МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ Е.А. Блинова

подпись инициалы и фамилия

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**ЗАДАНИЕ**

**к курсовой работе**

**по дисциплине** «Базы данных»

Специальность 1-40 01 01 «Программное обеспечение информационных технологий»

Курс: 3 Группа: 4

Студент: Точило Олег Вячеславович

**Тема: «**Web-приложение поиска и предложения работы»

**1. Срок сдачи студентом законченной работы**: «11» декабря 2023 г.

**2. Исходные данные к курсовой работе:**

**2.1**. Функционально должны быть выполнены следующие задачи:

* управление пользователями и паролями (добавление, удаление, изменение);
* управление ролями (администратор, обычный пользователь, гость);
* управление должниками (добавление, удаление, изменение);
* управление транзакциями (добавление, удаление, изменение);
* анализ доходов и расходов (объём долга каждого пользователя перед каждым другим пользователем, общий долг каждого пользователя, долг за указанный период);

**2.2. Требования.**

* База данных должна быть реализована в СУБД Oracle.
* Доступ к данным должен осуществляться только через соответствующие процедуры.
* Количество объектов БД (таблиц, представлений, индексов, пользователей и пр.) регламентируется задачей.
* Должен быть проведен импорт данных из JSON-файлов, экспорт данных в формат JSON.
* Необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости. Необходимо проанализировать планы запросов к таблице.
* Применить технологию базы данных согласно выбранной теме: подробно описать применяемые системные пакеты, утилиты или технологии; показать применение указанной технологии в базе данных.
* Листинги проекта должны содержать комментарии.

**3. Содержание расчетно-пояснительной записки**

* Введение
* Постановка задачи
* Проектирование базы данных.
* Разработка объектов базы данных
* Описание процедур импорта и экспорта
* Тестирование производительности
* Описание технологии и ее применения в базе данных
* Краткое описание приложения для демонстрации
* Руководство пользователя
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления, выполненного курсовой работы:**

* Пояснительная записка оформляется в MS Word.
* Оформление пояснительной записки должно соответствовать требованиям к оформлению пояснительной записки для курсовых работ.
* Листинги всех скриптов представляются в приложении.
* К записке необходимо приложить DVD-диск, который должен содержать пояснительную записку, листинги и файлы базы данных.

**Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсовой работы | Срок выполнения этапов работы | Примечание |
| 1 | Введение | 19.09.2023 |  |
| 2 | Аналитический обзор литературы по теме работы | 26.09.2023 |  |
| 3 | Изучение требований, определение вариантов использования | 03.10.2023 |  |
| 4 | Анализ и проектирование модели базы данных. Описание информационных объектов и ограничений целостности. | 10.10.2023 |  |
| 5 | Создание необходимых объектов | 24.10.2023 |  |
| 6 | Импорт и экспорт данных | 07.11.2023 |  |
| 7 | Описание используемой технологии | 14.11.2023 |  |
| 8 | Тестирование производительности | 21.11.2023 |  |
| 9 | Оформление пояснительной записки | 05.12.2023 |  |
| 10 | Сдача курсовой работы | 11.12.2023 |  |

**5. Дата выдачи задания** «07» сентября 2023 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.А. Нистюк

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

**Содержание**

[Введение 3](#_Toc165375559)

[1 Постановка задачи 4](#_Toc165375560)

[1.1 Обзор аналогичных решений 4](#_Toc165375561)

[1.2 Спецификация требований 4](#_Toc165375562)

[2 Проектирование web-приложения 7](#_Toc165375563)

[2.1 Проектирование базы данных 7](#_Toc165375564)

[2.1 Проектирование сервера web-приложения 11](#_Toc165375565)

[3 Разработка web-приложения 23](#_Toc165375566)

[3.1 Разработка таблиц 23](#_Toc165375567)

[3.2 Разработка хранимых процедур 23](#_Toc165375568)

[3.3 Создание индексов 25](#_Toc165375569)

[3.4 Создание пользователей 25](#_Toc165375570)

[4 Тестирование web-приложения 27](#_Toc165375571)

[5 Руководство пользователя 28](#_Toc165375572)

[Заключение 29](#_Toc165375573)

[Список используемых источников 30](#_Toc165375574)

[Приложение А 31](#_Toc165375575)

[Приложение Б 33](#_Toc165375576)

[Приложение В 43](#_Toc165375577)

[Приложение Г 44](#_Toc165375578)

Введение

В современном мире поиск и предложение работы становятся все более актуальными, поскольку технологические инновации меняют традиционные способы трудоустройства. Сегодня множество людей стремятся найти и предложить работу через специализированные приложения, которые облегчают этот процесс и создают связь между работодателями и соискателями.

