

```

```

ALTER TABLE "StudentGroups" ADD CONSTRAINT
"FKStudentGroupsPr711692" FOREIGN KEY ("CuratorId") REFERENCES
"Teachers" ("UserId");
ALTER TABLE "StudentGroupApplications" ADD CONSTRAINT
"FKStudentGroupRequestsPr711691" FOREIGN KEY ("GroupId")
REFERENCES "StudentGroups" ("Id");
ALTER TABLE "StudentGroupApplications" ADD CONSTRAINT
"FKStudentGroupRequestsPr711692" FOREIGN KEY ("StudentId")
REFERENCES "StudentProfiles" ("UserId");
ALTER TABLE "PracticePeriods" ADD CONSTRAINT
"FKPractisePeriodsPr711692" FOREIGN KEY ("SupervisorId")
REFERENCES "Teachers" ("UserId");
ALTER TABLE "PracticePeriods" ADD CONSTRAINT
"FKPractisePeriodsPr711690" FOREIGN KEY ("EducationalProgramId")
REFERENCES "EducationalPrograms" ("Id");
ALTER TABLE "PracticePeriodsGroup" ADD CONSTRAINT
"FKPracticePeriodsGroup373177" FOREIGN KEY ("StudentGroupId")
REFERENCES "StudentGroups" ("Id") ON UPDATE Cascade ON DELETE
Cascade;
ALTER TABLE "PracticePeriodsGroup" ADD CONSTRAINT
"FKPracticePeriodsGroup581449" FOREIGN KEY ("PracticePeriodId")
REFERENCES "PracticePeriods" ("Id");
ALTER TABLE "PracticeApplications" ADD CONSTRAINT
"FKPracticeApplications739148" FOREIGN KEY ("StudentId")
REFERENCES "StudentProfiles" ("UserId") ON UPDATE Cascade ON
DELETE Cascade;
ALTER TABLE "PracticeApplications" ADD CONSTRAINT
"FKPracticeApplications430548" FOREIGN KEY ("PracticeOfferId")
REFERENCES "PracticeOffers" ("Id");
ALTER TABLE "PracticeApplications" ADD CONSTRAINT
"FKPracticeApplications581449" FOREIGN KEY ("PracticePeriodId")
REFERENCES "PracticePeriods" ("Id");
ALTER TABLE "StudentPractices" ADD CONSTRAINT
"FKStudentPractices739148" FOREIGN KEY ("StudentId") REFERENCES
"StudentProfiles" ("UserId") ON UPDATE Cascade ON DELETE Cascade;
ALTER TABLE "StudentPractices" ADD CONSTRAINT
"FKStudentPractices430548" FOREIGN KEY ("PracticeOfferId")
REFERENCES "PracticeOffers" ("Id");
ALTER TABLE "StudentPractices" ADD CONSTRAINT
"FKStudentPractices581449" FOREIGN KEY ("PracticePeriodId")
REFERENCES "PracticePeriods" ("Id");
ALTER TABLE "PracticeSubmissions" ADD CONSTRAINT
"FKPracticeSubmissions581449" FOREIGN KEY ("StudentPracticeId")
REFERENCES "StudentPractices" ("Id");
ALTER TABLE "PracticeSubmissions" ADD CONSTRAINT
"FKPracticeSubmissions581440" FOREIGN KEY ("StatusId") REFERENCES
"PracticeSubmissionStatuses" ("Id");
ALTER TABLE "SubmissionComments" ADD CONSTRAINT
"FKSubmissionComments581440" FOREIGN KEY ("PracticeSubmissionId")
REFERENCES "PracticeSubmissions" ("Id");
ALTER TABLE "SubmissionComments" ADD CONSTRAINT
"FKSubmissionComments581441" FOREIGN KEY ("SenderId") REFERENCES
"Users" ("Id");

```

Приложение В
(рекомендуемое)

**Спецификация API серверного приложения информационной платформы
трудоустройства и стажировок студентов в ИТ-компаниях**

Таблица В.1 – Спецификация API контроллера AuthController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/auth/register/student	POST	Регистрация нового студента	-
/api/auth/register/employer	POST	Регистрация нового работодателя	-
/api/auth/register/teacher	POST	Регистрация нового преподавателя	-
/api/auth/login	POST	Аутентификация пользователя	-
/api/auth/refresh-token	POST	Обновление токена доступа	-

Таблица В.2 – Спецификация API контроллера CompanyController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/companies/{companyId}	GET	Получение информации о компании по её идентификатору	-

Таблица В.3 – Спецификация API контроллера CuratorGroupApplicationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/curators/me/group-applications	GET	Получение списка заявок в учебные группы текущего куратора	-
/api/curators/me/group-applications/{applicationId}/accept	PUT	Принятие заявки в учебную группу	-
/api/curators/me/group-applications/{applicationId}/reject	PUT	Отклонение заявки в учебную группу	-

Таблица В.4 – Спецификация API контроллера CuratorGroupsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/curators/me/groups	POST	Создание новой учебной группы куратором	-
/api/curators/me/groups	GET	Получение списка учебных групп текущего куратора	-
/api/curators/me/groups/{groupId}/students	GET	Получение списка студентов конкретной учебной группы куратора	-
/api/curators/me/groups/{groupId}/invite-code	PUT	Обновление кода приглашения для вступления в группу	-

Таблица В.5 – Спецификация API контроллера CuratorGroupStatisticsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/curators/me/groups/{groupId}/students/{studentId}	GET	Получение детальной информации о студенте учебной группы куратора	-
/api/curators/me/groups/{groupId}/students/{studentId}/resumes	GET	Получение резюме студента учебной группы куратора	-
/api/curators/me/groups/{groupId}/students/{studentId}/application-stats	GET	Получение статистики откликов студента учебной группы куратора	-

Таблица В.6 – Спецификация API контроллера CurrentEmployerController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me	GET	Получение профиля текущего работодателя	-
/api/employers/me	PUT	Обновление профиля текущего работодателя	-

Продолжение таблицы В.6

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me	DELETE	Удаление профиля текущего работодателя	-
/api/employers/me/logout	POST	Выход текущего работодателя из системы	-

Таблица В.7 – Спецификация API контроллера CurrentStudentController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me	GET	Получение профиля текущего студента	-
/api/students/me	PUT	Обновление данных профиля текущего студента	-
/api/students/me	DELETE	Удаление аккаунта текущего студента	-
/api/students/me/avatar	PUT	Загрузка/обновление аватара текущего студента	-
/api/students/me/avatar	DELETE	Удаление аватара текущего студента	-
/api/students/me/logout	POST	Выход текущего студента из аккаунта	-

Таблица В.8 – Спецификация API контроллера CurrentTeacherController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/teachers/me	GET	Получение профиля текущего преподавателя	-
/api/teachers/me	PUT	Обновление данных профиля текущего преподавателя	-

Продолжение таблицы В.8

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/teachers/me	DELETE	Удаление аккаунта текущего преподавателя	-
/api/teachers/me/avatar	PUT	Загрузка/обновление аватара текущего преподавателя	-
/api/teachers/me/avatar	DELETE	Удаление аватара текущего преподавателя	-
/api/teachers/me/logout	POST	Выход текущего преподавателя из системы	-

Таблица В.9 – Спецификация API контроллера EducationalProgramsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/educational-programs	GET	Получение списка образовательных программ	name – название программы для фильтрации

Таблица В.10 – Спецификация API контроллера EmployerApplicationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/applications	POST	Создание отклика на вакансию от лица работодателя	-
/api/employers/me/applications	GET	Получение списка откликов на вакансии текущего работодателя	VacancyId – идентификатор вакансии, Status – статус отклика, PageIndex – номер страницы, PageSize – размер страницы
/api/employers/me/applications/{applicationId}	PUT	Обновление статуса отклика работодателем	-

Таблица В.11 – Спецификация API контроллера EmployerChatsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/chats	GET	Получение списка чатов текущего работодателя	-
/api/employers/me/chats	POST	Инициация нового чата со студентом	-
