ВВЕДЕНИЕ

Современный мир труда требует быстрого приспособления к новым технологиям и навыкам. Эффективная платформа, учитывающая как уровень подготовки соискателя, так и актуальные технологические требования компаний, является ключевым инструментом для успешного взаимодействия между соискателями и работодателями. Подобная платформа поможет сократить время и ресурсы, затрачиваемые на поиск подходящих стажировок, а также способствует более точному соответствию потребностей студентов и ожиданий работодателей в области стажировок. Существующие интернет-платформы зачастую имеют ограниченное количество предлагаемых вакансий и не включают программы стажировок от небольших и средних компаний. Разработка усовершенствованной платформы, которая будет содержать широкий спектр программ стажировок, включая те, что предлагаются малыми фирмами, будет значительным шагом к обеспечению более эффективного и удобного поиска стажировочных мест для всех заинтересованных сторон.

Целью моей дипломной работы является улучшение процесса поиска стажировок по стеку технологий, которые актуальны в данный момент времени. Для этого будет создана платформа для поиска стажировок в различных компаниях с возможностью прохождения тестов по самым популярным технологиям, которая поможет стажерам в той или иной степени набраться уверенности и подать заявку на прохождение стажировки.

Для достижения поставленной цели были сформулированы следующие задачи:

1) Исследовать и проанализировать существующие аналоги по подбору вакансий на основе стека технологий.
2) Исследовать и проанализировать существующие тесты для определения уровня знаний по технологиям.

3) Проанализировать и подобрать алгоритмы для подбора вакансий по уровню знаний различных технологий.
4) Спроектировать серверную часть веб-приложения на основе языка Python, определить структуру базы данных, описать требования по функционалу веб-приложения.
5) Спроектировать клиентскую часть веб-приложения на базе языка TypeScript, создать макеты веб-приложения на Figma.
6) Разработать серверную часть веб-приложения с использованием фреймворка Django, а также технологии Firebase.
7) Разработать клиентскую часть веб-приложения с использованием фреймворка React и библиотекой Ant Design.
8) Соединить клиентскую и серверную части в одно приложение.
9) Протестировать веб-приложение, проанализировать пользовательский интерфейс, внести коррективы на основе полученных результатов.
10) Опубликовать веб-приложения на хостинге.

Данная платформа будет актуальна для людей, которые только начинают путь разработчика, так и для тех, кто хочет поменять сферу деятельности. Они могут использовать платформу InternShift для повышения самооценки в знании той или иной технологии, а также при поиске подходящих вакансий по их навыкам.

Хочется отметить, что разработка данной платформы по поиску стажировок на основе стека технологий с возможностью прохождения тестов по различным технологиям является актуальным на данный момент времени, так как сейчас много людей ищут вакансии и стажировки, чтобы начать свой путь в IT, но многие не знают где начать поиски. Поэтому данная платформа сможет улучшить процесс поиска, а также послужит своеобразным ориентиром в каком направлении стоит развиваться.

1 Теоретический обзор

В данной главе будет рассматриваться вопрос об актуальности создания платформы по поиску стажировок, рассмотрены существующие аналоги и исследованы методы подбора вакансий, которые использованы в веб-приложениях, будут изучены алгоритмы для определения наиболее подходящих по навыкам вакансий, а также исследованы методы по созданию тестов для определения примерного уровня знаний по различным технологиям.

1.1 Актуальность разработки платформы

В настоящее время стажировки становятся все более популярным способом приобретения опыта для студентов и молодых специалистов, а также для тех, кто собирается менять свой род деятельности. В свете растущей конкуренции на рынке труда, разработчики стремятся развивать свои навыки и расширять свой опыт через прохождение программ стажировок.

Многие компании, даже самые малые, начали предлагать программы стажировок для отбора потенциальных сотрудников. Какие плюсы при этом приносят программы стажировок для различных компаний? Во-первых, компания сможет проверить навыки будущих сотрудников на практике, тем самым происходит своеобразная оценка способностей стажеров. Во-вторых, стажировки предоставляют возможность обучения и развития для стажеров, то есть компания, которая работает с непопулярными, либо внутренними технологиями сможет привлекать разработчиков с похожей сферой деятельности, если рынок разработчиков, специализирующихся на определенных технологиях и языках программирования находится в дефиците.

Исходя из анализа компании ANCOR в 2023 году, основной проблемой в поиске кандидатов через стажировочные программы является дефицит кадров с необходимыми компетенциями, то есть большинство стажеров либо не уверены в своей компетенции и не подают заявки на такие программы, либо они не знают уровня своих знаний по определенным технологиям.

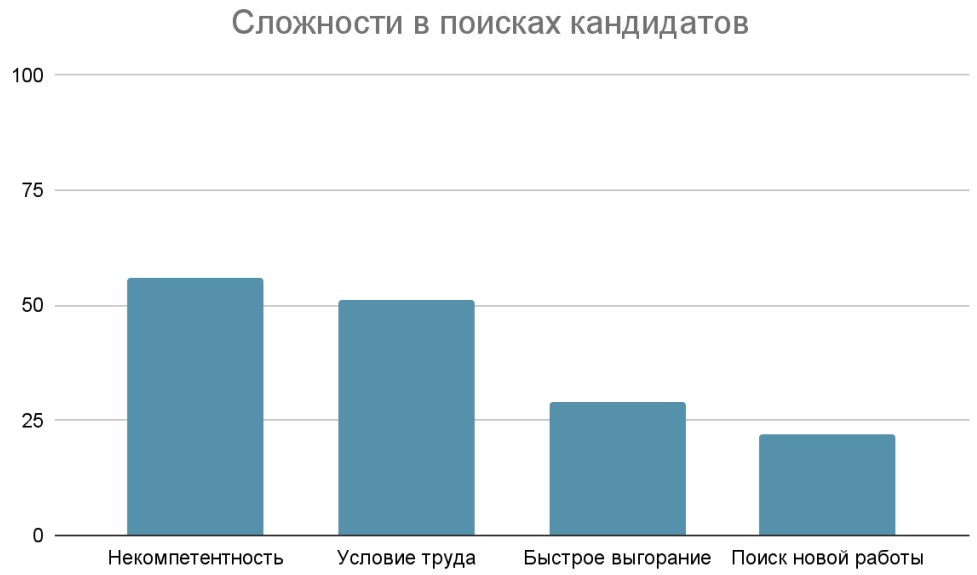

Рисунок 1.1

Таким образом, основной проблемой в поиске кандидатов на стажировку является некомпетентность самих стажеров. Как же можно решить данную проблему? В первую очередь, вопрос некомпетентности говорит нам о том, что есть пробелы в изучении той или иной технологии. Для решения этой проблемы я предлагаю платформу, которая специализируется на составлении тестов на знание технологий. Таким образом разработчик будет уверен в своих силах, а также сможет найти пробелы в знаниях и в дальнейшем улучшит свои навыки.

1.2 Обзор аналогов

Рассмотрим несколько агрегаторов вакансий, а также платформ для подбора вакансий по резюме. В обзоре я буду изучать только стажировочные программы.

1.2.1 Headhunter

HeadHunter - платформа для поиска работы и подбора персонала. Она предоставляет возможность работодателям эффективно находить потенциальных сотрудников, а также помогает соискателям в поиске подходящих вакансий. У сайта есть множество полезных функций:

- Поиск вакансий и резюме: Пользователи могут искать вакансии и резюме по различным критериям, таким как должность, город, зарплата и опыт работы.
- Публикация вакансий: Работодатели могут размещать вакансии на сайте, привлекая кандидатов со всех точек страны.
- Профили компаний: Компании могут создавать свои профили, предоставляя информацию о себе, своей деятельности и открытых вакансиях.
- Оценка кандидатов: Система оценки кандидатов позволяет работодателям быстро и эффективно оценить соответствие кандидатов требованиям вакансии.

Рассмотрим также преимущества и недостатки платформы в поиске стажировок:

Преимущества:

- Широкий охват: Данный ресурс является лидером среди агрегаторов вакансий в России, поэтому на нем можно найти множество различных стажировочных программ по различным компаниям.
- Удобство использования: Интуитивно понятный интерфейс делает сайт простым в использовании как для работодателей, так и для соискателей.

- Прохождение тестов: HeadHunter обладает огромной базой заданий по различным технологиям. Формат данного тестирования индивидуальный на каждого пользователя, поэтому можно раз в несколько месяцев повторно проходить тесты и изменять свой рейтинг знаний по различным технологиям.

Недостатки:

- Конкуренция: Из-за массивного использования сайта, конкуренция среди пользователей платформы очень большая, особенно на популярных вакансиях.
- Услуги: Некоторые функции доступны только за дополнительную плату.
- Не всегда актуальные вакансии: Некоторые вакансии могут быть неактуальными или уже закрытыми.
- Отсутствие статистики по тестированиям: Для обычного пользователя не видны результаты тестирования на знание технологий, поэтому невозможно увидеть свои пробелы в знании.

В заключении, можно отметить универсальный характер платформы, но HeadHunter больше ориентирован на поиск вакансий на определенные должности, поэтому количество актуальных стажировок в нем ограничено.

1.2.2 FutureToday

FutureToday - платформа для поиска программ стажировок, а также различных школ для профессиональной подготовки. 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.

Рассмотрим также преимущества и недостатки платформы в поиске стажировок:

Преимущества:

- Оформление: Красиво оформлена страница с подробной информацией по программам. Пользователь может подробно ознакомиться с различными направлениями стажировочных программ, а также узнать время их проведения.
- Актуальность: В каталоге вакансий расположены самые актуальные программы стажировок и обучения, можно сразу увидеть дату конца отбора на стажировки
- Быстрая отправка заявки: Можно сразу подавать заявки на некоторые программы стажировок, нужно лишь заполнить небольшую форму и оставить свой телефон, либо почту для обратной связи

Недостатки:

- Отсутствие фильтрации: На сайте есть поиск по названию, но нет базовой фильтрации по городу, направлению, формату проведения стажировки.
- Малое количество программ: На платформе опубликовано малое количество программ, в основном это программы больших компаний.
- Ручной поиск: На FutureToday размещены программы стажировок одной компании, то есть, если пользователь захочет выбрать программу по своему направлению, то он должен самостоятельно зайти и изучить дополнительную информацию о программе стажировок. В худшем случае, направление пользователя может отсутствовать в программах.

В общем, FutureToday является отличной платформой для поиска стажировок, но неоспоримым недостатком в нем является ручной поиск подходящего направления.

1.2.3 Jobby.AI

Jobby - это платформа для поиска вакансий с учетом навыков, сайт предоставляет широкий спектр программ стажировок в различных компаниях. Функционал у данного приложения обширный:

- Фильтрация вакансий: Присутствует поиск вакансий по ключевым словам, должностям, городам, фильтрация вакансий по зарплате, типу занятости, уровню опыта и другим параметрам.
- Карьерные консультации: Возможность общения с ментором и получение советов по различным вопросам
- Профиль пользователя: Возможность тщательного редактирования профиля. В профиле можно указывать технические навыки, по которым можно отфильтровать подходящие вакансии.
- Топ работодателей: Можно просматривать лучшие компании по разным сферам

- Можно рассмотреть также присутствующие преимущества и недостатки платформы.

Преимущества:

- Хорошая фильтрация: Пользователи могут точно настроить параметры поиска вакансий, что позволяет быстро находить подходящие предложения.
- Карьерные консультации: Наличие карьерных советов и консультаций помогает пользователям принимать осознанные решения о своей карьере и повышать свою конкурентоспособность на рынке труда.
- Персонализированный профиль: Пользователи могут создать детальный профиль с информацией о своем опыте работы и навыках, что поможет им выделиться среди других соискателей.
- Быстрые отклики: Возможность быстро откликнуться на понравившиеся вакансии, можно откликаться без резюме.

Недостатки:

- Конкуренция: Возможно, большое количество пользователей на платформе может создать высокую конкуренцию за популярные вакансии.
- Отсутствие верификации навыков: На платформе можно сразу указать профессиональные навыки, не проходя при этом никакой проверки.

В заключении, можно отметить хороший функционал сайта Jobby.AI, присутствует отличная фильтрация вакансий, а также редактирование профиля пользователя.

1.2.4 Superjob Students

Superjob - это платформа для поиска работы и подбора персонала. Данный сайт имеет ряд функций:

- Поиск вакансий и резюме: Пользователи могут искать вакансии и резюме по различным критериям, включая должность, город, зарплату и опыт работы.
- Карьерные тренды: SuperJob предоставляет информацию о карьерных трендах.
- Размещение вакансий и резюме: Работодатели могут размещать вакансии, а пользователи - свои резюме на платформе.
- Уведомления: Пользователи могут подписаться на уведомления о новых вакансиях, соответствующих их запросам.

