МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

РАЗРАБОТКА САЙТА ДЛЯ ПОИСКА СПЕЦИАЛИСТОВ И ВАКАНСИЙ

Студент _____ *31.05.2024* Эсенова А.К.
(подпись, дата)

Руководитель _____ *31.05.2024* Зинкин С.А.
(подпись, дата)

Нормоконтролёр _____ *31.05.2024* А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от 27.05.2024 г. №11)

Заведующий кафедрой _____ М.А. Митрохин
(подпись)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 24.06.24 № 7)

Секретарь ГЭК _____ Е.И. Гурин
(подпись)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«19» 04 2024г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Эсенова Азра Кубанычбековна гр.20ВВВ1 факультета вычислительной техники.

2.Тема работы: «Разработка сайта для поиска специалистов и вакансий»

Тема утверждена приказом ПГУ №405/о от "19" апреля 2024 г.

3. Руководитель бакалаврской работы д.т.н., профессор, каф. ВТ ПГУ Зинкин С.А.

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

Разработать онлайн сервис для размещения заказов на те или иные виды работ, создание функциональной и удобной веб-платформы, которая обеспечит эффективное взаимодействие между работодателями и соискателями. Платформа должна предоставлять пользователям возможность быстро находить вакансии и специалистов, а также обеспечивать безопасное и надежное хранение данных.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить универсальные инструменты онлайн сервисов. Необходимо изучить альтернативные инструменты, площадки, на которых возможно базирование фриланс услуг и разработки онлайн сервисов с использованием таких технологий как Javascript.

6. Календарный график выполнения работы

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	32	26.04.2024 – 28.05.2024	
2	Изучение конкурентов	32	29.04.2024 – 30.05.2024	
3	Постановка задачи	30	1.04.2024 – 3.05.2024	
4	Выбор технологии разработки	40	4.05.2024 – 8.05.2024	
5	Проектирование и реализация базы данных	66	9.05.2024 – 15.05.2024	
6	Разработка основной части программы	70	16.05.2024 – 23.05.2024	
7	Оформление пояснительной записки	54	24.05.2024 – 31.05.2024	

Дата выдачи задания "26" апреля 2024 г.

Руководитель бакалаврской работы _____ 26.04.24 Зинкин С.А.
(подпись, дата)

Задание к исполнению принял студент _____ 26.04.24 Эсенова А.К.
(подпись, дата)

Работу к защите допустить

Декан факультета _____ 24.06.2024 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 73 л., 10 рис., 6 табл., 12 источн., 3 прил. ОНЛАЙН-СЕРВИС, ОБРАБОТКА ЗАКАЗОВ, ДОБАВЛЕНИЕ УСЛУГ, БАЗА ДАННЫХ, JAVASCRIPT, POSTGRESQL

Цель работы – разработка такого сайта, который бы предоставлял возможность эффективного поиска специалистов и вакансий. Для достижения данной цели необходимо решить ряд задач, включающих анализ существующих сайтов для поиска специалистов и вакансий, проектирование и разработку базы данных для сайта, разработку пользовательского интерфейса и реализацию поиска специалистов и вакансий на сайте.

Для достижения поставленной задачи является создание онлайн-сервиса который, обеспечит выполнение следующих функций: добавление заказов, оставление заявок.

В конце работы разработана онлайн-сервис, полностью соответсвующая цели данной выпускной квалификационной работы. В ходе достижения результата, были использовованы следующие технологии: javascript, node js, react js, postgresql.

Содержание

Введение 6

1 Разработка веб-сайта для поиска специалистов и вакансий на JavaScript 8

1.1 Анализ предметной области 8

1.1.1 Обзор аналогов 9

1.1.2 Анализ существующих решений 13

1.1.3 Анализ классификаций веб-сайтов 15

1.1.4 Анализ и структура сайта 18

1.1.5. Постановка задачи 19

1.1.6 Выбор инструментов разработки 21

1.1.7 Выбранные технологии 24

1.2 Реализация 26

1.2.1 Регистрация хостинга 26

1.2.2 Работа с базой данных 29

1.2.3 Описание доступных функций 35

1.2.4 Описание программы 41

2 Безопасность жизнедеятельности 47

2.1 Правила безопасности при использовании ПК 47

2.1.1 Правила безопасности при использовании смартфона 50

2.1.2 Физическая безопасность 50

2.1.3 Здоровье и эргономика 50

2.1.4 Электробезопасность 51

2.1.5 Защита данных и конфиденциальности 51

2.1.6 Безопасное использование интернета 51

Заключение 53

Список использованных источников 55

Приложение А– UML–диаграммы 56

Приложение А.1 – UML–диаграмма последовательности 57

Приложение А.2 – UML – диаграмма прецедентов 58

Приложение А.3 – Архитектура хостинга веб-приложения на AWS 59
Приложение А.4 – Структура функциональных возможностей платформы ... 60
Приложение Б – Листинг программы 61
Приложение В – Презентация 69

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ПК – персональный компьютер

СУБД – система управления базами данных

ЯП – язык программирования

ID – уникальный идентификатор объекта

IDE – интегрированная среда разработки

SQL – язык структурированных запросов для работы с базой данных

UML (Unified Modeling Language) – унифицированный язык моделирования используемый при проектировании

Введение

В последние годы рынок труда претерпевает значительные изменения, обусловленные глобализацией, цифровизацией и быстрым развитием информационных технологий. Все большее количество компаний и соискателей предпочитают использовать онлайн-платформы для поиска специалистов и вакансий. Этот тренд объясняется несколькими ключевыми факторами.

Рост числа удаленных и фриланс-работ способствует распространению высокоскоростного интернета и технологий удаленного доступа, что позволяет все большему количеству компаний переходить на удаленный формат работы. Это открывает новые возможности для сотрудничества между работодателями и специалистами, находящимися в разных географических точках, делая веб-сайты для поиска работы основным инструментом для нахождения удаленных сотрудников и фрилансеров. Современные технологии позволяют автоматизировать и ускорить процесс подбора персонала. Использование веб-платформ для поиска специалистов дает работодателям возможность быстро находить кандидатов, соответствующих их требованиям, и эффективно управлять процессом найма. Для соискателей это означает оперативный поиск вакансий и отклик на них.

Веб-сайты для поиска специалистов и вакансий предоставляют компаниям доступ к большому количеству кандидатов, что увеличивает вероятность нахождения наиболее подходящего сотрудника. Для специалистов такие платформы открывают доступ к большему количеству вакансий, включая международные предложения. Удобство и доступность веб-платформ для поиска работы и специалистов, которые доступны круглосуточно и с любого устройства, подключенного к интернету, делают процесс поиска работы и подбора персонала удобным и доступным для всех участников.

Целью данной дипломной работы является создание функциональной и удобной веб-платформы для поиска специалистов и вакансий, которая обеспечит эффективное взаимодействие между работодателями и соискателями, а также продемонстрирует возможности современных веб-технологий на основе JavaScript.

В ходе выполнения дипломной работы будут использованы следующие методы:

— анализ существующих решений и технологий в области веб-разработки и онлайн-платформ для поиска работы;

— проектирование архитектуры веб-сайта, включая определение основных функций и интерфейсов;

— разработка веб-приложения с использованием JavaScript, React, Node.js и других современных технологий;

— тестирование системы для выявления и устранения возможных ошибок и недочетов;

— оценка производительности и удобства использования созданного веб-сайта.

Практическая значимость данной работы заключается в создании веб-платформы, которая может быть использована работодателями для эффективного поиска специалистов, а соискателями – для оперативного поиска подходящих вакансий. Реализация проекта позволит улучшить процессы подбора персонала и трудоустройства, сократить время на поиск работы и специалистов, а также повысить удобство и доступность данных процессов. Кроме того, данная работа способствует развитию навыков и компетенций в области веб-разработки, что является важным для профессионального роста в сфере информационных технологий.

1 Разработка веб-сайта для поиска специалистов и вакансий на JavaScript

1.1 Анализ предметной области

В последние десятилетия рынок труда претерпел значительные изменения благодаря глобализации и цифровизации. Традиционные методы поиска работы и подбора персонала постепенно уступают место онлайн-платформам, которые предлагают более быстрые и эффективные решения для обеих сторон. В этой главе мы рассмотрим основные аспекты и особенности предметной области онлайн-платформ для поиска специалистов и вакансий.

Онлайн-платформы для поиска работы стали незаменимым инструментом для множества компаний и соискателей. Они позволяют работодателям размещать вакансии, а соискателям - искать подходящие предложения и подавать заявки. Эти платформы могут быть как универсальными, предлагая широкий спектр вакансий в разных отраслях, так и специализированными, ориентированными на определенные профессиональные области.

Современные технологии, такие как JavaScript, React и Node.js, значительно упрощают процесс разработки сложных и интерактивных веб-приложений. Эти инструменты позволяют создавать масштабируемые и эффективные платформы, которые могут обрабатывать большие объемы данных и обеспечивать высокую производительность. JavaScript является одним из самых популярных языков программирования для веб-разработки. Он позволяет создавать динамичные и интерактивные пользовательские интерфейсы. React, библиотека JavaScript, упрощает разработку компонентов пользовательского интерфейса, позволяя создавать сложные и отзывчивые веб-приложения. Node.js, платформа для выполнения JavaScript на серверной стороне, обеспечивает высокую производительность и масштабируемость серверных приложений.

1.1.1 Обзор аналогов

Основная цель создания веб-сайта для поиска специалистов и вакансий предоставить удобный и эффективный инструмент для взаимодействия между работодателями и соискателями. Платформа должна удовлетворять следующие потребности: обеспечивать простой и интуитивно понятный интерфейс для пользователей, поддерживать быструю и точную фильтрацию и поиск вакансий и специалистов, обеспечивать безопасность данных пользователей, а также поддерживать масштабируемость для обработки большого количества пользователей и данных. Для достижения этих целей необходимо тщательно проанализировать существующие решения и определить ключевые требования и функции, которые должны быть реализованы в проекте.

Для более полного понимания предметной области важно провести сравнительный анализ существующих решений. В таблице ниже приведен пример сравнения популярных онлайн-платформ для поиска работы.

Таблица 1 – Сравнительный анализ популярных онлайн-платформ для поиска работы

Платформа	Основные функции	Технологии	Особенности
LinkedIn	Профили, вакансии, сеть контактов, рекомендации	JavaScript, React	Профессиональная сеть
Indeed	Поиск вакансий, резюме, отзывы о компаниях	JavaScript, Node.js	Широкий охват вакансий
Glassdoor	Вакансии, отзывы о компаниях,	JavaScript, React	Отзывы о компаниях и зарплаты

Платформа	Основные функции	Технологии	Особенности
	зарплаты		
Monster	Вакансии, советы по карьере, инструменты для работодателей	JavaScript, Node.js	Поддержка карьерного роста

Анализ этих платформ показывает, что использование современных технологий, таких как JavaScript, React и Node.js, является общепринятой практикой для создания эффективных и масштабируемых онлайн-сервисов.

На основе анализа предметной области и существующих решений можно выделить следующие ключевые требования к платформе: удобство использования, эффективный поиск, безопасность данных, масштабируемость и интерактивность. Эти требования обусловлены потребностью в интуитивно понятном интерфейсе для пользователей, быстрой и точной фильтрации и поиске вакансий и специалистов, защите данных пользователей от несанкционированного доступа, поддержке большого количества пользователей и данных, а также динамичных и отзывчивых интерфейсах.

На основе этих требований можно определить основные функции, которые должны быть реализованы на платформе: регистрация и авторизация пользователей, возможность создания и управления профилями для работодателей и соискателей, создание и управление вакансиями, поиск вакансий и специалистов с помощью различных фильтров, автоматические рекомендации на основе профилей пользователей, механизмы связи и обмена сообщениями между работодателями и соискателями, а также отзывы и рейтинги работодателей и специалистов.

Создание веб-сайта для поиска специалистов и вакансий на JavaScript является актуальной и востребованной задачей. Она отвечает текущим потребностям рынка труда и способствует развитию цифровых компетенций и инноваций в области веб-разработки. Реализация данного проекта будет полезной как для работодателей, так и для соискателей, обеспечивая эффективное взаимодействие между ними и способствуя развитию современного рынка труда.

Проектирование и разработка такого веб-сайта требует тщательного анализа предметной области, выбора подходящих технологий и инструментов, а также реализации ключевых функций и требований. Этот процесс включает в себя несколько этапов: анализ существующих решений, проектирование архитектуры веб-сайта, выбор инструментов разработки, реализация веб-приложения с использованием JavaScript, React, Node.js и других современных технологий, тестирование системы для выявления и устранения возможных ошибок и недочетов, а также оценку производительности и удобства использования созданного веб-сайта.

Практическая значимость данного проекта заключается в создании веб-платформы, которая может быть использована работодателями для эффективного поиска специалистов, а соискателями - для оперативного поиска подходящих вакансий. Реализация проекта позволит улучшить процессы подбора персонала и трудоустройства, сократить время на поиск работы и специалистов, а также повысить удобство и доступность данных процессов. Кроме того, данная работа способствует развитию навыков и компетенций в области веб-разработки, что является важным для профессионального роста в сфере информационных технологий.

Структура взаимодействия компонентов представлена на рисунке 1.