/api/employers/me/chats/{chatId}	GET	Получение сообщений конкретного чата работодателя	-
/api/employers/me/chats/{chatId}/read	POST	Отметка сообщений чата как прочитанных	-

Таблица В.12 – Спецификация API контроллера EmployerCompanyController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/company	GET	Получение информации о компании текущего работодателя	-
/api/employers/me/company	PUT	Обновление данных компании текущего работодателя	-
/api/employers/me/company/logo	PUT	Загрузка логотипа компании	-
/api/employers/me/company/logo	DELETE	Удаление логотипа компании	-

Таблица В.13 – Спецификация API контроллера EmployerPracticeApplicationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/practice-applications	GET	Получение списка заявок студентов на прохождение практики у работодателя	-

Продолжение таблицы В.13

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/practice-applications/{applicationId}	GET	Получение деталей конкретной заявки на практику	-
/api/employers/me/practice-applications/{applicationId}/accept	PUT	Принятие студента на практику по заявке	-
/api/employers/me/practice-applications/{applicationId}/reject	PUT	Отклонение заявки студента на практику	-

Таблица В.14 – Спецификация API контроллера

EmployerPracticeMaterialsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/practice-offers/{practiceOfferId}/files	POST	Загрузка файлов к предложению о практике	-
/api/employers/me/practice-offers/{practiceOfferId}/files/{fileId}	DELETE	Удаление файла из материалов практики	-

Таблица В.15 – Спецификация API контроллера

EmployerPracticeOffersController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/practice-offers	POST	Создание нового предложения о практике от работодателя	-

Продолжение таблицы В.15

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/practice-offers	GET	Получение списка предложений о практике текущего работодателя	-
/api/employers/me/practice-offers/{practiceOfferId}	PUT	Редактирование предложения о практике	-
/api/employers/me/practice-offers/{practiceOfferId}	DELETE	Удаление предложения о практике	-

Таблица В.16 – Спецификация API контроллера

EmployerStudentPracticesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/student-practices	GET	Получение списка практик подопечных студентов	-
/api/employers/me/student-practices/{studentPracticeId}	GET	Получение информации по конкретной практике студента	-
/api/employers/me/student-practices/{studentPracticeId}/report	GET	Скачивание файла отчета студента по практике	-
/api/employers/me/student-practices/{studentPracticeId}/solution	GET	Скачивание файла решения студента по практике	-
/api/employers/me/student-practices/{studentPracticeId}/submission/accept	PUT	Принятие решения студента по практике работодателем	-

Продолжение таблицы В.16

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/student-practices/{studentPracticeId}/submission/revision	PUT	Отправка отчета студента обратно на доработку	-

Таблица В.17 – Спецификация API контроллера

EmployerSubmissionCommentsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/student-practices/{studentPracticeId}/submission/comments	POST	Добавление комментария работодателя к ответу студента по практике	-

Таблица В.18 – Спецификация API контроллера EmployerVacanciesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/vacancies	POST	Создание вакансии текущим работодателем	-
/api/employers/me/vacancies	GET	Просмотр списка вакансий текущего работодателя	-
/api/employers/me/vacancies/{vacancyId}	PUT	Редактирование вакансии работодателя	-
/api/employers/me/vacancies/{vacancyId}	DELETE	Удаление (архивация) вакансии работодателя	-

Таблица В.19 – Спецификация API контроллера FavoriteVacanciesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/favorite-vacancies/{vacancyId}	POST	Добавление вакансии в избранное студентом	-
/api/students/me/favorite-vacancies/{vacancyId}	DELETE	Удаление вакансии из списка избранного	-
/api/students/me/favorite-vacancies	GET	Просмотр списка избранных вакансий студента	-

Таблица В.20 – Спецификация API контроллера ResumeRecommendationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/recommended-resumes	GET	Получение ленты рекомендации для работодателя	PageIndex – номер страницы, PageSize – размер страницы
/api/employers/me/vacancies/{vacancyId}/recommended-resumes	GET	Получение рекомендаций резюме под конкретную вакансию	PageIndex – номер страницы, PageSize – размер страницы

Таблица В.21 – Спецификация API контроллера ResumeViewsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/resumes/{resumeId}/views	GET	Просмотр списка компаний, просматривавших резюме студента	-
/api/employers/me/views-history	GET	История просмотров резюме текущим работодателем	-

Таблица В.22 – Спецификация API контроллера PracticeOffersController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/practice-offers/{practiceOfferId}	GET	Получение информации о конкретном предложении практики	-
/api/practice-offers	GET	Глобальный поиск предложений о практике с фильтрами	Search – поисковый запрос, SearchInTitle – искать в заголовке, SearchInDescription – искать в описании, SearchInCompanyName – искать в названии компании, IsRemote – флаг доступности удаленной практики, Region – регион, SpecializationId – идентификатор специализации, HidePracticesWithoutPlaces – скрыть предложения без свободных мест, PageIndex – номер страницы, PageSize – размер страницы
/api/companies/{companyId}/practice-offers	GET	Получение предложений о практике конкретной компании	-

Таблица В.23 – Спецификация API контроллера ResumesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/resumes/{resumeId}	GET	Получение информации по конкретному резюме студента	-

Продолжение таблицы В.23

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/resumes	GET	Глобальный поиск и фильтрация резюме студентов	SalaryFrom – зарплата от, SalaryTo – зарплата до, Region – регион, SpecializationId – идентификатор специализации, AgeFrom – возраст от, AgeTo – возраст до, SkillIds – массив идентификаторов навыков, MinWorkExperienceYears – опыт от, MaxWorkExperienceYears – опыт до, ResumeUpdatedDateFilter – фильтр даты обновления, UpdatedFrom – дата обновления от, UpdatedTo – дата обновления до, PageIndex – номер страницы, PageSize – размер страницы
/api/students/{studentId}/resumes	GET	Просмотр всех резюме конкретного студента	-

Таблица В.24 – Спецификация API контроллера

ResumeWorkExperiencesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/resumes/{resumeId}/work-experiences	POST	Добавление записи об опыте работы в резюме	-

Продолжение таблицы В.24

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/resumes/{resumeId}/work-experiences/{workExperienceId}	PUT	Редактирование записи об опыте работы	-
/api/students/me/resumes/{resumeId}/work-experiences/{workExperienceId}	DELETE	Удаление записи об опыте работы из резюме	-

Таблица В.25 – Спецификация API контроллера SkillsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/skills	GET	Поиск по справочнику профессиональных навыков	skillName – фильтр по названию навыка

Таблица В.26 – Спецификация API контроллера SpecializationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/specializations	GET	Поиск по справочнику специализаций	specializationName – фильтр по названию специализации