Рассмотрим преимущества и недостатки платформы.

Преимущества:

- Огромная база вакансий: SuperJob имеет большую базу вакансий и резюме.
- Простота использования: Интерфейс SuperJob интуитивно легок для использования пользователями.
- Рекомендации: Пользователи могут получить полезные рекомендации и советы по поиску работы.

Недостатки:

- Конкуренция: Из-за популярности платформы может возникнуть высокая конкуренция за популярные вакансии.
- Не всегда актуальные вакансии: Некоторые вакансии являются неактуальными, для определения которого надо обращаться к работодателю.

Таким образом, SuperJob является хорошим агрегатором вакансий, но количество программ стажировок на нем ограничено.

1.2.5 Выводы по обзору

Я изучил несколько самых популярных агрегаторов вакансий в России. Каждый из них обладает своими преимуществами и недостатками. Преимущества приложений можно почерпнуть для моего веб-приложения. Чтобы удобнее было рассматривать разницу между аналогами я приведу сравнительную характеристику, которая представлена на таблице 1.1.

Основными критериями для сравнения аналогов я взял:

- Количество стажировочных программ
- Персональная фильтрация по стеку технологий пользователя
- Удобство пользования
- Ориентированность на актуальные стажировочные программы

Оценка будет по 5-бальной шкале.

Критерий	HeadHunter	FutureToday	JobbyAI	SuperJob
Количество стажировочных программ	5	3	5	2
Персональная фильтрация по стеку технологий пользователя	5	0	3	0
Удобство пользования	4	3	5	3
Ориентированность на актуальные	2	5	5	3

стажировочные программы				

Таблица 1.1

1.3 Обзор алгоритмов для определения наиболее подходящих вакансий

Рассмотрим алгоритмы, которые могут использоваться для поиска релевантных вакансий по различным фильтрам.

- TF-IDF (Term Frequency-Inverse Document Frequency): Этот алгоритм вычисляет вес каждого слова в документе (в данном случае - вакансии) по его частоте в документе и обратной частоте его встречаемости во всех документах. С помощью TF-IDF можно оценить степень принадлежности вакансии запросу пользователя на основе ключевых слов.
- Word Embeddings (Word2Vec, GloVe, FastText): Эти алгоритмы преобразуют слова в векторы в многомерном пространстве таким образом, что семантически близкие слова имеют близкие векторы. После этого можно измерять расстояние между вектором слова и запроса, таким образом определяя ближайшее подходящее по смыслу вакансию.
- Машинное обучение (Random Forest, Gradient Boosting, Нейронные сети): Методы машинного обучения используются для обучения модели на основе данных о предпочтениях пользователей и характеристиках вакансий. Модель может предсказывать вероятность того, что вакансия будет релевантна пользователю на основе характеристики пользователя и его предпочтений.
- Коллаборативная фильтрация: Этот метод анализирует действия и предпочтения пользователя, а также действия других пользователей,

чтобы рекомендовать ему вакансии, которые могут быть интересны на основе выбора похожих по характеристикам пользователей. В данном случае можно использовать данный метод для того, чтобы определять релевантные вакансии.

Рассмотрим данные алгоритмы поподробнее.

1.3.1 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) - это мера, используемая для оценки важности слова в документе. Она основана на двух основных концепциях:

- TF (Term Frequency): Оценивает, насколько часто слово встречается в документе. Чем чаще слово встречается в документе, тем выше его вес.
- IDF (Inverse Document Frequency): Оценивает, насколько уникально слово для документа. Слова, которые встречаются во многих документах, имеют меньший вес IDF, а слова, которые встречаются редко, имеют более высокий вес IDF.

Высокий вес TF-IDF достигается для терминов, которые часто встречаются в документе, но редко в других.

В контексте поиска вакансий, TF-IDF может использоваться для оценки релевантности вакансии запросу пользователя на основе ключевых слов.
Чем будет больше значение TF-IDF, тем более релевантна будет вакансия по ключевому слову.

1.3.2 Word Embeddings

Word Embeddings - это методы представления слов в виде векторов в многомерном пространстве, где близкие по значению слова имеют близкие векторы. Рассмотрим методы алгоритмов:

- Word2Vec: Word2Vec создает векторы слов путем обучения нейронной сети на большом количестве полезного текста. Существует две основные архитектуры Word2Vec. Предсказывание контекста на основе слова и предсказывание слова на основе контекста. Обе модели выдают вектор представление слова.
- GloVe (Global Vectors for Word Representation): Этот метод использует статистические свойства матрицы встречаемости слов для создания векторов.
- FastText: Метод добавляет возможность работать с подсловами. То есть используется часть слов, вместо использования всего слова в целом.Это позволяет учитывать морфологические особенности слов.

Данные методы Word Embeddings позволяют создавать векторные представления слов, которые формируют представление о близости каждого слова.

1.3.3 Машинное обучение

- Случайный лес: Это деревья решений, в котором каждое дерево строится на основе случайной выборки данных. Каждое дерево совершает своеобразный прогноз, а затем окончательный результат определяется на основе среднего значения. Случайный лес хорошо работает с данными большого объема и способен обрабатывать множество признаков. Он также не требует масштабирования признаков и устойчив к переобучению.
- Градиентный бустинг: Это метод машинного обучения, который строит модель путем обучения новых моделей, которые исправляют ошибки предыдущих моделей. Основная идея заключается в том, чтобы итеративно строить деревья решений, минимизируя ошибки предыдущих деревьев.

1.3.4 Коллаборативная фильтрация

Коллаборативная фильтрация - это метод рекомендательной системы, который использует информацию о предпочтениях и поведении пользователей для предсказания релевантных вакансий. Основывается на анализе предпочтений других пользователей для предсказания.

Существует два основных подхода к коллаборативной фильтрации:

- Фильтрация на основе предпочтений других пользователей: Этот метод использует информацию о предпочтениях пользователей для поиска других пользователей с похожими интересами. После нахождения похожих пользователей система может рекомендовать вакансии, которые просмотрел другой пользователь.
- Фильтрация на основе элементов: В этом методе используется информация о схожести между вакансиями на основе предпочтений пользователей. Если пользователь проявил интерес к определенной вакансии, система может рекомендовать ему похожие вакансии на основе их схожести с выбранной вакансией.

Преимущества:

- Способность рекомендовать персонализированные элементы на основе предпочтений пользователя.
- Возможность работать с разреженными данными (когда не все пользователи оценили все элементы или не все элементы были оценены всеми пользователями).
- Не требует знания о содержании элементов (вакансий), а только информацию о действиях пользователей.

Недостатки:

- Проблема холодного старта: нет данных пользователя.
- Проблема разреженности данных: низкая активность пользователя.

Коллаборативная фильтрация все равно является мощным алгоритмом для поиска релевантных вакансий.

1.3.5 Выводу по обзору

В целом, каждый из этих алгоритмов и методов имеет свои преимущества и недостатки, и выбор конкретного зависит от конкретной задачи, характеристик данных и требований к модели.

Для моего проекта я выбрал коллаборативную фильтрацию. так как фильтрация будет основана на предпочтениях каждого пользователя, исходя из стека технологий на которых будет либо выполнен тест, либо просто отмечена как “выбрано”.

1.4 Обзор различных методов по созданию тестов

Способы создания тестов на знание различных технологий разнится от источника к источнику. Поэтому я выделил несколько, на мой взгляд, самых важных аспектов в создании тестов.

- Анализ реальных задач: Изучение реальных задач, которые происходили со специалистами.
- Экспертное мнение: Обращение за помощью к экспертам в соответствующей области, для получения обратной связи по поводу содержания вопросов, сложность вопросов и рекомендаций по улучшению теста.
- Актуализация тестового материала: Регулярное обновление тестового материала в соответствии с изменениями в требований на рынке труда.
- Создание разнообразных заданий: Включение разнообразных типов заданий в тест, для полного оценивания знаний пользователя.

- Проверка тестов на эффективность: Проведение тестов на малой выборке кандидатов, которые имеют определенный базис знаний в различных технологиях, для проверки эффективности, а также получения обратной связи от участников.

Такая методология может стать эффективной только при долгосрочном исследовании, так как каждый тест, созданный человеком, является субъективным видением того, что должен знать разработчик. Для проверки эффективности созданного теста стоит использовать следующие методы:

- Анализ статистических данных: Собрать данные о результате прохождения теста, такие как процент прохождения, средний балл, распределение результатов и т.д.
- Сравнение с реальным опытом работы: Сравнить результаты теста с реальным опытом работы специалистов в соответствующей области.
- Обратная связь: Нужно попросить участников теста оставить обратную связь о его содержании, сложности и полезности.
- Сравнение с альтернативными существующими тестами: Результаты тестирования нужно сравнить с результатами альтернативных тестов на оценку знаний и навыков.

Таким образом, используя данные методы, можно получить представление о методологии создания качественных тестов на знание технологий. Для этого необходимо время и ресурсы. Кроме того, критерии оценки знаний разработчика зависят от области в которой мы проводим тесты. Например: Разработчик знает отлично про многопоточность, но слабоват в базах данных, но работодатель требует знания в целом, а не по каждому пункту отдельно. Поэтому стажер может набрать необходимый балл после прохождения теста, но всё равно отставать в некоторых аспектах.

По итогу, после обзора методов создания тестов я определился с форматом проведения тестирования пользователей на знание технологий. Теперь я планирую предоставлять результаты в общем виде. Затем при улучшении приложения добавлять новый функционал.

1.5 Вывод по первой главе

В процессе исследования и обзора аналогов я выделил для себя множество аспектов и подходов к разработке платформы по поиску вакансий. Были также рассмотрены алгоритмы для фильтрации и рекомендации самых релевантных вакансий для каждого отдельного пользователя. На основе выделенного алгоритма будет реализована фильтрация вакансий на моем веб-приложении. Также рассмотрены методологии для создания тестов на знание различной технологии. Исходя из рассмотренных аспектов я также выделил для себя методы, которые будут использованы для проведения эффективного тестирования.

2 Проектирование приложения

В данной главе будут описаны этапы к проектированию приложения. Будет создана структура приложения по которой будет реализована разработка. А также будут описаны функциональные и нефункциональные требования к приложению.

2.1 Функциональные и нефункциональные требования

Для разработки веб-приложения распишем функциональные требования ориентированные на потребности пользователей.

2.1.1 Авторизация и регистрация

1) Система должна авторизовать пользователя при успешном прохождении аутентификации
2) Система должна поддерживать аутентификацию по email и паролю
3) Система должна поддерживать смену пароля авторизованного пользователя
4) Система должна поддерживать восстановление пароля пользователя
5) Система должна поддерживать выход из системы
6) Система должна поддерживать регистрацию пользователя по email и паролю
7) При регистрации система должна верифицировать почту пользователя
8) Система должна поддерживать уникальность email пользователей

2.1.2 Вакансии

1) Система должна поддерживать отображение вакансий на платформе
2) Система должна поддерживать фильтрацию вакансий
3) Система должна поддерживать поиск вакансий по ключевым словам
4) Система должна поддерживать отображение дополнительной информации о вакансии

5) Система должна поддерживать отображение подходящих для пользователя вакансий
6) Система должна отображать примерное проанализированное соответствие между вакансией и навыками пользователя
7) Система должна поддерживать создание/редактирование/удаление вакансий

2.1.3 Тесты

1) Система должна поддерживать прохождение тестов авторизованными пользователями
2) Система должна поддерживать корректное отображение тестов
3) Система должна сохранять результаты тестирования пользователя
4) Система должна поддерживать отображение всех доступных тестов для пользователя
5) Система должна поддерживать поиск тестов по ключевым словам
6) Система должна поддерживать отправку тестов на проверку
7) Система должна поддерживать корректное отображение вопросов для теста
8) Система должна поддерживать создание тестов в конструкторе тестов
9) Система должна поддерживать редактирование/удаление тестов

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 вакансиях компании
4) Система должна поддерживать фильтрацию вакансий компании по специальностям
5) Система должна поддерживать поиск по ключевым словам

6) Система должна поддерживать создание/редактирование/удаление компаний

2.1.5 Тренды

1) Система должна показывать отчет о самых популярных технологиях
2) Система должна корректно отображать аналитику технологий по различным компаниям
3) Система должна корректно отображать анализ количества вакансий по различным ресурсам
4) Система должна поддерживать добавление аналитики