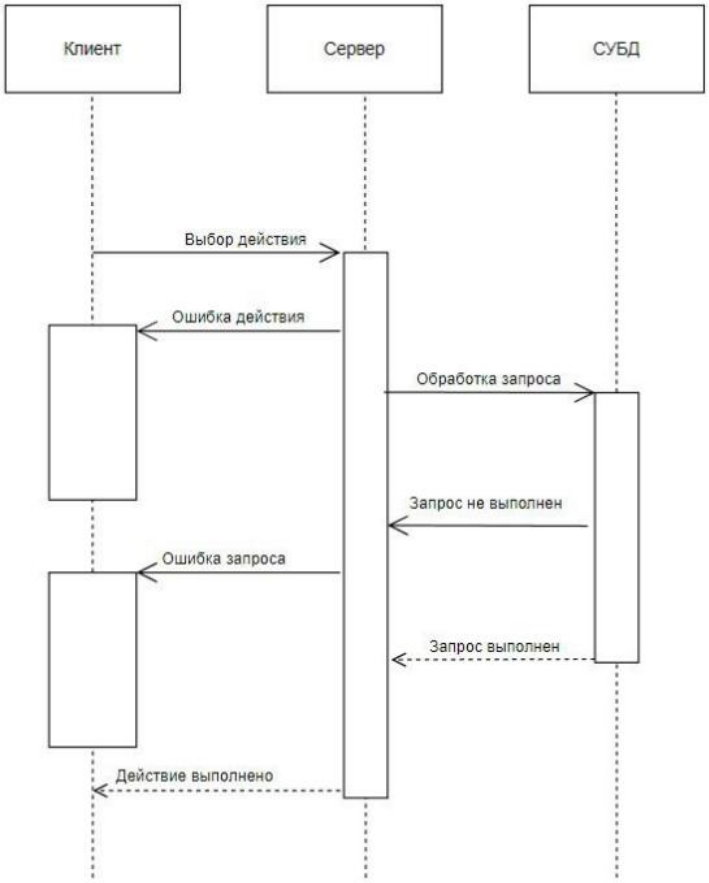


Рисунок 1 – UML-диаграмма последовательности

Для четкого понимания и отображения всех возможностей сайта, необходимо создать UML-диаграмму вариантов использования, которая представлена на рисунке 2.

Анализ существующих решений позволяет определить лучшие практики, недостатки и области для улучшения, которые могут быть использованы при разработке платформы для поиска специалистов и вакансий на JavaScript.

Существующие платформы можно разделить на две основные категории: универсальные платформы и специализированные платформы. Универсальные платформы охватывают широкий спектр отраслей и профессий, в то время как специализированные платформы фокусируются на конкретных профессиональных областях.

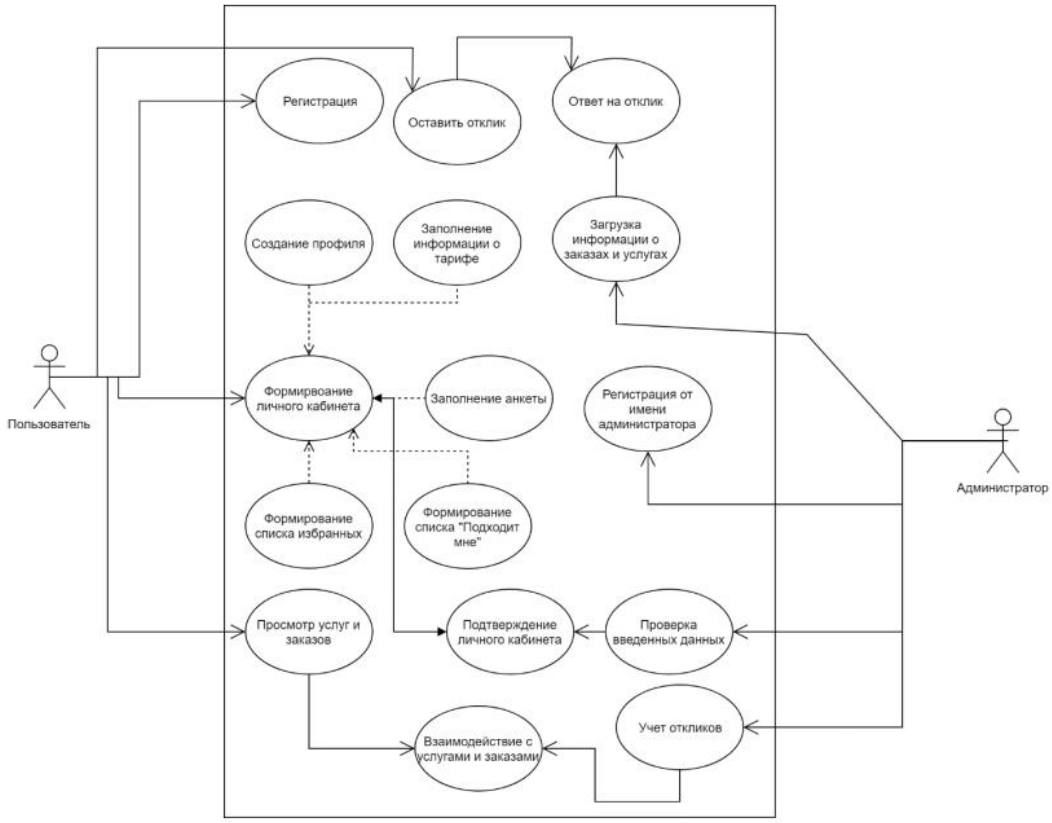


Рисунок 2 – UML-диаграмма вариантов использования

1.1.2 Анализ существующих решений

LinkedIn является одной из самых популярных профессиональных сетей. Она предоставляет пользователям возможность создавать детализированные профили, добавлять профессиональные контакты, искать вакансии и получать рекомендации. LinkedIn активно использует технологии JavaScript и React для создания интерактивного интерфейса и обеспечения быстрой работы. Основные преимущества LinkedIn включают большую сеть профессионалов, встроенные инструменты для коммуникации и возможность строить профессиональный бренд. Однако для некоторых пользователей интерфейс может казаться перегруженным, а доступ к некоторым функциям ограничен платными подписками.

Сравнительный анализ.

Для более наглядного представления основных характеристик и особенностей существующих платформ, приведем их в таблице 2.

Таблица 2. Сравнительный анализ существующих решений

Платформа	Основные функции	Технологии	Особенности	Недостатки
LinkedIn	Профили, вакансии, сеть контактов, рекомендации	JavaScript, React	Профессиональная сеть	Перегруженный интерфейс, платные подписки
Indeed	Поиск вакансий, резюме, отзывы о компаниях	JavaScript, Node.js	Широкий охват вакансий	Дубликаты вакансий
Glassdoor	Вакансии, отзывы о компаниях, зарплаты	JavaScript, React	Прозрачность отзывов о компаниях	Ограниченное количество отзывов
Monster	Вакансии, советы по карьере, инструменты для работодателей	JavaScript, Node.js	Поддержка карьерного роста	Устаревший интерфейс
AngelList	Вакансии, поиск финансирования, стартапы	JavaScript, React	Фокус на стартапах	Ограниченная область
Stack Overflow Jobs	Вакансии в IT, участие в сообществе	JavaScript, Node.js	Высококачественные вакансии	Ограниченная область вакансий

Анализ этих платформ показывает, что использование современных технологий, таких как JavaScript, React и Node.js, является общепринятой практикой для создания эффективных и масштабируемых онлайн-сервисов.

Основные функции и особенности таких платформ могут служить ориентиром для разработки новой платформы, учитывая их преимущества и недостатки.

1.1.3 Анализ классификаций веб-сайтов

Веб-сайты для поиска специалистов и вакансий можно классифицировать по различным признакам. Рассмотрение этих классификаций позволяет понять, какие типы сайтов существуют, каковы их особенности и каким образом они могут удовлетворять потребности пользователей. Основные классификации включают типы пользователей, модели взаимодействия, функциональные особенности и технологические платформы.

Первой классификацией можно считать деление по типам пользователей. Универсальные сайты ориентированы на широкий круг пользователей, включая представителей различных профессий и отраслей. Примеры таких платформ включают LinkedIn и Indeed, которые предлагают вакансии и возможности для всех категорий специалистов. Специализированные сайты, напротив, фокусируются на определенной профессиональной области или отрасли. Например, AngelList специализируется на стартапах и технологических компаниях, а Stack Overflow Jobs предназначен для специалистов в области программирования и IT.

Второй классификацией являются модели взаимодействия. Платформы B2C (Business to Consumer) связывают компании с индивидуальными соискателями. Примером такой платформы является Indeed, где компании размещают вакансии, а соискатели откликаются на них. Платформы B2B (Business to Business) предоставляют услуги для взаимодействия между компаниями. LinkedIn позволяет бизнесам не только нанимать специалистов, но и налаживать деловые контакты. Платформы C2C (Consumer to Consumer) менее распространены в контексте поиска работы, но могут включать

фриланс-биржи, где профессионалы предлагают свои услуги другим пользователям.

Третьей классификацией являются функциональные особенности. Сайты с объявлениями о вакансиях имеют основную функцию публикации и поиска объявлений о вакансиях. Примеры таких платформ включают Monster и Indeed. Социальные профессиональные сети позволяют пользователям создавать профили, налаживать профессиональные связи и участвовать в сообществах. LinkedIn является ярким примером такой платформы. Фриланс-биржи предоставляют платформы для фрилансеров, где клиенты размещают проекты, а фрилансеры подают заявки на их выполнение. Примеры включают Upwork и Freelancer. Отраслевые сайты ориентированы на определенные отрасли, такие как IT или медиа. Stack Overflow Jobs - пример отраслевого сайта для программистов.

Анализ классификаций веб-сайтов для поиска специалистов и вакансий показывает, что каждый тип платформ имеет свои преимущества и недостатки. Универсальные сайты предлагают широкий выбор вакансий для различных профессионалов, но могут быть перегружены информацией. Специализированные сайты предоставляют более таргетированные предложения, но имеют ограниченную аудиторию. Платформы B2C обеспечивают прямое взаимодействие между компаниями и соискателями, но могут сталкиваться с проблемой дублирования вакансий. Платформы B2B помогают налаживать деловые контакты, но меньше фокусируются на индивидуальных соискателях.

Фриланс-биржи предоставляют возможности для фрилансеров, но характеризуются высокой конкуренцией. Отраслевые сайты предлагают специфичные вакансии, но обладают узкой специализацией. Веб-платформы обеспечивают доступность через браузеры, но зависят от интернет-соединения. Мобильные приложения удобны в использовании, но могут

иметь ограниченный функционал. Гибридные платформы обеспечивают удобство и доступность, но требуют затрат на поддержку обеих версий.

Таблица ниже суммирует основные классификации веб-сайтов для поиска специалистов и вакансий:

Таблица 3. Классификация веб-сайтов для поиска специалистов и вакансий

Классификация	Типы сайтов	Примеры	Преимущества	Недостатки
Типы пользователей	Универсальные	LinkedIn, Indeed	Широкий выбор вакансий	Перегруженность информацией
	Специализированные	AngelList, Stack Overflow Jobs	Таргетированные предложения	Ограниченная аудитория
Модели взаимодействия	B2C	Indeed, Monster	Прямое взаимодействие	Возможны дубликаты вакансий
	B2B	LinkedIn	Налаживание деловых контактов	Меньше фокус на индивидуальных соискателях
	C2C	Upwork, Freelancer	Гибкость для фрилансеров	Риск низкого качества услуг
Функциональные особенности	Объявления о вакансиях	Indeed, Monster	Простой поиск вакансий	Ограниченные социальные функции
	Социальные профессиональные сети	LinkedIn	Профессиональные связи	Перегруженный интерфейс
	Фриланс-	Upwork	Возможности	Высокая

	биржи	, Freelancer	для фрилансеров	конкуренция
	Отраслевые	Stack Overflow Jobs	Специфичные вакансии	Узкая специализация
Технологические платформы	Веб-платформы	Indeed, Glassdoor	Доступность через браузеры	Зависимость от интернет-соединения
	Мобильные приложения	LinkedIn, Indeed	Удобство использования	Ограниченный функционал
	Гибридные платформы	LinkedIn, Indeed	Удобство и доступность	Затраты на поддержку обеих версий

Анализ классификаций веб-сайтов позволяет выделить ключевые аспекты, которые необходимо учитывать при разработке новой платформы для поиска специалистов и вакансий. Это поможет создать эффективное и удобное решение, отвечающее потребностям современных пользователей.

1.1.4 Анализ и структура сайта

Анализ и структура сайта для поиска специалистов и вакансий являются важными этапами в процессе разработки эффективной и функциональной платформы. В данной секции рассматриваются основные компоненты и архитектура сайта, а также ключевые элементы интерфейса, обеспечивающие удобство использования и удовлетворение потребностей пользователей.

Основные компоненты сайта:

— Главная страница. Главная страница является первым взаимодействием пользователя с сайтом. Она должна содержать краткое описание сервиса, ключевые функции и возможности сайта. На главной странице размещаются

ссылки на основные разделы сайта, такие как поиск вакансий, регистрация и авторизация.

1.1.5. Постановка задачи

Постановка задачи является важным этапом в разработке веб-сайта для поиска специалистов и вакансий на JavaScript. Она включает в себя определение целей и требований, а также описание основных функциональных возможностей и технических характеристик проекта. Постановка задачи помогает определить направления работы, необходимые ресурсы и критерии успеха.

Основная цель проекта — создание функциональной и удобной веб-платформы, которая обеспечит эффективное взаимодействие между работодателями и соискателями. Платформа должна предоставлять пользователям возможность быстро находить вакансии и специалистов, а также обеспечивать безопасное и надежное хранение данных.

Основные задачи проекта включают разработку интуитивно понятного интерфейса, создание эффективной системы поиска, реализацию регистрации и авторизации пользователей, создание профилей пользователей, внедрение системы отзывов и рейтингов, реализацию мессенджера для общения, интеграцию с внешними сервисами и обеспечение масштабируемости и производительности.

