

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	1
ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ.....	3
ВВЕДЕНИЕ.....	5
1 Теоретический обзор.....	7
1.1 Обзор предметной области.....	7
1.2 Обзор существующих решений по поиску стажировок.....	10
1.2.1 Headhunter.....	10
1.2.2 FutureToday.....	10
1.2.3 Jobby.AI.....	11
1.2.4 Superjob Students	12
1.2.5 Выводы по обзору платформ для поиска стажировок.....	12
1.3 Вывод по первой главе.....	13
2 Проектирование приложения.....	15
2.1 Функциональные и нефункциональные требования.....	15
2.1.1 Авторизация и регистрация.....	15
2.1.2 Стажировки.....	15
2.1.3 Тесты.....	15
2.1.4 Компании.....	16
2.1.5 Профиль.....	16
2.2 Проектирование базы данных.....	18
2.2.1 Обзор и выбор базы данных.....	18
2.2.2 Создание структуры и определение таблиц.....	21
2.3 Выбор технологий для разработки серверной части.....	24
2.3.1 Обзор и выбор архитектуры серверной части приложения.....	24
2.3.2 Выбор языка программирования для реализации серверной части приложения.....	26
2.3.3 Обзор и выбор фреймворка для создания приложения по выбранному языку программирования.....	27
2.4 Выбор технологий для разработки клиентской части.....	29
2.4.1 Обзор и выбор архитектурного подхода для разработки клиентской части приложения.....	29
2.4.2 Обзор и выбор языка программирования для клиентской части приложения.....	30
2.4.3 Выбор фреймворка для клиентской части приложения.....	31
2.5 Проектирование системы.....	32

2.6 Вывод по второй главе.....	33
3 Реализация приложения.....	35
3.1 Разработка модуля аутентификации.....	35
3.2 Разработка модуля поиска и создания стажировок.....	38
3.3 Разработка модуля сортировки стажировок с учетом навыков пользователя.....	44
3.4 Разработка модуля создания и прохождения тестовых заданий.....	52
3.5 Нагрузочное тестирование приложения.....	58
3.6 Вывод по третьей главе.....	60
ЗАКЛЮЧЕНИЕ.....	62
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	64

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Аутентификация - это процесс установления подлинности пользователя.

Авторизация - процесс предоставления прав доступа пользователю к определенному ресурсу.

БД (База данных) - система для хранения и управления данными.

Мониторинг - процесс наблюдения за состоянием системы, приложения или процесса. Используется для выявления проблем и предотвращения сбоев.

Хэширование - преобразование данных в набор символов фиксированной длины (хэш). Используется для проверки целостности данных, аутентификации и шифрования.

Ресурс - любой объект, доступный для использования в системе, например файл, страница сайта, база данных.

Даталогическая модель (Data model) - описание структуры данных, их взаимосвязей в базе данных.

Фреймворк (Framework) - структура для разработки программного обеспечения, предоставляющая набор инструментов.

Model-View-Controller (MVC) - архитектурный шаблон, разделяющий разработку веб-приложения на модель (данные в приложении), представление (интерфейс приложения) и контроллер (логика приложения).

HTML (HyperText Markup Language) - язык разметки документов для веб-страниц.

Рендеринг - процесс создания HTML, CSS и JavaScript кода на стороне клиента на основе данных полученных от сервера

Стек технологий - совокупность технологий, используемых для разработки и поддержки программного обеспечения.

Backend - серверная часть приложения, отвечающая за обработку данных, логику и взаимодействие с базами данных.

Frontend - клиентская часть приложения, отвечающая за взаимодействие с пользователем, интерфейс и отображение данных в браузере.

URL-адрес (Uniform Resource Locator) - адрес веб-ресурса в интернете.

Идентификатор - уникальный код, используемый для опознания объекта.

JSON-файл (JavaScript Object Notation) - формат представления данных в виде текста. Используется для обмена между сервером и клиентом.

ВВЕДЕНИЕ

В современном мире студентам необходимо приобретать актуальные навыки и знания о новых технологиях, чтобы быть конкурентоспособными на рынке труда. Одним из способов достичь этого является прохождение стажировки в компаниях, работающих с актуальными технологиями. Однако студенты часто сталкиваются с трудностями в поиске подходящих стажировок. При поиске стажировки студенты часто отправляют заявки во многие компании. Поэтому количество тестовых заданий может быть большим. Из-за этого возникает проблема отслеживания за статусом тестового задания

Создание веб-приложения для поиска стажировок, в котором будет система для прохождения тестовых заданий, может стать отличным решением для данной проблемы. Веб-сайт должен учитывать профессиональные навыки студента, актуальные технологические требования компаний и содержать широкий спектр программ стажировок. Особое внимание следует уделить процессу выполнения тестового задания для прохождения на стажировку, сделав его максимально прозрачным, чтобы можно было мониторить, какие тестовые задания сейчас у кандидата на стажировку активные, а какие уже прошли проверку от компании. Также система должна иметь конструктор тестов для компании, в котором будет спектр инструментов для разнообразия вариантов выполнения задач.

Такое приложение позволит:

- Обеспечить более точное соответствие между потребностями студентов и ожиданиями работодателей в области стажировок.
- Снизить нагрузку на студентов, связанную с загрузкой своих решений на различные платформы, каждая из которых имеет свой интерфейс, требования и сроки. То есть, система будет иметь удобный функционал отправки выполненных заданий, а также будет обеспечивать мониторинг статуса выполнения тестового задания.

Целью моей дипломной работы является улучшение процесса поиска стажировок и прохождения тестовых заданий. Для этого будет создано веб-приложение для поиска стажировок в различных компаниях с возможностью прохождения тестовых заданий. Это позволит студентам проходить тесты, результаты которых будут доступны принимающим компаниям.

Для достижения поставленной цели были сформулированы следующие задачи:

- 1) Исследовать и проанализировать существующие аналоги по подбору стажировок на основе стека технологий.
- 2) Исследовать и проанализировать существующие технологии для создания веб-приложения.
- 3) Спроектировать серверную часть веб-приложения на основе выбранного языка программирования, определить структуру базы данных, описать требования по функционалу веб-приложения.
- 4) Спроектировать клиентскую часть веб-приложения на базе выбранного языка программирования.
- 5) Разработать серверную часть веб-приложения с использованием выбранного фреймворка.
- 6) Разработать клиентскую часть веб-приложения с использованием выбранного фреймворка.

Данная платформа будет актуальна для людей, которые только начинают путь разработчика, так и для тех, кто хочет поменять сферу деятельности.

1 Теоретический обзор

1.1 Обзор предметной области

Стажировка – это образовательная программа, предлагаемая компанией для студентов или специалистов, желающих получить практический опыт в определенной сфере деятельности. Стажировка может быть оплачиваемой или неоплачиваемой и обычно имеет ограниченную продолжительность.

Есть еще понятие “испытательный срок”. Испытательный срок – это период, в течение которого работодатель оценивает навыки нового сотрудника занимаемой должности. Испытательный срок устанавливается в рамках трудового договора и предполагает выполнение реальных рабочих задач.

Стажировка и испытательный срок — это разные этапы профессионального пути, каждый из которых имеет свои особенности и преимущества.

Преимущества стажировки относительно испытательного срока:

- Фокус на обучении: Главная цель стажировки — обучение и развитие. Стажеры получают возможность познакомиться с компанией, ее процессами, а также получить практические навыки под руководством специалистов.
- Меньше ответственности и давления: Стажеры не несут полной ответственности за результаты своей работы, как сотрудники на испытательном сроке. Это позволяет им учиться, экспериментировать, задавать вопросы без страха совершения ошибки.
- Возможность попробовать себя в разных ролях: Некоторые программы стажировок предусматривают систему, по которой можно попробовать себя в различных позициях. Это позволяет стажерам получить более широкое представление о разных сферах деятельности.

- Нет жестких обязательств: Стажировка не предполагает обязательств по трудоустройству. Стажер может отказаться от предложения о работе, а компания – не продлевать сотрудничество после окончания стажировки.
- Снижение рисков для обеих сторон: Стажировка позволяет компании оценить навыки кандидата без рисков, связанных с наймом. Стажер, в свою очередь, может "примерить" на себя профессию и компанию, прежде чем принимать решение о дальнейшей работе на постоянной основе.

Испытательный срок, в свою очередь, направлен на оценку соответствия сотрудника занимаемой должности и предполагает выполнение реальных рабочих задач с полной ответственностью за результат.

Таким образом, стажировка - это более гибкий и менее рискованный способ получить практический опыт и определиться с дальнейшим профессиональным путем.

Чтобы пройти на стажировку студенты в большинстве случаев проходят ряд тестирований. Есть множество способов оценки специалистов, которые уникальны для каждой компании. Но для отсева большинства кандидатов используется тестовое задание, чтобы быстро оценить минимальную компетентность, а также для дальнейшего допуска уже на более узконаправленное тестирование.

Тестовое задание — это задача или набор задач, которые предлагаются кандидату на стажировку для оценки его профессиональных навыков, знаний и способностей.

На большинстве агрегаторов, на которых есть стажировки от малых и средних компаний, при отклике пользователя работодатель в большинстве случаев отправляет ссылку с инструкцией и техническим заданием для выполнения тестового задания. Большие компании в качестве тестового задания используют систему для решения алгоритмических задач.

В результате данного отбора решается вопрос о дальнейшем рассмотрении кандидата на прохождение стажировки в компании.

Надо учитывать, что для разных компаний требуются свои формы для проведения тестовых заданий, которые отражают реальные задачи при работе.

При выполнении тестовых заданий кандидаты могут попробовать себя в решении различных задач и при этом в любой момент отказаться от выполнения, если что-то не понравилось.

При рассмотрении тестовых заданий, которые предлагают большие компании, можно заметить, что они используют “контест” в качестве самой первой проверки на компетентность кандидата. Контест – это соревнование, в котором кандидатам предлагается решить набор задач за ограниченное время. Таким образом, компании автоматически отсеивают кандидатов, которые не набрали определенный балл в контесте.

Средние и малые компании, у которых нет большой системы для проверки написанного кода, используют более упрощенный вариант для проведения тестового задания. В первом случае, это обычный тест на знание технологий, то есть, примитивный тест, где нет никакого функционала, а просто выбор из 4 вариантов. Во втором случае, это техническое задание с инструкцией по выполнению, где нужно отправить выполненное кандидатом задание (это может быть код, либо какой-нибудь документ с описанием реализации) на почту, либо в систему хранения данных.

Таким образом, у каждой компании есть своя система для проведения тестового задания без каких-либо стандартизированных условий. Кандидату, для мониторинга всех тестовых заданий, необходимо запоминать что и где находится. Из этого возникает проблема раздробленности процесса выполнения тестовых заданий.

Для решения данной проблемы я предлагаю систему, которая имеет удобный функционал для поиска стажировок, а также возможность проходить тестовое задание и мониторить статус проверки выполнения.

1.2 Обзор существующих решений по поиску стажировок

Рассмотрим платформы, которые имеют функционал по поиску стажировок в различных компаниях.

1.2.1 Headhunter

HeadHunter - платформа для поиска работы и подбора персонала [1]. Она предоставляет возможность работодателям эффективно находить потенциальных сотрудников, а также помогает соискателям в поиске подходящих вакансий. У сайта есть функционал по поиску стажировок, то есть веб-сайт не фокусируется на поиске стажировок, а имеет возможность отображать доступные стажировки, которые предлагают компании.

Веб-сайт предлагает хорошую фильтрацию при поиске стажировок. Можно отметить в фильтре пункт “стажировка”, тогда будут показываться только актуальные стажировки, а также система будет показывать в первую очередь те стажировки, которые больше подходят пользователю. Указать профессиональные навыки можно в профиле пользователя.

Что касается прохождения тестового задания, то тут полностью отсутствует функционал прохождения тестовых заданий от компаний, но стоит отметить, что сайт предоставляет чат для общения с менеджерами по отбору специалистов, которые отправляют ссылку на систему, где можно проходить тестовое задание. В большинстве случаев это обычная файловая система, где нужно отправлять свое выполненное задание.

1.2.2 FutureToday.

FutureToday - платформа для поиска программ стажировок, а также различных школ для профессиональной подготовки [2]. У сайта есть

множество полезных функций для подбора стажировок, а также есть функционал для работодателей:

- Поиск программ стажировок: Пользователи могут искать актуальные стажировки на различные компании
- Подробная информация: На сайте можно выделить оформление программ по стажировкам и различным олимпиадам, каждая страница является своеобразной рекламной брошюрой с подробным описанием всех направлений стажировок.
- Рассылка: В FutureToday можно подписаться на рассылку актуальной информации по различным стажировочным программам.
- Рейтинг работодателей: На платформе есть рейтинг работодателей, который актуализируется на основе голосов студентов старших курсов, тем самым мы можем рассмотреть тенденцию рынка работодателей.

У веб-сайта отсутствует фильтрация по выбранному стеку технологий пользователя. А также нет системы для прохождения тестового задания, а также для большинства стажировок нужно переходить на сайт компании и там подавать заявку на прохождение стажировки.

1.2.3 Jobby.AI.

Jobby - это платформа для поиска вакансий с учетом навыков, сайт предоставляет широкий спектр программ стажировок в различных компаниях [3]. Также у веб-сайта есть функционал для поиска стажировок по выбранным технологиям и возможность указать, какими профессиональными навыками обладает пользователь.