2.1.6 Профиль

1) Система должна поддерживать редактирование профиля

Для неавторизованных пользователей будет доступна лишь страница с авторизацией и регистрацией

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

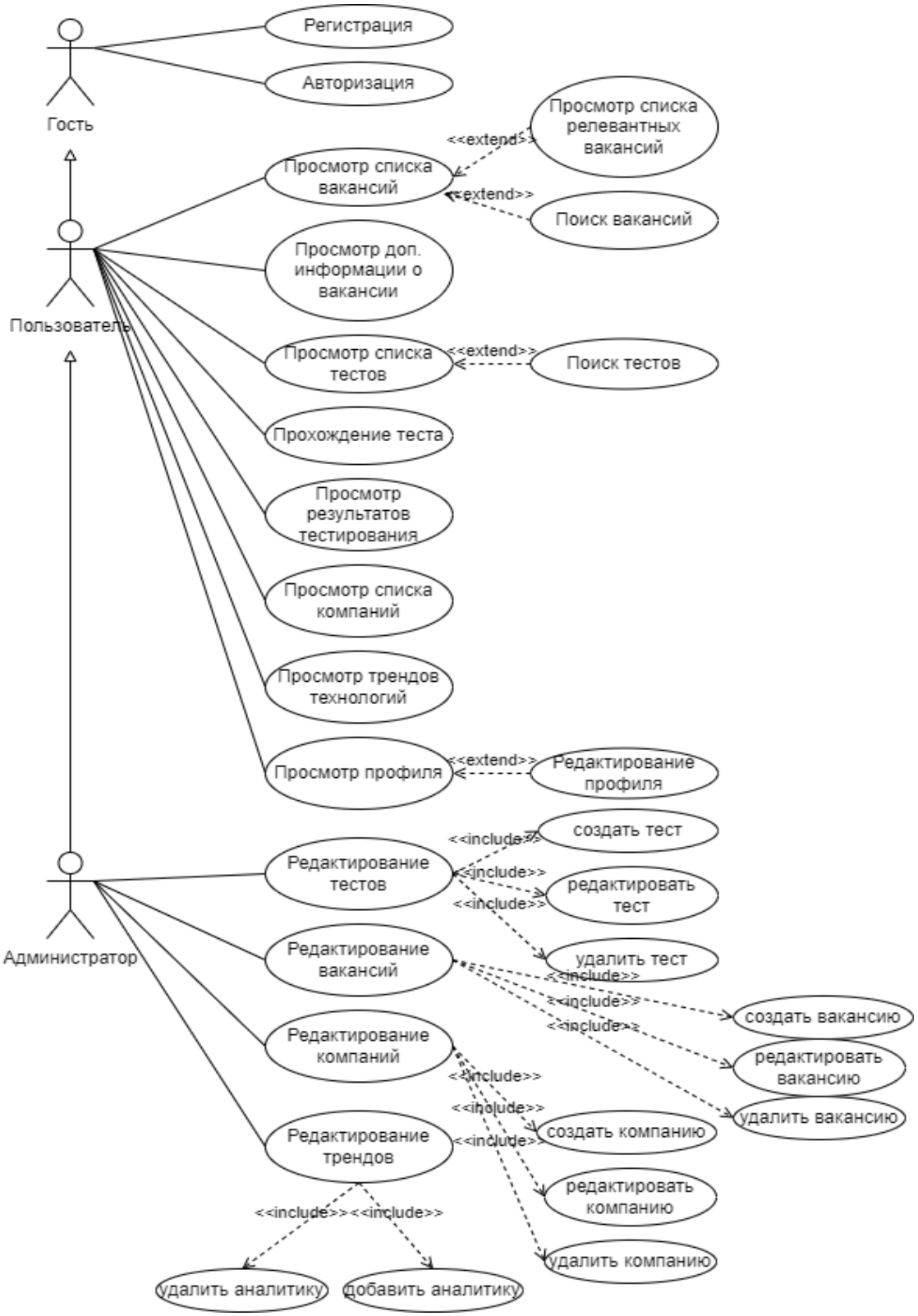
Гость
Регистрация
Авторизация
Пользователь
Просмотр списка вакансий
Просмотр списка релевантных вакансий
Поиск вакансий
Просмотр доп. информации о вакансии
Просмотр списка тестов
Поиск тестов
Прохождение теста
Просмотр результатов тестирования
Просмотр списка компаний
Просмотр трендов технологий
Просмотр профиля
Редактирование профиля
Администратор
Редактирование тестов
создать тест
редактировать тест
удалить тест
Редактирование вакансий
создать вакансию
редактировать вакансию
удалить вакансию
Редактирование компаний
создать компанию
редактировать компанию
удалить компанию
Редактирование трендов
удалить аналитику
добавить аналитику
<<extend>>
<<include>>

Рисунок 2.1

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

- Надежность: Будет реализован мониторинг приложения для непрерывного контроля за его работоспособностью и производительностью.
- Безопасность: Использование безопасных методов аутентификации, такие как хэширование паролей и использование CSRF-токенов, шифрование данных в БД.
- Условия эксплуатации: Приложение будет размещено на надежном хостинге с возможностью мониторинга и управления ресурсами.

2.2 Проектирование базы данных

Для реализация приложения я буду использовать базу данных PostgreSQL. База данных состоит из 9 таблиц:

1) User: На данной таблице будет храниться информация о пользователя - почта пользователя, хешированный пароль
2) Skill: Таблица с технологиями, нужен для хранения различных технологий
3) Company: На этой таблице будет храниться информация о компаниях - название, адрес, почта и описание
4) Vacancy: Будет храниться информация о вакансиях.
5) Test: По каждым технологиям будет выделен тест с названием и описанием
6) VacancySkill: Таблица для хранения информации об вакансии и какие навыки будут нужны для конкретной вакансии
7) Trend: Таблица для хранения аналитики по различным технологиям

8) SkillUser: Промежуточная таблица для отношения многие-ко-многим, нужна для хранения информации о результатах тестирования пользователя

Для наглядного отображения была построена даталогическая модель на рисунке 2.2

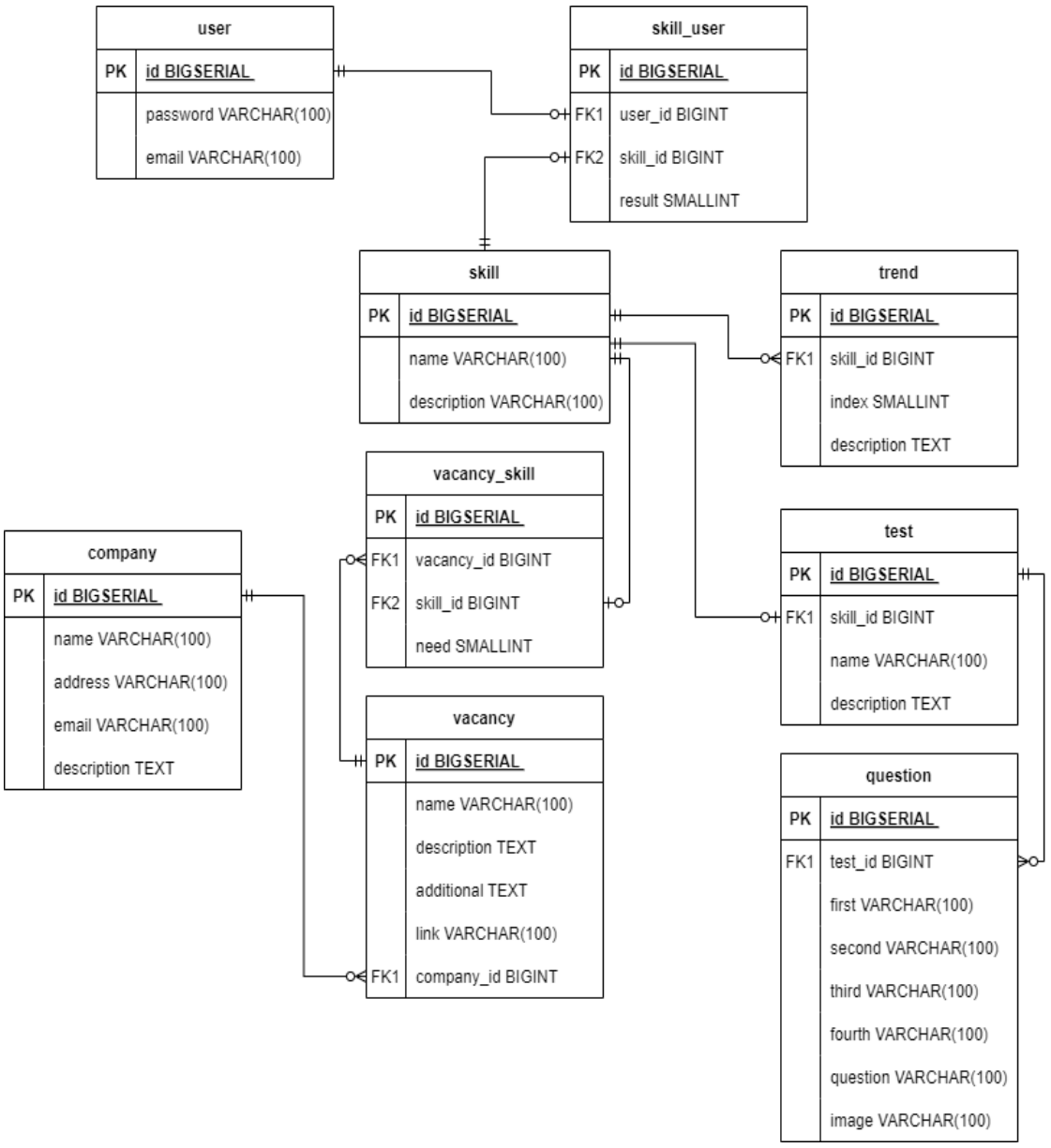

Рисунок 2.2

2.3 Проектирование серверной части

Серверная часть приложения будет реализована на языке Python с использованием фреймворка Django. Данный фреймворк является мощным инструментом для создания веб-приложений. Основным преимуществом является связь с различными библиотеками на языке Python для анализа данных, а также с встроенными инструментами безопасности.

Для разработки веб-приложения будет использоваться архитектурный шаблон MTV - Model-Template-View:

- Модели (Model) представляют бизнес-логику и взаимодействуют с базой данных.
- Представления (View) отвечают за обработку запросов и взаимодействие с моделями, а также за отображение данных пользователю.
- Шаблоны (Template) отвечают за представление данных пользователю, они содержат HTML-код с встроенными переменными и логикой.

Схема работы шаблона MTV приведена на рисунке 2.3

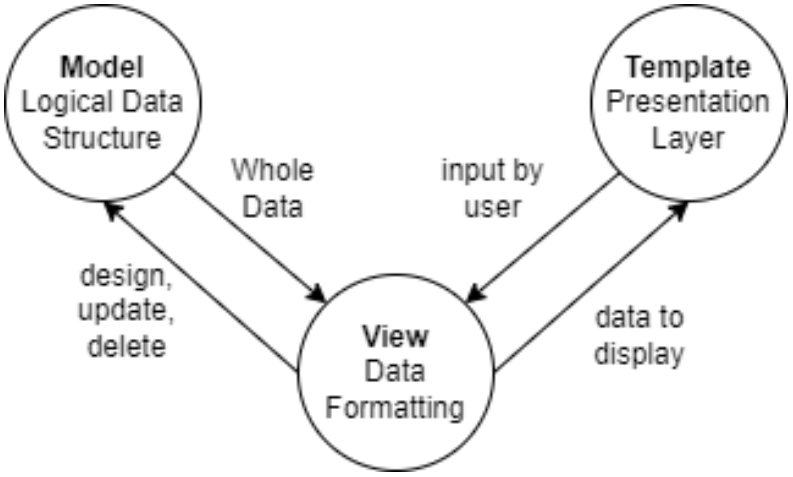

Рисунок 2.3

2.4 Проектирование клиентской части

Клиентская часть приложения будет реализована на языке TypeScript с использованием фреймворка React. Далее рассмотрим архитектуру и макеты веб-приложения.

2.4.1 Архитектура клиентской части приложения

Для архитектуры клиентской части приложения была использована модель FSD - Feature-Sliced Design. Проект на FSD состоит из слоев (layers), каждый слой состоит из слайсов (slices) и каждый слайс состоит из сегментов (segments).

Слои стандартизированы и расположены вертикально. Модули на одном слое могут взаимодействовать лишь с модулями, находящимися на слоях строго ниже.