Разработанное приложение предоставляет удобную и эффективную платформу для взаимодействия между работодателями и соискателями. Оно предлагает. Приложение позволяет соискателям создавать профили и свои резюме и представлять свои навыки, образование и опыт работы. Они могут просматривать доступные вакансии и отправлять свои заявки на эти вакансии. Работодатели могут размещать вакансии и просматривать профили и резюме откликнувшихся кандидатов. Помимо базовых функций поиска и предложения работы, приложение предоставляет возможность оценивать и оставлять отзывы о работодателях и соискателях.

В приложении разграничены возможности гостей, соискателей, работодателей и администраторов. Так, гости могут только просматривать публичные вакансии, страницы компаний и отзывы на них; соискателям доступно изменение своих персональных данных, изменение списка резюме и возможность отклика на доступные вакансии; работодатели могут изменять список своих вакансий, просматривать список откликов, принимать и отклонять их; работодателям и соискателям доступна возможность оставлять отзывы друг на друга, но только при условии, что соискатель откликнулся на вакансию работодателя, и последний принял отклик; администраторам доступны возможности по удалению учётных записей, подтверждению статусов компаний и удалению комментариев.

В ходе выполнения курсового проектирования будут решены следующие задачи:

– анализ литературы по теме работы;

– изучение требований и определение вариантов использования;

– анализ и проектирование модели базы данных, описание информационных объектов и ограничений целостности;

– создание необходимых объектов;

– реализация импорта и экспорта данных;

– описание используемой технологии;

– тестирование производительности;

– создание руководства пользователя.

Целями базы данных являются хранение, организация и обеспечение безопасности данных, а задачами являются структуризация данных в таблицы и использование индексов для ускоренного доступа к данным.

1 Постановка задачи

1.1 Обзор аналогичных решений

В качестве первого аналогичного решения была рассмотрена платформа Fiverr, на которой предлагаются услуги фрилансеров со всего мира. Она специализируется на небольших задачах, а не на поиске постоянного места работы. Пользователи могут найти исполнителей по различным областям, таким как дизайн, письменное творчество, программирование и многое другое. Основные особенности Fiverr включают простой процесс заказа, возможность просмотра портфолио и отзывов о соискателях и ориентация на заказчиков, то есть соискатели должны размещать свои резюме, а заказчики должны выбирать из имеющихся на данный момент соискателей.

В качестве второго аналогичного решения была рассмотрена платформа Upwork, предлагающая более широкий спектр услуг и проектов для фрилансеров. Платформа предоставляет доступ к множеству заказов в различных областях, включая разработку ПО, маркетинг и так далее. Upwork обеспечивает более сложный процесс поиска и найма, включая возможность просмотра профилей, проведения собеседований и оценки работы. Также на платформе присутствует система обратной связи и защиты платежей. К преимуществам платформы относится возможность размещения как вакансий, так и публичных резюме, а также возможность предлагать работу как на основе проектной занятости, так и на полный рабочий день.

В качестве последнего аналогичного решения была рассмотрена платформа HeadHunter – платформа с мощных функционалом по поиску работы и работников в странах СНГ. Она предлагает широкий функционал по размещению вакансий и резюме, а также обширные возможности фильтрации оп множеству критериев. Для связи соискателя и работодателя предусмотрен чат.

Недостатком всех рассмотренных аналогов является невозможность узнать, была ли изменена вакансия, без перезагрузки страницы. В худшем случае соискатель может откликнуться не на ту вакансию, на которую собирался, так как работодатель изменил её, пока соискатель находился на её странице.

На основе рассмотренных аналогичных решений были сформированы следующие требования к программному продукту:

– возможность размещать вакансии;

– возможность оценивать соискателей и работодателей;

– автоматическое обновление данных о просматриваемой вакансии в случае её изменения.