Веб-сайт не предоставляет функционала для прохождения тестовых заданий. Но стоит отметить, что есть возможность напрямую отправить заявку для прохождения стажировки.

1.2.4 Superjob Students .

Superjob - это платформа для поиска работы и подбора персонала, но также сайт обладает функциями для поиска стажировок [4]. Данный сайт не акцентируется на поиске стажировок, а дает возможность работодателем публиковать доступные их компанией стажировки.

Платформа имеет определенный фильтр для отображения стажировок. Что касается функционала по прохождению тестового задания, то она отсутствует. Пользователи могут напрямую откликнуться по предложениям на стажировку и ждать обратной связи. После этого менеджер по подбору специалистов отправляет ссылку на тестовое задание, которое нужно пройти кандидату.

1.2.5 Выводы по обзору платформ для поиска стажировок.

Я рассмотрел несколько самых популярных агрегаторов вакансий в России, которые имеют функционал для поиска стажировок. Исходя из обзора, можно понять, что все рассмотренные веб-сайты не имеют функционала для прохождения тестового задания от компаний. Каждый из них имеет функционал для общения между менеджерами по подбору персонала и пользователями, чтобы в дальнейшем установить связь и провести отбор на стажировку в своей системе.

Чтобы удобнее было рассматривать характеристику между существующими решениями для поиска стажировок, я приведу сравнительную характеристику, которая представлена на таблице 1.1.

Основные критерии:

- Есть функционал по поиску стажировок
- Функционал для проведения тестового задания
- Фильтрация стажировок по профессиональным навыкам

- Мониторинг статуса выполнения тестового задания

Таблица 1.1 - Сравнительная характеристика аналогов

Критерий	HeadHunter	FutureToday	JobbyAI	SuperJob
Есть функционал по поиску стажировок	Да	Да	Да	Да
Функционал для проведения тестового задания	Нет	Нет	Нет	Нет
Фильтрация стажировок по профессиональным навыкам	Да	Нет	Да	Нет
Мониторинг статуса выполнения тестового задания	Нет	Нет	Нет	Нет

Исходя из данной сравнительной характеристики, можно сделать вывод о том, что реализация системы, которая будет включать в себя функционал по прохождению тестовых заданий, а также мониторинг статуса проверки выполнения будет конкурентоспособна среди рассмотренных платформ.

1.3 Вывод по первой главе

В данной главе была рассмотрена предметная область, а также обзор существующих решений по поиску стажировок. Был изучен функционал каждой платформы, а также выделен недостаток в виде системы для прохождения тестового задания, которая является индивидуальной для каждой компании. Из этого вытекает проблема при отслеживании кандидатом каждой отправленной заявки на прохождение стажировки. При получении обратной связи компания отправляет ссылку, где находится тестовое задание.

Зачастую, при поиске стажировок кандидат отправляет множество заявок в различные компании, поэтому появляются трудности при мониторинге статуса выполнения тестового задания, а также место куда нужно отправлять выполненное задание нужно вручную запоминать или сохранять. На основе проведенного обзора можно сделать вывод об актуальности создания веб-приложения для поиска стажировок с упором на систему для выполнения тестового задания.

2 Проектирование приложения

2.1 Функциональные и нефункциональные требования

Для разработки веб-приложения распишем функциональные и нефункциональные требования.

2.1.1 Авторизация и регистрация

- 1) Система должна авторизовать пользователя при успешном прохождении аутентификации
- 2) Система должна поддерживать аутентификацию по login и паролю
- 3) Система должна поддерживать выход из системы
- 4) Система должна поддерживать регистрацию пользователя по login, email и паролю
- 5) Система должна поддерживать уникальность login пользователей

2.1.2 Стажировки

- 1) Система должна поддерживать отображение стажировок на платформе
- 2) Система должна поддерживать сортировку стажировок по профессиональным навыкам пользователя
- 3) Система должна поддерживать поиск стажировок по названию стажировки
- 4) Система должна поддерживать отображение дополнительной информации о стажировке.
- 5) Система должна поддерживать отображение подходящих для пользователя стажировок
- 6) Система должна поддерживать создание/редактирование/удаление стажировок

2.1.3 Тесты

- 1) Система должна поддерживать прохождение тестов от компаний авторизованными пользователями
- 2) Система должна поддерживать корректное отображение тестов

- 3) Система должна сохранять результаты тестирования пользователя
- 4) Система должна поддерживать отправку тестов на проверку
- 5) Система должна поддерживать корректное отображение вопросов для теста
- 6) Система должна поддерживать создание тестов в конструкторе тестов компаниями
- 7) Система должна поддерживать редактирование/удаление вопросов теста компаниями

2.1.4 Компании

- 1) Система должна корректно отображать компании
- 2) Система должна поддерживать отображение дополнительной информации о компании
- 3) Система должна поддерживать отображение всех доступных стажировок компании
- 4) Система должна поддерживать сортировку стажировок компании по технологиям
- 5) Система должна поддерживать создание/редактирование/удаление компаний

2.1.5 Профиль

- 1) Система должна поддерживать редактирование навыков пользователя

Для неавторизованных пользователей будет доступна лишь страница с авторизацией и регистрацией

Для более детального ознакомления была создана диаграмма вариантов использования, рисунок 2.1.

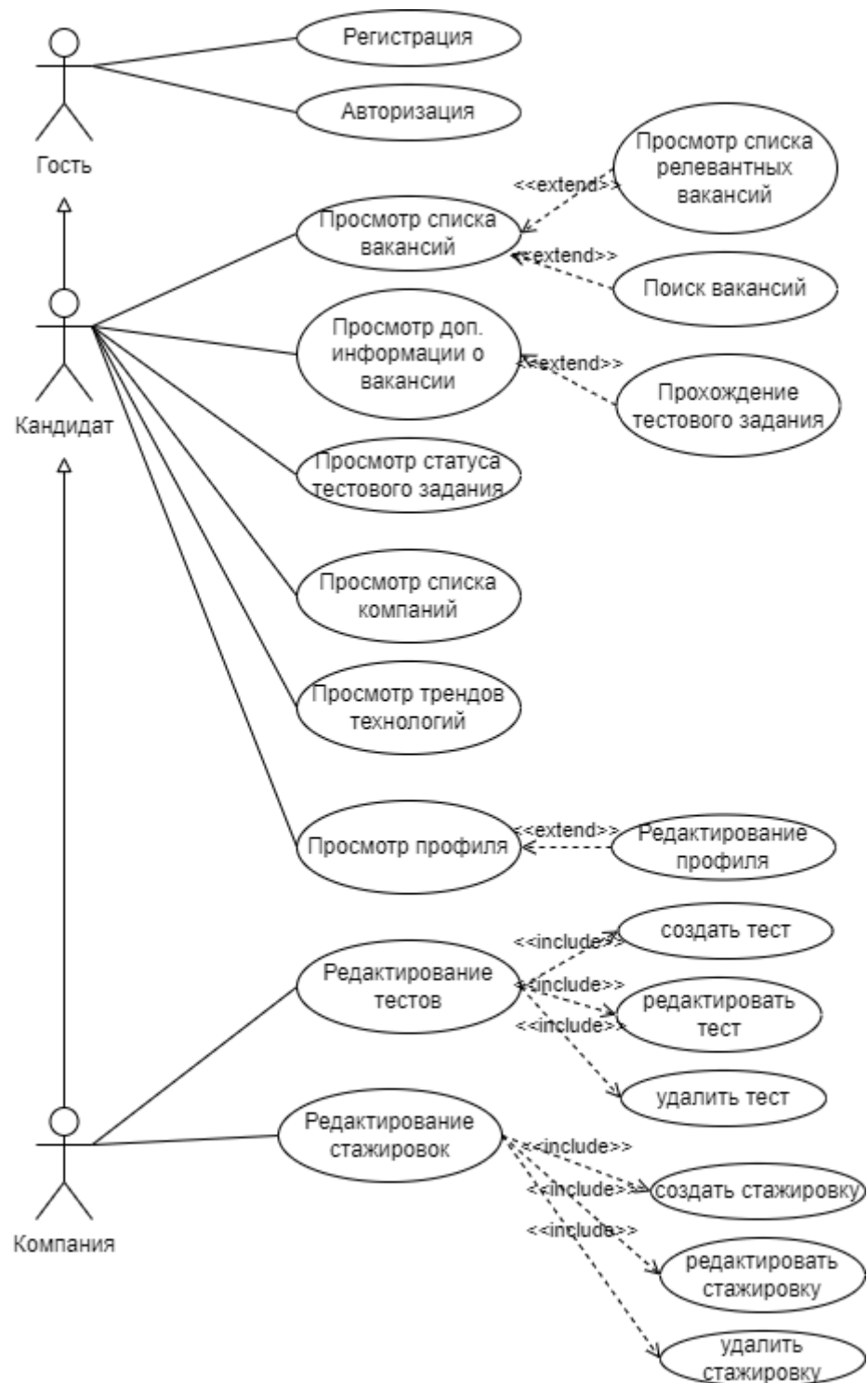


Рисунок 2.1 - диаграмма вариантов использования

Также распишем нефункциональные требования для системы в целом.
В требованиях будут описания свойства системы:

- Надежность: Будет реализован мониторинг приложения для контроля за работоспособностью и производительностью приложения.
- Безопасность: Использование безопасных методов аутентификации, такие как хэширование паролей и использование токенов, шифрование данных в БД.

2.2 Проектирование базы данных

База данных является основным компонентом для хранения данных системы, поэтому выбор технологий для создания стабильного и защищенного хранилища является основным пунктом при проектировании. В данном этапе будет обзор и выбор базы данных, а также будет описана структура и определение таблиц.

2.2.1 Обзор и выбор базы данных

При выборе баз данных нужно определиться с моделью базы, которая подойдет под мое веб-приложение. Можно выделить два типа базы данных: реляционная, нереляционная. Реляционная модель базы данных используется в основном для хранения данных, которые можно отобразить в виде таблицы. Нереляционная модель базы данных не использует привычную табличную систему. В этих базах данных используется модель хранения, оптимизированная под конкретные требования. Например: можно хранить данные “ключ-значения”, мультимедиа.

Под мое приложение подходит реляционная модель баз данных, так как я буду использовать структурированные данные: тесты, пользователи, вопросы. Все эти таблицы будут иметь определенную структуру. Рассмотрим самые популярные реляционные базы данных:

PostgreSQL [5]

- База данных с открытым исходным кодом.
- Сильные стороны: Надежность, расширяемость, строгая поддержка SQL стандартов, богатый набор функций.

- Слабые стороны: Может уступать в производительности при операциях чтения.
- Идеально подходит для: Проектов, где важна надежность, соответствие стандартам и расширяемость, а также для работы со сложными схемами данных.

MySQL [6]

- База данных с открытым исходным кодом.
- Сильные стороны: Простота использования, высокая производительность при операциях чтения, широкая распространенность.
- Слабые стороны: Ограниченная функциональность по сравнению с PostgreSQL, некоторые отклонения от SQL стандартов.
- Идеально подходит для: Проекты, где важна скорость чтения данных и простота настройки.

MariaDB [7]

- База данных с открытым исходным кодом.
- Сильные стороны: Высокая совместимость с MySQL, активная разработка, позиционируется как более производительная и функциональная альтернатива MySQL.
- Слабые стороны: Меньшая распространенность по сравнению с MySQL.
- Идеально подходит для: Проектов, использующих MySQL, но нуждающихся в расширенной функциональности или большей производительности.

Oracle Database [8]

- Используется коммерческий код

- Сильные стороны: Широкий набор функций, высокая надежность, масштабируемость, отличная производительность.
- Слабые стороны: Высокая стоимость, сложность в настройке и поддержке.
- Идеально подходит для: Крупных приложений, критически важных систем, где требуется максимальная надежность и производительность.

Microsoft SQL [9]

- Используется коммерческий код
- Сильные стороны: Широкая функциональность, хорошая интеграция с другими продуктами Microsoft, высокая производительность.
- Слабые стороны: Высокая стоимость, привязка к платформе Microsoft.
- Идеально подходит для: Проектов, разрабатываемых на платформе Microsoft, где важна интеграция с другими продуктами Microsoft.

SQLite [10]

- База данных с открытым исходным кодом.
- Сильные стороны: Встраиваемая база данных, простота использования, не требует настройки, идеальна для локальных приложений.
- Слабые стороны: Ограниченная функциональность, не подходит для крупных проектов.
- Идеально подходит для: Встраивания в приложения, мобильной разработки, проектов с ограниченными ресурсами.

По проведенному обзору можно выделить базу данных PostgreSQL, он обладает богатым функционалом, а также является простым в использовании. Но при этом PostgreSQL имеет проблемы при операциях чтения данных, которые нивелируются количеством стажировок на данный момент, количество записей в таблицах будет ограничено. Исходя из сильных и слабых сторон каждой рассмотренной базы данных, а также из моего личного опыта при использовании некоторых рассмотренных решений, я выбрал базу данных PostgreSQL.