Разработка интуитивно понятного интерфейса включает создание удобного пользовательского интерфейса, который позволяет легко находить и откликаться на вакансии. Интерфейс должен обеспечивать простоту навигации и доступ к основным функциям сайта. Создание эффективной системы поиска предполагает разработку мощного поискового механизма с возможностью фильтрации по различным критериям, таким как местоположение, зарплата, опыт работы и навыки. Важно внедрить систему рекомендаций, которая будет предлагать пользователям подходящие вакансии на основе их профилей и предпочтений.

Реализация регистрации и авторизации пользователей включает обеспечение возможности регистрации новых пользователей и авторизации существующих. Необходимо внедрить защиту данных пользователей через использование шифрования и других методов безопасности. Создание профилей пользователей предполагает разработку системы профилей для соискателей и работодателей, включающей информацию о резюме, навыках, опыте работы и вакансиях. Профили должны быть легко настраиваемыми и редактируемыми.

Технические характеристики проекта включают использование JavaScript, React и других современных библиотек и фреймворков для разработки фронтенда. Основные задачи клиентской части – обеспечение интерактивного пользовательского интерфейса и взаимодействие с серверной частью через API. Серверная часть разрабатывается с использованием Node.js и Express, ее основные задачи – обработка запросов от клиентской части, взаимодействие с базой данных и обеспечение бизнес-логики приложения. Для хранения данных о пользователях, вакансиях, отзывах и другой информации можно использовать реляционные базы данных, такие как PostgreSQL, или нереляционные базы данных, такие как MongoDB, в зависимости от конкретных требований проекта. API интерфейсы для взаимодействия между клиентской и серверной частями могут быть реализованы с использованием RESTful API или GraphQL для обеспечения гибкости и расширяемости. Системы безопасности включают шифрование данных пользователей, аутентификацию и авторизацию через JWT (JSON Web Tokens), а также защиту от SQL-инъекций и XSS-атак. В инфраструктуре проекта могут быть использованы облачные сервисы, такие как AWS или Heroku, для хостинга, а также Docker для упрощения развертывания и масштабирования приложения.

Успех проекта можно оценить по следующим критериям: удовлетворенность пользователей, функциональность, производительность,

безопасность и масштабируемость. Платформа должна быть удобной и полезной для пользователей, что можно измерить через отзывы и рейтинги. Платформа должна обеспечивать все заявленные функции, включая регистрацию, авторизацию, поиск, систему рекомендаций, профили, отзывы и мессенджер. Сайт должен работать быстро и эффективно даже при большом количестве пользователей и данных. Данные пользователей должны быть надежно защищены от несанкционированного доступа. Платформа должна быть способна масштабироваться для обработки увеличивающегося объема данных и пользователей без значительных потерь в производительности.

Постановка задачи для разработки веб-сайта для поиска специалистов и вакансий на JavaScript определяет четкие цели, основные задачи и технические характеристики проекта. Она служит основой для дальнейшей работы над проектом, помогая определить направления работы, необходимые ресурсы и критерии успеха.

1.1.6 Выбор инструментов разработки

Выбор инструментов разработки является критически важным этапом при создании веб-сайта для поиска специалистов и вакансий на JavaScript. Правильный выбор инструментов и технологий позволяет не только ускорить процесс разработки, но и обеспечить надежность, масштабируемость и высокую производительность платформы. В данном разделе будут рассмотрены основные инструменты и технологии, которые планируется использовать в проекте.

Фронтенд.

— JavaScript: Основной язык программирования для разработки клиентской части веб-приложений. JavaScript позволяет создавать динамичные и интерактивные интерфейсы, обеспечивая высокий уровень отзывчивости и взаимодействия с пользователем.

— React: Популярная библиотека для создания пользовательских интерфейсов. React позволяет создавать компоненты, которые можно повторно использовать в различных частях приложения, что упрощает разработку и поддержку кода. React также обеспечивает высокую производительность за счет использования виртуального DOM.

— Redux: Библиотека для управления состоянием приложения. Redux используется в сочетании с React для обеспечения предсказуемости состояния приложения и упрощения его тестирования. Он помогает управлять состоянием в сложных приложениях, где данные могут передаваться между различными компонентами.

— Axios: Библиотека для выполнения HTTP-запросов. Axios позволяет легко взаимодействовать с серверной частью приложения, отправлять запросы и получать ответы. Она поддерживает промисы и имеет удобный API для работы с асинхронными запросами.

Бэкенд.

— Node.js: Платформа для выполнения JavaScript на серверной стороне. Node.js обеспечивает высокую производительность и масштабируемость за счет асинхронной обработки событий. Она отлично подходит для разработки серверных приложений, обрабатывающих большое количество одновременных запросов.

— Express: Фреймворк для Node.js, который упрощает создание веб-серверов и API. Express предоставляет легковесную и гибкую структуру, позволяющую быстро разрабатывать серверные приложения. Он поддерживает маршрутизацию, обработку запросов и ответы, а также интеграцию с различными базами данных.

— PostgreSQL: Реляционная база данных с открытым исходным кодом. PostgreSQL известна своей надежностью, производительностью и поддержкой стандартов SQL. Она обеспечивает хранение и управление

данными, необходимыми для работы приложения, включая информацию о пользователях, вакансиях и отзывах.

— MongoDB: Нереляционная база данных, использующая документно-ориентированную модель данных. MongoDB обеспечивает гибкость и масштабируемость, что делает ее подходящей для приложений, работающих с большими объемами неструктурированных данных. В проекте MongoDB может использоваться для хранения данных, которые не требуют жесткой схемы, таких как пользовательские профили и отзывы.

— Инструменты для разработки и деплоя:

— Git: Система управления версиями, которая позволяет отслеживать изменения в коде и управлять различными версиями проекта. Git обеспечивает совместную работу команды разработчиков, позволяет интегрировать изменения и быстро восстанавливать предыдущие версии кода при необходимости.

— GitHub: Платформа для хостинга и управления репозиториями Git. GitHub предоставляет инструменты для совместной работы над проектами, включая управление задачами, обзор кода и интеграцию с CI/CD (Continuous Integration/Continuous Deployment) системами.

— Docker: Платформа для контейнеризации приложений. Docker позволяет упаковать приложение и его зависимости в контейнеры, что обеспечивает консистентность среды выполнения и упрощает развертывание на различных платформах. Контейнеры Docker помогают масштабировать приложение и обеспечивают его изоляцию от других процессов.

— AWS: Облачная платформа, предоставляющая инфраструктуру и сервисы для разработки, деплоя и управления приложениями. AWS (Amazon Web Services) предлагает широкий спектр услуг, включая вычислительные ресурсы, базы данных, хранение данных и инструменты для мониторинга. AWS обеспечивает масштабируемость и высокую доступность приложения.

— Инструменты для обеспечения безопасности

— JWT (JSON Web Tokens): Стандарт для создания токенов доступа, используемых для аутентификации и авторизации. JWT позволяет безопасно передавать информацию между клиентом и сервером в виде зашифрованного токена. Токены JWT используются для контроля доступа к защищенным ресурсам и API.

— Helmet: Миддлвэр для Express, который помогает защитить приложение от веб-уязвимостей. Helmet добавляет различные заголовки безопасности в ответы сервера, что помогает предотвратить атаки, такие как XSS (Cross-Site Scripting) и Clickjacking.

— bcrypt: Библиотека для хэширования паролей. bcrypt используется для безопасного хранения паролей пользователей, обеспечивая защиту данных даже в случае утечки базы данных.

— Выбор инструментов разработки включает использование современных технологий, таких как JavaScript, React, Node.js и Express, для создания эффективной и масштабируемой платформы. Базы данных PostgreSQL и MongoDB обеспечивают хранение и управление данными. Инструменты для разработки, такие как Git, GitHub и Docker, упрощают управление проектом и развертывание приложения. Облачная платформа AWS предоставляет необходимые ресурсы для масштабируемости и высокой доступности. Инструменты безопасности, такие как JWT, Helmet и bcrypt, обеспечивают защиту данных пользователей и безопасность приложения.

1.1.7 Выбранные технологии

Выбор технологий для разработки веб-сайта играет ключевую роль в обеспечении его функциональности, производительности и масштабируемости. Основные выбранные технологии для разработки веб-сайта для поиска специалистов и вакансий на JavaScript включают в себя фронтенд, бэкенд, базы данных, инструменты для разработки и деплоя, а также технологии для обеспечения безопасности.

Для разработки фронтенда выбраны следующие технологии. JavaScript является основным языком программирования для создания интерактивных и динамичных веб-приложений. Он обеспечивает возможность создания отзывчивых пользовательских интерфейсов и взаимодействия с серверной частью через API. React - это популярная библиотека для создания пользовательских интерфейсов. Она позволяет создавать компоненты, которые можно повторно использовать в различных частях приложения, что упрощает разработку и поддержку кода.

Для разработки бэкенда выбраны следующие технологии. Node.js - это платформа для выполнения JavaScript на серверной стороне. Node.js обеспечивает высокую производительность и масштабируемость за счет асинхронной обработки событий. Он отлично подходит для создания серверных приложений, которые обрабатывают большое количество одновременных запросов. Express - это фреймворк для Node.js, который упрощает создание веб-серверов и API. Express предоставляет легковесную и гибкую структуру, позволяющую быстро разрабатывать серверные приложения. Он поддерживает маршрутизацию, обработку запросов и ответы, а также интеграцию с различными базами данных.

Для управления данными выбраны две базы данных. PostgreSQL - это реляционная база данных с открытым исходным кодом. Она известна своей надежностью, производительностью и поддержкой стандартов SQL. PostgreSQL обеспечивает хранение и управление структурированными данными, необходимыми для работы приложения, включая информацию о пользователях, вакансиях и отзывах. MongoDB - это нереляционная база данных, использующая документно-ориентированную модель данных. MongoDB обеспечивает гибкость и масштабируемость, что делает ее подходящей для приложений, работающих с большими объемами неструктурированных данных. В проекте MongoDB может использоваться

для хранения данных, которые не требуют жесткой схемы, таких как пользовательские профили и отзывы.

Для разработки и деплоя приложения выбраны следующие инструменты. Git - это система управления версиями, которая позволяет отслеживать изменения в коде и управлять различными версиями проекта. Git обеспечивает совместную работу команды разработчиков, позволяет интегрировать изменения и быстро восстанавливать предыдущие версии кода при необходимости. GitHub - это платформа для хостинга и управления репозиториями Git.

Выбор технологий включает использование современных инструментов, таких как JavaScript, React, Node.js и Express, для создания эффективной и масштабируемой платформы. Базы данных PostgreSQL и MongoDB обеспечивают надежное хранение и управление данными. Инструменты для разработки, такие как Git, GitHub и Docker, упрощают управление проектом и развертывание приложения. Облачная платформа AWS предоставляет необходимые ресурсы для обеспечения масштабируемости и высокой доступности. Инструменты безопасности, такие как JWT, Helmet и bcrypt, гарантируют защиту данных пользователей и безопасность приложения.

1.2 Реализация

1.2.1 Регистрация хостинга

Регистрация хостинга является важным этапом в реализации веб-сайта для поиска специалистов и вакансий. Выбор подходящего хостинг-провайдера и настройка инфраструктуры обеспечивают надежность, производительность и масштабируемость платформы. В этом разделе рассматриваются основные шаги по регистрации хостинга и настройке окружения.

Выбор хостинг-провайдера.

Для обеспечения высокой производительности и масштабируемости веб-сайта необходимо выбрать надежного хостинг-провайдера. На рынке существует множество провайдеров, предоставляющих облачные услуги. В данном проекте целесообразно рассмотреть использование облачных платформ, таких как Amazon Web Services (AWS), Google Cloud Platform (GCP) и Microsoft Azure. Облачные хостинг-провайдеры предлагают широкий спектр услуг, включая вычислительные ресурсы, базы данных, хранилища и инструменты для мониторинга.

Таблица 4. Сравнение облачных хостинг-провайдеров

Параметр	AWS	GCP	Microsoft Azure
Услуги	Вычисления, базы данных, хранение	Вычисления, базы данных, хранение	Вычисления, базы данных, хранение
Надежность	Высокая	Высокая	Высокая
Масштабируемость	Высокая	Высокая	Высокая
Стоимость	Средняя	Средняя	Средняя
Бесплатный тариф	Да (Free Tier)	Да	Да

Регистрация аккаунта.

— Процесс регистрации аккаунта на облачной платформе обычно включает следующие шаги:

— Переход на веб-сайт выбранного хостинг-провайдера (например, aws.amazon.com).

— Нажатие кнопки «Создать аккаунт» или «Sign Up».

— Ввод личных данных, таких как имя, адрес электронной почты и номер телефона.

— Выбор тарифного плана. Многие хостинг-провайдеры предлагают бесплатные или пробные тарифы, которые можно использовать на начальном этапе разработки.

— Подтверждение регистрации через электронную почту или SMS.

— Ввод платежной информации. Даже если выбран бесплатный тариф, провайдер может запросить данные кредитной карты для верификации.

— Настройка виртуальных машин.

После регистрации аккаунта необходимо настроить виртуальные машины (VM), которые будут использоваться для развертывания веб-приложения. Рассмотрим основные шаги для настройки виртуальных машин на примере AWS:

- создание экземпляра EC2:

1) Перейти в консоль управления AWS и выбрать сервис EC2 (Elastic Compute Cloud).

2) Нажать «Запустить экземпляр» (Launch Instance).

3) Выбрать Amazon Machine Image (AMI). Для данного проекта можно выбрать Ubuntu Server или Amazon Linux.

4) Выбрать тип экземпляра. На начальном этапе разработки подойдет тип t2.micro, который предоставляется бесплатно в рамках AWS Free Tier.