1.2 Спецификация требований

База данных должна быть реализована в СУБД PostgreSQL.

Всем пользователям, включая гостей, должна быть доступна возможность просматривать публичные вакансии, страницы компаний и отзывы о них и жаловаться на отзывы.

Соискателю должна быть доступна возможность добавлять резюме, изменять и удалять их, откликаться и удалять свои отклики на публичные вакансии и оставлять отзывы на работодателей, которые приняли его отклик.

Работодателю должна быть доступна возможность добавлять вакансии, изменять и удалять их, получать список откликов на свои вакансии, принимать и отклонять их, просматривать данные о соискателях, откликнувшихся на его вакансии, просматривать отзывы о них и оставлять отзывы о тех соискателях, чьи отклики он принял и запрашивать подтверждение компании у администратора.

Работодателям и соискателям должна быть доступна возможность редактировать данные о себе, менять пароль, удалять свои отзывы и запрашивать удаление своей учётной записи.

Администратору должна быть доступна возможность блокировать возможность пользователя оставлять отзывы о другом пользователе, удалять комментарии и учётные записи и подтверждать компании. Диаграмма вариантов использования представлена на рисунке 1.1.

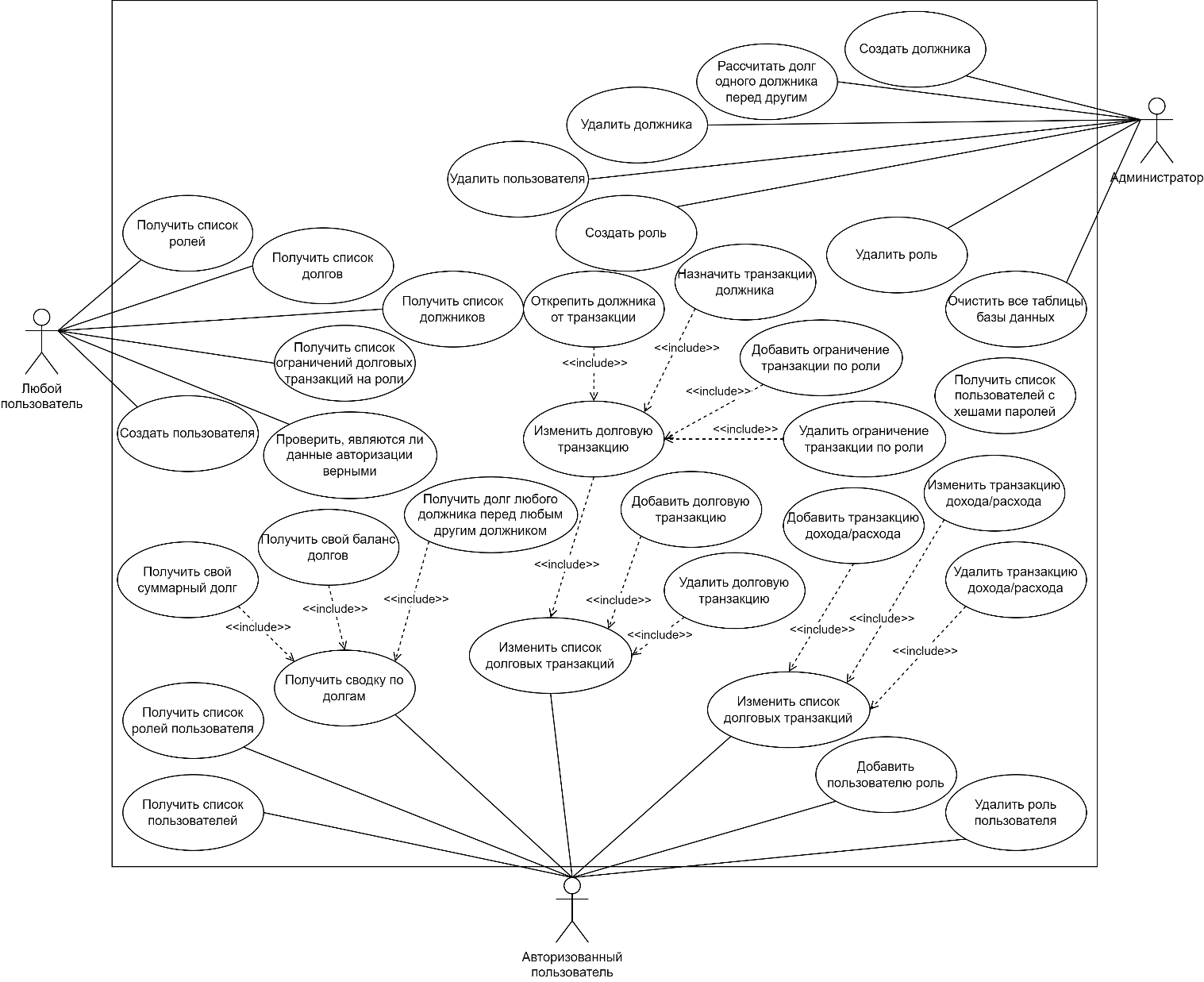


Рисунок 1.1 – Диаграмма вариантов использования базы данных

В категорию «Любые пользователи» входят гости, обычные пользователи и администраторы. В категорию «Авторизованные пользователи» входят обычные пользователи и администраторы.

2 Проектирование web-приложения

2.1 Проектирование базы данных

Логическая схема базы данных представлена на рисунке 2.1.

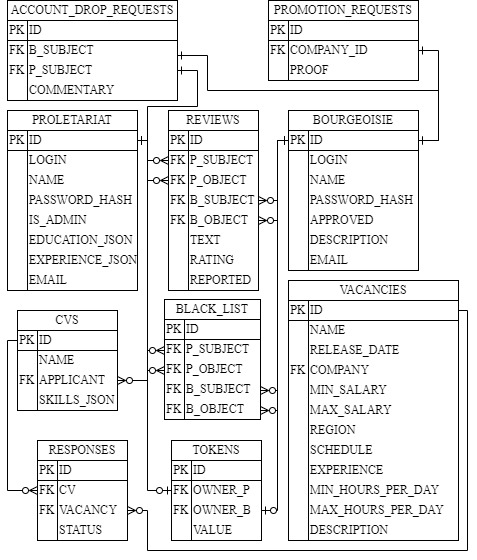


Рисунок 2.1 – Логическая схема базы данных