2.2.2 Создание структуры и определение таблиц

База данных состоит из 9 таблиц:

- 1) User: На данной таблице будет храниться информация о пользователя - почта пользователя, хешированный пароль, login - пользователя
- 2) Skill: Таблица с технологиями, нужен для хранения различных технологий
- 3) Internship: Будет храниться информация о стажировках - название стажировки, дата начала, дата окончания отбора, а также будет привязан к компании .
- 4) SkillUser: Информация о профессиональных навыках пользователя
- 5) Test: По каждой стажировке будет выделен тест с названием.
- 6) SkillInternship: Таблица для хранения информации о стажировке и какие навыки будут нужны для конкретной стажировки
- 7) Question: Таблица для хранения вопросов в тестовом задании. Хранятся вопросы двух типов: вопрос по знанию технологии, задание с отправкой файла
- 8) UserTestResult: Таблица хранит информацию о прохождении тестового задания конкретным пользователем

9) UserTestQuestionResult: Информация о выполненном вопросе из тестового задания, где есть поля об отправленном ответе

10) Group: Информация о роли пользователя

Для наглядного отображения была построена даталогическая модель на рисунке 2.2

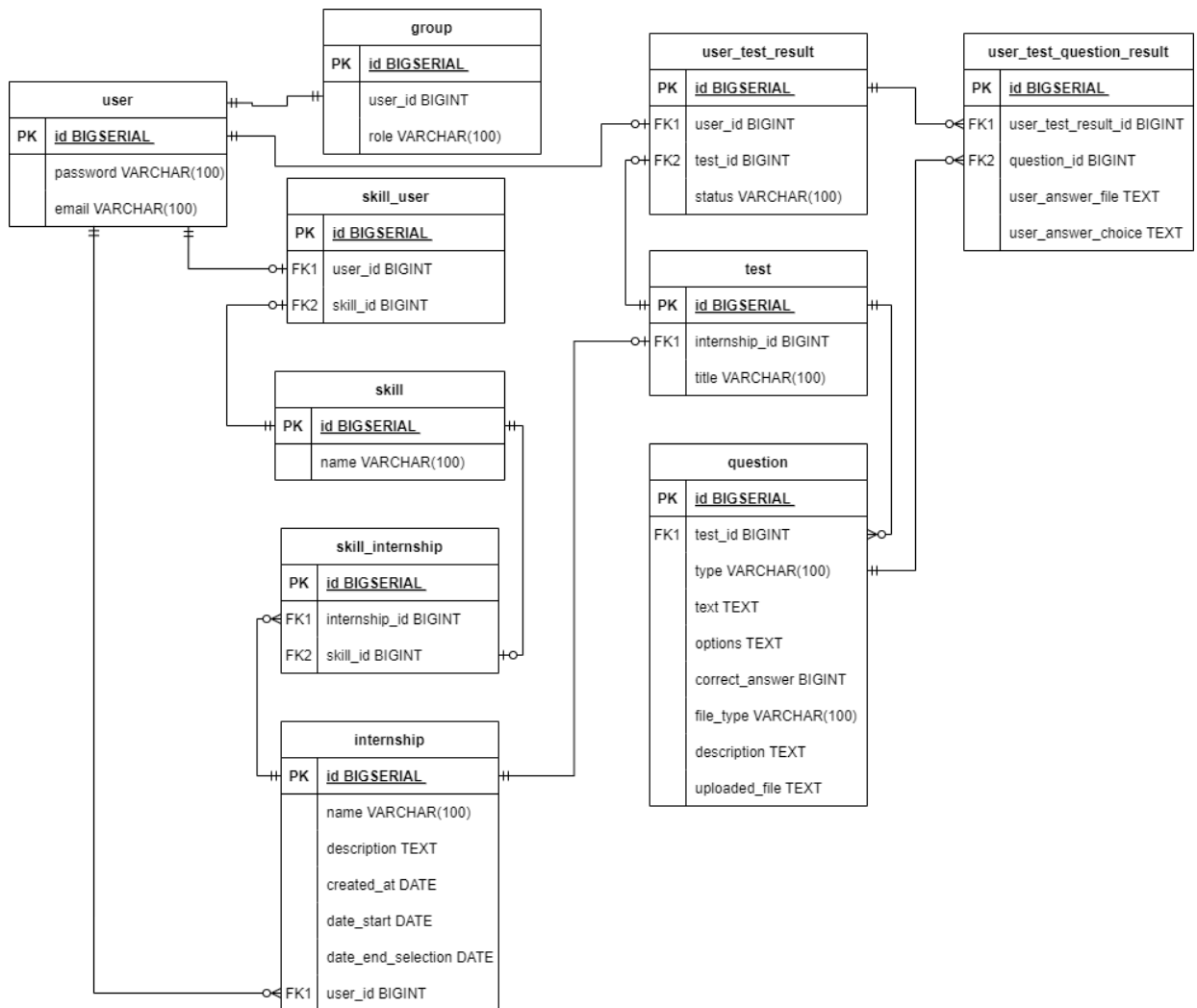


Рисунок 2.2 - даталогическая модель базы данных

Приложение должно хранить информацию о пользователях, которые будут разделены на роли: candidates и companies. Для этого будут созданы 2 таблицы для хранения этой информации: User, Group.

Так как приложение должно предоставлять пользователям сортировку стажировок на основе профессиональных навыков, то в базе данных должна быть таблица Skill, где будет поле - name, название технологии. Также

пользователь должен сохранять информацию о имеющихся у него навыках, для этого будет создана таблица SkillUser - которая реализует отношение многие-ко-многим с таблицами Skill и User.

Приложение должно предоставлять карточки стажировок, для этого будет создана таблица Internship для хранения стажировок. Поля таблицы:

- name - название стажировки
- description - дополнительная информация о стажировке
- created_at - дата создания карточки стажировки, нужна для сортировки
- date_start - дата начала стажировки
- date_end_selection - дата окончания отбора на стажировку
- user_id - поле привязки к компании (так как в таблице user будут данные кандидатов и компаний)

К каждой стажировке необходимо привязывать теги навыков для определения наиболее релевантных стажировок, поэтому будет создана таблица SkillInternship, где будут храниться необходимые навыки для каждой стажировки.

Приложение должно предоставлять возможность прохождения и создания тестовых заданий, поэтому необходима таблица test, которая будет привязана к каждой стажировке отношением один-к-одному.

Так как тесты будут включать в себя различные типы вопросов, то при создании таблицы Question должны быть учтены поля для хранения типа вопроса. В моем случае будет 2 типа вопросов: по знанию технологии, с отправкой файла. Поэтому таблица Question будет иметь следующие поля:

- test_id - привязка к тесту по идентификатору
- type - будет храниться информация о типе вопроса (“multiple_choice” - вопрос по знанию технологии, “file_upload” - задание с загрузкой файла)

- text - описание вопроса
- options - будет храниться информация для вопроса на знание, если был тип вопроса “multiple_choice”. Формат хранения данных: строка с 4 значениями.
- correct_answer - номер правильного ответа на вопрос о знании технологии. (от 1 до 4)
- file_type - тип отправленного файла с техническим заданием на тип вопроса “file_upload”
- description - описание технического задания
- uploaded_file - данные об адресе хранения прикрепленного файла

Также будут созданы две таблицы: UserTestResult и UserTestQuestionresult для хранения данных об ответе пользователя.

2.3 Выбор технологий для разработки серверной части

Серверная часть приложения - это сторона приложения которая отвечает за процессы и функционал системы. Пользователь не видит эту часть, а взаимодействует с ним путем отправления различных запросов с клиентской части.

2.3.1 Обзор и выбор архитектуры серверной части приложения

При проектировании серверной части приложения я буду отталкиваться от требований к системе. Исходя от требований, можно понять, что количество различных запросов к серверу будет ограничено. Рассмотрим архитектуры для проектирования системы:

1. Монолитная архитектура - весь код серверной части находится в одном приложении.
 - Преимущества: Простая для понимания и разработки. Легко развертывается.

- Недостатки: Сложность обслуживания и масштабирования. Изменения в одной части кода могут повлиять на работу всего приложения. Сложно внедрять новые технологии.
 - Пример: Небольшие веб-сайты, приложения с ограниченной функциональностью.
2. Микросервисная архитектура - приложение разбивается на независимые модули (микросервисы), взаимодействующие друг с другом.
- Преимущества: Легко масштабируется, каждый микросервис может быть развернут независимо. Повышенная отказоустойчивость, сбой одного сервиса не влияет на другие. Возможность использовать разные технологии для разных сервисов.
 - Недостатки: Сложность разработки и управления. Требует большого опыта. Более сложное тестирование.
 - Пример: Крупные приложения, системы с высокой нагрузкой, платформы электронной коммерции.
3. API-ориентированная архитектура (API-first) - серверная часть строится вокруг API, предоставляющего доступ к функциям и данным.
- Преимущества: Упрощает взаимодействие между различными сервисами и клиентами. Позволяет создавать экосистему вокруг приложения.
 - Недостатки: Требует тщательного проектирования API.
 - Пример: Публичные API, платформы для разработчиков, микросервисные приложения.

Исходя из рассмотренных архитектур серверной части приложения самым подходящим является монолитная, ввиду ограниченного количества разных запросов для отправки на сервер. Можно было разделить функционал

по микросервисам, но тогда для каждого сервиса должна быть выделена отдельная база данных и сформированы методы для общения между сервисами. В контексте моего приложения данная архитектура избыточна. Поэтому монолитная архитектура больше всего подойдет для проектирования серверной части.

2.3.2 Выбор языка программирования для реализации серверной части приложения

Для разработки серверной части приложения был выбран язык Python. Язык Python известен своей простотой и читаемостью кода. Его синтаксис интуитивно понятен, что позволяет снизить вероятность ошибок. Это особенно важно при разработке серверной части, где ошибки могут привести к серьезным проблемам.

Я выделил основные преимущества выбора языка Python:

- Python обладает богатым набором библиотек и фреймворков, которые облегчают разработку серверной части приложений. Данные инструменты предоставляют готовые решения для работы с базами данных, маршрутизации запросов, аутентификации.
- Скорость разработки - из-за количества инструментов, которые позволяют значительно ускорить процесс разработки, можно полностью сосредоточиться на разработке логики приложения, а не на решение низкоуровневых задач
- Документация - из-за популярности языка Python в просторе интернета хранится множество полезной документации по разработке, а также готовых решений для определенных задач.
- Масштабируемость - фреймворки Python поддерживают разработку масштабируемых приложений. Они позволяют легко добавлять новые функции и модули по мере роста приложения и его пользовательской базы.

2.3.3 Обзор и выбор фреймворка для создания приложения по выбранному языку программирования

Рассмотрим основные фреймворки языка Python для создания веб-приложений:

1. Django [12] - это высокоуровневый веб-фреймворк на Python. Данный фреймворк предоставляет всё необходимое для создания полноценных и безопасных веб-приложений прямо из коробки.

- Преимущества: предоставляет всё необходимое для создания полноценных веб-приложений (ORM, админская панель). Хорошая документация. MVC архитектура, которая способствует структурированию кода. Встроенные средства безопасности.
- Недостатки: Может быть слишком громоздким для небольших проектов.

Фреймворк подходит как для больших веб-приложений, так и для малых.

2. Flask [13] - это микро-фреймворк Python для веб-разработки, который предоставляет базовые инструменты для создания веб-приложений. Фреймворк не навязывают никакую структуру при создании веб-приложений.

- Преимущества: Гибкость и минимализм, дает больше контроля над структурой приложения. Легкий в изучении, хорошо подходит для небольших проектов. Большое количество расширений.
- Недостатки: Требуется больше ручной настройки, чем Django. Меньше встроенных функций, некоторые нужно реализовывать самостоятельно.

Фреймворк подходит для разработки простых веб-приложений, микросервисов.

3. Pyramid [14] - это мощный фреймворк для создания веб-приложений. Предоставляет гибкость и модульность, таким образом, предоставляет разработчику создавать приложения с различными архитектурами.

- Преимущества: Гибкость и модульность, позволяет выбирать нужные компоненты. Хорошая производительность. Подходит для проектов любого размера.
- Недостатки: Более сложный порог входа, чем у Django или Flask.

Фреймворк подходит для разработки проектов, требующих высокой гибкости и производительности.

4. FastAPI [15] - это современный фреймворк Python для создания API, который отличается своей скоростью, простотой и функциональностью.

- Преимущества: Высокая производительность, автоматическая генерация документации. Поддержка асинхронного программирования. Встроенная валидация данных.
- Недостатки: Относительно новый фреймворк, сообщество ещё развивается.

Фреймворк подходит для создания REST API, микросервисов, приложений, где особый упор идет на производительность.

Для разработки системы для поиска стажировок и прохождения тестовых задач фреймворк Django может предоставить готовый функционал для организации аутентификации пользователя. В контексте моего приложения пользователь одновременно может быть компанией и кандидатом. В Django встроено две таблицы User и Group, которая реализует данный момент. Это позволит значительно уменьшить время разработки. А также в Django встроена MVC архитектура. Исходя из данных преимуществ был выбран фреймворк Django.

2.4 Выбор технологий для разработки клиентской части

2.4.1 Обзор и выбор архитектурного подхода для разработки клиентской части приложения

Рассмотрим подходы к разработке клиентской части приложения:

1. Традиционная веб-разработка - страницы создаются с использованием чистого HTML [16], CSS [17], JavaScript [18]. При такой разработке запросы на серверную часть отправляются при каждом действии пользователя.

- Преимущества: Простота реализации, совместимость с браузерами.
- Недостатки: Низкая скорость взаимодействия, сложность в реализации динамических компонентов.