Таблица В.27 – Спецификация API контроллера StudentGroupController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/group	GET	Получение информации об учебной группе студента	-

Таблица В.28 – Спецификация API контроллера StudentApplicationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/applications	POST	Создание отклика студента на вакансию работодателя	-
/api/students/me/applications	GET	Получение списка всех откликов и приглашений текущего студента	ResumeId – идентификатор резюме, Status – статус отклика, PageIndex – номер страницы, PageSize – размер страницы
/api/students/me/applications/{applicationId}	PUT	Изменение или принятие статуса приглашения/отклика студентом	-

Таблица В.29 – Спецификация API контроллера StudentChatsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/chats	GET	Получение списка чатов студента	-
/api/students/me/chats	POST	Создание нового диалога с работодателем по поводу вакансии	-
/api/students/me/chats/{chatId}	GET	Получение списка сообщений внутри конкретного чата студента	-
/api/students/me/chats/{chatId}/read	POST	Отметка сообщений в чате как прочитанных студентом	-

Таблица В.30 – Спецификация API контроллера

StudentGroupApplicationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/group-applications	POST	Подача заявки на вступление в группу по коду приглашения	-
/api/students/me/group-applications	GET	Получение отправленных студентом заявок на вступление в группу	-
/api/students/me/group-applications/{applicationId}	DELETE	Отмена поданной заявки на вступление в группу	-

Таблица В.31 – Спецификация API контроллера

StudentPracticeApplicationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/practice-applications	POST	Подача заявки студентом на конкретное предложение практики	-
/api/students/me/practice-applications/current	GET	Просмотр текущих активных заявок студента на практику	-
/api/students/me/practice-applications/{applicationId}	DELETE	Отмена поданной заявки на практику	-

Таблица В.32 – Спецификация API контроллера StudentPracticePeriodsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/practice-periods	GET	Получение периодов прохождения практики студента	-

Продолжение таблицы В.32

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/practice-periods/{practicePeriodId}	GET	Получение детальной информации о конкретном периоде практики	-

Таблица В.33 – Спецификация API контроллера StudentPracticesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/practices/current	GET	Получение информации по текущей практике студента	-
/api/students/me/practices/current/materials/{materialId}	GET	Скачивание учебных материалов по текущей практике	-
/api/students/me/practices/current/submission	PUT	Отправка отчета, файлов решений или ссылки на выполненную работу	-
/api/students/me/practices/{studentPracticeId}/report	GET	Получение файла собственного отчета по идентификатору практики	-
/api/students/me/practices/{studentPracticeId}/solution	GET	Получение файла собственного решения по идентификатору практики	-

Таблица В.34 – Спецификация API контроллера StudentProfilesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/{id}	GET	Поиск профиля студента по идентификатору	-
/api/students	GET	Поиск профиля студента по адресу электронной почты	email – электронная почта для точного поиска

Таблица В.35 – Спецификация API контроллера StudentResumesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/resumes	POST	Создание нового резюме студентом	-
/api/students/me/resumes	GET	Получение списка собственных резюме студента	-
/api/students/me/resumes/{resumeId}	PUT	Обновление данных конкретного резюме	-
/api/students/me/resumes/{resumeId}	DELETE	Удаление резюме студента	-
/api/students/me/resumes/{resumeId}/copy	POST	Дублирование (копирование) существующего резюме	-

Таблица В.36 – Спецификация API контроллера TeacherPracticePeriodsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/teachers/me/practice-periods/groups	GET	Просмотр групп по текущей практике	-
/api/teachers/me/practice-periods/{practicePeriodId}/groups/{groupId}	GET	Получение информации о периоде практики конкретной группы	-

Таблица В.37 – Спецификация API контроллера TeacherProfilesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/teachers/{id}	GET	Поиск профиля преподавателя по идентификатору	-

Таблица В.38 – Спецификация API контроллера

TeacherStudentPracticesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/teachers/me/student-practices	GET	Просмотр списка практик всех студентов, закрепленных за преподавателем	-
/api/teachers/me/student-practices/{studentPracticeId}	GET	Детальная информация о практике конкретного студента куратором	-
/api/teachers/me/student-practices/{studentPracticeId}/report	GET	Скачивание отчета по практике студента для проверки преподавателем	-
/api/teachers/me/student-practices/{studentPracticeId}/solution	GET	Скачивание файла решения студента преподавателем	-
/api/teachers/me/student-practices/{studentPracticeId}/grade	PUT	Выставление итоговой оценки преподавателем за практику	-

Таблица В.39 – Спецификация API контроллера

TeacherSubmissionCommentsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/teachers/me/student-practices/{studentPracticeId}/submission/comments	POST	Добавление комментария/замечания преподавателя к отправленной работе	-

Таблица В.40 – Спецификация API контроллера UniversitiesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/universities	GET	Поиск по справочнику университетов	universityName – фильтр по названию университета

Таблица В.41 – Спецификация API контроллера
VacancyRecommendationsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/students/me/recommended-vacancies	GET	Получение ленты рекомендаций для студента	PageIndex – номер страницы, PageSize – размер страницы
/api/students/me/resumes/{resumeId}/recommended-vacancies	GET	Получение персональных рекомендаций вакансий под конкретное резюме	PageIndex – номер страницы, PageSize – размер страницы

Таблица В.42 – Спецификация API контроллера VacancyViewsController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/employers/me/vacancies/{vacancyId}/views	GET	Получение списка студентов, просматривавших вакансию	-
/api/students/me/views-history	GET	Просмотр студентом собственной истории просмотров вакансий	-

Таблица В.43 – Спецификация API контроллера VacanciesController

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/vacancies/{vacancyId}	GET	Получение информации по конкретной вакансии	-

Продолжение таблицы В.43

URL запроса	HTTP метод	Описание	Query-параметры запроса
/api/vacancies	GET	Глобальный поиск и фильтрация доступных вакансий в системе	Search – строка поиска, SearchInTitle – искать в заголовке, SearchInDescription – искать в описании, SearchInCompanyName – искать в названии компании, SalaryFrom – зарплата от, SalaryTo – зарплата до, IsRemote – флаг доступности удаленной работы, Region – регион, MinWorkExperienceYears – минимальный опыт, MaxWorkExperienceYears – максимальный опыт, SpecializationId – идентификатор специализации, SkillIds – массив идентификаторов навыков, PageIndex – номер страницы, PageSize – размер страницы
/api/companies/{companyId}/vacancies	GET	Получение списка открытых вакансий конкретной компании	-

Приложение Г

(обязательное)

**Код серверной части информационной платформы трудоустройства и
стажировок студентов в ИТ-компаниях**

Листинг Г.1 – FileStorageService.cs