База данных содержит десять таблиц, хранящих информацию о пользователях, резюме, вакансиях, откликах и прочих данных. Листинг создания таблиц представлен в приложении А.

Таблица PROLETARIAT хранит информацию о соискателях. Описание её столбцов представлено в таблице 2.1.

Таблица 2.1 – Описание таблицы PROLETARIAT

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор соискателя, первичный ключ |
| login | varchar(20) | логин соискателя |
| name | varchar(50) | имя соискателя |
| password\_hash | varchar(60) | хеш пароля соискателя |
| is\_admin | char(1) | является ли пользователь администратором |
| education\_json | varchar(200) | образование соискателя в формате JSON |
| experience\_json | varchar(500) | опыт работы соискателя в формате JSON |
| email | varchar(30) | адрес электронной почты соискателя |

Таблица CVS хранит информацию о резюме соискателей. Описание её столбцов представлено в таблице 2.2.

Таблица 2.2 – Описание таблицы CVS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор резюме, первичный ключ |
| name | varchar(30) | название резюме |
| applicant | integer | идентификатор соискателя, внешний ключ |
| skills\_json | varchar(100) | массив навыков в формате JSON |

Таблица BOURGEOISIE хранит информацию о работодателях. Описание её столбцов представлено в таблице 2.3.

Таблица 2.3 – Описание таблицы BOURGEOISIE

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор работодателя, первичный ключ |
| login | varchar(20) | логин работодателя |
| name | varchar(70) | имя работодателя |
| password\_hash | varchar(60) | хеш пароля работодателя |
| approved | char(1) | показатель того, подтверждена ли компания |
| description | varchar(2000) | описание работодателя |
| email | varchar(30) | адрес электронной почты работодателя |

Таблица VACANCIES хранит информацию о вакансиях. Описание её столбцов представлено в таблице 2.4.