- shared — переиспользуемый код, не имеющий отношения к специфике приложения/бизнеса.
- entities (сущности) — бизнес-сущности.
- features (фичи) — взаимодействия с пользователем.
- widgets (виджеты) — композиционный слой для соединения сущностей и фич в самостоятельные блоки.
- pages (страницы) — слой для сборки полноценных страниц из сущностей, фич и виджетов.
- app — настройки для всего приложения.

Затем есть слайсы, разделяющие код по предметной области. Они группируют логически связанные модули, что облегчает навигацию по кодовой базе. Слайсы не могут использовать другие слайсы на том же слое, что обеспечивает высокий уровень связности (cohesion) при низком уровне зацепления (coupling).

В свою очередь, каждый слайс состоит из сегментов. Это маленькие модули, главная задача которых — выделить код внутри слайса по техническому назначению. Схема модели FSD приведена на рисунке 2.4

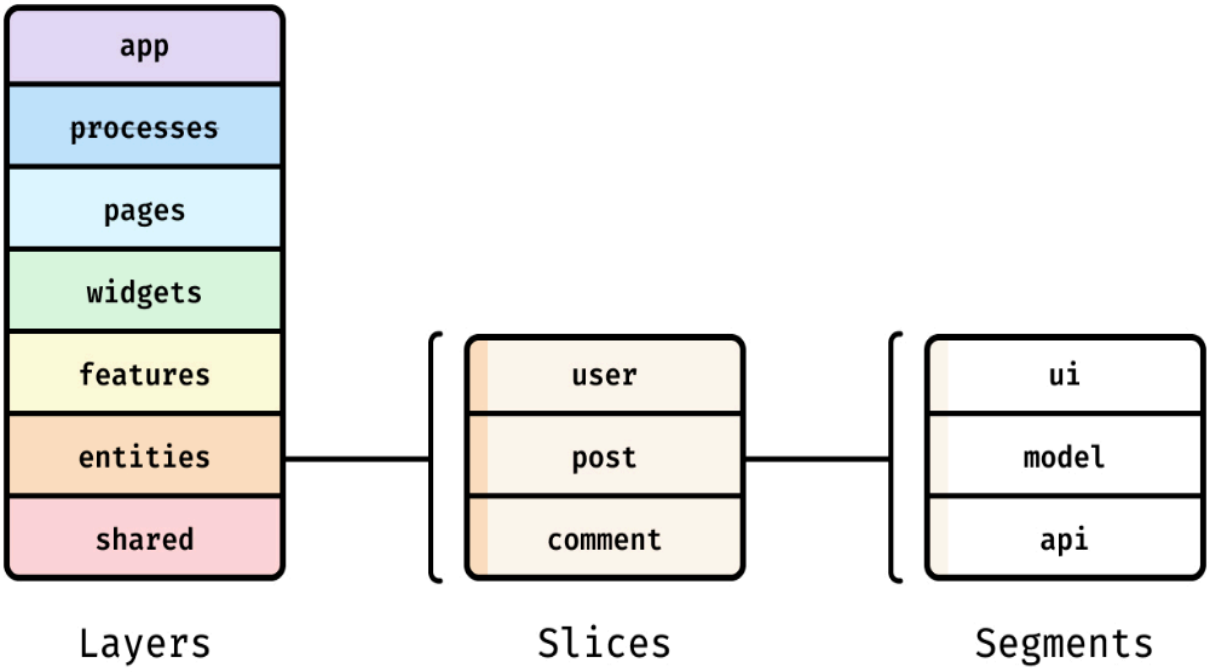

Рисунок 2.4

Преимущества данной архитектуры:

- Единообразие: Код распределяется по слоям, предметной области и техническому назначению. Благодаря этому архитектура стандартизируется и становится более простой.
- Переиспользование логики: Каждый компонент архитектуры имеет свое назначение и список зависимостей. Благодаря этому появляется возможность адаптировать модуль под разные цели.
- Устойчивость к изменениям: Один модуль не может использовать другой модуль, расположенный на том же слое или на слоях выше. Благодаря этому компоненты изолированы.

2.4.2 Макет приложения

Для создания макета веб-приложения я использую платформу Figma. Данный веб-сайт обладает мощным инструментарием для создания прототипов приложений.

InternShift будет состоять из 5 основных страниц и нескольких вспомогательных. Основной страницей является - страница с вакансиями. На нем можно увидеть функции поиска и фильтрации по различным категориям. А также на каждой карточке вакансий можно будет увидеть описание и основной стек технологий. Макет страницы представлен на рисунке 2.5.

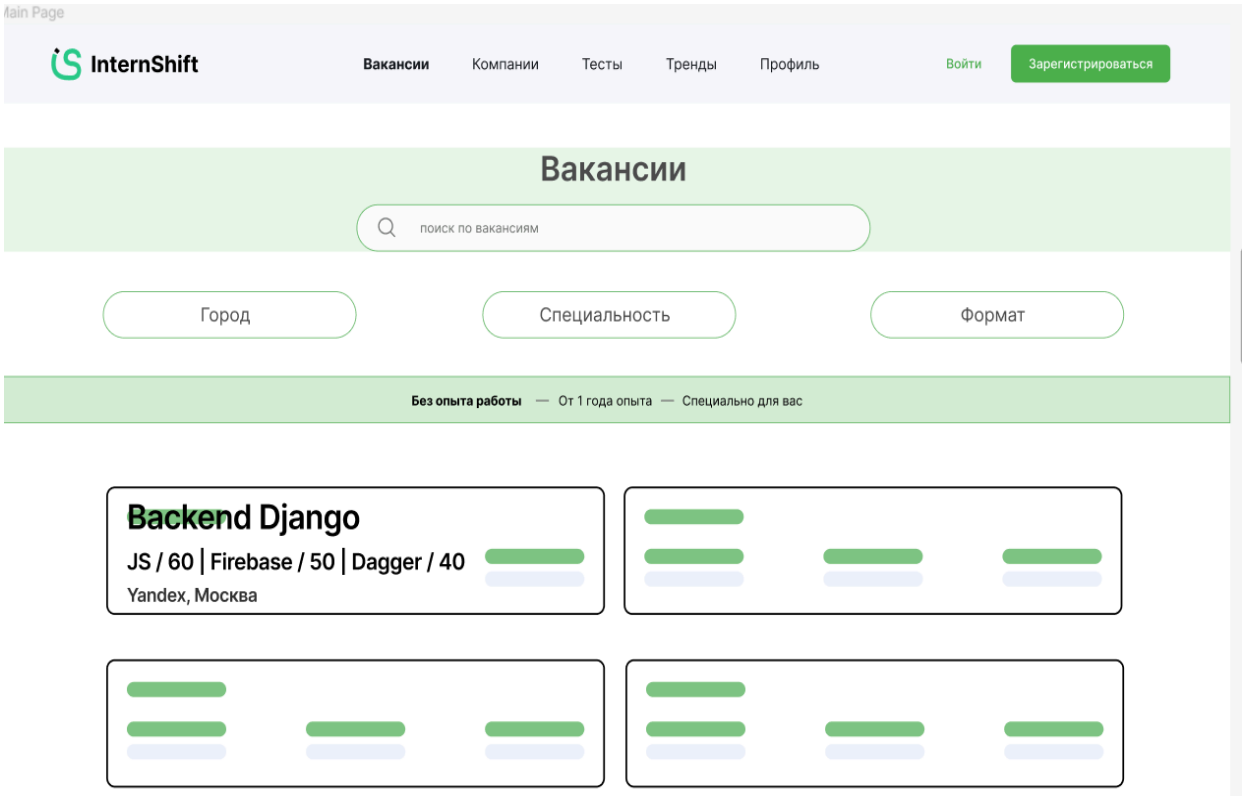

Рисунок 2.5

Страница с компаниями будет хранить информацию о компаниях, сортировка будет проводиться по количеству доступных вакансий, но будет поиск по названиям компаний и фильтрация по различным категориям. Макет страницы приведен на рисунке 2.6.

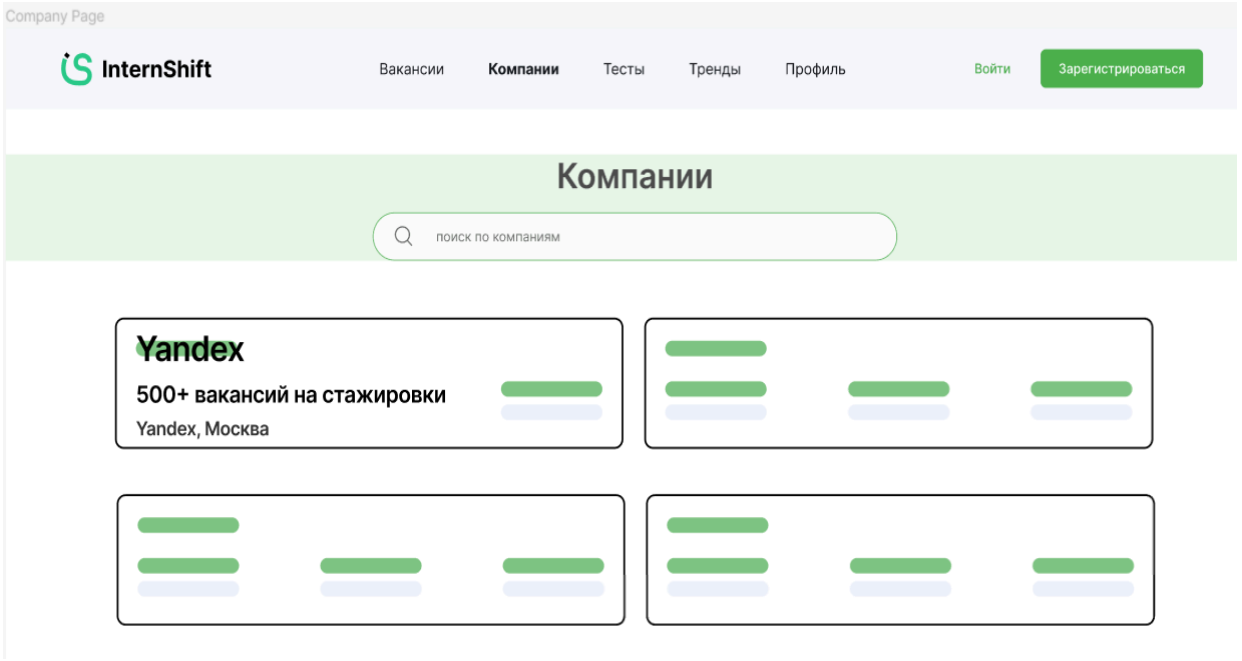

Рисунок 2.6

Страница с тестами будет содержать множество карточек с различными технологиями и языками программирования. На карточке будет отражена информация о статусе прохождения теста. Макет страницы приведен на рисунке 2.7.

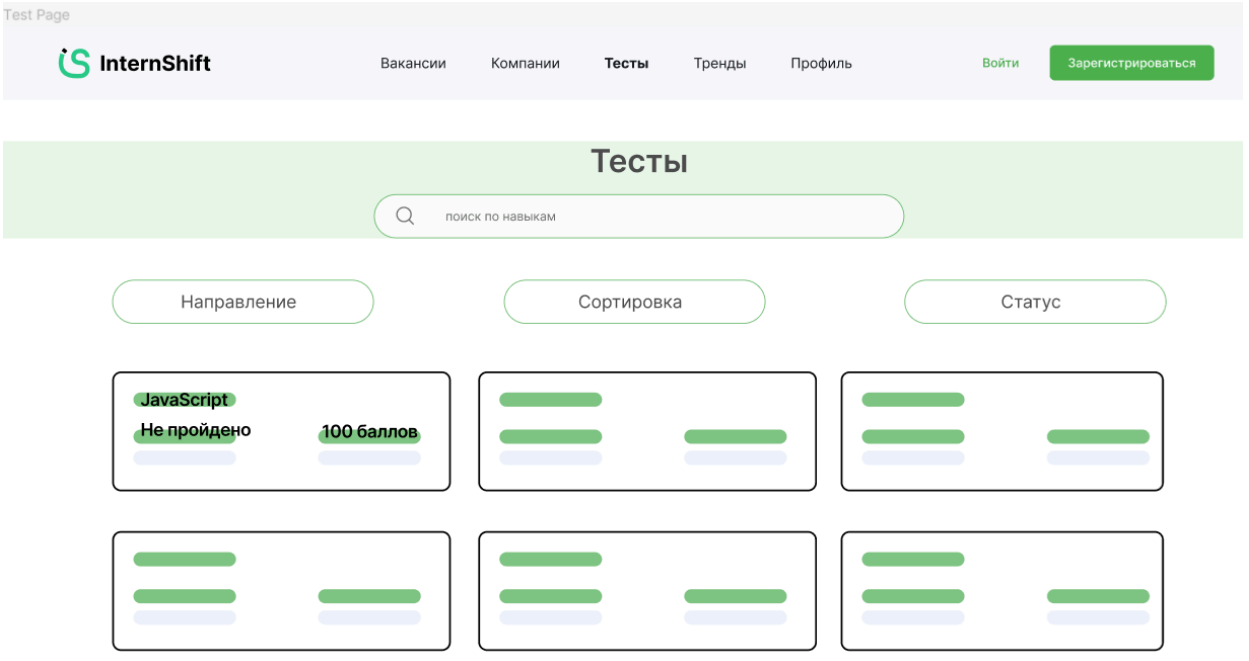

Рисунок 2.7

Страница трендов будет содержать информацию о самых популярных технологиях в различных вакансиях и самых распространенных вакансиях на рынке. Информация будет взята из самых популярных агрегаторов вакансий. Макет страницы представлен на рисунке 2.8.