```

using InternshipPlatform.Application.Dtos.File;
using InternshipPlatform.Application.Exceptions.File;
using InternshipPlatform.Application.Interfaces.Services;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.StaticFiles;

namespace InternshipPlatform.Infrastructure.Services
{
    public class FileStorageService(IWebHostEnvironment env) :
    IFileStorageService
    {
        private const string FilesFolder = "storage";

        private string PracticeMaterialsFolder =>
        Path.Combine(FilesFolder, "practices");

        private string StudentReportsFolder =>
        Path.Combine(FilesFolder, "students", "reports");

        private string StudentSolutionsFolder =>
        Path.Combine(FilesFolder, "students", "solutions");

        private async Task<SavedFileResult> SaveFile(Stream stream,
            string originalFileName,
            string folder,
            string extension)
        {
            var originalSafeFileName =
            Path.GetFileName(originalFileName);

            var storageRoot =
            Path.GetFullPath(Path.Combine(env.ContentRootPath, FilesFolder));
            var targetFolder =
            Path.GetFullPath(Path.Combine(env.ContentRootPath, folder));

            if (!targetFolder.StartsWith(storageRoot,
            StringComparison.Ordinal))
                throw new InvalidOperationException("Некорректный
            путь к папке.");

            Directory.CreateDirectory(targetFolder);

            var storedFileName = $"{Guid.NewGuid():N}{extension}";
            var fullPath = Path.Combine(targetFolder,
            storedFileName);

            await using var fileStream = new FileStream(
                fullPath,
                FileMode.CreateNew,
                FileAccess.Write,
                FileShare.None);

```

```

        await stream.CopyToAsync(fileStream);

        var relativePath =
Path.GetRelativePath(env.ContentRootPath, fullPath)
        .Replace(Path.DirectorySeparatorChar, '/');

        return new SavedFileResult
        {
            OriginalFileName = originalSafeFileName,
            RelativePath = relativePath
        };
    }

    public Task<SavedFileResult> SavePracticeMaterial(Stream
stream, string originalFileName, string extension)
    {
        extension = extension.ToLowerInvariant();