Таблица 2.4 – Описание таблицы VACANCIES

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор вакансии, первичный ключ |
| name | varchar(30) | название вакансии |
| release\_date | date | дата публикации вакансии |
| company | integer | идентификатор компании, внешний ключ |
| active | char(1) | показатель того, доступна ли вакансия публично |
| min\_salary | integer | минимальная зарплата |
| max\_salary | integer | максимальная зарплата |
| region | varchar(20) | регион поиска соискателей |
| schedule | integer | график работы |
| experience | integer | требуемый опыт работы |
| min\_hours\_per\_day | integer | минимальное количество рабочих часов в неделю |
| max\_hours\_per\_day | integer | максимальное количество рабочих часов в неделю |
| description | varchar(1000) | описание вакансии |

Таблица RESPONSES хранит информацию об откликах на вакансии. Описание её столбцов представлено в таблице 2.5.

Таблица 2.5 – Описание таблицы RESPONSES

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор отклика, первичный ключ |
| cv | integer | идентификатор резюме, внешний ключ |
| vacancy | integer | идентификатор вакансии, внешний ключ |
| status | char(1) | показатель статуса отклика |

Таблица REVIEWS хранит информацию об отзывах пользователей друг на друга. Описание её столбцов представлено в таблице 2.6.

Таблица 2.6 – Описание таблицы REVIEWS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор отзыва, первичный ключ |
| p\_subject | integer | идентификатор соискателя, оставившего отзыв, внешний ключ |
| p\_object | integer | идентификатор соискателя, о котором был оставлен отзыв, внешний ключ |
| b\_subject | integer | идентификатор работодателя, оставившего отзыв, внешний ключ |
| b\_object | integer | идентификатор работодателя, о котором был оставлен отзыв, внешний ключ |
| text | varchar(100) | текст отзыва |
| rating | integer | оценка |
| reported | char(1) | показатель того, была ли оставлена жалоба на отзыв |

Таблица PROMOTION\_REQUESTS хранит информацию о запросах на подтверждение компании. Описание её столбцов представлено в таблице 2.7.

Таблица 2.7 – Описание таблицы PROMOTION\_REQUESTS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор запроса, первичный ключ |
| company\_id | int | идентификатор компании, внешний ключ |
| proof | varchar(125000) | текст предоставленного доказательства |

Таблица ACCOUNT\_DROP\_REQUESTS хранит информацию о запросах на удаление учётных записей. Описание её столбцов представлено в таблице 2.8.

Таблица 2.8 – Описание таблицы ACCOUNT\_DROP\_REQUESTS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор запроса, первичный ключ |
| p\_subject | integer | идентификатор соискателя, внешний ключ |
| b\_subject | integer | идентификатор компании, внешний ключ |
| commentary | varchar(255) | комментарий к запросу |

Таблица TOKENS хранит информацию о refresh-токенах пользователей. Описание её столбцов представлено в таблице 2.9.

Таблица 2.9 – Описание таблицы TOKENS

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор токена, первичный ключ |
| owner\_p | integer | идентификатор соискателя, внешний ключ |
| owner\_b | integer | идентификатор компании, внешний ключ |
| value | varchar(256) | значение токена |

Таблица BLACK\_LIST хранит информацию о пользователях, которым запрещено оставлять отзывы о некоторых других пользователях. Описание её столбцов представлено в таблице 2.10.

Таблица 2.10 – Описание таблицы BLACK\_LIST

|  |  |  |
| --- | --- | --- |
| Название столбца | Тип данных | Описание |
| id | integer | идентификатор записи, первичный ключ |
| p\_subject | integer | идентификатор соискателя, которому запрещается оставлять отзыв, внешний ключ |
| p\_object | integer | идентификатор соискателя, о котором запрещено оставлять отзыв, внешний ключ |
| b\_subject | integer | идентификатор работодателя, которому запрещается оставлять отзыв, внешний ключ |
| b\_object | integer | идентификатор работодателя, о котором запрещено оставлять отзыв, внешний ключ |

2.2 Проектирование сервера web-приложения

Для обработки запросов применяется четыре роутера, каждый из которых обрабатывает запросы к определённым адресам. Так, masterRouter обрабатывает все запросы, для которых не нужна авторизация, такие как получение списка вакансий, жалоба на отзыв и так далее. Список представляемых обработчиков представлен в таблице 2.11.