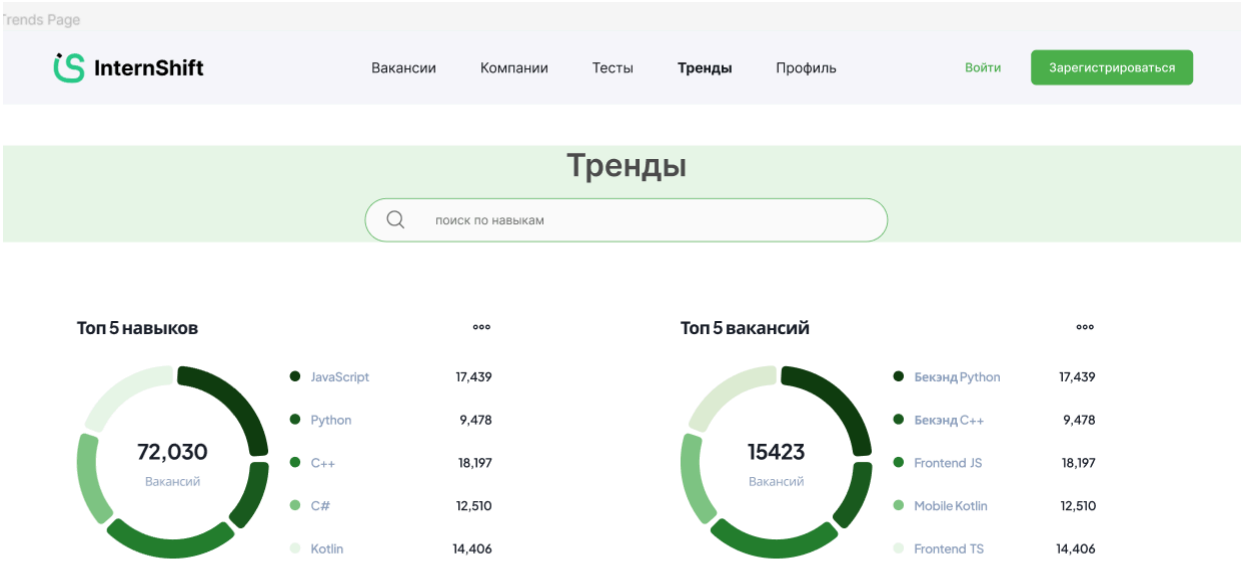

Рисунок 2.8

Страница профиля пользователя будет содержать информацию о пользователе и результатов тестирования. При нажатии на карточку с навыками можно будет переходить на страницу с выбранным тестом. Макет страницы представлен на рисунке 2.9.

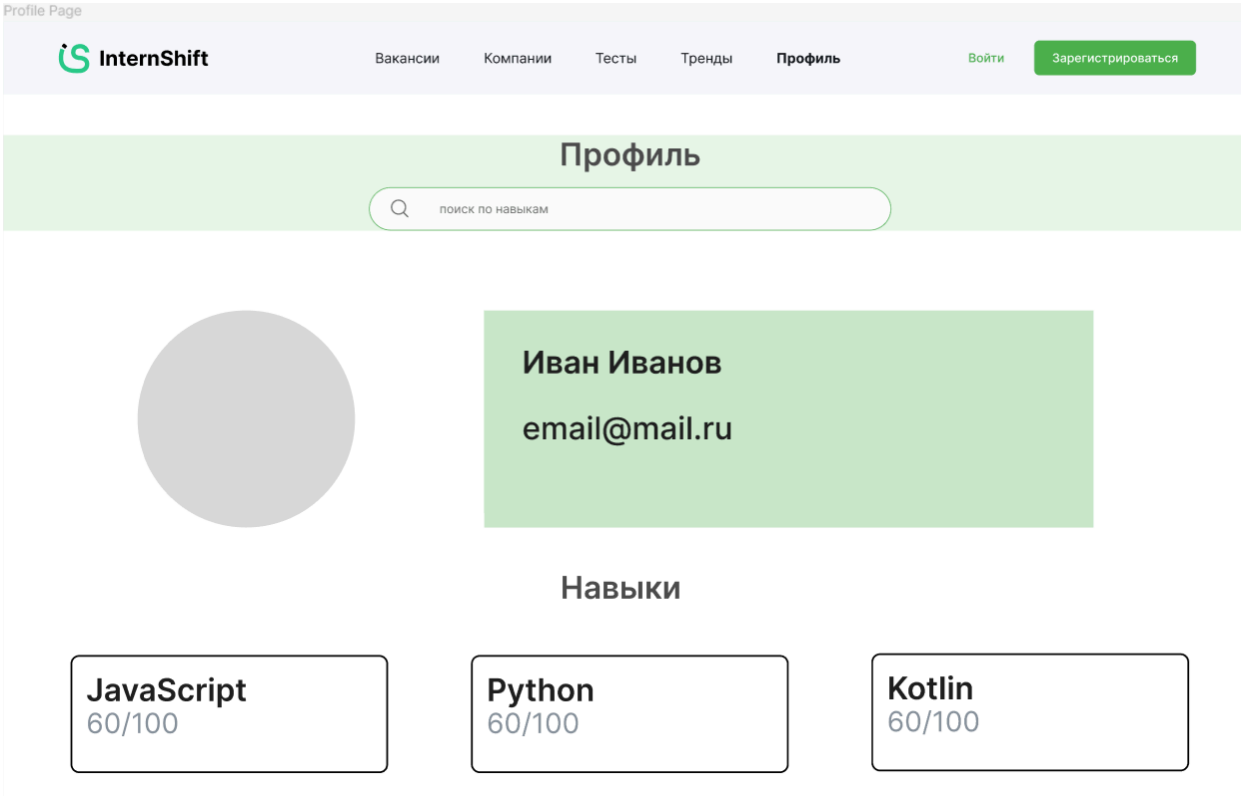

Рисунок 2.9

2.5 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 клиентской и серверной части приложения. Для хранения данных была сформирована даталогическая модель базы данных.

3 Реализация приложения

В данной главе будет описаны этапы по разработке всего приложения. Глава будет разделена на 3 основных этапа: Backend, Frontend, Аналитика. По каждому этапу будет подробное описание всех основных компонентов. Листинг кода будет только по самым важным частям приложения с подробным описанием функции всех компонентов.

3.1 Разработка серверной части приложения

В данном разделе будет рассмотрены основные детали разработки серверной части приложения. Медиа-контент присутствующий в приложении будет храниться в базе данных как URL-адрес соответствующего контента, в моем случае изображения в вопросах тестов. Так как количество вопросов со всех тестов будет очень большим, то рационально не хранить изображения хорошего качества в базе данных, так как это займет огромное количество памяти. Поэтому я решил сейчас использовать данную структуру в базе данных. Также разработка была проведена в виртуальной среде Python. Она позволяет изолировать проект от других проектов в системе, то есть все пакеты устанавливаемые в проекте будут доступны только для данного приложения.

3.1.1 Реализация базы данных

При разработке базы данных во фреймворке Django используем готовую систему моделей данных. Модель данных представляет собой класс Python, который определяет структуру и типы данных полей базы данных.

Сначала определим модели данных. Для каждой таблицы создаем отдельный класс и определяем нужные нам поля, а также отношения между

таблицами. Для таблицы User я буду использовать готовую таблицу из библиотеки django.contrib.auth.models. Модели данных приведены на рисунках 3.1.1 - 3.1.3

internshift > models.py > Company

```python
from django.db import models
from django.contrib.auth.models import User

class Company(models.Model):
    name = models.CharField(max_length=100)
    address = models.CharField(max_length=200)
    email = models.EmailField()
    description = models.TextField(max_length=300)

    def __str__(self):
        return self.name

class Vacancy(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
    additional = models.TextField(blank=True)
    link = models.URLField()
    company = models.ForeignKey(Company, on_delete=models.CASCADE)

    def __str__(self):
        return self.name
```

Рисунок 3.1.1

```python
class Skill(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()

    def __str__(self):
        return self.name

class SkillUser(models.Model):
    skill = models.ForeignKey(Skill, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    result = models.CharField(max_length=100)

    def __str__(self):
        return f"{self.user.username}'s test for {self.skill.name}"

class SkillVacancy(models.Model):
    skill = models.ForeignKey(Skill, on_delete=models.CASCADE)
    vacancy = models.ForeignKey(Vacancy, on_delete=models.CASCADE)
    need = models.IntegerField()

    def __str__(self):
        return str(self.id)
```

Рисунок 3.1.2

```python
class Trend(models.Model):
    skill = models.ForeignKey(Skill, on_delete=models.CASCADE)
    index = models.IntegerField()
    description = models.TextField()

    def __str__(self):
        return f"Trend for {self.skill.name}"

class Test(models.Model):
    skill = models.ForeignKey(Skill, on_delete=models.CASCADE)
    name = models.CharField(max_length=100)
    description = models.TextField(max_length=300)

    def __str__(self):
        return f"Test for {self.skill.name}"

class Question(models.Model):
    test = models.ForeignKey(Test, on_delete=models.CASCADE)
    first = models.CharField(max_length=100)
    second = models.CharField(max_length=100)
    third = models.CharField(max_length=100)
    fourth = models.CharField(max_length=100)
    question = models.CharField(max_length=100)
    image = models.CharField(max_length=200)
```

Рисунок 3.1.3

После реализации моделей данных используем команду makemigrations и migrate для того, чтобы создать таблицы в подключенной базе данных.

3.1.2 Реализация функций взаимодействия с вакансиями

Реализуем основные функции для взаимодействия с таблицей Vacancy, это будет: получение списка вакансий, получение вакансии по идентификатору изменение, удаление. Реализация функций приведена на рисунках 3.2.1 - 3.2.3.

```python
#Работа с вакансиями

# Получение списка всех вакансий
def vacancy_list(request):
    vacancies= Vacancy.objects.all()
    vacancy_json = [{'id': vacancy.id, 'name': vacancy.name, 'description': vacancy.description,
                     'additional': vacancy.additional, 'link': vacancy.link,
                     'company': vacancy.company.name} for vacancy in vacancies]
    return JsonResponse({'vacancies': vacancy_json})

# Получение вакансии по ID или удалить по ID
def get_or_delete_vacancy(request, vacancy_id):
    try:
        vacancy = Vacancy.objects.get(id=vacancy_id)
        if request.method == 'GET':
            vacancy_json = {'id': vacancy.id, 'name': vacancy.name, 'description': vacancy.description,
                            'additional': vacancy.additional, 'link': vacancy.link,
                            'company': vacancy.company.name}
            return JsonResponse(vacancy_json)
        elif request.method == 'DELETE':
            vacancy.delete()
            return JsonResponse({'message': 'Vacancy deleted successfully'})
    except Skill.DoesNotExist:
        return JsonResponse({'error': 'Vacancy not found'}, status=404)
```

Рисунок 3.2.1

```python
#Добавление вакансии
@require_POST
def add_vacancy(request):
    # Получение данных JSON из запроса
    data = json.loads(request.body)
    name = data.get('name')
    description = data.get('description')
    additional = data.get('additional')
    link = data.get('link')
    company_id = data.get('company')
    if company_id:
        try:
            company = Company.objects.get(id=company_id)
            vacancy = Vacancy.objects.create(name=name, description=description, additional=additional, link=link, company=company)
            return JsonResponse({'message': 'Vacancy created successfully'})
        except Company.DoesNotExist:
            return JsonResponse({'error': 'Company with specified id does not exist'}, status=400)
    else:
        return JsonResponse({'error': 'No company id provided in JSON'}, status=400)
```