5) Настроить параметры экземпляра, такие как количество экземпляров, сеть и подстеть.

6) Настроить хранилище. Можно оставить настройки по умолчанию или изменить размер диска в зависимости от требований приложения.

7) Добавить теги для удобства управления ресурсами.

8) Настроить группы безопасности (Security Groups), которые контролируют доступ к экземпляру. Необходимо открыть порты 80 (HTTP) и 443 (HTTPS) для доступа к веб-сайту, а также порт 22 (SSH) для удаленного доступа.

9) Проверить и запустить экземпляр;

Регистрация хостинга и настройка инфраструктуры являются важными шагами в процессе реализации веб-сайта для поиска специалистов и вакансий. Правильный выбор хостинг-провайдера и настройка виртуальных машин, базы данных, домена и SSL-сертификата обеспечивают надежность, производительность и безопасность платформы. Рисунок 3 иллюстрирует архитектуру хостинга веб-приложения на AWS, включающую использование EC2 для вычислений, RDS для базы данных, S3 для хранения статических файлов, Route 53 для управления доменом и ACM для управления SSL-сертификатами.

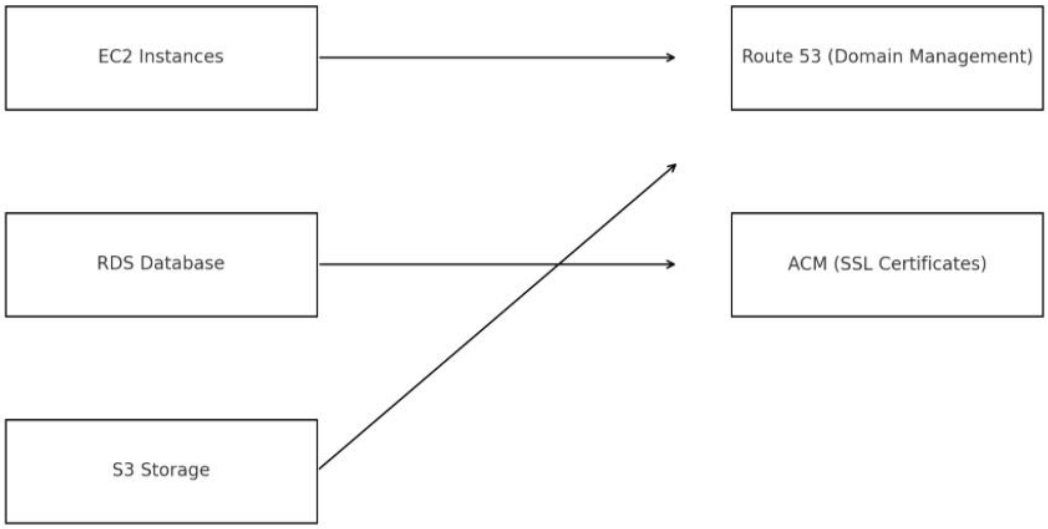


Рисунок 3 – Архитектура хостинга веб-приложения на AWS

1.2.2 Работа с базой данных

Работа с базой данных является ключевым компонентом разработки веб-сайта для поиска специалистов и вакансий. Эффективное управление данными позволяет обеспечивать надежное хранение и быстрый доступ к информации, необходимой для функционирования платформы. В этом разделе рассматриваются основные шаги по настройке и управлению базой данных для нашего сайта.

Для данного проекта важно выбрать базу данных, которая обеспечит надежность, масштабируемость и высокую производительность. В данном проекте используются две основные базы данных:

– PostgreSQL: Реляционная база данных с открытым исходным кодом, известная своей надежностью, производительностью и поддержкой стандартов SQL. PostgreSQL обеспечивает хранение и управление структурированными данными, такими как информация о пользователях, вакансиях и отзывах.

– MongoDB: Нереляционная база данных, использующая документно-ориентированную модель данных. MongoDB обеспечивает гибкость и масштабируемость, что делает ее подходящей для хранения данных, которые не требуют жесткой схемы, таких как пользовательские профили и отзывы.

Таблица 5. Сравнение баз данных PostgreSQL и MongoDB

Параметр	PostgreSQL	MongoDB
Тип базы данных	Реляционная	Нереляционная (NoSQL)
Модель данных	Табличная	Документно-ориентированная
Язык запросов	SQL	MongoDB Query Language
Структурированность данных	Высокая	Гибкая
Масштабируемость	Вертикальная и горизонтальная	Горизонтальная
Производительность	Высокая	Высокая при больших объемах данных
Поддержка транзакций	Полная	Ограниченная

Настройка базы данных PostgreSQL на AWS RDS.

Для настройки базы данных PostgreSQL на AWS RDS необходимо выполнить следующие шаги:

— Перейти в консоль управления AWS и выбрать сервис RDS (Relational Database Service).

— Нажать «Создать базу данных» (Create Database).

— Выбрать тип базы данных PostgreSQL и версию.

— Настроить параметры базы данных, такие как имя базы данных, имя пользователя и пароль.

— Настроить параметры развертывания, такие как тип инстанса, хранилище и сеть.

— Настроить параметры безопасности и доступности, такие как группы безопасности и режим Multi-AZ для обеспечения высокой доступности.

— Нажать «Создать базу данных» для завершения настройки.

— Настройка MongoDB на AWS.

— Для настройки MongoDB можно использовать сервисы, такие как MongoDB Atlas, который предоставляет управляемую облачную базу данных MongoDB:

— Перейти на веб-сайт MongoDB Atlas и зарегистрироваться.

— Создать новый кластер MongoDB, выбрав облачного провайдера (например, AWS) и регион.

— Настроить параметры кластера, такие как тип инстансов, размер и масштабируемость.

— Создать пользователя базы данных и настроить права доступа.

— Настроить сеть и параметры безопасности, добавив IP-адреса, с которых будет разрешен доступ к базе данных.

— Нажать «Создать кластер» для завершения настройки.

Управление данными.

После настройки базы данных необходимо организовать управление данными, включающее создание схем, индексов и обеспечение целостности

данных. Рассмотрим основные аспекты управления данными для PostgreSQL и MongoDB.

– Создание схемы данных в PostgreSQL

Создание схемы данных в PostgreSQL включает определение таблиц, полей и их типов, а также создание индексов и установление связей между таблицами. Пример создания таблиц и индексов для нашего сайта:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE jobs (
    id SERIAL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    company VARCHAR(100) NOT NULL,
    location VARCHAR(100),
    salary INTEGER,
    user_id INTEGER REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_users_username ON users (username);
CREATE INDEX idx_jobs_title ON jobs (title);
```

Управление данными в MongoDB.

В MongoDB данные хранятся в коллекциях в виде документов JSON. Пример создания коллекции и вставки документа для нашего сайта:

```
db.users.insertOne({
    username: «johndoe»,
```

```
    email: «johndoe@example.com»,
    password_hash: «hashed_password»,
    created_at: new Date()
});

db.jobs.insertOne({
    title: «Senior Developer»,
    description: «Responsible for developing high-quality applications.»,
    company: «Tech Corp»,
    location: «San Francisco, CA»,
    salary: 120000,
    user_id: ObjectId(«user_id»),
    created_at: new Date()
});
```

Взаимодействие с базой данных.

Для взаимодействия с базой данных необходимо реализовать API, которое будет обрабатывать запросы от клиентской части приложения. API должно поддерживать основные операции CRUD (Create, Read, Update, Delete) для управления данными. Пример API на Node.js и Express:

```
const express = require('express');
const bodyParser = require('body-parser');
const { Pool } = require('pg');
const { MongoClient } = require('mongodb');

const app = express();
app.use(bodyParser.json());

// PostgreSQL setup
const pool = new Pool({
  user: 'dbuser',
  host: 'database.server.com',
```

```
  database: 'mydb',
  password: 'secretpassword',
  port: 5432,
});

// MongoDB setup
const mongoClient = new MongoClient('mongodb://localhost:27017', { useNewUrlParser: true, useUnifiedTopology: true });
let mongoDb;
mongoClient.connect(err => {
  if (err) throw err;
  mongoDb = mongoClient.db('mydb');
});

// API endpoints for PostgreSQL
app.get('/users', async (req, res) => {
  const result = await pool.query('SELECT * FROM users');
  res.json(result.rows);
});

app.post('/users', async (req, res) => {
  const { username, email, password_hash } = req.body;
  const result = await pool.query('INSERT INTO users (username, email, password_hash) VALUES ($1, $2, $3) RETURNING *', [username, email, password_hash]);
  res.json(result.rows[0]);
});

// API endpoints for MongoDB
app.get('/profiles', async (req, res) => {
  const profiles = await mongoDb.collection('profiles').find().toArray();
  res.json(profiles);
});
```

```
app.post('/profiles', async (req, res) => {
  const profile = req.body;
  const result = await mongoDb.collection('profiles').insertOne(profile);
  res.json(result.ops[0]);
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Работа с базой данных включает выбор подходящих технологий, настройку базы данных, управление данными и реализацию API для взаимодействия с базой данных. PostgreSQL и MongoDB обеспечивают надежное и гибкое хранение данных, необходимых для функционирования веб-сайта для поиска специалистов и вакансий.

1.2.3 Описание доступных функций

Веб-сайт для поиска специалистов и вакансий должен обладать широким спектром функций, обеспечивающих удобство использования и удовлетворение потребностей как работодателей, так и соискателей. В этом разделе описываются основные функции, доступные на платформе, включая функции для работодателей и соискателей.

Основные функции для пользователей

1. Регистрация и авторизация:

– Регистрация: Пользователи могут создать учетную запись, указав основную информацию (имя, адрес электронной почты, пароль). Для работодателей также необходимо указать информацию о компании.

– Авторизация: Зарегистрированные пользователи могут войти в систему, используя адрес электронной почты и пароль.

– Восстановление пароля: Пользователи могут восстановить забытый пароль через электронную почту.

2. Профили пользователей:

– Создание и редактирование профиля: Соискатели могут добавлять и редактировать информацию о себе, включая резюме, навыки, опыт работы и образование. Работодатели могут добавлять информацию о компании, ее миссии и текущих вакансиях.

– Просмотр профилей: Пользователи могут просматривать профили других пользователей. Работодатели могут просматривать профили соискателей, а соискатели - профили компаний.

3. Поиск вакансий и специалистов:

– Поиск вакансий: Соискатели могут искать вакансии по различным критериям, включая должность, местоположение, зарплату, тип занятости и требуемые навыки.

– Поиск специалистов: Работодатели могут искать специалистов по навыкам, опыту работы и другим параметрам.

– Фильтры поиска: Пользователи могут применять фильтры для уточнения результатов поиска.

4. Система рекомендаций:

– Персонализированные рекомендации: Система предоставляет пользователям рекомендации на основе их профилей, предпочтений и предыдущей активности. Соискателям предлагаются вакансии, а работодателям - подходящие кандидаты.

5. Публикация и управление вакансиями:

– Создание вакансий: Работодатели могут создавать вакансии, указывая подробную информацию о должности, требованиях и условиях работы.

– Управление вакансиями: Работодатели могут редактировать, публиковать, приостанавливать и удалять вакансии.

– Просмотр откликов: Работодатели могут просматривать отклики соискателей на их вакансии и управлять ими.

6. Отклики на вакансии:

– Отклик на вакансию: Соискатели могут откликаться на интересующие их вакансии, отправляя свое резюме и сопроводительное письмо.

– Управление откликами: Соискатели могут управлять своими откликами, просматривать статус откликов и получать уведомления о новых откликах.

7. Мессенджер:

– Обмен сообщениями: Работодатели и соискатели могут общаться друг с другом через встроенный мессенджер. Возможность отправки текстовых сообщений и обмена файлами.

– Уведомления о сообщениях: Пользователи получают уведомления о новых сообщениях и могут настроить параметры уведомлений.

8. Система отзывов и рейтингов:

– Оставление отзывов: Соискатели могут оставлять отзывы о работодателях и наоборот. Отзывы могут содержать текстовые комментарии и оценки.

– Просмотр отзывов: Пользователи могут просматривать отзывы и рейтинги других пользователей, что помогает принимать более обоснованные решения.

Основные функции для администраторов.

1. Управление пользователями:

– Просмотр и управление профилями пользователей: Администраторы могут просматривать профили всех пользователей и управлять ими (блокировать, удалять и т.д.).

– Модерация контента: Администраторы могут проверять и модерировать опубликованные вакансии и профили на соответствие правилам платформы.

2. Аналитика и отчеты:

– Сбор и анализ данных: Система собирает данные о деятельности пользователей, таких как количество регистраций, откликов, просмотров и сообщений.

– Генерация отчетов: Администраторы могут генерировать отчеты для анализа эффективности работы платформы и принятия стратегических решений.

Рисунок 4 иллюстрирует структуру функциональных возможностей платформы, разделенных на функции для пользователей и администраторов. Пользователи могут регистрироваться, управлять профилями, искать вакансии и специалистов, обмениваться сообщениями и оставлять отзывы. Администраторы могут управлять пользователями, анализировать данные и управлять контентом платформы.