        HashSet<string> allowedExtensions =
[..AllowedTextFileExtensions, ..AllowedArchiveExtensions];

        if (!allowedExtensions.Contains(extension))
            throw new
InvalidFileTypeException(allowedExtensions);

        return SaveFile(stream, originalFileName,
PracticeMaterialsFolder, extension);
    }

    public Task<SavedFileResult> SaveStudentReport(Stream
stream, string originalFileName, string extension)
    {
        extension = extension.ToLowerInvariant();

        if (!AllowedTextFileExtensions.Contains(extension))
            throw new
InvalidFileTypeException(AllowedTextFileExtensions);

        return SaveFile(stream, originalFileName,
StudentReportsFolder, extension);
    }

    public Task<SavedFileResult> SaveStudentSolution(Stream
stream, string originalFileName, string extension)
    {
        extension = extension.ToLowerInvariant();

        if (!AllowedArchiveExtensions.Contains(extension))
            throw new
InvalidFileTypeException(AllowedArchiveExtensions);

```

```

        return SaveFile(stream, originalFileName,
StudentSolutionsFolder, extension);
    }

    public Task<FileDownloadResult?> GetFile(string
relativePath)
    {
        var storageRoot =
Path.GetFullPath(Path.Combine(env.ContentRootPath, FilesFolder));

        var normalized = relativePath
            .TrimStart('/')
            .Replace('/', Path.DirectorySeparatorChar);

        var fullPath =
Path.GetFullPath(Path.Combine(env.ContentRootPath, normalized));

        if (!fullPath.StartsWith(storageRoot,
StringComparison.Ordinal))
            throw new InvalidOperationException("Некорректный
путь к файлу.");

        if (!File.Exists(fullPath))
            return Task.FromResult<FileDownloadResult?>(null);

        var stream = new FileStream(
            fullPath,
            FileMode.Open,
            FileAccess.Read,
            FileShare.Read);

        var provider = new FileExtensionContentTypeProvider();

        if (!provider.TryGetContentType(fullPath, out var
contentType))
            contentType = "application/octet-stream";

        return Task.FromResult<FileDownloadResult?>(new
FileDownloadResult
        {
            Stream = stream,
            ContentType = contentType
        });
    }

    public Task DeleteFileIfExists(string? relativePath)
    {
        if (string.IsNullOrEmpty(relativePath))
            return Task.CompletedTask;

        var storageRoot =
Path.GetFullPath(Path.Combine(env.ContentRootPath, FilesFolder));

```

```

        var normalized = relativePath
            .TrimStart('/')
            .Replace('/', Path.DirectorySeparatorChar);

        var fullPath =
Path.GetFullPath(Path.Combine(env.ContentRootPath, normalized));

        if (!fullPath.StartsWith(storageRoot,
StringComparison.Ordinal))
            throw new InvalidOperationException("Некорректный
путь к файлу.");

        if (File.Exists(fullPath))
            File.Delete(fullPath);

        return Task.CompletedTask;
    }
}

```

Листинг Г.2 – JobApplicationService.cs

```

using InternshipPlatform.Application.Dtos.Chat;
using InternshipPlatform.Application.Dtos.JobApplication;
using InternshipPlatform.Application.Dtos.Pagination;
using InternshipPlatform.Application.Exceptions.JobApplication;
using InternshipPlatform.Application.Exceptions.Resume;
using InternshipPlatform.Application.Exceptions.Vacancy;
using InternshipPlatform.Application.Interfaces;
using InternshipPlatform.Application.Interfaces.Notifiers;
using InternshipPlatform.Application.Interfaces.Repositories;
using InternshipPlatform.Application.Interfaces.Services;
using InternshipPlatform.Application.Mappers;
using InternshipPlatform.Application.Values;

namespace InternshipPlatform.Application.Services
{
    public class JobApplicationService(
        IJobApplicationRepository applicationRepository,
        IChatService chatService,
        IResumeRepository resumeRepository,
        IVacancyRepository vacancyRepository,
        IEmployerProfileRepository employerProfileRepository,
        IStudentProfileRepository studentProfileRepository,
        IJobApplicationNotifier jobApplicationNotifier,
        IUnitOfWork unitOfWork) : IJobApplicationService
    {
        public async Task<int>
CreateJobApplicationByStudent(CreateJobApplicationRequest request)
        {
            if (!await
resumeRepository.IsStudentOwnsResume(request.UserId,
request.ResumeId))

```

```

        throw new ResumeNotFoundException();

        var vacancy = await
vacancyRepository.GetVacancyById(request.VacancyId);

        if (vacancy is null || !vacancy.IsActive)
            throw new VacancyNotFoundException();

        if (await
applicationRepository.HasStudentActiveApplicationOnVacancy(request
.ResumeId, request.VacancyId))
            throw new
ActiveApplicationAlreadyExistsException();

        var chatId = await
chatService.GetOrCreateStudentChat(new StartStudentChatRequest
        {
            StudentId = request.UserId,
            VacancyId = request.VacancyId,
            Message = request.WelcomeMessage
        });

        var application = request.ToDomain(Roles.Student,
chatId);
        await
applicationRepository.AddJobApplication(application);

        await unitOfWork.SaveChangesAsync();

        var employerEmail = await
employerProfileRepository.GetEmployerEmailByCompanyId(vacancy.Comp
anyId);

        if (!string.IsNullOrEmpty(employerEmail))
            await
jobApplicationNotifier.NotifyEmployerAboutStatusChangedAsync(emplo
yerEmail, vacancy.Title, JobApplicationStatuses.Pending);

        return application.Id;
    }

    public async Task<int>
CreateJobApplicationByEmployer(CreateJobApplicationRequest
request)
    {
        if (!await
vacancyRepository.IsEmployerOwnsVacancy(request.UserId,
request.VacancyId))
            throw new VacancyNotFoundException();

        var resume = await
resumeRepository.GetResumeById(request.ResumeId);

```

```

        if (resume is null || !resume.IsActive)
            throw new ResumeNotFoundException();

        if (await
applicationRepository.HasStudentActiveApplicationOnVacancy(request
.ResumeId, request.VacancyId))
            throw new
ActiveApplicationAlreadyExistsException();