Рисунок 3.2.2

```python
#Обновление вакансии
def update_vacancy(request, vacancy_id):
    data = json.loads(request.body)
    try:
        vacancy = Vacancy.objects.get(id=vacancy_id)
        if 'name' in data:
            vacancy.name = data['name']
        if 'description' in data:
            vacancy.description = data['description']
        if 'additional' in data:
            vacancy.additional = data['additional']
        if 'link' in data:
            vacancy.link = data['link']
        if 'company' in data:
            company_id = data['company']
            company = Company.objects.get(id=company_id)
            vacancy.company = company
        vacancy.save()
        return JsonResponse({'message': 'Vacancy updated successfully'})
    except Vacancy.DoesNotExist:
        # Если вакансия с указанным идентификатором не найдена, возвращаем ошибку
        return JsonResponse({'error': 'Vacancy with specified id does not exist'}, status=404)
    except Company.DoesNotExist:
        # Если компания с указанным id не найдена, возвращаем ошибку
        return JsonResponse({'error': 'Company with specified id does not exist'}, status=400)
```

Рисунок 3.2.3

Для создания зависимостей между вакансиями и необходимыми навыками, реализуем функцию, которая принимает на вход идентификатор навыка и вакансии, а также необходимое количество баллов. Реализация функции приведена на рисунке 3.2.4

```python
#Работа по соединению вакансии с навыками

def create_vacancy_skill(request):
    data = json.loads(request.body)
    vacancy = data.get("vacancy_id")
    skill = data.get("skill_id")
    need = data.get("need")
    try:
        vacancySkill = SkillVacancy.objects.get(vacancy=vacancy, skill=skill)
        return JsonResponse({'message': 'VacancySkill already exists'})
    except SkillVacancy.DoesNotExist:
        try:
            newVacancy = Vacancy.objects.get(id=vacancy)
            newSkill = Skill.objects.get(id=skill)
            vacancySkill = SkillVacancy.objects.create(vacancy=newVacancy, skill=newSkill, need=need)
            return JsonResponse({'vacancy': newVacancy.name, 'skill':newSkill.name, 'need':need})
        except Skill.DoesNotExist:
            return JsonResponse({'error': 'Skill not found'}, status=404)
        except Vacancy.DoesNotExist:
            return JsonResponse({'error': 'Vacancy not found'}, status=404)
```

Рисунок 3.2.4

3.1.3 Реализация функций взаимодействия с тестированием

Сначала реализуем основные функции работы с таблицей Skill. Нам понадобится методы для создания, изменения, удаления, а также получения списка, либо конкретного навыка по идентификатору. А также сделаем функцию для фильтрации навыков по названию. Реализация функций приведена на рисунках 3.3.1

```python
# Получение списка всех навыков
def skill_list(request):
    skills = Skill.objects.all()
    skills_json = [{'id': skill.id, 'name': skill.name, 'description': skill.description} for skill in skills]
    return JsonResponse({'skills': skills_json})

# Получение навыка по ID или удалить по ID
def get_or_delete_skill(request, skill_id):
    try:
        skill = Skill.objects.get(id=skill_id)
        if request.method == 'GET':
            skill_json = {'id': skill.id, 'name': skill.name, 'description': skill.description}
            return JsonResponse(skill_json)
        elif request.method == 'DELETE':
            skill.delete()
            return JsonResponse({'message': 'Skill deleted successfully'})

    except Skill.DoesNotExist:
        return JsonResponse({'error': 'Skill not found'}, status=404)

```

Рисунок 3.3.1

```python
#Добавление навыка
@require_POST
def add_skill(request):
    # Получение данных JSON из запроса
    data = json.loads(request.body)
    name = data.get('name')
    description = data.get('description')
    skill = Skill.objects.create(name=name, description=description)
    return JsonResponse({'id': skill.id, 'name': skill.name, 'description': skill.description})

#Фильтрация навыков
def filter_skills(request):
    json_data = json.loads(request.body)

    if 'name' in json_data:
        name_to_filter = json_data['name']
        filtered_skills = Skill.objects.filter(name=name_to_filter)
        filtered_skills_list = [{'name': skill.name, 'description': skill.description} for skill in filtered_skills]
        return JsonResponse({'skills': filtered_skills_list})
    else:
        return JsonResponse({'error': 'Key "name" is missing in JSON'}, status=400)
```

Рисунок 3.3.2

Для проверки и сохранения результатов тестирования мы будем получать на вход данные о пользователе и навыке, а также список выбранных ответов по каждому вопросу. Реализация функции для проверки тестирования приведена на рисунке 3.3.3.

```python
@require_POST
def check_test(request):
    json_data = json.loads(request.body)
    questions = json_data.get('questions', [])
    user_id = json_data.get('user_id')
    user, created = User.objects.get_or_create(username=user_id)
    if questions:
        results = []
        for question_data in questions:
            answer_to_check = question_data.get('answer')
            question_matches = Question.objects.get(id=question_data['question_id'])
            if question_matches.answer == answer_to_check:
                results.append({'question_id': question_data['question_id'], 'match': True})
            else:
                results.append({'question_id': question_data['question_id'], 'match': False})
        total = 0
        correct = 0
        for i in results:
            total += 1
            if i['match'] == True:
                correct += 1
        percent_correct = (correct / total) * 100
        normalized_score = int((percent_correct * 100) // 100)
        skill_id = json_data.get('skill_id')
        skill, _ = Skill.objects.get_or_create(id=skill_id)
        skill_user, _ = SkillUser.objects.get_or_create(skill=skill, user=user)
        skill_user.result = str(normalized_score)
        skill_user.save()

        return JsonResponse({'result': normalized_score})
    else:
        return JsonResponse({'error': 'No questions provided in JSON'}, status=400)
```

Рисунок 3.3.3

Пример JSON-файла для отправки на тестирование приведен на рисунке 3.3.4.

Рисунок 3.3.4

После проведения тестирования результаты будут создаваться, либо перезаписываться в таблице SkillUser. Таким образом мы будем хранить результаты о проведенном тестировании.

3.1.4 Реализация функций взаимодействия с компаниями

Реализуем основные функции для работы с таблицей компаний: получение списка компаний, получение конкретной компании по идентификатору, удаление, обновление. На рисунках приведены реализации функций 3.4.1 - 3.4.3.

```python
# Получение списка всех компаний
def company_list(request):
    companies= Company.objects.all()
    companies_json = [{'id': company.id, 'name': company.name, 'description': company.description,
                       'address': company.address, 'email': company.email} for company in companies]
    return JsonResponse({'companies': companies_json})

# Получение вакансии по ID или удалить по ID
def get_or_delete_company(request, company_id):
    try:
        company = Company.objects.get(id=company_id)
        if request.method == 'GET':
            company_json = {'id': company.id, 'name': company.name, 'description': company.description,
                            'address': company.address, 'email': company.email}
            return JsonResponse(company_json)
        elif request.method == 'DELETE':
            company.delete()
            return JsonResponse({'message': 'Company deleted successfully'})
    except Company.DoesNotExist:
        return JsonResponse({'error': 'Company not found'}, status=404)
```

Рисунок 3.4.1

```python
#Добавление компании
@require_POST
def add_company(request):
    # Получение данных JSON из запроса
    data = json.loads(request.body)
    name = data.get('name')
    description = data.get('description')
    address = data.get('address')
    email = data.get('email')
    try:
        company = Company.objects.get(name=name)
        return JsonResponse({'message': 'Company already exists'})
    except Company.DoesNotExist:
        company = Company.objects.create(name=name, description=description, address=address, email=email)
        return JsonResponse({'id': company.id, 'name': company.name,
                             'description': company.description, 'address': company.address, 'email': company.email})
```

Рисунок 3.4.2

```python
    #Обновление компании
    def update_company(request, company_id):
        data = json.loads(request.body)
        try:
            company = Company.objects.get(id=company_id)
            if 'name' in data:
                company.name = data['name']
            if 'description' in data:
                company.description = data['description']
            if 'address' in data:
                company.address = data['address']
            if 'email' in data:
                company.email = data['email']
            company.save()
            return JsonResponse({'message': 'Company updated successfully'})
        except Company.DoesNotExist:
            return JsonResponse({'error': 'Company with specified id does not exist'}, status=400)
```

Рисунок 3.4.3

3.1.5 Реализация функций взаимодействия с профилем

Для работы на странице профиля нам понадобится две функции: функция для выведения информации о пользователе, функция для вывода списка навыков пользователя. Список навыков формируется на основе пройденных пользователем тестов. Реализации функций приведена на рисунке 3.5.1.

```python
#Работа с профилем
def get_user(request, user_id):
    try:
        data = User.objects.get(id=user_id)
        user_json = {
            "email": data.email,
        }
        return JsonResponse(user_json)
    except User.DoesNotExist:
        return JsonResponse({'error': 'User with specified id does not exist'}, status=404)

def get_user_skills(request, user_id):
    try:
        user = User.objects.get(id=user_id)
        skill_users = SkillUser.objects.filter(user=user)
        skill_users_list = [[skill_user.skill.name, skill_user.result] for skill_user in skill_users]
        # Возвращаем список в формате JSON
        return JsonResponse({'skill_users': skill_users_list})
    except User.DoesNotExist:
        return JsonResponse({'error': 'User with specified id does not exist'}, status=404)
```

Рисунок 3.5.1

Пример вывода списка навыков пользователя приведен на рисунке 3.5.2.

Рисунок 3.5.2

3.1.6 Реализация функций для аутентификации пользователя

Для реализации аутентификации пользователя используем готовые функции фреймворка Django. Реализации функций приведены на рисунках 3.6.1 - 3.6.2

```
def login_view(request):
    if request.method == 'POST':
        # Получить данные JSON из запроса
        data = json.loads(request.body)
        username = data.get('login')
        password = data.get('password')
        # Аутентификация пользователя
        user = authenticate(request, username=username, password=password)
        if user is not None:
            # Вход пользователя
            login(request, user)
            return JsonResponse({'message': 'Login successful'})
        else:
            return JsonResponse({'message': 'Invalid credentials'}, status=401)
    else:
        return JsonResponse({'message': 'Method not allowed'}, status=405)
```

Рисунок 3.6.1

```
def register_view(request):
    if request.method == 'POST':
        # Получить данные JSON из запроса
        data = json.loads(request.body)
        username = data.get('login')
        password = data.get('password')
        email = data.get('email')
        # Проверка, что все данные присутствуют
        if not (username and password and email):
            return JsonResponse({'message': 'Missing required fields'}, status=400)
        # Создание нового пользователя
        user = User.objects.create_user(username=username, password=password, email=email)
        return JsonResponse({'message': 'Registration successful'})
    else:
        return JsonResponse({'message': 'Method not allowed'}, status=405)

def logout_view(request):
    logout(request)
    return JsonResponse({'message': 'Logged out successfully'})
```