Рисунок 4 – Структура функциональных возможностей платформы

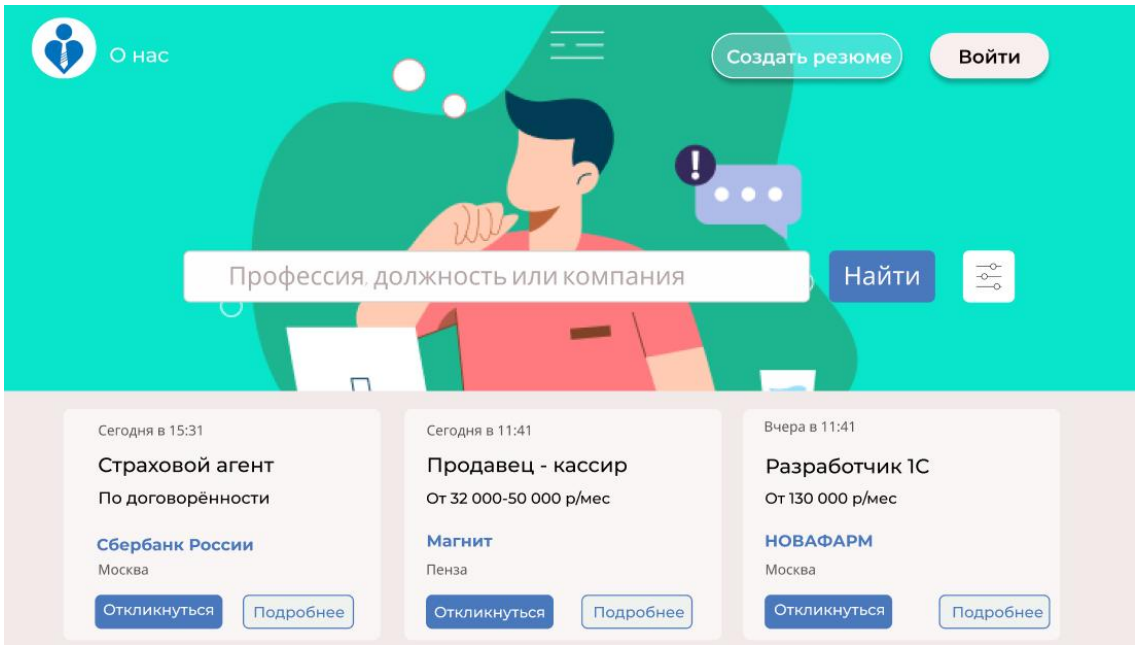


Рисунок 5 – Старт работы сайта

При выборе какой-либо услуги необходимо нажать на кнопку «Подробнее» (рисунок 6).

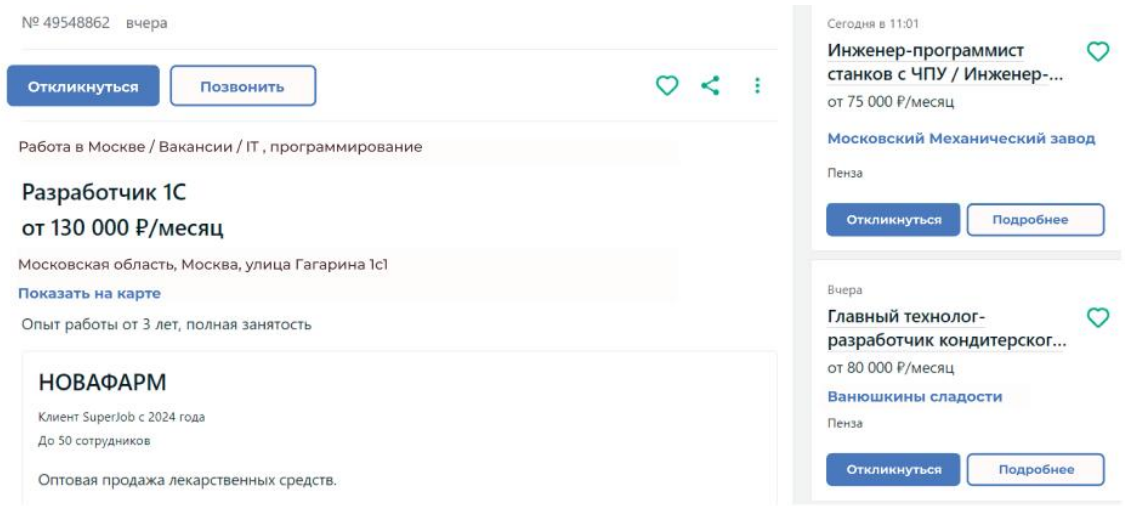


Рисунок 6 – Просмотр услуги

Для того чтобы откликнуться на услугу, необходимо нажать на соответствующую кнопку (рисунок 7).

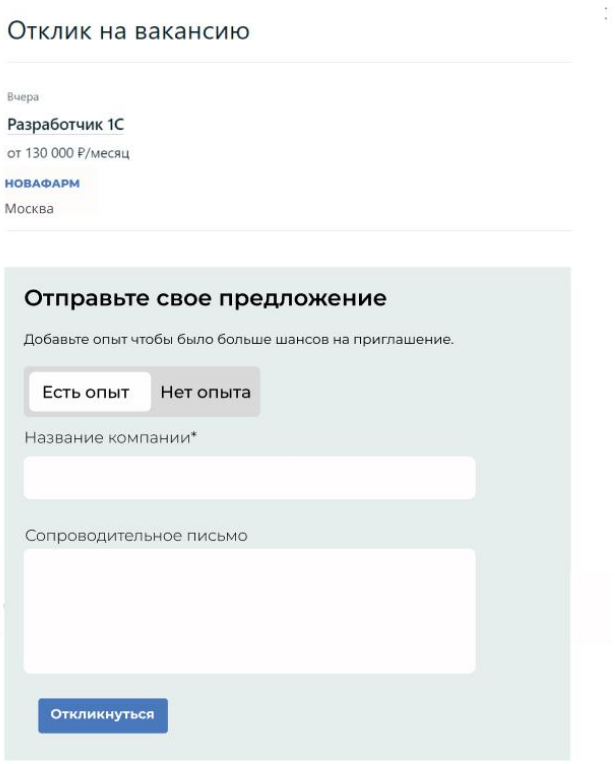


Рисунок 7– Отклик на вакансию

После заполнения данных о пользователе, необходимо нажать «Откликнуться» (рисунок 8)

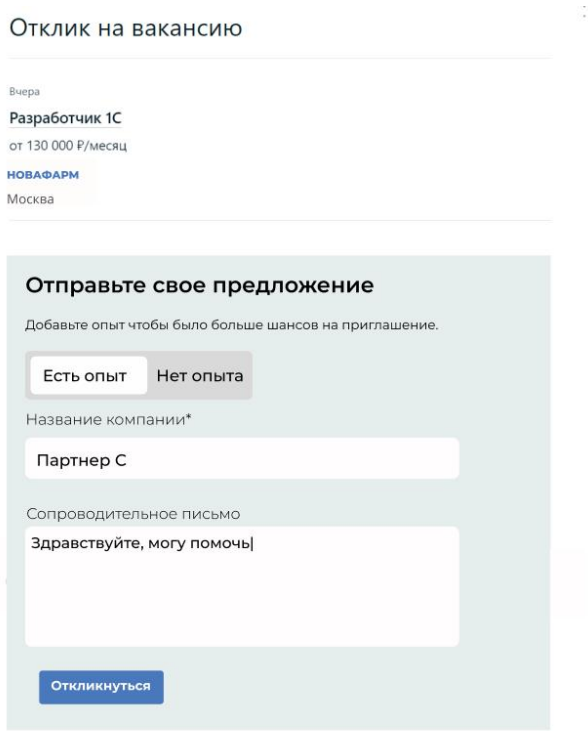


Рисунок 8 – Заполненное поле

Таблица 6. Ключевые функции платформы

Функция	Описание
Регистрация и авторизация	Создание учетных записей, вход в систему, восстановление пароля
Профили пользователей	Создание и редактирование профилей, просмотр профилей других пользователей
Поиск вакансий и специалистов	Поиск по различным критериям, применение фильтров
Система рекомендаций	Персонализированные рекомендации вакансий и специалистов
Публикация и управление вакансиями	Создание, редактирование, управление вакансиями, просмотр откликов
Отклики на вакансии	Отклики на вакансии, управление откликами
Мессенджер	Обмен сообщениями и файлами, уведомления о новых сообщениях
Система отзывов и рейтингов	Оставление и просмотр отзывов и рейтингов
Управление пользователями	Просмотр и управление профилями пользователей, модерация контента
Аналитика и отчеты	Сбор данных, генерация отчетов
Управление контентом	Редактирование статических страниц, публикация новостей и объявлений

Описание доступных функций веб-сайта для поиска специалистов и вакансий включает функции для пользователей и администраторов, обеспечивая удобство использования и эффективность работы платформы.

1.2.4 Описание программы

Программа для веб-сайта по поиску специалистов и вакансий реализована с использованием современных технологий и включает в себя клиентскую и серверную части, а также базы данных для хранения информации о пользователях, вакансиях и профилях. В этом разделе

рассматриваются основные компоненты программы, их функции и взаимодействие.

Программа имеет многослойную архитектуру, включающую следующие основные компоненты: клиентская часть (Frontend), серверная часть (Backend) и базы данных.

Пример кода для обработки запросов и взаимодействия с базой данных в Node.js и Express:

```
const express = require('express');
const bodyParser = require('body-parser');
const { Pool } = require('pg');
const { MongoClient } = require('mongodb');

const app = express();
app.use(bodyParser.json());

// PostgreSQL setup
const pool = new Pool({
  user: 'dbuser',
  host: 'database.server.com',
  database: 'mydb',
  password: 'secretpassword',
  port: 5432,
});

// MongoDB setup
const mongoClient = new MongoClient('mongodb://localhost:27017', { useNewUrlParser: true, useUnifiedTopology: true });
let mongoDb;
mongoClient.connect(err => {
  if (err) throw err;
  mongoDb = mongoClient.db('mydb');
```

```
});

// API endpoints for PostgreSQL
app.get('/users', async (req, res) => {
  const result = await pool.query('SELECT * FROM users');
  res.json(result.rows);
});

app.post('/users', async (req, res) => {
  const { username, email, password_hash } = req.body;
  const result = await pool.query('INSERT INTO users (username, email, password_hash) VALUES ($1, $2, $3) RETURNING *', [username, email, password_hash]);
  res.json(result.rows[0]);
});

// API endpoints for MongoDB
app.get('/profiles', async (req, res) => {
  const profiles = await mongoDb.collection('profiles').find().toArray();
  res.json(profiles);
});

app.post('/profiles', async (req, res) => {
  const profile = req.body;
  const result = await mongoDb.collection('profiles').insertOne(profile);
  res.json(result.ops[0]);
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Базы данных.

Для хранения данных используются PostgreSQL и MongoDB. PostgreSQL применяется для хранения структурированных данных, таких как пользователи и вакансии. MongoDB используется для хранения гибких данных, таких как профили и отзывы.

Структура базы данных PostgreSQL:

```
 CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE jobs (
    id SERIAL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    company VARCHAR(100) NOT NULL,
    location VARCHAR(100),
    salary INTEGER,
    user_id INTEGER REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_users_username ON users (username);
CREATE INDEX idx_jobs_title ON jobs (title);
```

Структура базы данных MongoDB:

```
db.users.insertOne({
    username: «johndoe»,
    email: «johndoe@example.com»,
```

```
        password_hash: «hashed_password»,
        created_at: new Date()
    });

    db.jobs.insertOne({
        title: «Senior Developer»,
        description: «Responsible for developing high-quality applications.»,
        company: «Tech Corp»,
        location: «San Francisco, CA»,
        salary: 120000,
        user_id: ObjectId(«user_id»),
        created_at: new Date()
    });
```

Взаимодействие с базой данных.

Для взаимодействия с базой данных необходимо реализовать API, которое будет обрабатывать запросы от клиентской части приложения. API должно поддерживать основные операции CRUD (Create, Read, Update, Delete) для управления данными. Пример API на Node.js и Express:

```
const express = require('express');
const bodyParser = require('body-parser');
const { Pool } = require('pg');
const { MongoClient } = require('mongodb');

const app = express();
app.use(bodyParser.json());

// PostgreSQL setup
const pool = new Pool({
  user: 'dbuser',
  host: 'database.server.com',
  database: 'mydb',
  password: 'secretpassword',
```

```
  port: 5432,
});

// MongoDB setup
const mongoClient = new MongoClient('mongodb://localhost:27017', { useNewUrlParser: true, useUnifiedTopology: true });
let mongoDb;
mongoClient.connect(err => {
  if (err) throw err;
  mongoDb = mongoClient.db('mydb');
});

// API endpoints for PostgreSQL
app.get('/users', async (req, res) => {
  const result = await pool.query('SELECT * FROM users');
  res.json(result.rows);
});

app.post('/users', async (req, res) => {
  const { username, email, password_hash } = req.body;
  const result = await pool.query('INSERT INTO users (username, email, password_hash) VALUES ($1, $2, $3) RETURNING *', [username, email, password_hash]);
  res.json(result.rows[0]);
});