2. Одностраничные приложения - весь интерфейс подключается один раз, взаимодействия с пользователем происходят через JavaScript.

- Преимущества: Высокая скорость взаимодействия. Динамические интерфейсы. Подходит для приложений с большим количеством данных.
- Недостатки: Более сложная разработка. Требуется более мощный браузер.

3. Многостраничные приложения - приложение состоит из нескольких HTML-страниц, которые загружаются, исходя из действий пользователя.

- Преимущества: Простота реализации.
- Недостатки: Низкая скорость взаимодействия, не подходит для динамических приложений.

Для разработки клиентской части приложения для поиска стажировок с возможностью прохождения тестовых заданий мне нужна архитектура, где есть возможность загрузки различных страниц на основе действий пользователя. По предоставленным функциональным требованиям можно понять, что в клиентской части будет много динамического контента. Исходя из этого, я буду использовать многостраничное приложение.

2.4.2 Обзор и выбор языка программирования для клиентской части приложения

1. JavaScript:

- Преимущества: Фактический стандарт для веб-разработки. Широко распространен, большое сообщество, много ресурсов. Возможность создавать как простые, так и сложные интерфейсы. Поддерживает объектно-ориентированное, функциональное программирование. Много библиотек и фреймворков (React, Angular, Vue.js) для упрощения разработки.
- Недостатки: Может быть сложным для новичков. Не всегда легко отлаживать.

Язык подходит для любых веб-приложений.

2. TypeScript [19]:

- Преимущества: Статически типизированный язык, что упрощает отладку и делает код более надежным. Совместимость с JavaScript, можно использовать существующие библиотеки. Поддержка ООП, что делает код более структурированным. Большое количество инструментов для разработки и отладки.
- Недостатки: Может быть более сложным для новичков, чем JavaScript. Не все браузеры поддерживают TypeScript непосредственно, требуется компиляция в JavaScript.

Подходит для сложных веб-приложений, где важна надежность кода.

3. Dart [20]:

- Преимущества: Статически типизированный язык с поддержкой ООП. Создан Google и используется в Flutter Web. Высокая производительность и эффективность.
- Недостатки: Не так широко распространен, как JavaScript или TypeScript. Меньше библиотек и фреймворков по сравнению с JavaScript.

Подходит для разработки веб-приложений с помощью Flutter.

При выборе языка программирования можно отталкиваться от базы данных. Так как у меня реляционная модель базы данных, то стоит больше сделать упор на типизацию. JavaScript имеет множество полезных инструментов для разработки клиентской части, но уступает в типизации данных в сравнении с TypeScript. TypeScript также имеет множество библиотек, которые изначально были созданы для JavaScript. Исходя из этого был выбран язык TypeScript для создания клиентской части приложения.

2.4.3 Выбор фреймворка для клиентской части приложения

TypeScript — это язык программирования, который добавляет статическую типизацию к JavaScript. Это позволяет создавать более масштабируемые и устойчивые приложения. Для создания клиентской части приложения для поиска стажировок я выбрал фреймворк React. [21] Основными преимуществами данного фреймворка являются:

- Гибкость и настраиваемость - можно создавать различные интерактивные интерфейсы.

- Библиотеки - огромная база библиотек с полезными инструментами для облегчения разработки.
- Совместимость - совместим почти со всеми библиотеками для React JavaScript. Это позволяет использовать уже проверенные библиотеки.

2.5 Проектирование системы

Для визуализации архитектуры взаимодействия между фронтендом на React (TypeScript), бэкендом на Django и базой данных PostgreSQL, можно использовать диаграмму компонентов. Такая диаграмма позволяет показать основные компоненты системы и их взаимодействие. Включим следующие компоненты:

- Клиентское приложение: React (TypeScript) отправляет запросы к серверу и получает ответы для отображения данных.
- Веб-сервер: Nginx, обрабатывающий HTTP-запросы и передающий их к Django.
- Django сервер: обрабатывает бизнес-логику и взаимодействует с базой данных
- База данных: PostgreSQL, которая хранит данные приложения.

Рисунок диаграммы компонентов представлен на рисунке 2.3.

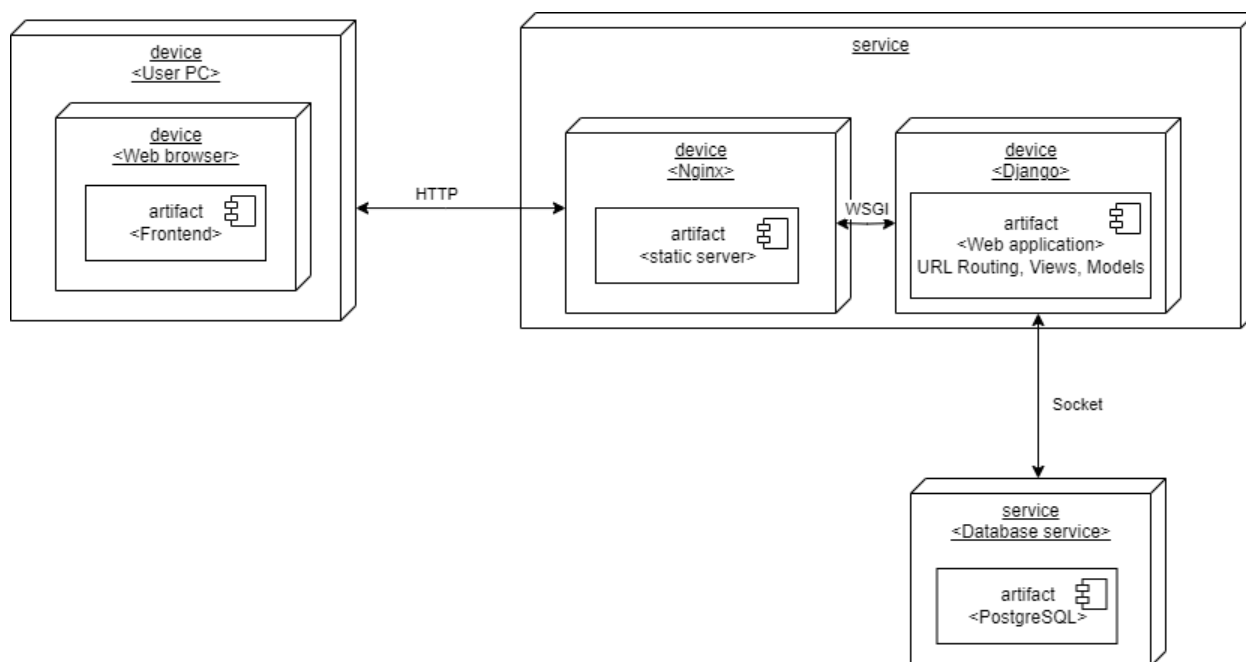


Рисунок 2.3 - диаграмма компонентов системы

Клиентское приложение (React TypeScript) - отправляет HTTP-запросы к серверу для получения или отправки данных. Также принимает HTTP-ответы и обновляет интерфейс пользователя.

Веб-сервер - принимает входящие HTTP-запросы и перенаправляет их к серверному приложению Django.

Серверное приложения Django - обрабатывает бизнес-логику приложения. Использует URL-routing для определения необходимых функций для обработки запросы. Views - обрабатывают запрос, взаимодействуют с моделями и формируют ответ. Models - представляют структуру данных и взаимодействуют с базой данных.

База данных PostgreSQL - хранит данные приложения, обрабатывает запросы от серверного приложения для чтения и записи данных.

2.6 Вывод по второй главе

В данной главе было проведено проектирование всего веб-приложения. Были описаны функциональные и нефункциональные требования к веб-приложению. Исходя из функциональных требований была создана

диаграмма вариантов использования, а также спроектирована структура приложения представленная в виде диаграммы компонентов. Для хранения данных была сформирована даталогическая модель базы данных.

3 Реализация приложения

3.1 Разработка модуля аутентификации

Для начала реализуем модуль аутентификации. Чтобы была возможность для регистрации как компании, так и пользователя, который ищет стажировки, мы будем использовать систему ролей. Для студента это роль - `candidates`, для компании - `companies`.

Для взаимодействия с базой данных Django предоставляет инструмент, который позволяет взаимодействовать с базой данных с помощью объектов Python вместо написания SQL-запросов напрямую. Вместо создания таблиц SQL я определяю структуру данных с помощью классов Python, называемых моделями. Каждый атрибут модели представляет собой поле в таблице базы данных. Django ORM автоматически генерирует SQL-код для создания таблиц в базе данных на основе моих моделей. Для модуля аутентификации были использованы готовые классы `User` и `Groups` из библиотеки `django.contrib.auth.models`.

Для обеспечения безопасности данных я использовал токен для аутентификации пользователя. Токен, в данном контексте - это строка символов, которая подтверждает личность пользователя и предоставляет доступ к ресурсам API без необходимости повторного ввода логина и пароля при каждом запросе. На стороне клиента хранится только токен, который действителен в течении некоторого времени, при истечении срока действия токена он обновляется при следующем входе. Токен генерируется на этапе регистрации пользователя.

Для отображения бизнес-процесса авторизации я приведу диаграмму последовательности на рисунке 3.1 в нотации UML.

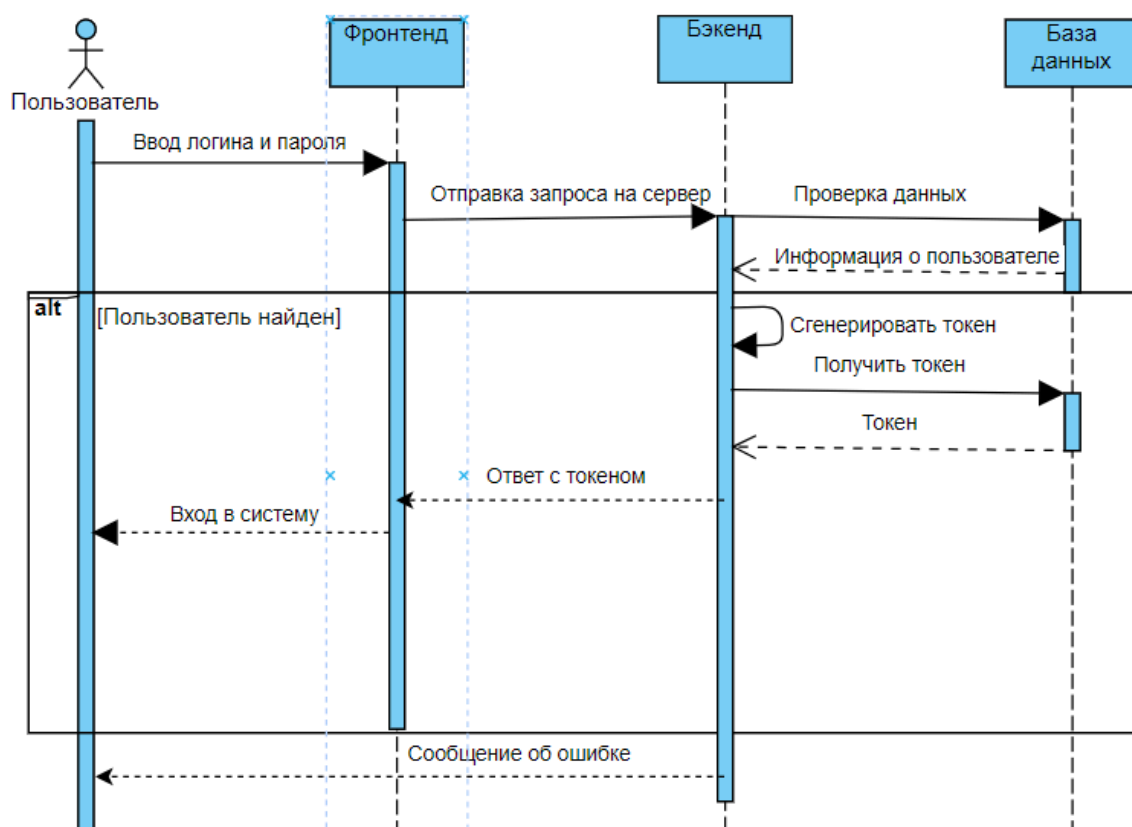
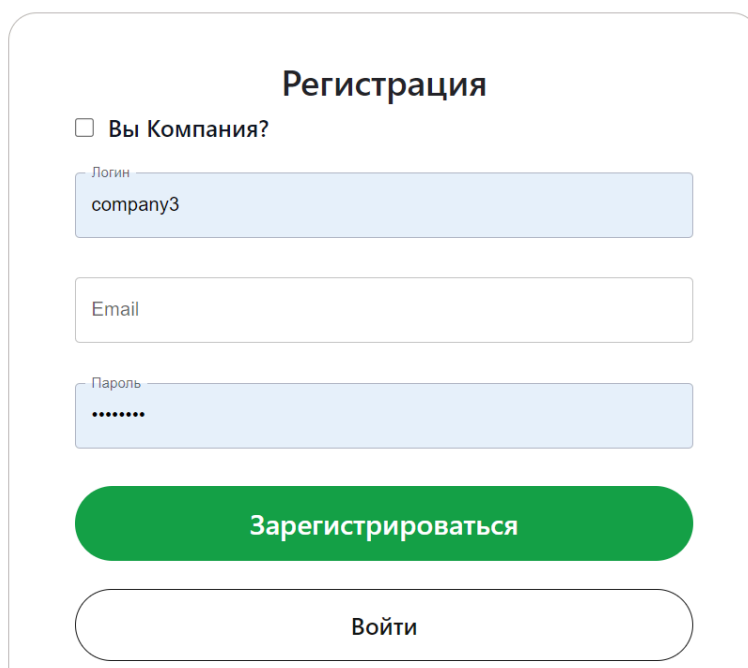