Рисунок 3.6.2

3.2 Разработка клиентской части приложения

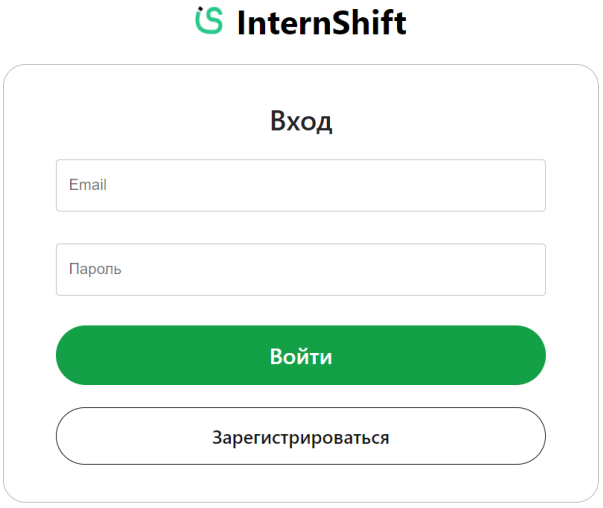

Рисунок 3.7 - Страница входа пользователя

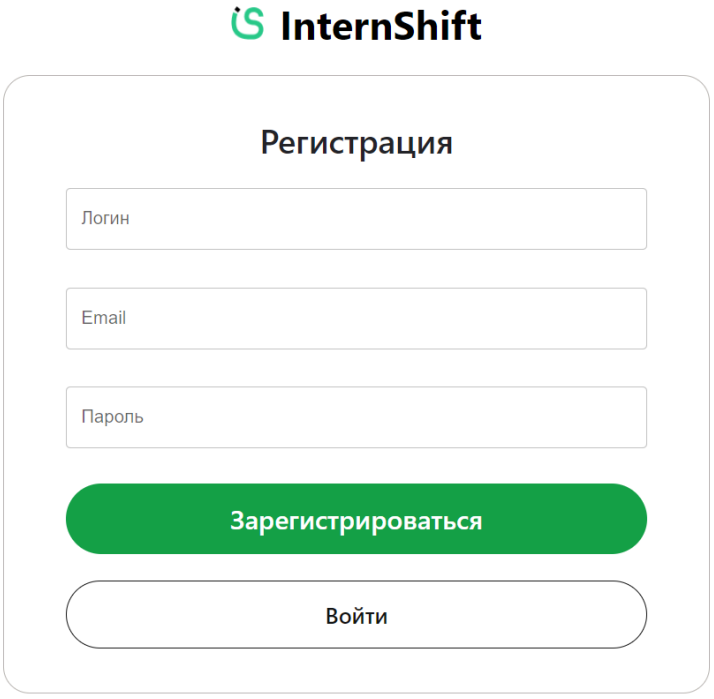

Рисунок 3.8 - Страница регистрации пользователя

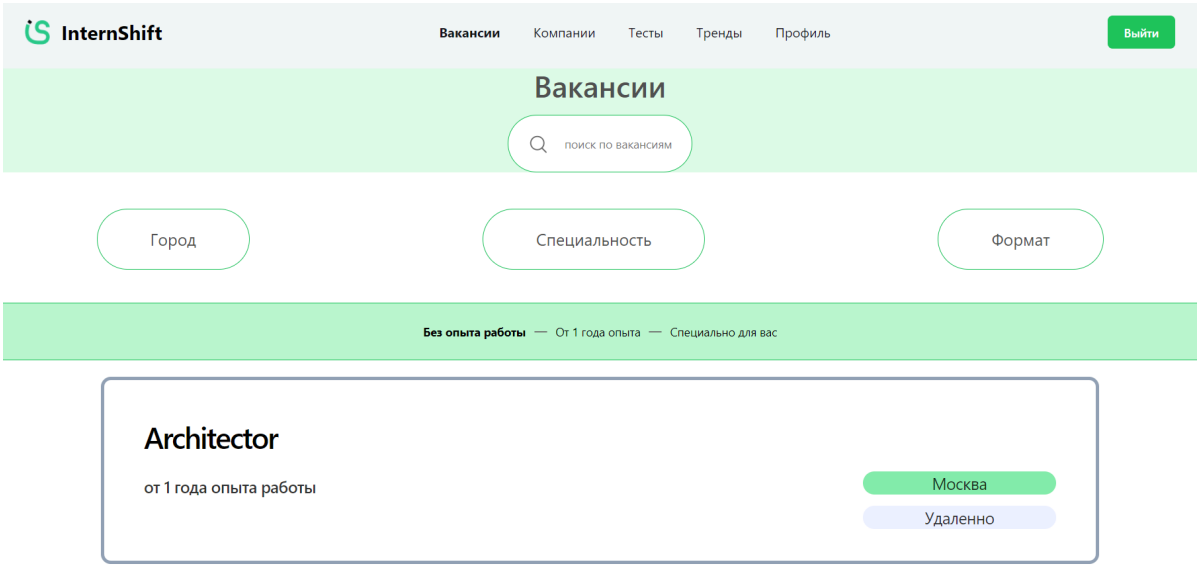

Рисунок 3.9 - Страница с вакансиями

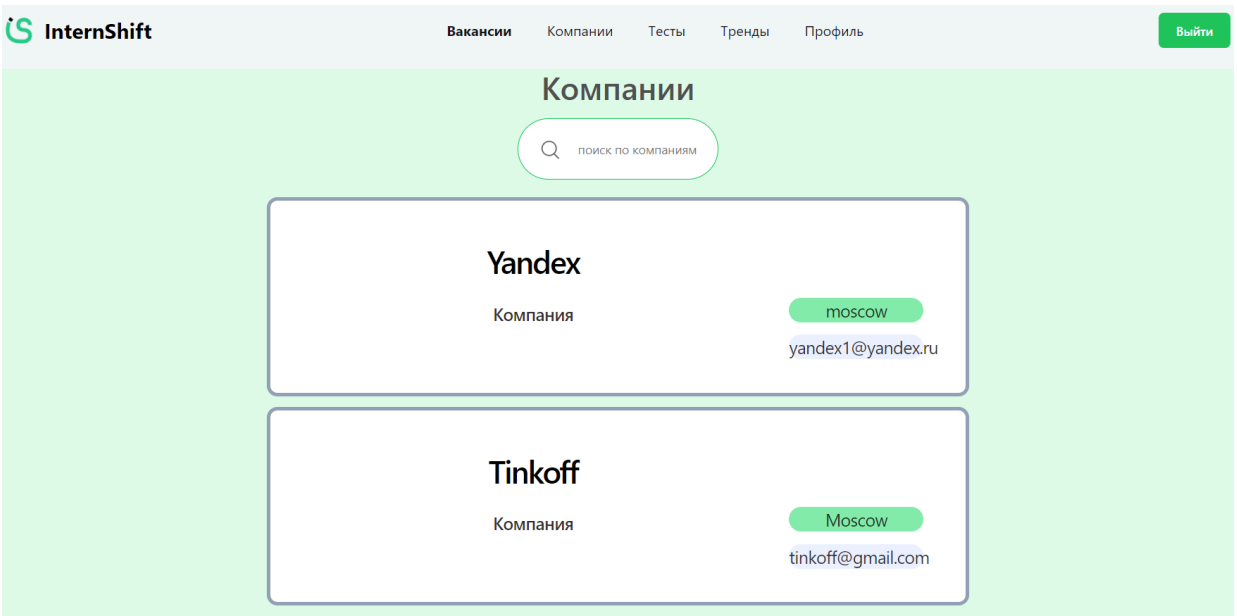

Рисунок 3.10 - Страница с компаниями

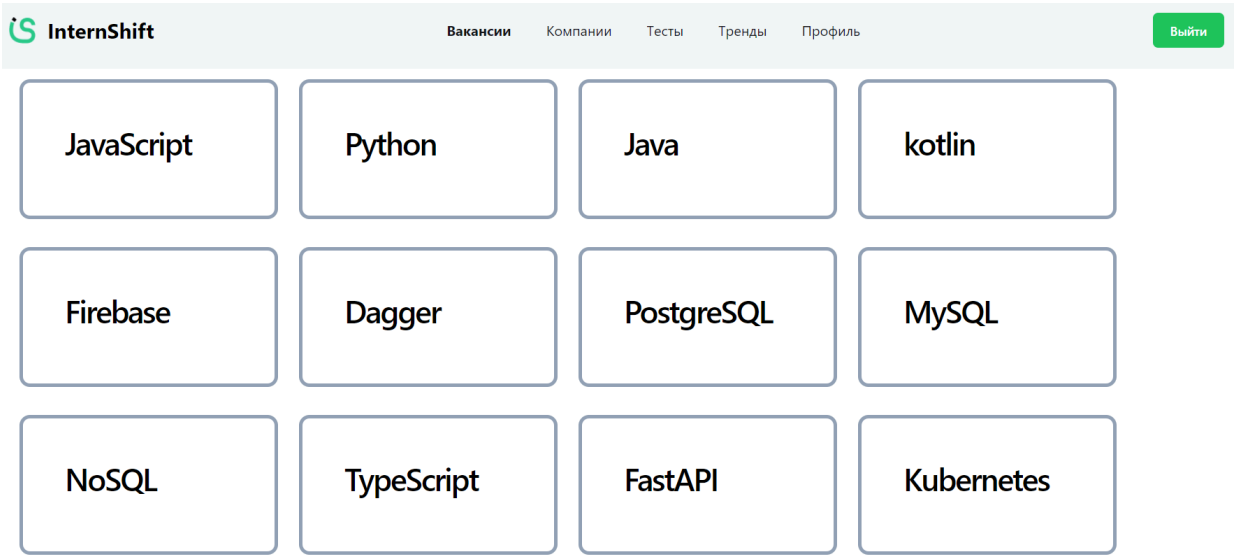

Рисунок 3.11 - Страница с тестами

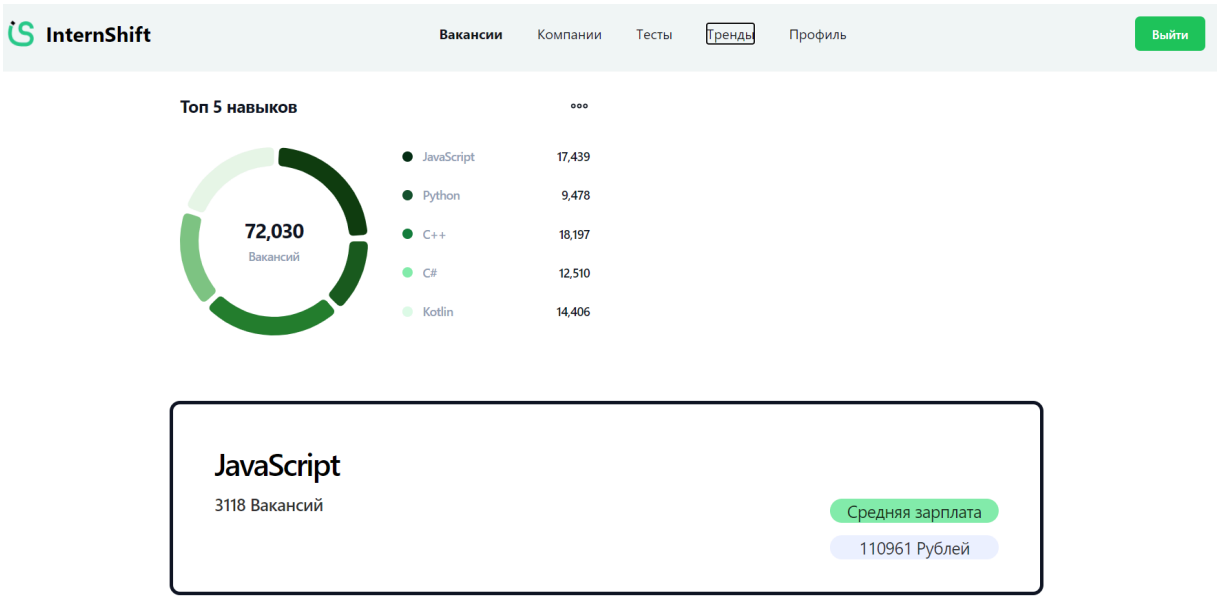

Рисунок 3.12 - Страница с трендами по вакансиям

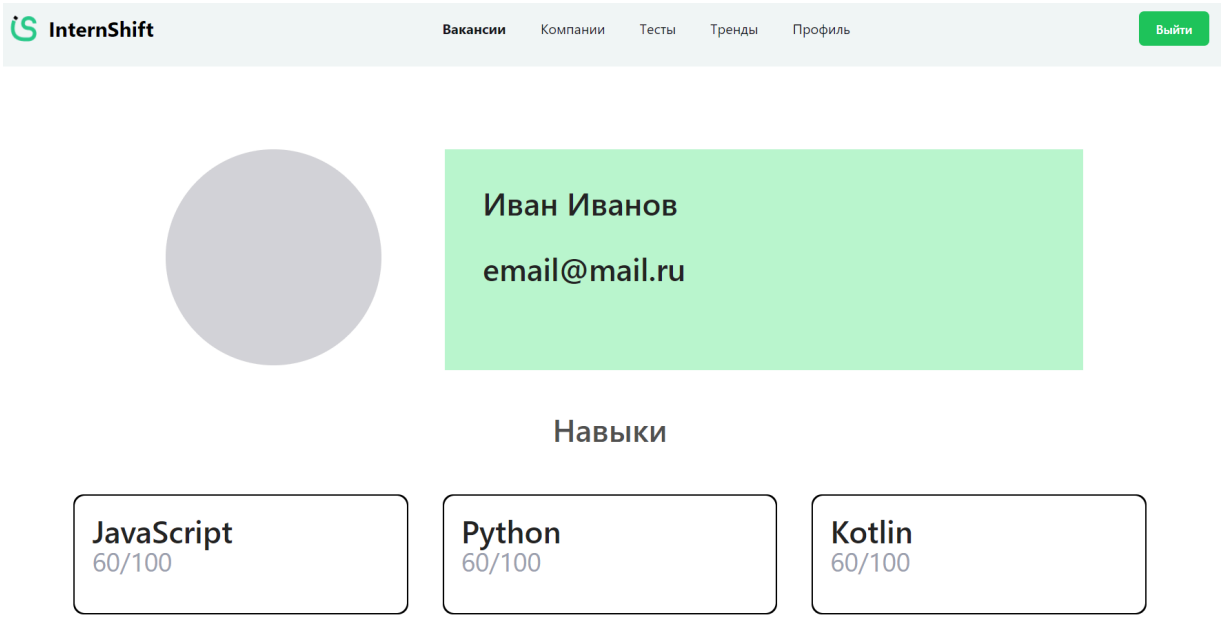

Рисунок 3.13 - Страница профиля пользователя

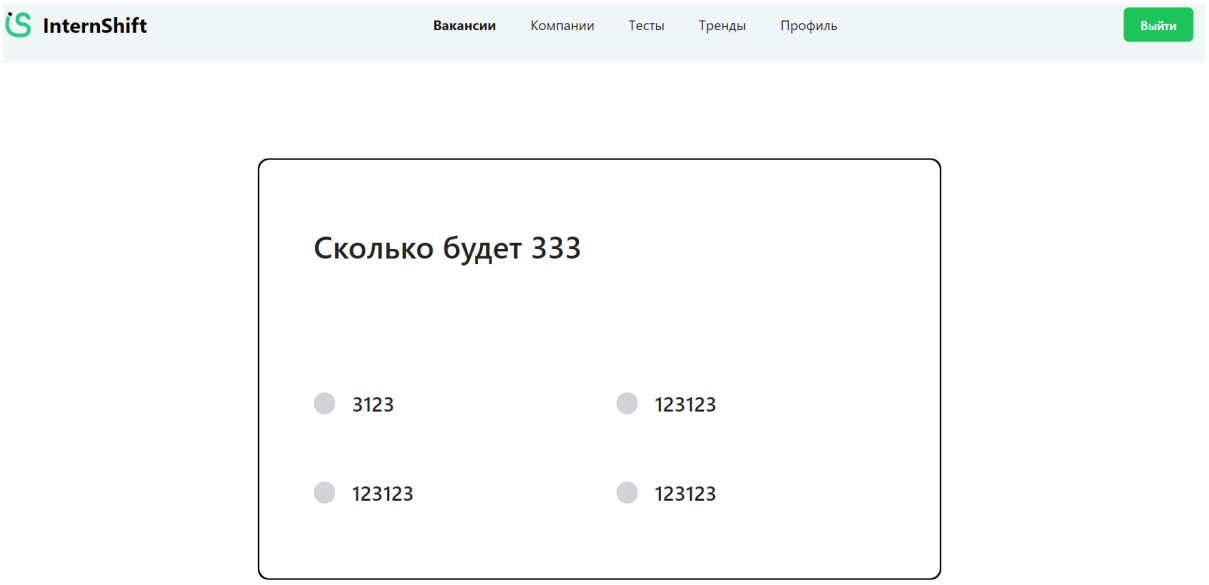

Рисунок 3.14 - Страница теста на знание технологии

3.3 Разработка сервиса для сбора и анализа данных с платформы HeadHunter

Для сбора данных и автоматического сбора вакансий с платформы HeadHunter был разработан скрипт для загрузки всех данных с API агрегатора. Далее была реализована фильтрация и отбор только нужных вакансий с ключевым словом “Стажировка”. Реализация функций приведена на рисунках 3.15 -

```python
def get_page(request: str, area_id: int, period,
             only_with_salary=False, order_by='relevance', page=0) -> tuple:

    params = {'text': request,
              'area': area_id,
              'page': page,
              'per_page': 100,
              'only_with_salary': only_with_salary,
              'order_by': order_by,
              'period': period}

    with get('https://api.hh.ru/vacancies', params=params) as r:
        json_object = r.json()
        return json_object['items'], json_object['found']
```

Рисунок 3.15

```python
def get_job_info(job):
    data, count = get_page(job, region, last_day)
    salary_sum = 0.0
    salary_count = 0
    for elem in data:
        if "salary" not in elem:
            continue
        if not elem["salary"]:
            continue
        salary = elem["salary"]["from"] or elem["salary"]["to"]
        try:
            salary_sum += salary * rates[elem["salary"]["currency"]]
            salary_count += 1
        except Exception:
            continue

    salary_avg = 0.0
    if salary_count:
        salary_avg = round(salary_sum / salary_count)
    return {"hh_salary": salary_avg, "hh_count": count}
```

Рисунок 3.16

```
def get_vacancies_for(job, reg):
    data, count = get_page(job, reg, last_day)
    jobs_json = []
    for elem in data:
        temp = {}
        if elem["name"]:
            temp["name"] = elem["name"]
        if elem["snippet"]["requirement"]:
            temp["description"] = elem["snippet"]["requirement"]
        if elem["schedule"]:
            temp["format"] = elem["schedule"]["name"]
        if elem["area"]:
            temp["town"] = elem["area"]["name"]
        if elem["experience"]["name"]:
            temp["additional"] = elem["experience"]["name"]
        if elem["employer"]["name"]:
            temp["company"] = elem["employer"]["name"]