// API endpoints for MongoDB
app.get('/profiles', async (req, res) => {
  const profiles = await mongoDb.collection('profiles').find().toArray();
  res.json(profiles);
});

app.post('/profiles', async (req, res) => {
```

```
    const profile = req.body;
    const result = await mongoDb.collection('profiles').insertOne(profile);
    res.json(result.ops[0]);
  });

  app.listen(3000, () => {
    console.log('Server is running on port 3000');
  });
```

Программа для веб-сайта по поиску специалистов и вакансий включает в себя клиентскую часть на основе React, серверную часть на основе Node.js и Express, а также базы данных PostgreSQL и MongoDB для надежного хранения данных. Клиентская часть отвечает за взаимодействие с пользователем и представление данных, серверная часть обрабатывает запросы и выполняет бизнес-логику, а базы данных обеспечивают надежное хранение и быстрый доступ к информации.

2 Безопасность жизнедеятельности

2.1 Правила безопасности при использовании ПК

Безопасность при использовании персонального компьютера (ПК) является важным аспектом как в личной, так и в профессиональной жизни. Следование правилам безопасности помогает предотвратить несчастные случаи, повысить производительность и сохранить здоровье пользователя. В этом разделе рассматриваются основные правила безопасности при работе с ПК.

Организация рабочего места играет ключевую роль в обеспечении комфортной и безопасной работы за ПК. Монитор должен быть расположен на уровне глаз на расстоянии 50-70 см от пользователя, чтобы

минимизировать напряжение глаз. Клавиатура и мышь должны находиться на удобной высоте, чтобы руки находились в естественном положении, предотвращая напряжение в запястьях. Важно использовать кресло с регулировкой по высоте и поддержкой поясницы для правильной осанки. Хорошее освещение рабочего места также важно для предотвращения утомляемости глаз. Использование естественного освещения предпочтительно, но при необходимости можно использовать искусственные источники света, избегая отражений на экране монитора.

На рисунке 9 изображена организация рабочего места, которая отвечает указанным требованиям.

Рабочее место с дисплеем должно быть эргономичным, то есть максимально удобным и не наносящим вреда здоровью.

При правильно организованном рабочем месте должна обеспечиваться правильная посадка пользователя ПЭВМ, изображённая на рисунке 10.

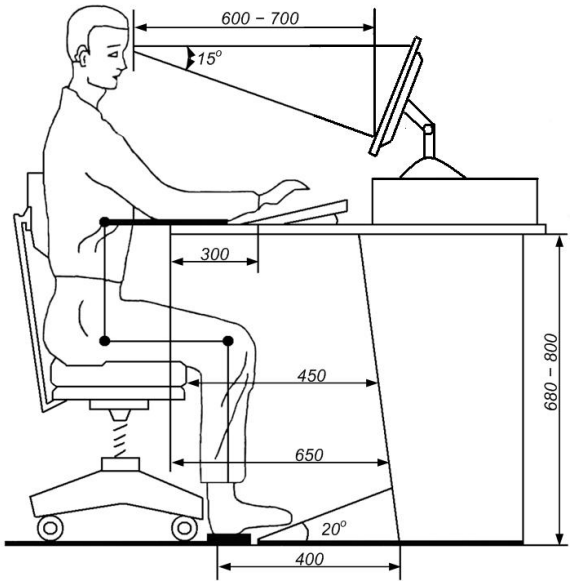


Рисунок 9 – Организация рабочего места

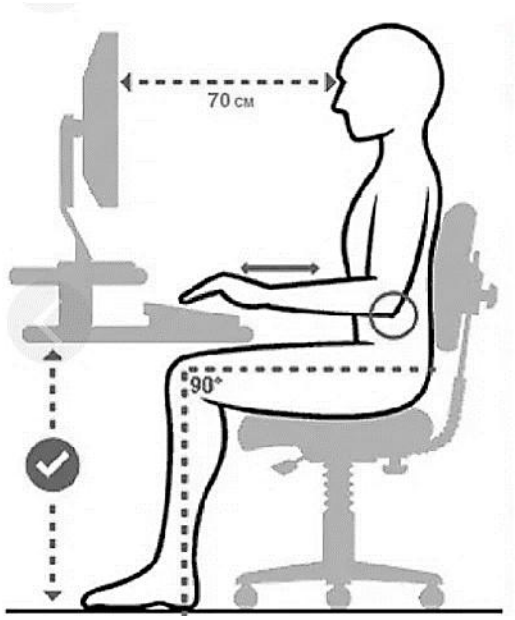


Рисунок 10– Правильная посадка за ПК

Регулярные перерывы во время работы за компьютером помогают снизить утомляемость глаз и мышц. Рекомендуется делать перерывы каждые 1-2 часа. Во время перерывов полезно выполнять простые упражнения для глаз и тела, чтобы улучшить кровообращение и снять напряжение. Поддержание чистоты рабочего места также важно для обеспечения здоровья и комфорта. Регулярно очищайте клавиатуру, мышь и экран монитора от пыли и загрязнений, используя специальные средства для очистки электроники.

Использование антивирусного программного обеспечения является важным шагом для защиты ПК от вирусов и вредоносных программ. Установите и регулярно обновляйте антивирусное ПО и периодически выполняйте полное сканирование системы на наличие угроз. Создание резервных копий важных данных на внешние носители или в облачные хранилища поможет предотвратить их потерю. Используйте надежные пароли и шифрование для защиты данных. Безопасное использование интернета включает избегание посещения подозрительных веб-сайтов и скачивания файлов из ненадежных источников. Для повышения уровня

безопасности учетных записей рекомендуется использовать двухфакторную аутентификацию.

Соблюдение правил безопасности при использовании ПК помогает предотвратить несчастные случаи, сохранить здоровье и повысить продуктивность работы. Важно регулярно напоминать себе и коллегам об этих правилах для поддержания безопасной рабочей среды.

2.1.1 Правила безопасности при использовании смартфона

Использование смартфона стало неотъемлемой частью повседневной жизни, как в личных, так и в профессиональных целях. Однако неправильное использование смартфона может привести к различным рискам для здоровья и безопасности. В этом разделе рассматриваются основные правила безопасности при использовании смартфона.

2.1.2 Физическая безопасность

Избегайте использования смартфона за рулем. Это может отвлекать внимание и привести к аварии. Лучше используйте гарнитуру или систему громкой связи для совершения звонков. Также берегите смартфон от падений и повреждений, используя защитные чехлы и стекла, чтобы минимизировать риск повреждения устройства при падении. Держите смартфон вдали от краев столов и других мест, откуда он может упасть.

2.1.3 Здоровье и эргономика

Соблюдайте правильную осанку при использовании смартфона. Не используйте устройство в согнутом положении или с наклоненной головой, чтобы избежать напряжения шеи и спины. Держите смартфон на уровне глаз, чтобы минимизировать нагрузку на шейный отдел позвоночника. Регулярно делайте перерывы каждые 20-30 минут, чтобы дать глазам отдохнуть и предотвратить зрительное напряжение. Во время перерывов полезно выполнять упражнения для рук и пальцев, чтобы предотвратить развитие туннельного синдрома. Регулируйте яркость экрана на комфортный уровень,

чтобы избежать перенапряжения глаз. Используйте режим ночного чтения или фильтр синего света в темное время суток.

2.1.4 Электробезопасность

Используйте только оригинальные или сертифицированные зарядные устройства для зарядки смартфона, чтобы избежать риска короткого замыкания или пожара. Не используйте поврежденные или изношенные кабели для зарядки. Заряжайте смартфон на твердой, непожароопасной поверхности вдали от горючих материалов. Не оставляйте смартфон на зарядке без присмотра на длительное время и не заряжайте его под подушкой или одеялом.