Рисунок 3.1 - Процесс авторизации пользователя

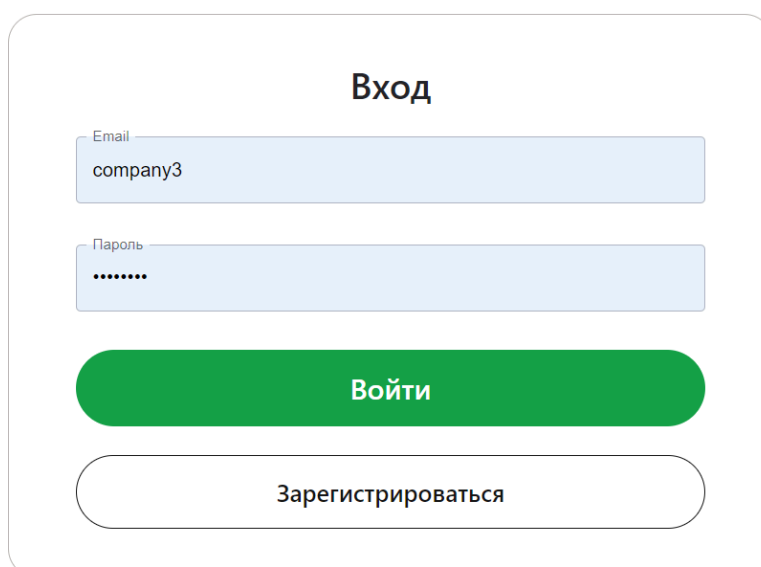
При успешной авторизации пользователя каждый последующий запрос отправляется с заголовком `Authorization: Token <Токен пользователя>`. При регистрации у пользователя реализован чекбокс, где пользователь выбирает роль. Если человек выбирает, что он “компания”, то запрос отправиться на endpoint для регистрации компании, иначе для регистрации кандидата. Интерфейс регистрации для с возможностью выбора роли пользователя приведен на рисунке 3.2.



The registration form is titled "Регистрация". It includes a checkbox labeled "Вы Компания?". Below this are three input fields: "Логин" (containing "company3"), "Email", and "Пароль" (masked with dots). At the bottom are two buttons: a green "Зарегистрироваться" button and a white "Войти" button.

Рисунок 3.2 - страница регистрации

При успешной регистрации пользователь переадресуется на страницу входа, рисунок 3.3. После заполнения полей и нажатии на кнопку “зарегистрироваться” отправляется запрос на авторизацию пользователя.



The login form is titled "Вход". It includes two input fields: "Email" (containing "company3") and "Пароль" (masked with dots). At the bottom are two buttons: a green "Войти" button and a white "Зарегистрироваться" button.

Рисунок 3.3 - Страница входа

Если авторизация прошла успешно, то пользователь будет переадресован на главную страницу. При последующем посещении веб-сайта будет проведена проверка на актуальность токена аутентификации. Если время жизни токена не истекло, то пользователь сразу будет переадресован на главную страницу, иначе пользователь должен пройти повторную авторизацию.

Система ролей нужна для отображения нужных компонентов на клиентской стороне. Для компаний был реализован функционал для создания тестов и карточек стажировок, а также для мониторинга прохождения тестовых заданий. Для кандидатов был реализован функционал для поиска релевантных стажировок по навыкам и прохождения тестовых заданий с мониторингом статуса выполнения.

3.2 Разработка модуля поиска и создания стажировок.

После успешной авторизации пользователя мы переходим на главную страницу приложения. По описанным функциональным требованиям система должна обладать функционалом для поиска, а также фильтрацией стажировок по навыкам пользователя. Поэтому реализуем для начала данный функционал.

Для начала создадим модели Skill и SkillUser. В модели Skill хранится поле с названием навыка, а в модели SkillUser реализовано отношение многие-ко-многим с таблицами User и Skill. Данная связь позволяет одному пользователю обладать несколькими навыками и одному навыку быть привязанным к нескольким пользователям.

Навыками в данном случае являются популярные технологии и языки программирования. Чтобы заполнить таблицу Skill языками программирования я взял 50 самых популярных языков программирования,

исходя из рейтинга TIOBE [22]. TIOBE формирует свой рейтинг на основе количества запросов с “выбранный язык программирования” на различных платформах. Я вручную заполнил таблицу Skill в админской панели от Django. Теперь необходимо заполнить таблицу популярными библиотеками и фреймворками, которые используют в настоящее время. Для выбора веб-фреймворков и технологий я исходил из статистик на сайте [statista.com](https://www.statista.com) [23].

После заполнения данными таблицы Skill, я перехожу на разработку функционала для выбора навыков пользователя.

Изначально при регистрации у пользователя не будет выбрано никаких навыков. При открытии страницы профиля проводится проверка роли пользователя, если роль пользователя - кандидат, то отображается форма для выбора профессиональных навыков. В серверной части у функции для обработки запросов пользователя будут декораторы от Django Rest Framework. Декораторы позволяют "обернуть" одну функцию другой, добавляя новый функционал без изменения ее кода. В моем случае я использую 3 декоратора:

- `@api_view` - позволяет выполнять запросы только определенного метода HTTP. (например: GET-запрос, POST-запрос)
- `@authentication_classes` - нужен для указания механизмов аутентификации, которые должны применяться к определенному представлению или набору представлений.
- `@permission_classes` - отвечает за управление доступом к представлениям на основе прав доступа или ролей пользователя. (мне необходима только проверить авторизован ли пользователь на основе заголовка Authentication в HTTP-запросе.

Для обработки запроса на получение списка навыков используем функцию `get_skills`. Она создает список навыков из таблицы Skill, а затем отправляет JSON - файл клиенту. Функция приведена на рисунке 3.4.

```

@api_view(['POST'])
@authentication_classes([SessionAuthentication, TokenAuthentication])
@permission_classes([IsAuthenticated])
def get_skills(request):
    skills = Skill.objects.all()
    skills_json = [{'name': skill.name} for skill in skills]
    return Response({'skills': skills_json})

```

Рисунок 3.4

Теперь на клиентской части сделаем страницу профиля пользователя с формой для редактирования навыков. После выполнения запроса на получение списка навыков мы должны это корректно отобразить в форме. Так как количество навыков большое, то мы будем группировать навыки и отображать их отдельно в dropdown. Реализация страницы приведена на рисунке 3.5:

The screenshot shows the 'Профиль' (Profile) page of the InternShift application. The header includes the InternShift logo and navigation links: Стажировки, Компании, Тесты, Тренды, and Профиль. A 'Выйти' (Logout) button is in the top right. The main content area is divided into two sections: 'Данные пользователя:' (User Data) and 'Навыки' (Skills). The 'Данные пользователя:' section shows 'Логин: user1' and 'Почта: user@mail.ru'. The 'Навыки' section features a dropdown menu labeled 'ВЫБРАТЬ НАВЫКИ' (Select Skills). The dropdown is open, displaying a list of skills: Solidity, Logo, PL/SQL, Forth, Awk, CFML, Bash, Transact-SQL, LabVIEW, and TypeScript.

Рисунок 3.5 - Страница профиля пользователя

При нажатии на кнопку редактировать навыки мы попадаем на модальное окно с возможностью выбрать профессиональные навыки. Модальное окно с открытым dropdown приведена на рисунке 3.6.

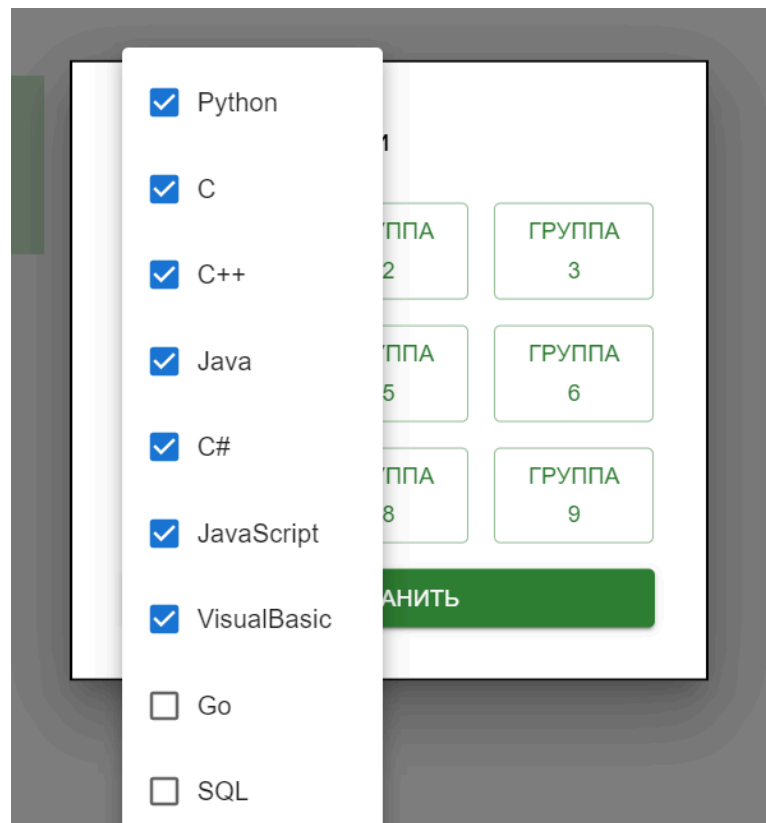


Рисунок 3.6 - Модальное окно с выбором навыков

В конце модального окна я реализовал кнопку “Сохранить”, при нажатии которого на сервер отправляется запрос об изменении таблицы SkillUser. Я показываю в чекбоксах уже выбранными те навыки, которые были изначально выбраны пользователем. При первой регистрации все чекбоксы невыбранные. То есть записей в таблице SkillUser по пользователю отсутствуют. После редактирования навыков, а также при повторном открытии модального окна записи подтягиваются с сервера и уже некоторые чекбоксы становятся закрашенными. Я реализовал функцию на клиентской части, которая на вход принимает все нажатые пользователем навыки, а затем отправляем запрос на изменение на сервер. Пользователь также может оставить чекбокс пустым, тогда те навыки, которые были изначально привязаны к пользователю будут удалены. Функция приведена на рисунке 3.7

```

@api_view(['POST'])
@authentication_classes([SessionAuthentication, TokenAuthentication])
@permission_classes([IsAuthenticated])
def save_skills(request):
    data = json.loads(request.body)
    submitted_skills = set(data.get('skills', []))
    user_skills = SkillUser.objects.filter(user=request.user)
    existing_skills = set(user_skill.skill.name for user_skill in user_skills)
    skills_to_delete = existing_skills - submitted_skills
    user_skills.filter(skill__name__in=skills_to_delete).delete()
    skills_to_add = submitted_skills - existing_skills
    new_skill_users = []
    for skill_name in skills_to_add:
        skill = Skill.objects.get_or_create(name=skill_name)[0]
        new_skill_users.append(SkillUser(user=request.user, skill=skill))
    SkillUser.objects.bulk_create(new_skill_users)

    return Response({'message': 'Навыки успешно обновлены пользователю'})

```

Рисунок 3.7 - функция для изменения записей SkillUser

Я получаю список навыков от фронтенд стороны, затем ищу совпадения по идентификатору пользователя. Если в списке отсутствуют навыки, которые есть в таблице SkillUser, то я их удаляю. Те навыки, которые отсутствуют в таблице SkillUser, но присутствуют в списке я добавляю. Таким образом происходит процесс обновления навыков пользователя.

Для отображения навыков только для пользователей с ролью “Candidates” используем тернарный оператор, который проверяет привязан ли пользователей к данной роли. Запрос совершается при переадресации пользователя на страницу профиля. С сервера я прошу информацию о привязанных ролях. Если это компания, то навыки не будут отображаться на странице пользователя. Тернарный оператор для вывода функционала приведен на рисунке 3.8.

```

{userInfo?.group == "companies" ? <div></div> :
  <div>
    <div className="mt-12 text-4xl font-semibold leading-10 text-center">
      Навыки
    </div>
    <div>
      <Button onClick={handleOpen}>Редактировать навыки</Button>
      <Modal

```

Рисунок 3.8 - условие для отображения функционала навыков

Теперь реализуем страницу стажировок. На этой странице выводится список стажировок с тегами. Теги, в данном контексте - это навыки привязанные к стажировке. По ним производится фильтрация карточек стажировок. Если пользователь не выбрал навыки в профиле, то первыми показываются последние созданные стажировки. Если у пользователя выбраны определенные навыки, то сортировка проводится на основе количества совпадений по привязанным навыкам и первыми отображаются стажировки с наименьшим количеством тегов. Функционал по привязке навыков я сохранил с прошлой функции, то есть к каждой стажировке реализована форма для заполнения тегов. Данная форма идентична форме для заполнения навыков пользователя.

Для кандидатов виден список доступных релевантных стажировок, а для компаний дополнительно реализована форма для создания карточек стажировок. Также для компаний есть фильтр для отображения только собственных созданных стажировок. При нажатии на карточку стажировки, пользователь переходит на страницу с дополнительной информацией. На данной странице реализована функция для создания тестового задания привязанному к стажировке, а также кнопка для пользователя с возможностью проходить тестовое задание.