Таблица 2.11 – список представляемых обработчиков роутера masterRouter

|  |  |  |
| --- | --- | --- |
| Адрес | Метод | Описание |
| /public-vacancies | GET | Возвращает список публичных вакансий с заданным смещением относительно начала таблицы. Если в строке запроса находятся параметры из списка фильтров, их значения применяются для фильтрации вакансий |
| /public-companies | GET | Возвращает список названий и идентификаторов всех компаний. Если не получен параметр поисковой строки skipRating, также вычисляется рейтинг каждой компании |
| /company-reviews | GET | Возвращает идентификатор компании, её название, список отзывов о ней, её рейтинг, идентификатор отзыва, который о данной компании оставил запрашивающий пользователь, и показатель того, может ли запрашивающий пользователь оставить отзыв о данной компании, чтобы скрипт на странице мог определить, добавлять ли на неё кнопку создания отзыва или нет |
| /report-review | PUT | Создаёт жалобу на отзыв. Если жалоба уже существовала, возвращает сообщение об этом. Если жалобы не существовало, возвращает сообщение об успешном создании жалобы |
| /logout | GET | Производит очистку cookie клиента от идентификатора пользователя, его типа пользователя, access-токена и refresh-токена |

Роутер proletariatRouter предназначен для обработки запросов соискателей. Все запросы к нему должны начинаться с /prol Список представляемых обработчиков представлен в таблице 2.12.

Таблица 2.12 – список представляемых обработчиков роутера proletariatRouter

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Требует авторизации | Метод | Описание |
| /login | Нет | GET | Если пользователь уже авторизован, возвращает сообщение об ошибке. Производит авторизацию, генерирует пару токенов и устанавливает идентификатор пользователя, тип пользователя, access-токен и refresh-токен в cookie пользователя. В случае ошибки производит выход пользователя из учётной записи |
| /register | Нет | PUT | Осуществляет проверку на существование пользователя с предоставленным логином. В случае, если логин занят, возвращает сообщение об ошибке. В противном случае создаёт нового пользователя и возвращает сообщение об успешном выполнении регистрации |
| /personal | Да | GET | Возвращает идентификатор пользователя, его имя, объект, содержащий сведения о его образовании, объект, содержащий сведения о его опыте работы, адрес его электронной почты и показатель того, был выполнен запрос на удаление учётной записи или нет |
| /personal | Да | POST | Обновляет персональные данные пользователя. Притом, если исходное значение электронной почты было правильным, а новое не проходит проверку регулярным выражением, то возвращается сообщение об ошибке. В противном случае возвращается сообщение об успешном обновлении персональных данных |
| /password | Да | PATCH | Ожидает получение старого и нового пароля из тела запроса, получает идентификатор пользователя из cookie-файла и ищет пользователя по идентификатору. В случае, если такого пользователя не существует, возвращает сообщение об ошибке. Сравнивает хеш пришедшего старого пароля с хешем пароля из базы данных. В случае, если они не совпадают, возвращает сообщение об ошибке. Устанавливает хеш нового пароля в базу данных и возвращает сообщение об успешной смене пароля |