2.1.5 Защита данных и конфиденциальности

Используйте сложный пароль или биометрическую аутентификацию (отпечаток пальца, распознавание лица) для защиты доступа к смартфону. Не используйте простые и легко угадываемые пароли. Регулярно обновляйте операционную систему и приложения до последних версий, чтобы защитить смартфон от уязвимостей и вредоносного ПО. Включите автоматическое обновление, если такая функция доступна. Устанавливайте приложения только из надежных источников, таких как официальные магазины приложений (Google Play, App Store). Проверяйте разрешения, которые запрашивают приложения, и избегайте установки приложений, которые требуют избыточных прав доступа. Используйте встроенные функции шифрования для защиты данных на смартфоне. Включите шифрование SD-карты, если вы используете внешнюю память.

2.1.6 Безопасное использование интернета

Избегайте использования общественных Wi-Fi сетей для выполнения конфиденциальных операций, таких как интернет-банкинг. Используйте VPN для защиты данных при подключении к общественным сетям. Не открывайте подозрительные ссылки или вложения в электронных письмах и сообщениях.

Установите антивирусное ПО, чтобы сканировать загружаемые файлы на наличие вредоносного ПО.

Следование этим правилам безопасности при использовании смартфона поможет предотвратить риски для здоровья и безопасности, обеспечив комфортное и безопасное использование устройства.

Заключение

Разработка веб-сайта для поиска специалистов и вакансий на JavaScript является актуальной задачей, обусловленной современными тенденциями в сфере трудоустройства и веб-технологий. В данной работе были рассмотрены основные этапы создания такой платформы, начиная от анализа предметной области и выбора технологий, заканчивая реализацией и вопросами безопасности.

Анализ предметной области показал, что растущая глобализация, цифровизация и распространение удаленной работы создают необходимость в эффективных онлайн-платформах для поиска специалистов и вакансий. Эти платформы должны быть удобными, функциональными и безопасными, чтобы удовлетворять потребности как работодателей, так и соискателей.

Выбор инструментов разработки основывался на использовании современных технологий, таких как JavaScript, React, Node.js и Express. Эти технологии обеспечивают высокую производительность, масштабируемость и гибкость веб-приложения. PostgreSQL и MongoDB были выбраны в качестве баз данных для хранения структурированных и неструктурированных данных соответственно.

Реализация проекта включала разработку клиентской и серверной частей, настройку баз данных, обеспечение безопасности данных и создание удобного пользовательского интерфейса. Важным аспектом работы стало соблюдение правил безопасности при использовании ПК и смартфона, что способствует защите здоровья и данных пользователей.

В процессе разработки были описаны основные функции платформы, такие как регистрация и авторизация пользователей, создание и редактирование профилей, поиск вакансий и специалистов, система рекомендаций, публикация и управление вакансиями, отклики на вакансии, мессенджер и система отзывов и рейтингов. Эти функции обеспечивают

всестороннее взаимодействие между работодателями и соискателями, способствуя эффективному процессу подбора персонала и трудоустройства.

Кроме того, особое внимание было уделено вопросам безопасности данных, включая использование паролей, биометрической аутентификации, антивирусного ПО, шифрования данных и безопасного использования интернета. Соблюдение этих правил помогает предотвратить риски, связанные с потерей данных и вредоносными атаками.

В заключение можно отметить, что разработанный веб-сайт для поиска специалистов и вакансий на JavaScript отвечает современным требованиям и обеспечивает функциональность, удобство и безопасность для всех участников процесса. Реализация данного проекта способствует развитию цифровых компетенций, улучшению процессов подбора персонала и трудоустройства, а также поддержке инноваций в области веб-разработки.

Список использованных источников

1. Айзекс А. Dynamic HTML BHV-Санкт-Петербург 1998 [Электронный ресурс]
2. Ганчаров А. Самоучитель HTML. Питер 2020. [Электронный ресурс]
3. Дарнелл Р. HTML 4 Энциклопедия пользователя ДиаСофт 1999
4. Денисов Internet Explorer 5 Справочник Питер 1999
5. Дюк Дж. PHP и MySQL для начинающих / Дж. Дюк. - СПб.: Питер, 2016.
6. Зеланд Ю.В. MySQL: подробное руководство / Ю.В. Зеланд. - М.: Диалектика / Вильямс, 2015.
7. Кингсли-Хью Э., JavaScript: учебный курс. – СПб.: Питер, 2022
8. Петюшкин А.В., HTML. Экспресс-курс. – СПб.: БХВ - Петербург, 2003
9. Уэллс Л. WordPress для профессионалов / Л. Уэллс. - М.: Диалектика / Вильямс, 2017.
10. Харгрейв Д. WordPress: разработка плагинов / Д. Харгрейв. - СПб.: Питер, 2019.
11. Холм Я., Оулстенсет Р. WordPress. Профессиональная разработка веб-сайтов / Я. Холм, Р. Оулстенсет. - М.: ДМК Пресс, 2018.
12. Хоумер А. Dynamic HTML Справочник Питер 1999

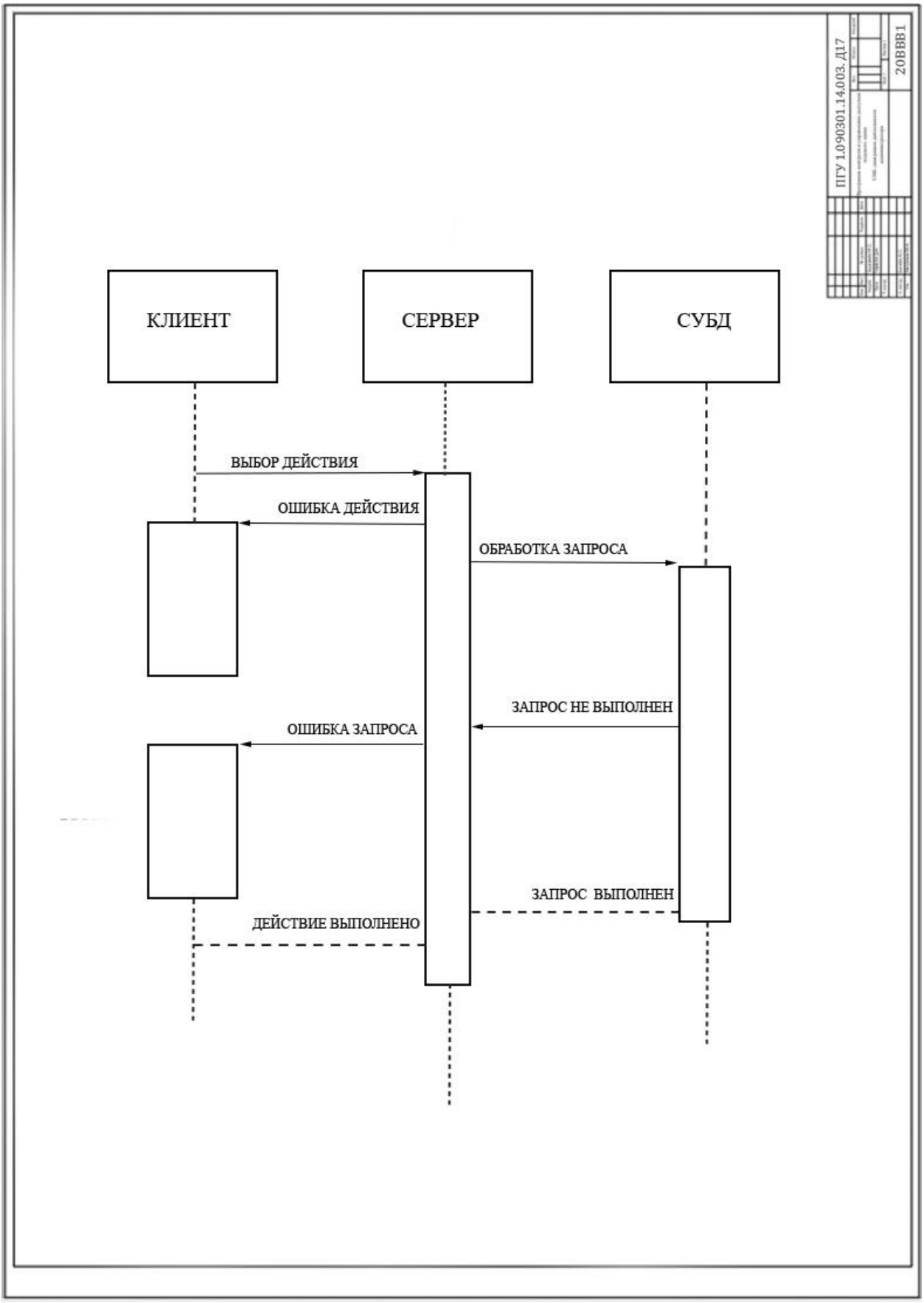
Приложение А

(Обязательное)

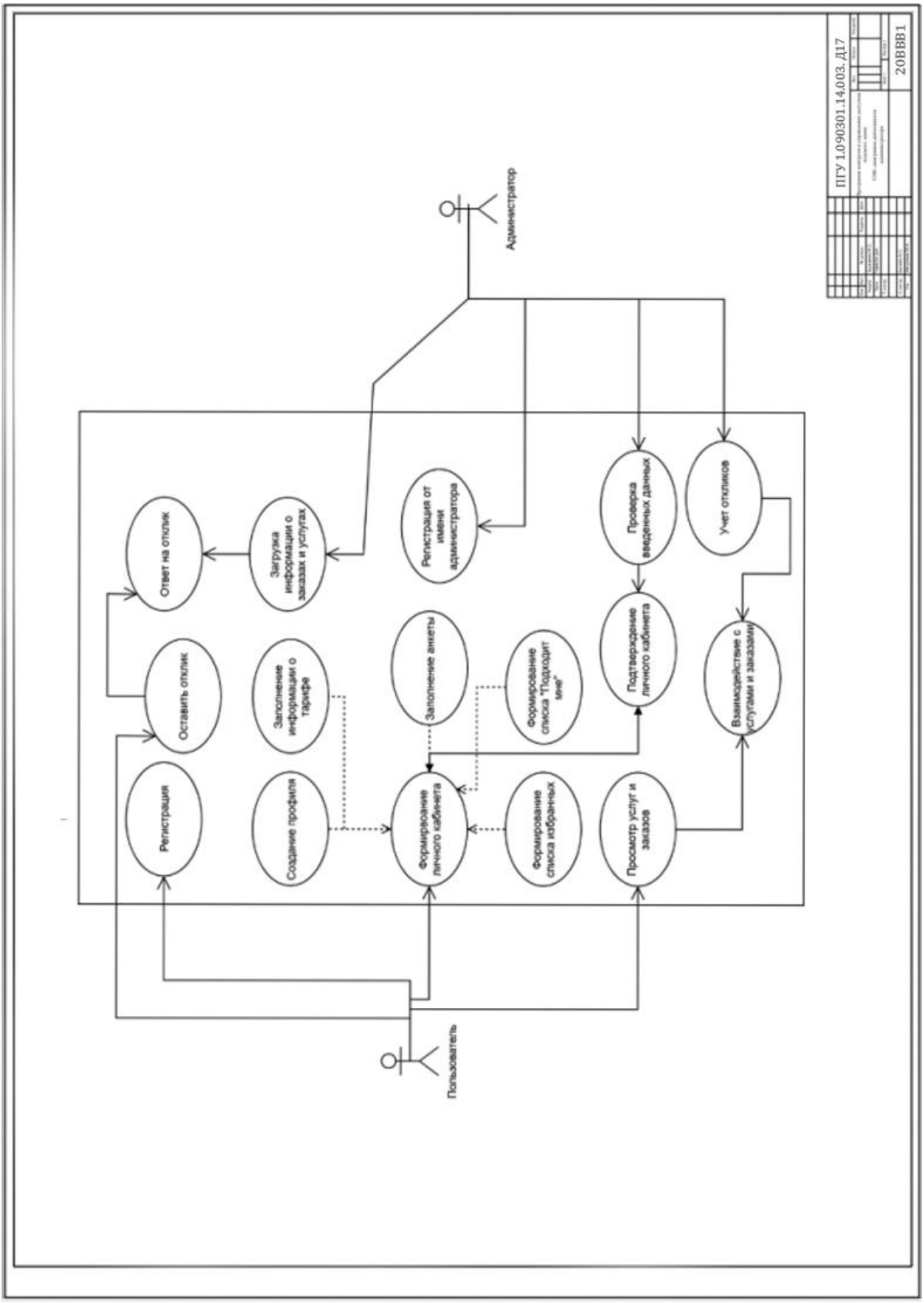
Разработка сайта для поиска специалистов и вакансий

UML-диаграммы

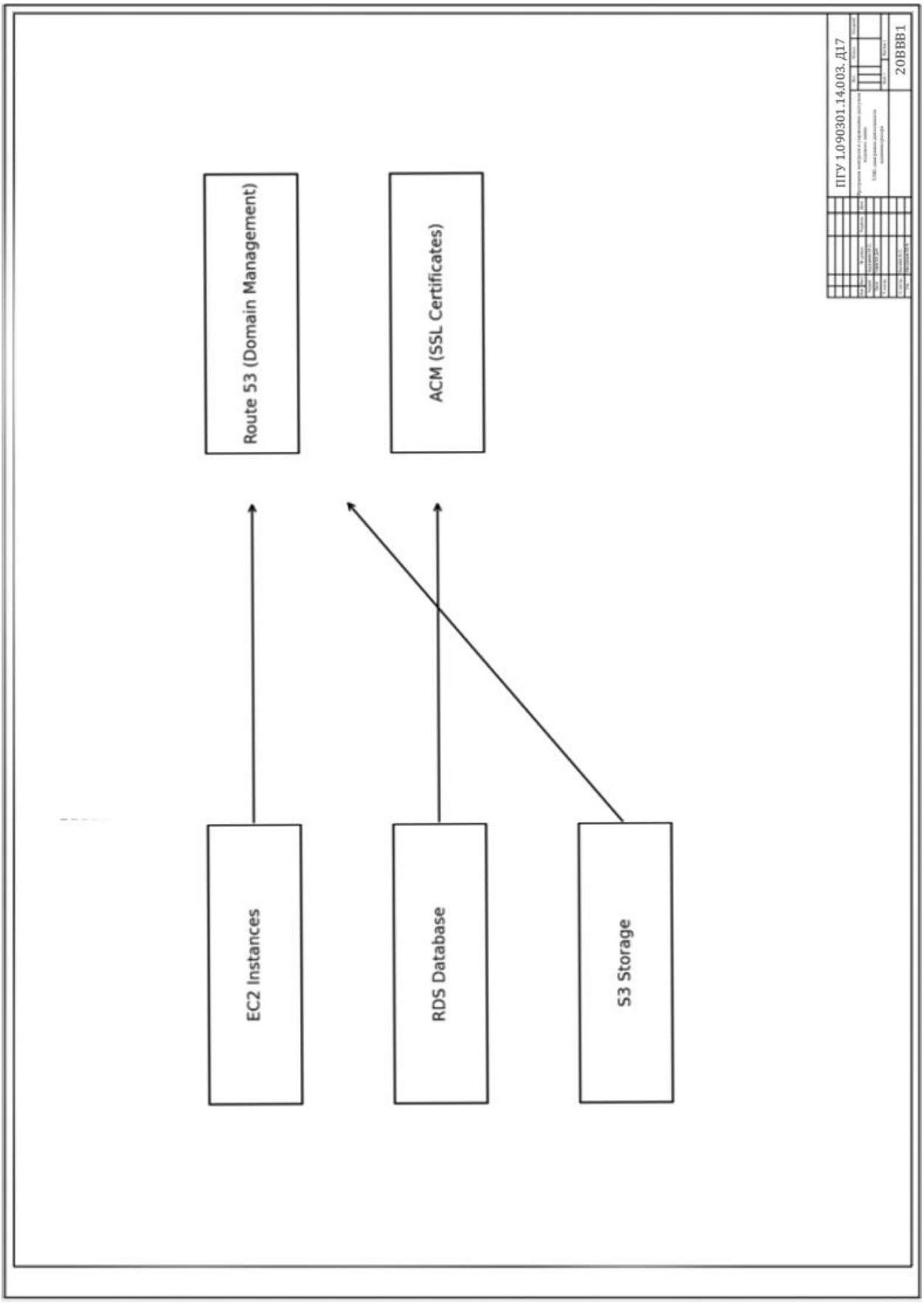
Приложение А.1 – UML–диаграмма последовательности



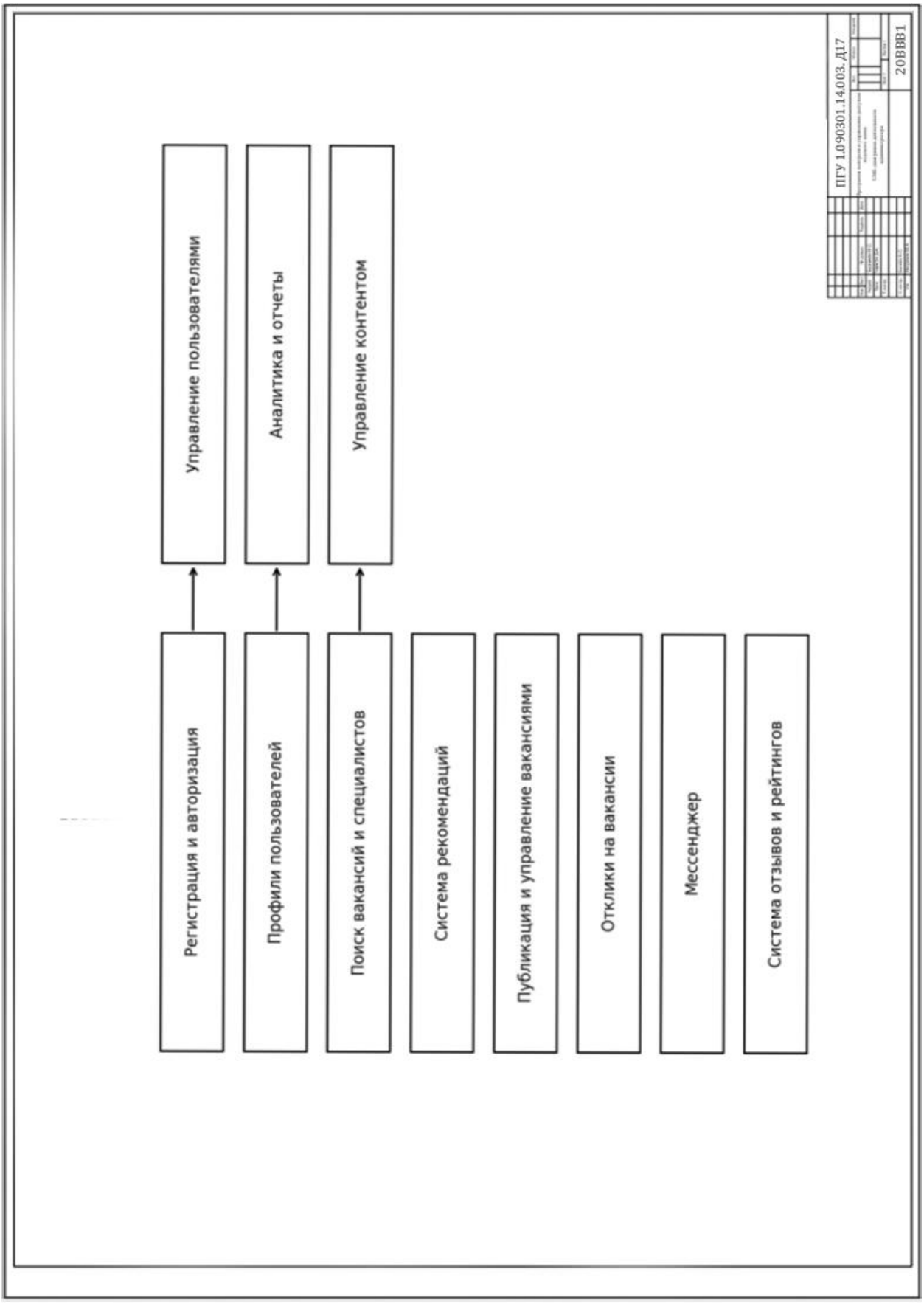
Приложение А.2 – UML – диаграмма прецедентов



Приложение А.3 – Архитектура хостинга веб-приложения на AWS



Приложение А.4 – Структура функциональных возможностей платформы



Приложение Б

(Обязательное)

Разработка сайта для поиска специалистов и вакансий

Листинг программы

Клиентская часть (Frontend)

Клиентская часть разработана с использованием React. Ниже приведен пример кода компонентов для регистрации пользователей и поиска вакансий.

Компонент регистрации пользователя (Register.js):