Начинаем с создания таблиц Internship и SkillInternship. Internship - таблица которая хранит информацию о стажировке: дата отбора, краткая информация, компания, которая проводит стажировку. Так как при регистрации был выбор между компанией и кандидатом, то компаниями, в данном контексте, служат пользователи с ролью “companies”. SkillInternship реализует отношение многие-ко-многим между таблицей Skill и Internship и хранит теги навыков, по которым проводится сортировка. Я разработал функцию, которая на вход принимает JSON-файл с информацией о

стажировке и навыками, затем создает новые записи в соответствующих таблицах.

Была также разработана форма имеющая поля для заполнения данных о стажировке и с тегами навыков. Теги можно настроить в дополнительном модальном окне. Форма создания стажировки приведена на рисунке 3.9.

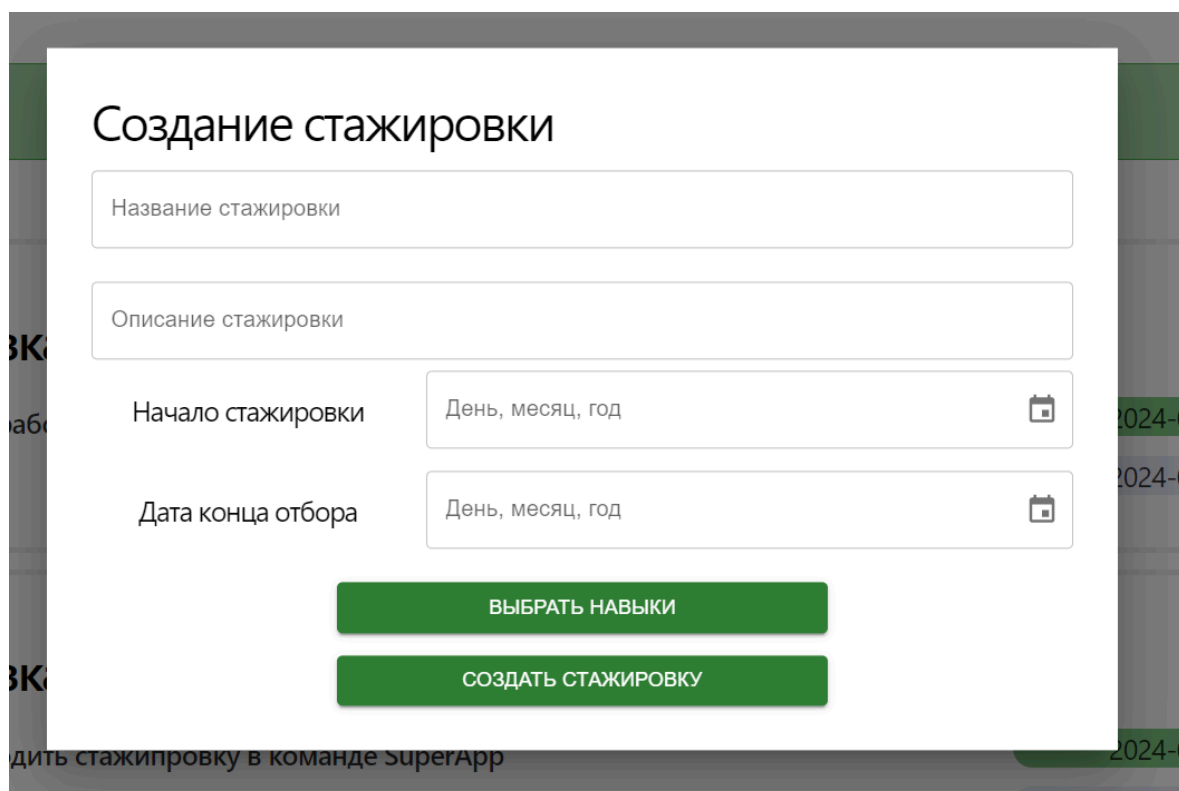


Рисунок 3.9 - форма создания стажировки

3.3 Разработка модуля сортировки стажировок с учетом навыков пользователя

Отображение стажировок должна быть эффективной и корректно отображать самые подходящие по профессиональным навыкам. На данный момент действует следующий алгоритм для отображения списка стажировок:

- Получаем список стажировок из базы данных
- Получаем список навыков пользователя, который отправил запрос

- Получаем список навыков привязанных к стажировкам
- Сравниваем совпадения по навыкам стажировки и пользователя
- Сортируем их по количеству совпадений

Можно увидеть, что я три раза обращаюсь к базе данных и алгоритмическая сложность при двойном цикле у меня $O(n^2)$. В агрегаторе вакансий hh.ru на данный момент 7400 стажировок, которые можно получить по фильтру “стажировка”. Я отталкиваюсь от этого количества стажировок, так как в остальных агрегаторах он гораздо меньше. Проведем исследование скорости выполнения запроса и отображения карточек на веб-сайте, учитывая разные количества тегов привязанные к стажировкам. Для заполнения данными базы данных я разработал скрипт, который отправляет запросы на создание стажировок к серверу с токеном аутентификации компании. То есть в данном случае я использую для создания стажировок только одну компанию.

Код скрипта приведен на рисунке 3.10.

```
for i in range(100):
    selected_skills = random.sample(skills, 5)
    url = "http://127.0.0.1:8000/auth/add_internship"
    headers = {
        "Authorization": "Token 508a36d745e29ef9dce7c60eaeb5dd2cb2a388c8",
        "Content-Type": "application/json"
    }

    data = {
        "name": "Стажировка" + str(i),
        "description": "Тест количества тегов",
        "date_start": "2024-05-29",
        "date_end_selection": "2024-05-29",
        "skills": selected_skills
    }
    json_data = json.dumps(data)
    response = send_request(url, headers, json_data)
    if response.status_code == 200:
        print("Запрос успешен!")
    else:
        print(f"Ошибка: {response.status_code}")
```

Рисунок 3.10

В данном примере показан скрипт для создания 100 стажировок, сначала привяжем к ним 5 случайных навыков. Для более реалистичной симуляции работы приложения и получения релевантных показателей, добавим в него информацию еще о 7300 стажировках. Сначала возьмем показатели времени отображения страницы стажировок с hh.ru, так как симуляцию я сравнивал с данной платформой. Я составил таблицу 2 для отображения и сравнения нынешнего состояния отображения страницы запросов. Далее данные по платформе hh.ru использованы повторно. Данные из hh.ru были взяты из браузера Google Chrome во вкладке Network, по времени отображении карточек стажировок.

Таблица 2 - время отображения страницы стажировок, первый алгоритм фильтрации

Время отображения hh.ru, с	Время отображения страниц, с	Время обработки в бэкенде, с
40	66	44,4
34,44	47,52	25,89
40,2	51,48	29,64
41,45	49,3	28,03
43,04	56,09	35,65

Как можно заметить, время обработки запроса занимает очень много времени, а также время отрисовки компонентов занимает в среднем примерно 25 секунд. Теперь приступим к оптимизации производительности приложения. Во-первых, при отображении компонента страницы стажировок я заметил, что некоторые запросы отправляются дважды на сервер. После проведения изменений на клиентской стороне я сделал замеры времени отображения еще раз.

Таблица 3 - время отображения страницы стажировок, после оптимизации клиентской части приложения

Время отображения hh.ru, с	Время отображения страниц, с	Время обработки в бэкенде, с
40	27	25,18
34,44	30,06	28,45
40,2	24,13	20,63
41,45	25,67	22,34
43,04	27,08	24,35

По таблице можно заметить большое улучшение при отображении карточек стажировок после проведенных изменений. Теперь перейдем к оптимизации алгоритма для отображения стажировок. На данный момент у меня такой алгоритм:

- Запрашиваем все стажировки из базы данных.
- Инициализируем пустой список для хранения данных о стажировках.
- Для каждой стажировки:
 - Инициализируем пустой список для хранения навыков.
 - Запрашиваем все связи "навык-стажировка" для текущей стажировки.
 - Для каждой связи добавляем название навыка в список навыков.
 - Добавляем данные о стажировке в список стажировок.
- Запрашиваем все навыки пользователя из базы данных.
- Инициализируем пустой список для хранения навыков пользователя.
- Для каждого навыка пользователя добавляем название навыка в список.
- Сортируем список стажировок по количеству совпадающих навыков с навыками пользователя.
- Возвращаем отсортированный список стажировок в ответе.

Основным моментом потери времени при обработке запроса является момент, когда мы ищем навыки привязанные к каждой стажировке. Для оптимизации запросов к базе данных можно использовать функцию `prefetch_related` от фреймворка Django, которая позволяет одним запросом

получить связанные данные, таким образом избегая повторного запроса к базе данных. Скрипт для запроса к базе данных, Листинг 1.

```
internships = Internship.objects.prefetch_related(
    Prefetch('skillinternship_set',
    queryset=SkillInternship.objects.select_related('skill')))
```

Листинг 1 - запрос к базе данных для получения стажировок

При выполнении данного запроса мы сразу получаем поле 'skillinternship_set', где хранится массив навыков привязанных к определенной стажировке. Также можем оптимизировать процесс получения навыков пользователя и сортировки. Для этого я буду одним запросом получать множество навыков пользователя, затем при сортировке нужно сравнивать множество навыков стажировки и множество навыков пользователя. В данном случае я использую пересечение двух множеств. Код приведен на Листинге 2.

```
internships_json = [
    {
        'id': internship.id,
        'name': internship.name,
        'description': internship.description,
        'date_start': internship.date_start,
        'date_end_selection': internship.date_end_selection,
        'skills': [si.skill.name for si in
internship.skillinternship_set.all()]
    }
    for internship in internships
]
user_skills=
set(SkillUser.objects.filter(user=request.user).values_list('skill__name',
flat=True))
sorted_internships = sorted(
    internships_json,
    key=lambda internship: -len(set(internship['skills']) & user_skills)
)
```

Листинг 2 - функция для сортировки стажировок по навыкам пользователя

Сейчас получаем данный алгоритм для составления списка стажировок на основе навыков пользователя:

- Запрашиваем все стажировки из базы данных и предзагружаем связанные навыки одним запросом, используя `prefetch_related`.
- Инициализируем пустой список для хранения данных о стажировках.
- Для каждой стажировки:
 - Собираем все связанные навыки для текущей стажировки.
 - Добавляем данные о стажировке в список стажировок.
- Запрашиваем все навыки пользователя из базы данных и конвертируем их в множество.
- Сортируем список стажировок по количеству совпадающих навыков с навыками пользователя.
- Возвращаем отсортированный список стажировок в ответе.

Также немного оптимизируем процесс отображения карточек стажировок на клиентской стороне. На данный момент я отображаю все полученные 7400 стажировок пользователю, поэтому время рендеринга всех компонентов получается примерно 15 секунд, что тоже замедляет время отображения стажировок. Для оптимизации я использовал компонент `Pagination` из библиотеки `Material Design`, который позволяет организовать постраничный вывод стажировок.

После всех исправлений и оптимизаций я сделал замеры времени обработки запроса на серверной части и отображения карточек стажировок на клиентской стороне. Результаты приведены на таблице 4.

Таблица 4 - время отображения карточек стажировок после второй оптимизации

Время отображения hh.ru, с	Время отображения страниц, с	Время обработки в бэкенде, с
40	3,4	1,23

34,44	2,95	1,53
40,2	2,47	1,32
41,45	2,28	1,36
43,04	2,89	1,35

По результатам оптимизации функций для отображения стажировок мы можем увидеть значительное уменьшение времени обработки запроса на сервере. Такая разница времени отображения относительно платформы hh.ru обуславливается тем, что мое приложение полностью ориентировано только на стажировки. Поэтому количество записей в базе данных у меня будет всегда малым, а hh.ru при обработке запроса обращается к базе данных, где хранится более миллиона записей. Соответственно время обработки запроса увеличивается.

Но в результате я добился более эффективного процесса поиска стажировок. Сейчас нужно сделать исследование зависимости времени отображения от количества тегов привязанных к стажировке, потому что все замеры были сделаны при 5 тегах для каждой стажировки. Теперь сделаем исследование при 10, 15 тегах и рассмотрим результаты отображения стажировок, таблицы 5-6.

Таблица 5 - время отображения стажировок при 10 тегах

Время отображения hh.ru, с	Время отображения страниц, с	Время обработки в бэкенде, с
40	3,3	1,48
34,44	2,9	1,54
40,2	2,3	1,47
41,45	2,43	1,46
43,04	2,34	1,42

Таблица 6 - время отображения стажировок при 15 тегах

Время отображения hh.ru, с	Время отображения страниц, с	Время обработки в бэкенде, с
-------------------------------	---------------------------------	---------------------------------

40	3,4	1,48
34,44	2,8	1,47
40,2	2,5	1,57
41,45	2,47	1,51
43,04	2,51	1,60

Построим диаграмму среднего времени выполнения запроса на разных этапах изменения кода, рисунок 3.11.