        if elem["alternate_url"]:
            temp["link"] = elem["alternate_url"]
        jobs_json.append(temp)
    return jobs_json
```

Рисунок 3.17

3.4 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 был разработан скрипт для вывода статистики вакансий с веб-сайта hh.ru и автоматического заполнения таблиц базы данных с вакансиями и компаниями.

4 Тестирование и проведение экспериментальных исследований

Для проверки удобства и практического использования данного веб-приложения было проведено исследование. Целью исследования было определить удобство использования веб-приложения для поиска стажировок с тестами, оценить его эффективность в отборе вакансий и получить обратную связь от пользователей.

В исследовании участвовало 30 человек от 20 до 23 лет. При тестировании пользователи оценивали удобство использования веб-сайта, а также проходили тесты на знание технологий. После окончания тестирования пользователи проходили опрос с вопросами:

1. Использовали бы ли вы данный веб-сайт для поиска стажировок для различных компаний?
2. Какая функция удобнее относительно похожих сайтов?
3. Является ли функционал по отслеживанию навыков полезным?

По результатам проведения опроса, приведенных на рисунках 4.1 и 4.2, можно сказать, что данное приложение было бы полезным при поиске стажировок различных компаний.

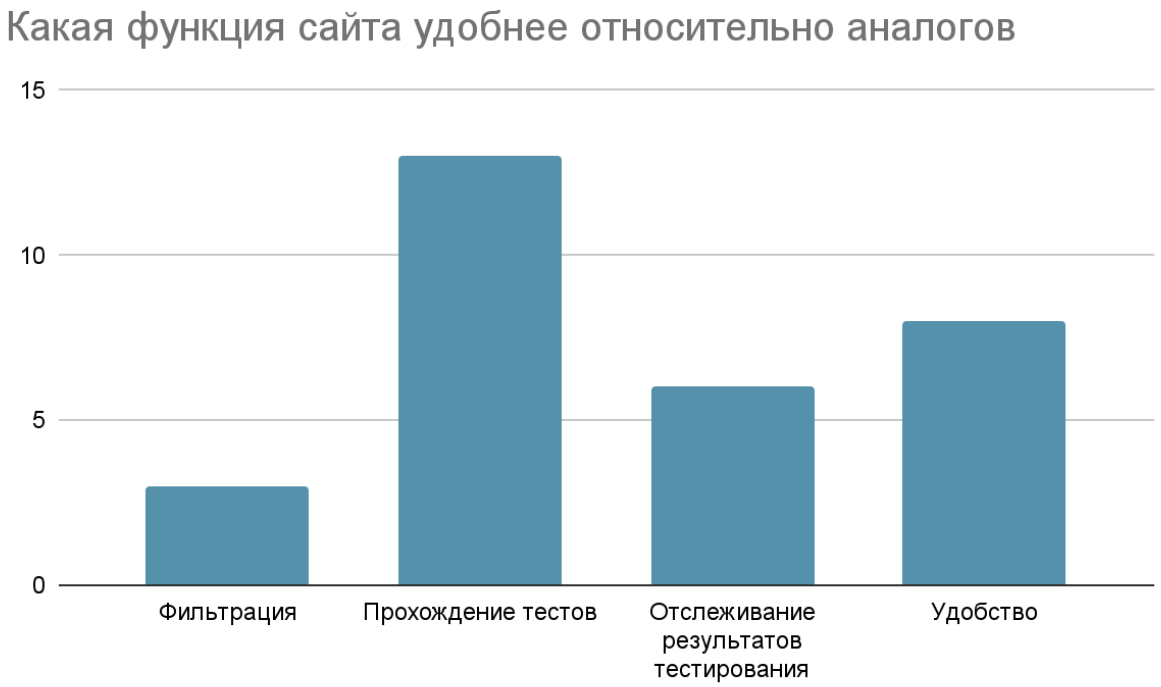

Рисунок 4.1

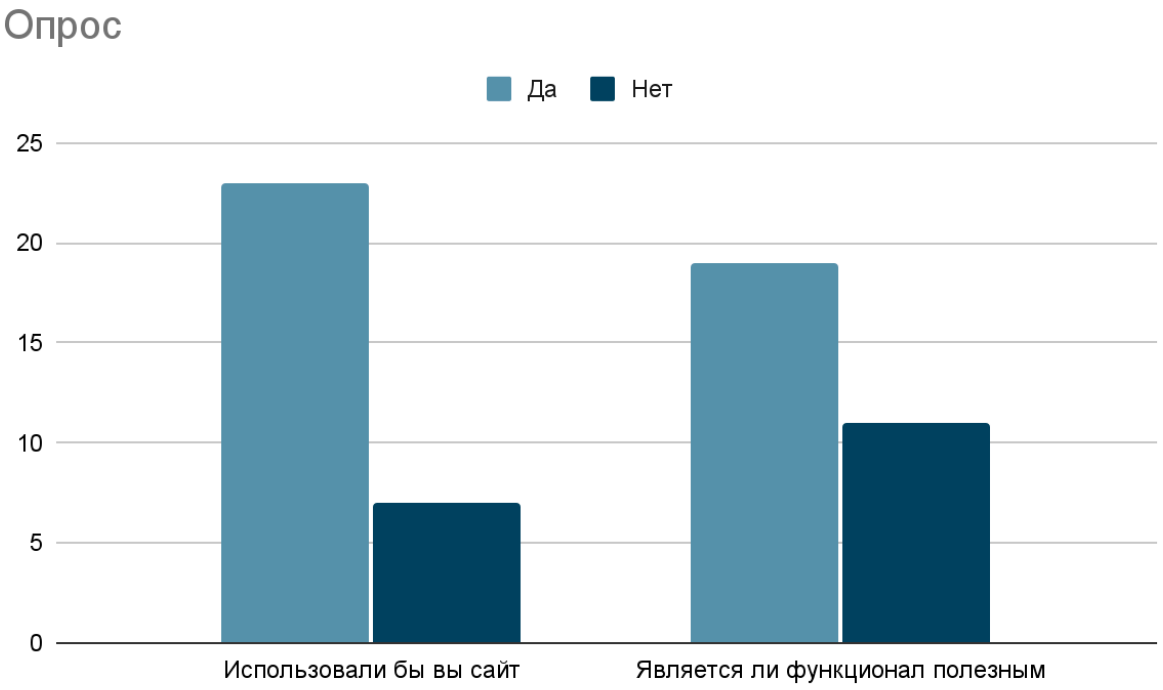

Рисунок 4.2

ЗАКЛЮЧЕНИЕ

В данной дипломной работе была разработана веб-платформа InternShift, предназначенная для оптимизации процесса поиска стажировок в сфере компьютерных технологий. Платформа предлагает пользователям удобный поиск вакансий по актуальным технологиям, возможность пройти тесты на знание выбранных технологий, а также получить персонализированные рекомендации по вакансиям, соответствующим их уровню знаний.

Проведенное исследование показало, что InternShift обладает следующими преимуществами:

- Удобный интерфейс: Пользователи отметили интуитивно понятный дизайн и простоту использования платформы.
- Актуальные вакансии: Платформа предоставляет доступ к огромному количеству стажировочных программ, в том числе от небольших компаний. Вакансии подгружаются с различных агрегаторов.
- Персонализация: Система позволяет пользователям проходить тесты на знание различных технологий и получать вакансии соответствующие уровню знаний пользователя.
- Повышение уверенности пользователя: Тесты помогают пользователям объективно оценивать свои шансы на прохождение собеседований по стажировкам, а также помогают выявить пробелы в знаниях, которые необходимо заполнить.

Результаты исследования подтвердили, что InternShift является перспективным инструментом для поиска стажировок, который может повысить шансы студентов на успешное трудоустройство.

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 увеличения количества пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