```
import React, { useState } from 'react';
import axios from 'axios';

const Register = () => {
  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post('/api/users', {
        username,
        email,
        password,
      });
      console.log('User registered:', response.data);
    } catch (error) {
      console.error('Error registering user:', error);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Username:
        <input type=«text» value={username} onChange={(e) =>
setUsername(e.target.value)} required />
      </label>
```

```
            <label>
              Email:
              <input type=«email» value={email} onChange={(e) =>
setEmail(e.target.value)} required />
            </label>
            <label>
              Password:
              <input type=«password» value={password} onChange={(e)
=> setPassword(e.target.value)} required />
            </label>
            <button type=«submit»>Register</button>
          </form>
        );
      };

      export default Register;
```

Компонент поиска вакансий (JobSearch.js):

```
      import React, { useState, useEffect } from 'react';
      import axios from 'axios';

      const JobSearch = () => {
        const [jobs, setJobs] = useState([]);
        const [searchTerm, setSearchTerm] = useState('');

        useEffect(() => {
          const fetchJobs = async () => {
            try {
              const response = await
axios.get(`/api/jobs?search=${searchTerm}`);
              setJobs(response.data);
            } catch (error) {
              console.error('Error fetching jobs:', error);
            }
```

```
    };
    fetchJobs();
  }, [searchTerm]);

  return (
    <div>
      <input
        type=«text»
        placeholder=«Search for jobs»
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
      />
      <ul>
        {jobs.map((job) => (
          <li key={job.id}>{job.title} - {job.company}</li>
        ))}
      </ul>
    </div>
  );
};

export default JobSearch;
```

Серверная часть (Backend)

Серверная часть разработана с использованием Node.js и Express. Пример кода для обработки запросов и взаимодействия с базами данных PostgreSQL и MongoDB приведен ниже.

Настройка серверной части (server.js):

```
const express = require('express');
const bodyParser = require('body-parser');
const { Pool } = require('pg');
```

```
const { MongoClient } = require('mongodb');

const app = express();
app.use(bodyParser.json());

// PostgreSQL setup
const pool = new Pool({
  user: 'dbuser',
  host: 'database.server.com',
  database: 'mydb',
  password: 'secretpassword',
  port: 5432,
});

// MongoDB setup
const mongoClient = new MongoClient('mongodb://localhost:27017', { useNewUrlParser: true, useUnifiedTopology: true });
let mongoDb;
mongoClient.connect(err => {
  if (err) throw err;
  mongoDb = mongoClient.db('mydb');
});

// API endpoints for PostgreSQL
app.get('/api/users', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM users');
    res.json(result.rows);
  } catch (error) {
    console.error('Error fetching users:', error);
    res.status(500).send('Server error');
  }
```

```
});

app.post('/api/users', async (req, res) => {
  const { username, email, password } = req.body;
  try {
    const result = await pool.query('INSERT INTO users (username, email, password_hash) VALUES ($1, $2, $3) RETURNING *', [username, email, password]);
    res.json(result.rows[0]);
  } catch (error) {
    console.error('Error registering user:', error);
    res.status(500).send('Server error');
  }
});

// API endpoints for MongoDB
app.get('/api/profiles', async (req, res) => {
  try {
    const profiles = await mongoDb.collection('profiles').find().toArray();
    res.json(profiles);
  } catch (error) {
    console.error('Error fetching profiles:', error);
    res.status(500).send('Server error');
  }
});

app.post('/api/profiles', async (req, res) => {
  const profile = req.body;
  try {
    const result = await mongoDb.collection('profiles').insertOne(profile);
    res.json(result.ops[0]);
  } catch (error) {
```

```
    console.error('Error creating profile:', error);
    res.status(500).send('Server error');
  }
});

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Базы данных

Структура базы данных PostgreSQL:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE jobs (
    id SERIAL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    company VARCHAR(100) NOT NULL,
    location VARCHAR(100),
    salary INTEGER,
    user_id INTEGER REFERENCES users(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_users_username ON users (username);
CREATE INDEX idx_jobs_title ON jobs (title);
```

Структура базы данных MongoDB:

```
db.users.insertOne({
    username: «johndoe»,
    email: «johndoe@example.com»,
    password_hash: «hashed_password»,
    created_at: new Date()
});

db.jobs.insertOne({
    title: «Senior Developer»,
    description: «Responsible for developing high-quality applications.»,
    company: «Tech Corp»,
    location: «San Francisco, CA»,
    salary: 120000,
    user_id: ObjectId(«user_id»),
    created_at: new Date()
});
```

Приложение В

(Обязательное)

Разработка сайта для поиска специалистов и вакансий

Презентация

Пензенский
государственный
университет

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Разработка сайта для поиска специалистов и вакансий

Выполнила: студентка группы 20ВВВ1 Эсенова А.К.
Научный руководитель д.т.н., профессор Зинкин С.А.

Фриланс

Цель работы
Разработка сайта – интернет-площадки оказания услуг

Назначение разработки
Разрабатываемая интернет-площадка предназначена для поиска специалистов работодателями и поиска работы соискателями

Функции системы

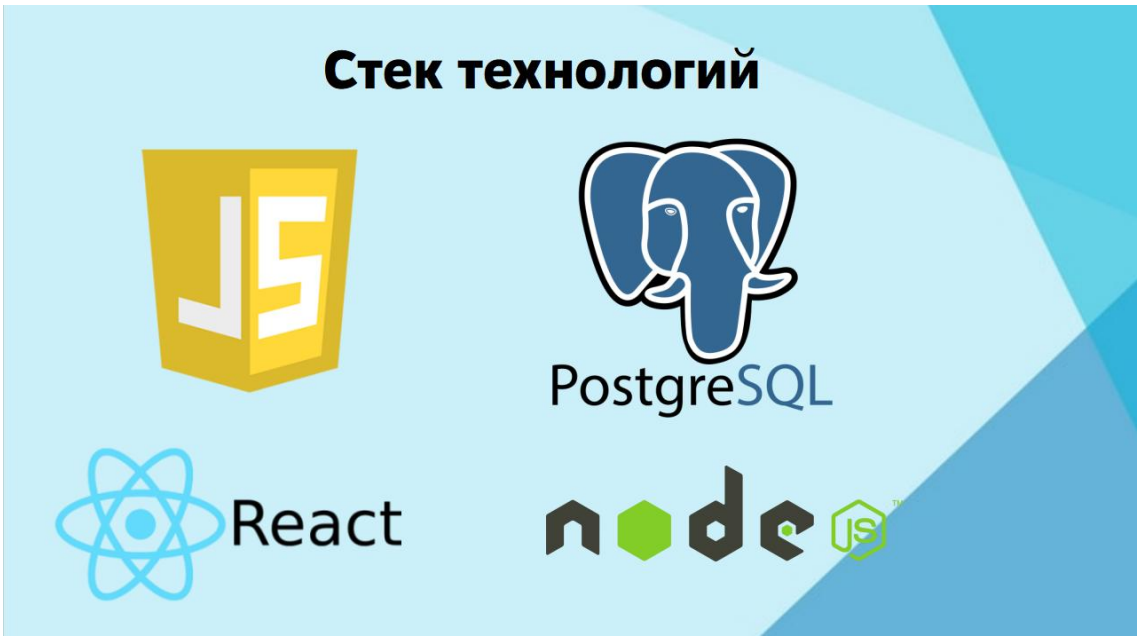
- • Добавление анкет специалистов
- • Добавление заказов на проведение работ
- • Формирование каталога услуг и проектов
- • Внедрение системы отзывов и рейтингов
- • Оставление заявок
- • Отклик на заявки

Входные данные

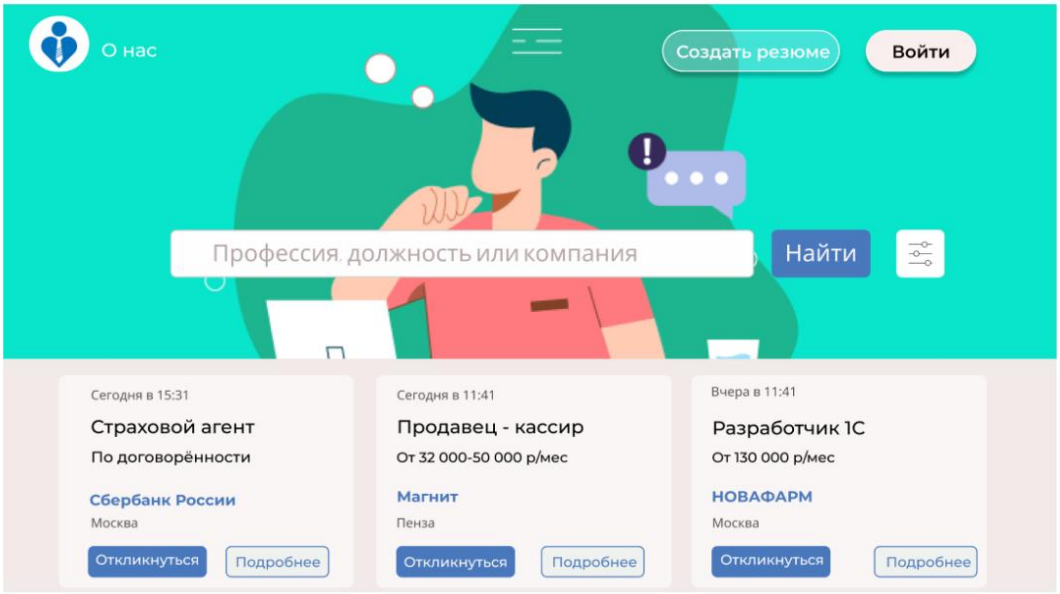
Входными данными для сайта
являются анкеты и заказы

Выходные данные

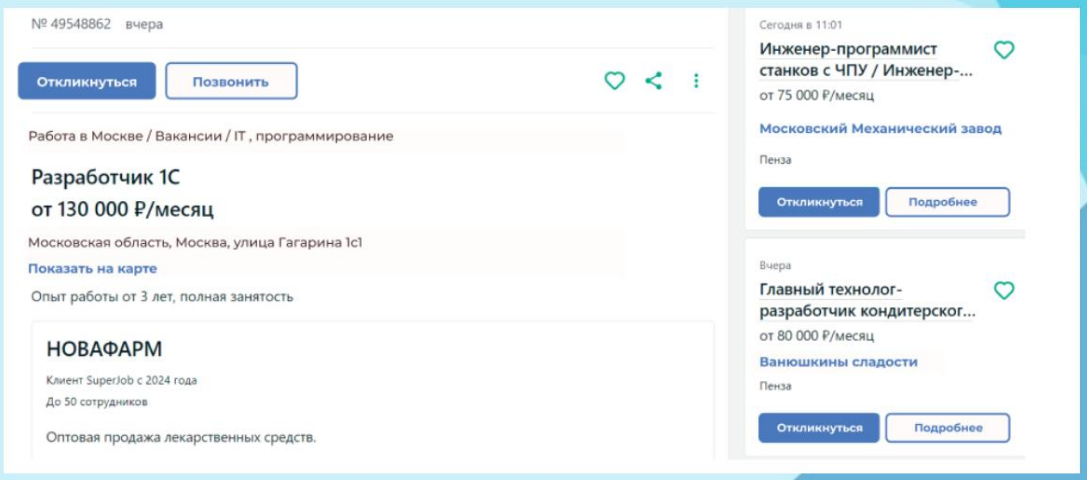
Выходными данными являются ответные заявки на услуги и проекты



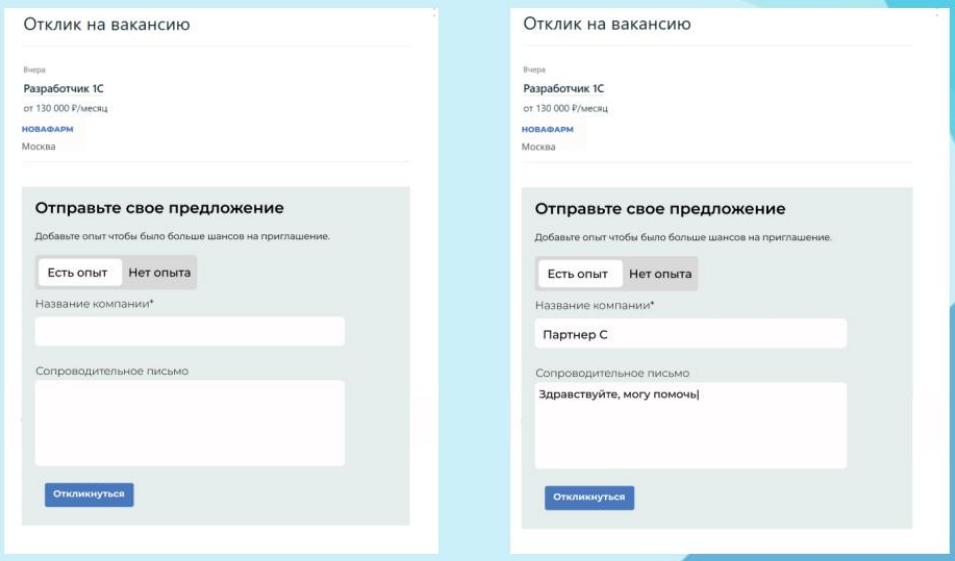
Главная страница сайта



Конкретный проект



Отклик на вакансию



Заключение

• Выполнено проектирование

• Разработана база данных

• Результатом выполнения данной выпускной квалификационной работы можно считать успешно, реализованный в соответствии с заданием создания интернет площадка оказания фриланс услуг.