Сравнение среднего времени выполнения запроса

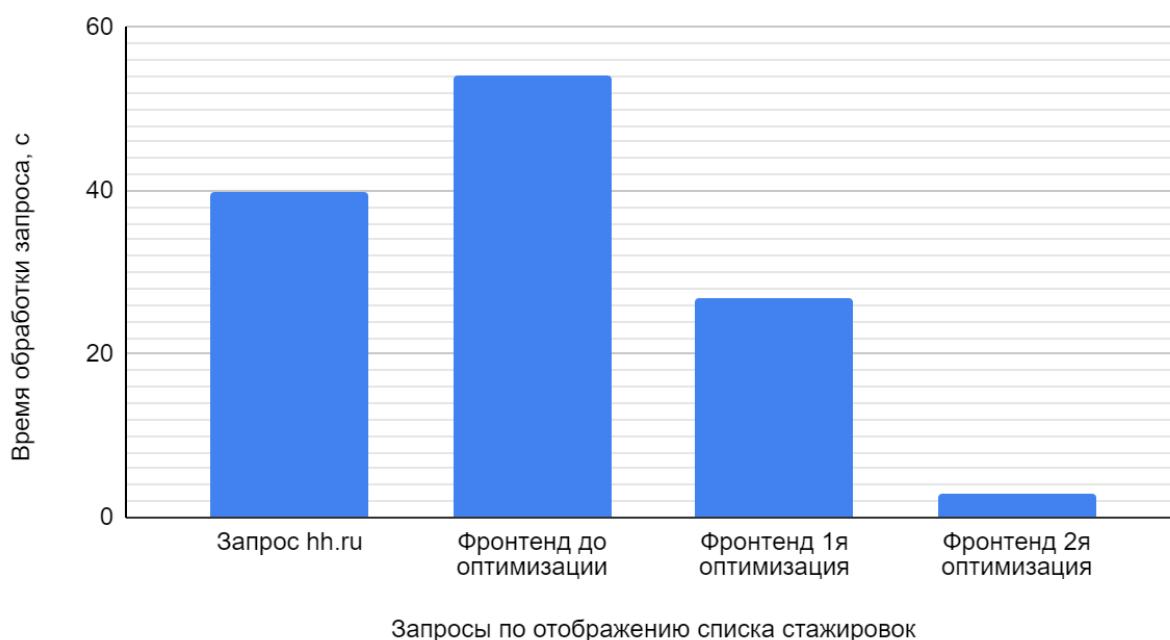


Рисунок 3.11 - диаграмма сравнения времени выполнения запроса

Исходя из диаграммы мы можем увидеть уменьшение времени выполнения запроса на вывод стажировок подходящих данному пользователю. Это доказывает эффективность поиска стажировок.

Также построим диаграмму среднего времени обработки запроса при различных количествах тегов. Рисунок 3.12.

Анализ изменения времени обработки от количества привязанных навыков

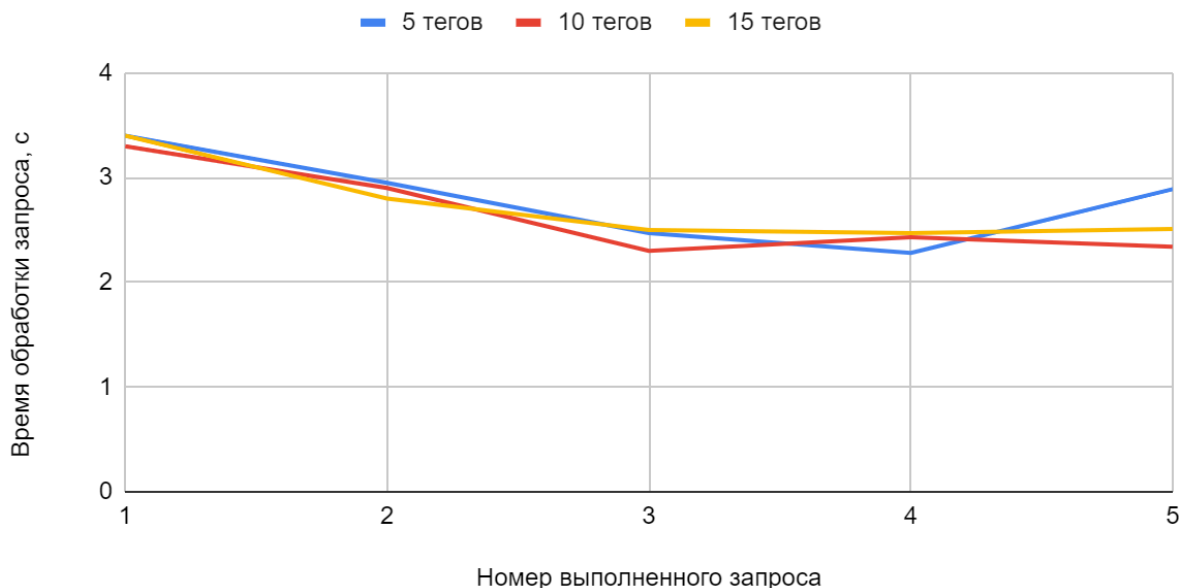


Рисунок 3.12 - анализ изменения времени обработки запроса

По графику можно увидеть, что количество тегов особо не влияет на скорость обработки запроса на список релевантных стажировок. Можно сделать вывод, что система отображает страницу с релевантными стажировками эффективно вне зависимости от количества привязанных навыков к стажировкам.

Также на странице со стажировками был реализован поиск по названию стажировки.

3.4 Разработка модуля создания и прохождения тестовых заданий

После реализации эффективного поиска релевантных стажировок. Я могу переходить к разработке системы для проведения тестовых заданий. Конструктор тестов предлагает добавлять вопросы двух типов: добавление задач на знание технологий, добавление задания с прикреплением технического задания. В техническом задании есть поле с описанием, а также

реализована возможность загружать собственные файлы формата .pdf или .docx. Кандидат ответом на техническое задание, также может прикреплять свои решения файлом формата .docx или .pdf.

Реализуем конструктор теста на клиентской стороне. Страница конструктора теста приведена на рисунке 3.13.

IS InternShift Стажировки Компании Тесты Профиль Выйти

Конструктор теста

Название теста

ДОБАВИТЬ ВОПРОС

123123
multiple-choice

Тест 1
file-upload

ОТПРАВИТЬ ТЕСТ

Рисунок 3.13 - страница конструктора теста

Рассмотрим функционал данного конструктора:

- Можно настраивать название теста
- Можно добавлять новый вопрос, а также редактировать и удалять после создания
- Можно отправить отредактированный тест.

Была создана возможность выбирать тип вопроса: выбор из 4, техническое задание, где можно прикреплять свой файл. Интерфейс двух вариантов вопросов приведены на рисунках 3.14 - 3.15.

Новый вопрос

Текст вопроса
Вопрос с 4 вариантами

Тип вопроса
Тест с 4 вариантами ответа ▼

Варианты ответов:

Вариант 1
Что будет 1

Вариант 2
Что будет 2

Вариант 3
Что будет 3

Рисунок 3.14 - вопрос выбор из 4 вариантов

Новый вопрос

Текст вопроса
Техн. задание

Тип вопроса
Задание с отправкой файла ▼

Тип файла
.pdf ▼

Выберите файл Innokent...101 (1).pdf

Выбранный файл: Innokentev_P34101 (1).pdf

Описание задания
Прочитайте и отправьте

ДОБАВИТЬ ВОПРОС В ТЕСТ

Рисунок 3.15 - вопрос с загрузкой файла

После создания, тест отправляется на сервер и обрабатывается. Также была создана функция, для изначальной подгрузки уже подготовленных вопросов. Данные отправлялись на сервер в формате FormData вместо json, так как файл большого размера сложно отправлять в таком формате. После отправки на сервер файл сохраняется в локальной директории. В таком виде хранятся файлы, рисунок 3.16:

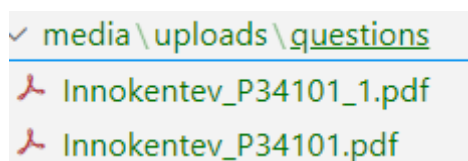


Рисунок 3.16 - хранение файлов локально

При нажатии на карточку стажировки пользователь автоматически переходит на страницу с информацией о стажировке. После он может нажать на кнопку для прохождения тестового задания, рисунок 3.17

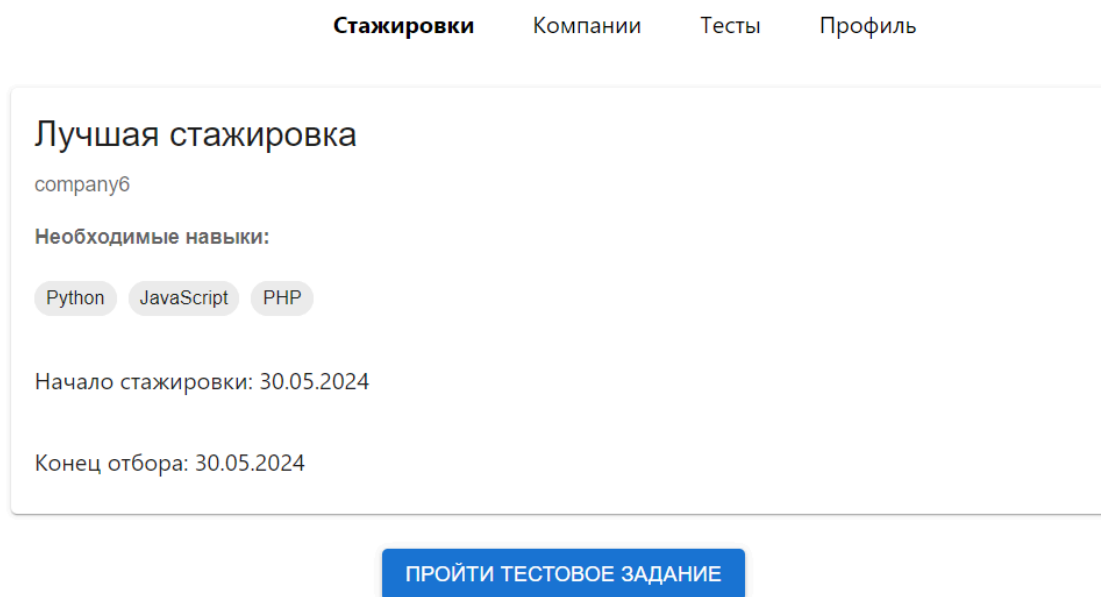


Рисунок 3.17 - страница с информацией о стажировке

После нажатия на кнопку “пройти тестовое задание” пользователь переходит на страницу с тестом, Рисунок 3.18.

Тестирование кандидата

Вопрос 1

test5

Выбери ответ:

☐ 1

☐ 2

☐ 3

☐ 4

Вопрос 2

Найди

Выбери файл:

Файл не выбран

Файл задания:

СКАЧАТЬ ЗАДАНИЕ

Рисунок 3.18 - страница с прохождением теста

После выполнения тестирования пользователь отправляет свое решение. Также после прохождения компания может отмечать сделан ли тест, либо нет, то есть провести проверку. Это находится на вкладке редактирования теста, снизу появляются карточки с решениями, направленными пользователями. На тестовые вопросы с выбором реализованы два поля: ответ пользователя, правильное решение. Проверяющий может проверить все вопросы и вынести вердикт по прохождению тестированию. Также внизу карточки для проверки есть компонент “select”, где проверяющий выбирает какой статус выполнения проверки поставить кандидату: В обработке, Не принято, Принято. После выбора статуса на сервер идет запрос об обновлении. Карточка для проверки приведена на рисунке 3.19.

Результаты тестов кандидатов

Логин пользователя: company5
Почта пользователя: company5@mail.ru

test5
Ответ пользователя: 1
Правильный ответ: 2

Найди

СКАЧАТЬ ФАЙЛ ОТВЕТА

test5

СКАЧАТЬ ФАЙЛ ОТВЕТА

Пояс1

СКАЧАТЬ ФАЙЛ ОТВЕТА

Статус

Не принято ▼

Рисунок 3.19 - компонент для проверки теста пользователя

После отправки своего решения на проверку, у пользователя появляется карточка со статусом проверки теста. Он отображается на вкладке тесты, рисунок 3.20

InternShift

СтажировкиКомпанииТестыПрофиль

Выйти

Тестирование кандидата
Статус: Принято

Рисунок 3.20 - страница с пройденными тестами пользователя.

Также была разработана страница с компаниями. Так как компании, в контексте приложения, это пользователи с ролью “companies”, то на странице отображается список пользователей. Все элементы списка являются кнопками, при нажатии которых пользователь переадресуется на страницу выбранной компании. Страница компании приведена на рисунке 3.21.