| /review | Да | PUT | Проверяет, имеет ли право соискатель оставлять отзыв на данную компанию, проверяя наличие нужной записи в чёрном списке. В случае существования такой записи возвращает сообщение об ошибке. Затем проверяет, может ли пользователь оставить отзыв на данную компанию, проверяя наличие принятого отклика, оставленного данным соискателем на вакансию данной компании. Если такого отклика нет, возвращает сообщение об ошибке. В противном случае создаёт отзыв и возвращает сообщение об успешном его создании |
| /review | Да | DELETE | Проверяет, существует ли отзыв с данным идентификатором от данного пользователя. Если такой отзыв существует, удаляет его. В противном случае возвращает сообщение об ошибке |
| /cv | Да | GET | Если строка запроса не содержит параметров, возвращает список всех резюме соискателя. В случае, если установлен параметр id, возвращает резюме с заданным идентификатором данного соискателя, если оно существует. В противном случае возвращает сообщение о том, что резюме не найдено. В случае, если установлен параметр поисковой строки vacancy, но не установлен id, возвращает список резюме, которыми можно откликнуться на данную вакансию |
| /cv | Да | POST | Проверяет, существует ли резюме с данным идентификатором и данным соискателем. Если не существует, возвращает сообщение об ошибке. Если название резюме у данного соискателя уже занято другим резюме, возвращает сообщение об ошибке. Обновляет данные резюме и возвращает сообщение об успешном выполнении операции |
| /cv | Да | PUT | Проверяет наличие у пользователя электронной почты. В случае её отсутствия возвращает сообщение об ошибке, так как она является единственным способом связи между работодателем и соискателем. Проверяет, занято ли название резюме у данного соискателя. В случае, если резюме с таким названием уже существует, возвращает сообщение об ошибке. Создаёт новое резюме с заданными данными и возвращает сообщение об успешном создании резюме |
| /cv | Да | DELETE | Получает идентификатор пользователя из cookie-файла. Проверяет, существует ли резюме с указанным в теле запроса названием и принадлежащее данному соискателю. Если резюме с указанным названием существует, то удаляет его и возвращает сообщение об успешном удалении |
| /responses | Да | GET | Возвращает список всех откликов данного соискателя и общее число откликов данного соискателя |
| /responses | Да | PUT | Проверяет наличие у текущего пользователя правильной электронной почты. Затем проверяет существование резюме по идентификатору резюме и идентификатору соискателя, полученному из cookie-файла. Затем проверяет доступность вакансии и, если она недоступна, возвращает сообщение об ошибке. Затем проверяет существование такого отклика по идентификаторам резюме и вакансии и, если такого отклика не существует, создаёт отклик и возвращает сообщение об успехе |
| /responses | Да | DELETE | Проверяет наличие у текущего соискателя резюме с данным идентификатором и отклика с данными идентификаторами резюме и вакансии. Если такого отклика не существует, возвращает сообщение об ошибке. В противном случае удаляет его и возвращает сообщение об успешном удалении |
| /drop-requests | Да | PUT | Проверяет, был ли запрос на удаление учётной записи уже отправлен текущим соискателем. Если такой запрос уже существует, возвращает сообщение об ошибке. В противном случае создаёт запрос и возвращает сообщение об его успешном создании |