        var chatId = await
chatService.GetOrCreateEmployerChat(new StartEmployerChatRequest
{
            EmployerId = request.UserId,
            StudentId = resume.StudentId,
            VacancyId = request.VacancyId,
            Message = request.WelcomeMessage
        });

        var application = request.ToDomain(Roles.Employer,
chatId);
        await
applicationRepository.AddJobApplication(application);

        await unitOfWork.SaveChangesAsync();

        var vacancy = await
vacancyRepository.GetVacancyById(request.VacancyId)
            ?? throw new VacancyNotFoundException();

        var studentEmail = await
studentProfileRepository.GetStudentEmailById(resume.StudentId);

        if (!string.IsNullOrEmpty(studentEmail))
            await
jobApplicationNotifier.NotifyStudentAboutStatusChangedAsync(studen
tEmail, vacancy, JobApplicationStatuses.InterviewInvited);

        return application.Id;
    }

    public async
Task<PagedResponse<StudentApplicationResponse>>
GetStudentApplications(GetStudentApplicationsParameters
parameters)
    {
        if (parameters.ResumeId.HasValue
            && !await
resumeRepository.IsStudentOwnsResume(parameters.StudentId,
parameters.ResumeId.Value))
            throw new ResumeNotFoundException();

        var applications = await
applicationRepository.GetStudentApplications(parameters);

```

```

        return applications.ToPagedResponse(parameters, a =>
a.ToStudentApplication());
    }

```

```

    public async
Task<PagedResponse<EmployerApplicationResponse>>
GetEmployerApplications(GetEmployerApplicationsParameters
parameters)
    {
        if (parameters.VacancyId.HasValue
            && !await
vacancyRepository.IsEmployerOwnsVacancy(parameters.EmployerId,
parameters.VacancyId.Value))
            throw new VacancyNotFoundException();

        var applications = await
applicationRepository.GetEmployerApplications(parameters);

        return applications.ToPagedResponse(parameters, a =>
a.ToEmployerApplication());
    }

```

```

    private bool IsValidStatusTransition(string role,
JobApplicationStatuses currentStatus, JobApplicationStatuses
newStatus)
    {
        if (currentStatus == newStatus)
            return false;

        return (currentStatus, newStatus, role) switch
        {
            (JobApplicationStatuses.Pending,
JobApplicationStatuses.InterviewInvited, Roles.Employer) => true,
            (JobApplicationStatuses.Pending,
JobApplicationStatuses.Rejected, Roles.Employer) => true,
            (JobApplicationStatuses.Pending,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,

            (JobApplicationStatuses.InterviewInvited,
JobApplicationStatuses.OfferReceived, Roles.Employer) => true,
            (JobApplicationStatuses.InterviewInvited,
JobApplicationStatuses.Rejected, Roles.Employer) => true,
            (JobApplicationStatuses.InterviewInvited,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,

            (JobApplicationStatuses.OfferReceived,
JobApplicationStatuses.Accepted, Roles.Student) => true,
            (JobApplicationStatuses.OfferReceived,
JobApplicationStatuses.Rejected, Roles.Employer) => true,
            (JobApplicationStatuses.OfferReceived,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,

```

```

        (JobApplicationStatuses.Accepted,
JobApplicationStatuses.Employed, Roles.Employer) => true,
        (JobApplicationStatuses.Accepted,
JobApplicationStatuses.Withdrawn, Roles.Student) => true,
        _ => false
    };
}

public async Task
UpdateApplicationStatus(UpdateApplicationStatusRequest request)
{
    var application = request.Role switch
    {
        Roles.Student => await
applicationRepository.GetStudentApplicationForUpdate(
    request.UserId,
    request.ApplicationId),

        Roles.Employer => await
applicationRepository.GetEmployerApplicationForUpdate(
    request.UserId,
    request.ApplicationId),

        _ => null
    } ?? throw new JobApplicationNotFoundException();

    var currentStatus =
(JobApplicationStatuses)application.ApplicationStatusId;

    if (!IsValidStatusTransition(request.Role!,
currentStatus, request.ApplicationStatus))
        throw new InvalidJobApplicationStatusException();

    application.ApplicationStatusId =
(int)request.ApplicationStatus;
    application.LastStatusDate =
DateOnly.FromDateTime(DateTime.UtcNow);

    if (request.ApplicationStatus ==
JobApplicationStatuses.Rejected ||
        request.ApplicationStatus ==
JobApplicationStatuses.Withdrawn)
        await
chatService.CloseChatByApplicationId(request.UserId,
request.Role!, request.ApplicationId);

    await unitOfWork.SaveChangesAsync();

    var vacancy = await
vacancyRepository.GetVacancyById(application.VacancyId)
        ?? throw new VacancyNotFoundException();

```

```

        if (request.Role == Roles.Employer)
        {
            var studentEmail = await
resumeRepository.GetStudentEmailByResumeId(application.ResumeId);
            if (!string.IsNullOrEmpty(studentEmail))
                await
jobApplicationNotifier.NotifyStudentAboutStatusChangedAsync(studen
tEmail, vacancy, request.ApplicationStatus);
        }
        else if (request.Role == Roles.Student)
        {
            var employerEmail = await
employerProfileRepository.GetEmployerEmailByCompanyId(vacancy.Comp
anyId);
            if (!string.IsNullOrEmpty(employerEmail))
                await
jobApplicationNotifier.NotifyEmployerAboutStatusChangedAsync(emplo
yerEmail, vacancy.Title, request.ApplicationStatus);
        }
    }
}

```

Листинг Г.3 – NotificationConsumer.cs

```

using Confluent.Kafka;
using Microsoft.Extensions.Options;
using NotificationService.Dto;
using NotificationService.Interfaces;
using NotificationService.Json;

namespace NotificationService.Services
{
    public class NotificationConsumer(
        IOptions<KafkaSettings> kafkaOptions,
        IServiceScopeFactory scopeFactory) : BackgroundService
    {
        private const string Topic = "PlatformNotifications";

        protected override Task ExecuteAsync(CancellationToken
stoppingToken)
        {
            var config = new ConsumerConfig
            {
                GroupId = "InternshipPlatform",
                BootstrapServers =
kafkaOptions.Value.BootstrapServers,
                AutoOffsetReset = AutoOffsetReset.Earliest,
            };