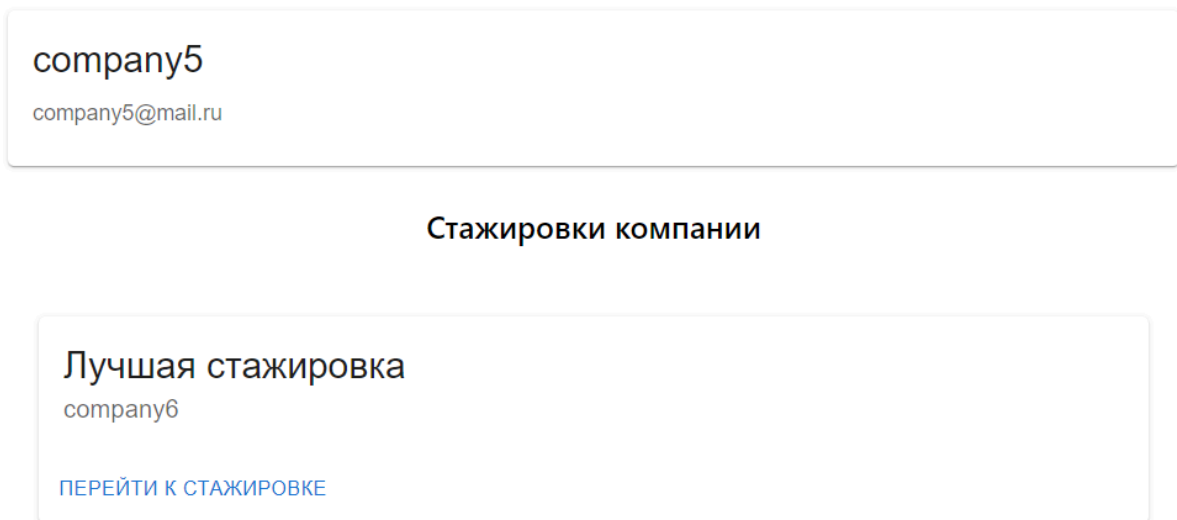


Рисунок 3.21

На странице компании отображена основная информация о компании, а также список созданных компанией стажировок. При нажатии на кнопку “Перейти к стажировке” пользователь будет переадресован на страницу с выбранной стажировкой.

3.5 Нагрузочное тестирование приложения

Для проверки стабильной работы веб-приложения при нагрузке системы было проведено нагрузочное тестирование. Нагрузочное тестирование (load testing) — это вид тестирования производительности, который используется для проверки, как система работает под определенной нагрузкой. Для работы была использована библиотека Locust. Она предоставляет функционал для проведения тестирования.

Основным запросом для проверки стал запрос на получение списка отсортированных стажировок, так как он самый требовательный в системе. При первом тестировании я получил следующий результат, рисунок 3.22.

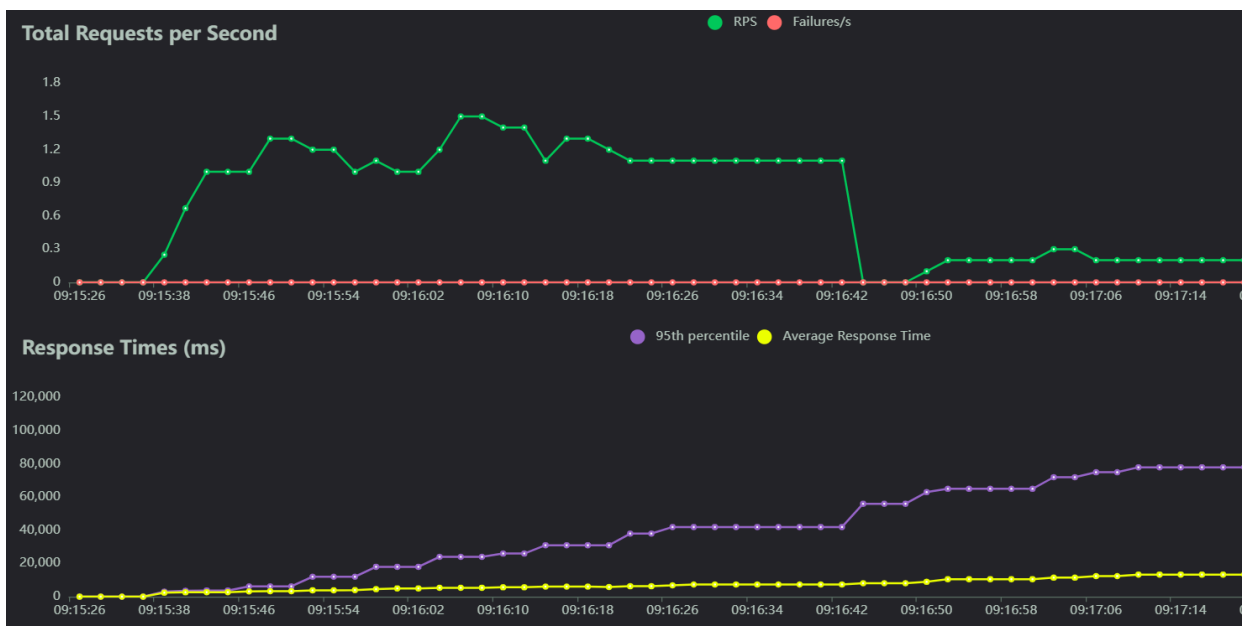


Рисунок 3.22 - первое нагрузочное тестирование

Тестирование было проведено на 100 одновременных пользователей. Для каждого пользователя выполнялся запрос на вход и запрос на получение списка отсортированных стажировок. По второму графику на рисунке 3.22 можно увидеть увеличение времени обработки запроса с увеличением количества одновременно отправленных запросов. Так как система должна обеспечивать стабильную работу при высоких нагрузках, то можно сделать вывод, что система имеет проблему при обработке запроса на получение отсортированного списка стажировок.

Для решения этой проблемы используем кэширование повторяющихся запросов. Так как при переходе на страницу стажировок каждый раз отправляется запрос на получение списка стажировок, то количество одинаковых запросов отправляемых серверу увеличивается, что замедляет процесс обработки запроса. Кэширование - это процесс сохранения результатов выполнения запроса, чтобы повторно использовать их при последующих запросах. Цель этого процесса — уменьшить время отклика и нагрузку на систему при многократном выполнении однотипных запросов.

Для кэширования я использовал библиотеку “cache” из фреймворка Django. Она помогает сохранять результаты выполнения запросов на определенное количество времени. В данном случае, результаты запросов сохранялись на 15 минут.

При повторном нагрузочном тестировании с использованием кэширования повторяющихся запросов был получен следующий результат, рисунок 3.23.

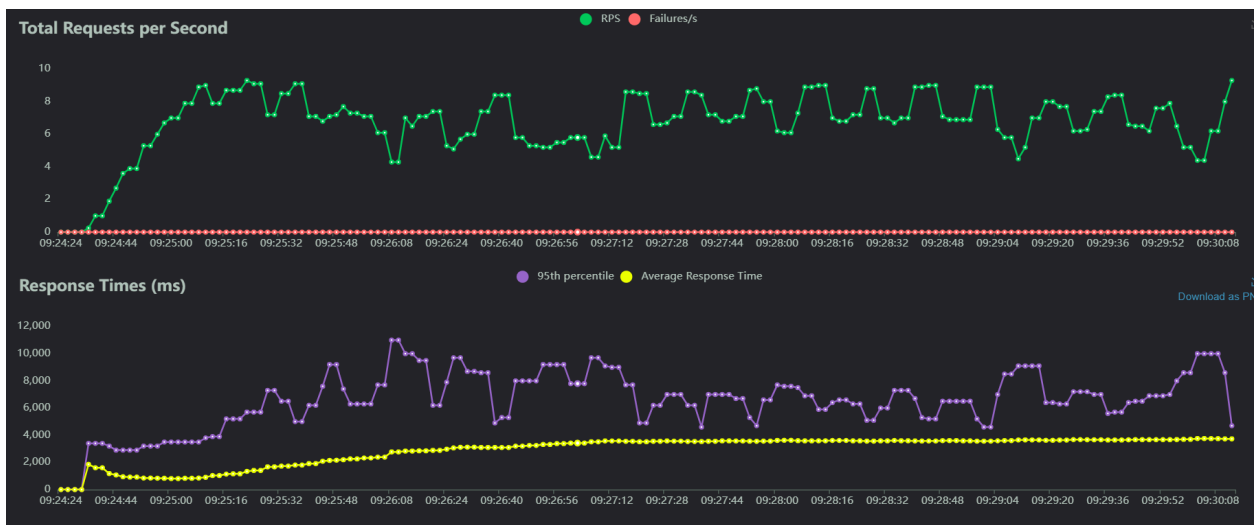


Рисунок 3.23 - второе нагрузочное тестирование

После проведения второго можно увидеть уменьшение времени обработки запроса относительно первого тестирования. Среднее время обработки запроса составляет 8 секунд.

3.6 Вывод по третьей главе

В данной главе была проведена разработка веб-приложения. Описаны основные функции серверной части приложения и приведены рисунки интерфейса клиентской части. Также было проведено исследование на эффективность отображения стажировок относительно сайта hh.ru, после выполненного исследования я внёс коррективы в алгоритм отображения стажировок, тем самым добился лучших результатов при отображении релевантных стажировок. Был реализован основной функционал по

проведению тестирования пользователя, а также разработан конструктор тестов. Пользователи могут проходить тестирование и после отправки решения смотреть статус проверки выполненного тестового задания. Было проведено нагрузочное тестирование при 100 одновременных пользователей, по результатам которого был оптимизирован код для обработки запроса на получение списка стажировок.

ЗАКЛЮЧЕНИЕ

В данной дипломной работе была разработана веб-платформа InternShift, предназначенная для эффективного поиска стажировок с учетом стека технологий. Платформа предлагает пользователям удобный поиск стажировок по технологиям, которые пользователь выбирает на этапе редактирования профиля. Система предлагает возможность пройти тестовое задание от компании на знание технологий, а также задачи с возможностью отправки файлов.

В ходе работы над данным веб-приложением была рассмотрена предметная область и рассмотрены существующие решения, где был функционал по поиску стажировок. Обзор показал, что у решений полностью отсутствует система для прохождения тестовых заданий на их платформе, таким образом, есть актуальность в реализации собственной системы с дополнительным функционалом.

Были решены все поставленные задачи:

1. Рассмотрены существующие аналоги по подбору стажировок.
2. Рассмотрены существующие технологии для создания веб-приложения.
3. Определена структура базы данных, а также выбрана архитектура для разработки серверной части приложения.
4. Определена архитектура для разработки клиентской части.
5. Разработана серверная часть веб-приложения с использованием фреймворка Django.
6. Разработана клиентская часть веб-приложения с использованием фреймворка React.

Было проведено нагрузочное тестирование системы, по результатам которого был оптимизирован код для уменьшения времени обработки запросов.

Такое приложение можно внедрить в различные платформы, как отдельный сервис, только ориентированное под систему для прохождения тестовых заданий.

Что можно добавить в будущем:

- Интеграция с социальными сетями: Можно сделать интеграции для авторизации с помощью социальных сетей.
- Разработка мобильного приложения: Можно создать мобильное приложение для более удобного мониторинга статуса отправленного заявления, а также для возможности проходить тестирование с мобильного устройства.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. HeadHunter [Электронный ресурс]. – URL: <https://dev.hh.ru/> (дата обращения – 12.04.2024).
2. FutureToday [Электронный ресурс]. – URL: <https://fut.ru/> (дата обращения – 24.04.2024).
3. JobbyAI [Электронный ресурс]. – URL: <https://jobby.ai/> (дата обращения – 24.04.2024).
4. SuperJob [Электронный ресурс]. – URL: <https://www.superjob.ru/> (дата обращения – 24.04.2024).
5. PostgreSQL [Электронный ресурс]. – URL: <https://www.postgresql.org/docs/16/> (дата обращения – 03.04.2024).
6. MySQL [Электронный ресурс]. – URL: <https://dev.mysql.com/doc/> (дата обращения – 03.04.2024).
7. MariaDB [Электронный ресурс]. – URL: <https://mariadb.com/kb/en/documentation/> (дата обращения – 03.04.2024).
8. Oracle [Электронный ресурс]. – URL: <https://docs.oracle.com/en/database/> (дата обращения – 03.04.2024).
9. Microsoft SQL [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/sql/> (дата обращения – 03.04.2024).
10. SQLite [Электронный ресурс]. – URL: <https://www.sqlite.org/docs.html> (дата обращения – 03.04.2024).
11. Python [Электронный ресурс]. – URL: <https://docs.python.org/3/> (дата обращения – 22.03.2024).
12. Django [Электронный ресурс]. – URL: <https://docs.djangoproject.com/en/5.0/> (дата обращения – 03.04.2024).

- 13.Flask [Электронный ресурс]. – URL: <https://flask.palletsprojects.com/en/3.0.x/> (дата обращения – 03.04.2024).
- 14.Pyramid [Электронный ресурс]. – URL: <https://docs.pylonsproject.org/projects/pyramid/en/latest/> (дата обращения – 03.04.2024).
- 15.FastAPI [Электронный ресурс]. – URL: <https://fastapi.tiangolo.com/> (дата обращения – 03.04.2024).
- 16.HTML [Электронный ресурс]. – URL: <https://developer.mozilla.org/en-US/docs/Web/HTML> (дата обращения – 18.04.2024).
- 17.CSS [Электронный ресурс]. – URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата обращения – 18.04.2024).
- 18.JavaScript [Электронный ресурс]. – URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (дата обращения – 18.04.2024).
- 19.TypeScript [Электронный ресурс]. – URL: <https://www.typescriptlang.org/docs/> (дата обращения – 01.04.2024).
- 20.Dart [Электронный ресурс]. – URL: <https://dart.dev/> (дата обращения – 01.04.2024).
- 21.React [Электронный ресурс]. – URL: <https://react.dev/> (дата обращения – 03.04.2024).
22. Индекс TIOBE [Электронный ресурс]. – URL: <https://www.tiobe.com/tiobe-index/> (дата обращения – 15.04.2024).

23. Statista.com [Электронный ресурс]. – URL:
<https://www.statista.com/markets/418/topic/484/software/> (дата обращения
– 15.04.2024).