            Task.Run(async () =>
            {

```

```

        using (var consumer = new ConsumerBuilder<Ignore,
NotificationEvent>(config)
            .SetValueDeserializer(new
JsonDeserializer<NotificationEvent>())
            .Build())
        {
            consumer.Subscribe(Topic);

            while (!stoppingToken.IsCancellationRequested)
            {
                var consumerResult =
consumer.Consume(TimeSpan.FromSeconds(5));

                if (consumerResult == null)
                    continue;

                using var scope =
scopeFactory.CreateScope();

                var emailSender = scope.ServiceProvider
                    .GetRequiredService<IEmailSender>();

                var emailMessage =
consumerResult.Message.Value;

                await
emailSender.SendEmailAsync(emailMessage);

                consumer.Commit(consumerResult);
            }
        }, stoppingToken);

        return Task.CompletedTask;
    }
}

```

Приложение Д
(рекомендуемое)

**Результаты тестирования серверной части информационной платформы
трудоустройства и стажировок студентов в ИТ-компаниях**

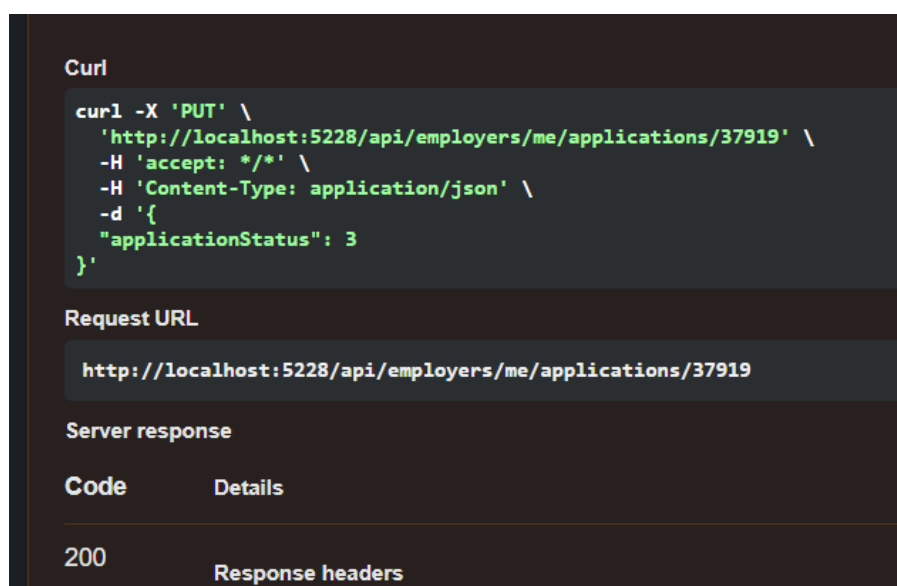


Рисунок Д.1 – Демонстрация отклонения отклика работодателем

```
{
  "id": 37919,
  "vacancyId": 6306,
  "vacancyTitle": "Стажёр frontend-разработчик",
  "companyId": 1260,
  "companyName": "000 \"Колесников Торг\"",
  "companyLogoPath": null,
  "lastStatusDate": "2026-06-04",
  "applicationStatus": "Отклонено",
  "chatId": 37919
}
```

Рисунок Д.2 – Демонстрация обновления статуса отклика на «Отклонено»

```
{
  "id": 37919,
  "vacancyId": 6306,
  "vacancyTitle": "Стажёр frontend-разработчик",
  "companyId": 1260,
  "companyName": "000 \"Колесников Торг\"",
  "companyLogoPath": null,
  "messages": [],
  "isClosed": true
}
```

Рисунок Д.3 – Демонстрация закрытия чата

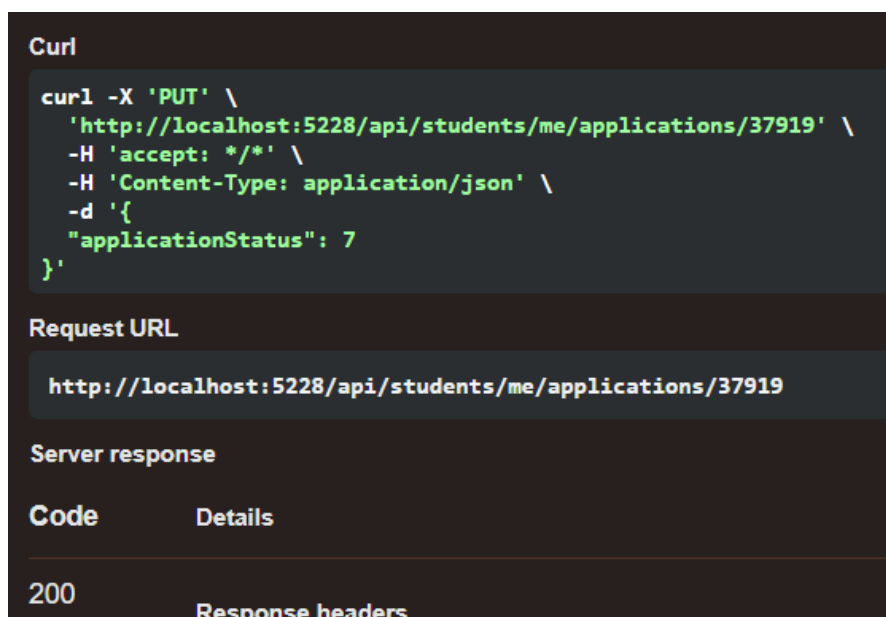


Рисунок Д.4 – Демонстрация отзыва отклика студентом

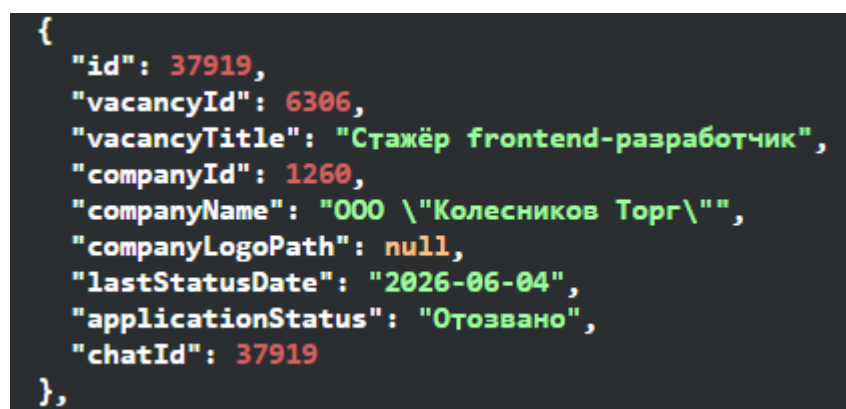


Рисунок Д.5 – Демонстрация обновления статуса отклика на «Отозвано»

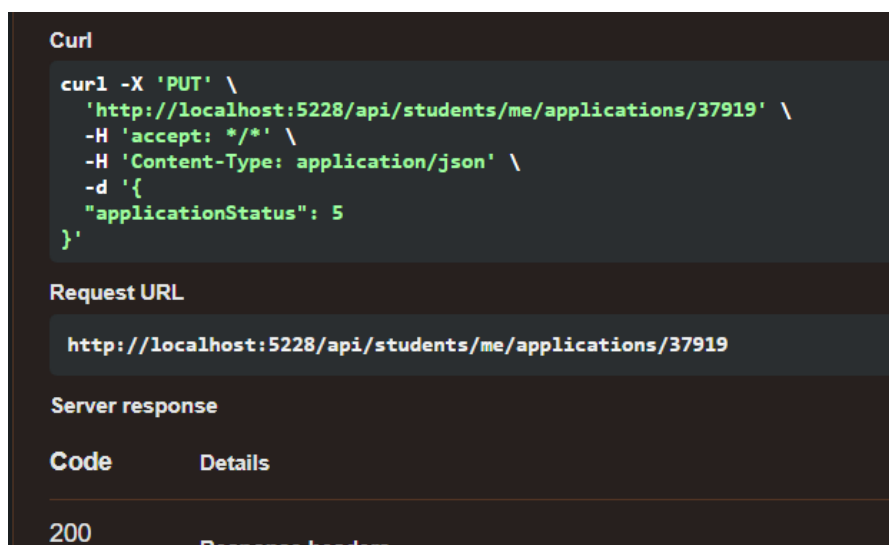


Рисунок Д.6 – Демонстрация принятия студентом приглашения на работу

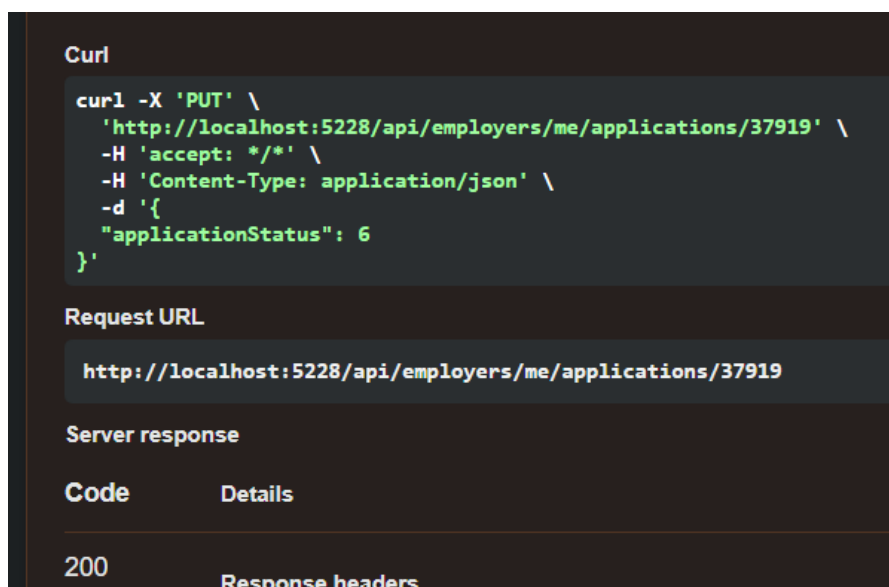


Рисунок Д.7 – Демонстрация подтверждения трудоустройства работодателем

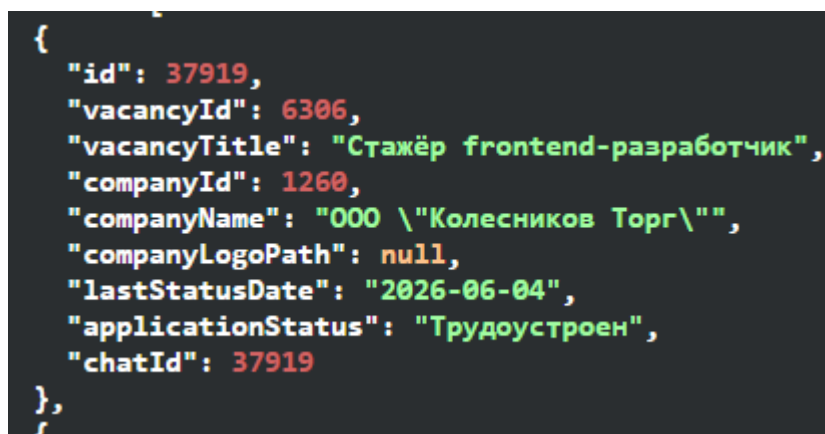


Рисунок Д.8 – Демонстрация обновления статуса отклика у студента на «Трудоустроен»