Роутер bourgeoisieRouter предназначен для обработки запросов работодателей и запросов на получение публичной информации о работодателе. Список представляемых обработчиков представлен в таблице 2.13.

Таблица 2.13 ­ список представляемых обработчиков роутера bourgeoisieRouter

|  |  |  |  |
| --- | --- | --- | --- |
| Адрес | Требует авторизации | Метод | Описание |
| /login | Нет | GET | Если пользователь уже авторизован, возвращает сообщение об ошибке. Производит авторизацию, генерирует пару токенов и устанавливает идентификатор пользователя, тип пользователя, access-токен и refresh-токен в cookie пользователя. В случае ошибки производит выход пользователя из учётной записи |
| /register | Нет | PUT | Осуществляет проверку на существование пользователя с предоставленным логином. В случае, если логин занят, возвращает сообщение об ошибке. В противном случае создаёт нового пользователя и возвращает сообщение об успешном выполнении регистрации |
| /personal | Да | GET | Возвращает идентификатор работодателя, его имя, описание, адрес его электронной почты показатель того, был выполнен запрос на удаление учётной записи, показатель того, была ли компания подтверждена и показатель того, был ли отправлен |
| /personal | Да | POST | Обновляет персональные данные пользователя. Притом, если исходное значение электронной почты было правильным, а новое не проходит проверку регулярным выражением, то возвращается сообщение об ошибке. В противном случае возвращается сообщение об успешном обновлении персональных данных |
| /password | Да | PATCH | Ожидает получение старого и нового пароля из тела запроса, получает идентификатор пользователя из cookie-файла и ищет пользователя по идентификатору. В случае, если такого пользователя не существует, возвращает сообщение об ошибке. Сравнивает хеш пришедшего старого пароля с хешем пароля из базы данных. В случае, если они не совпадают, возвращает сообщение об ошибке. Устанавливает хеш нового пароля в базу данных и возвращает сообщение об успешной смене пароля |
| /review | Да | GET | Предназначен для получения отзывов о соискателе по его идентификатору. Проверяет, отправлял ли данный соискатель отклик на любую из вакансий текущего работодателя и принял ли работодатель этот отклик. Если такого принятого отклика нет, возвращает сообщение об ошибке, так как работодатель не должен видеть отзывов о любом соискателе. В противном случае возвращает идентификатор, имя и рейтинг соискателя, список отзывов о соискателе, каждый элемент которого содержит идентификатор и имя автора отзыва и идентификатор, текст и значение оценки отзыва, показатель того, разрешено ли текущему работодателю оставить отзыв о данном соискателе и идентификатор отзыва текущего соискателя о данном работодателе. |
| /review | Да | PUT | Проверяет, имеет ли право соискатель оставлять отзыв на данную компанию, проверяя наличие нужной записи в чёрном списке. В случае существования такой записи возвращает сообщение об ошибке. Затем проверяет, может ли пользователь оставить отзыв на данную компанию, проверяя наличие принятого отклика, оставленного данным соискателем на вакансию данной компании. Если такого отклика нет, возвращает сообщение об ошибке. В противном случае создаёт отзыв и возвращает сообщение об успешном его создании |
| /review | Да | DELETE | Проверяет, существует ли отзыв с данным идентификатором от данного пользователя. Если такой отзыв существует, удаляет его. В противном случае возвращает сообщение об ошибке |
| /applicants-list | Да | GET | Возвращает список идентификаторов и имён соискателей, которые отправили отклик на вакансию текущего работодателя и чьи отклики были одобрены, для отображения в списке доступных для создания отзыва соискателей |
| /icon | Да | PUT | Проверяет заголовки запроса на тип содержимого и в случае, если запрос содержит не изображение в формате JPEG, возвращает сообщение об ошибке. Проверяет заголовки запроса на размер изображения и в случае превышения допустимого размера возвращает сообщение об ошибке. Записывает содержимое тела запроса в изображение текущего работодателя на стороне сервера. Возвращает сообщение об успешной замене изображения |
| /info | Нет | GET | Возвращает публичные данные работодателя: идентификатор, название, описание и адрес электронной почты ­ работодателя по его идентификатору |
| /responses | Да | GET | Возвращает список откликов текущего работодателя, каждый элемент которого содержит идентификатор отклика, статус отклика, объект с данными вакансии и объект, содержащий идентификатор резюме, название резюме, список навыков и объект, содержащий идентификатор, имя, данные об образовании и данные об опыте работы соискателя, и количество всех откликов |
| /responses | Да | POST | Проверяет существование отклика с заданным идентификатором у данного работодателя. В случае, если такого отклика не существует, возвращает сообщение об ошибке. Проверяет, можно ли менять статус текущего отклика и производится ли попытка сбросить его статус, и в таком случае возвращает сообщение об ошибке. Обновляет статус отклика по идентификатору и возвращает сообщение об успешном обновлении данных |
| /vacancy | Да | GET | Проверяет, установлен ли параметр id строки запроса. Если он установлен, возвращает вакансию по идентификатору вакансии и идентификатору текущей компании. В противном случае возвращает список всех вакансий текущего работодателя по его идентификатору |
| /vacancy | Да | PUT | Проверяет, установлен ли адрес электронной почты у работодателя. Если он не установлен, возвращает сообщение об ошибке. Проверяет, занято ли данное имя вакансии другой вакансией данного работодателя, производит валидацию полей вакансии, создаёт её и возвращает сообщение об успешном создании |
| /vacancy | Да | POST | Проверяет существование вакансии по идентификатору вакансии, переданному в теле запроса, и идентификатору работодателя, полученному из cookie-файла. Проверяет, не занято ли новое название вакансии другой вакансией данного работодателя. В случае ошибки возвращает сообщение об ошибке. Обновляет данные о вакансии и возвращает сообщение об успешном обновлении |
| /vacancy | Да | DELETE | Проверяет, существует ли вакансия с заданным в теле запроса идентификатором и текущим работодателем. Если такой вакансии не существует, возвращает сообщение об ошибке. Удаляет вакансию по идентификатору, генерирует событие changed и возвращает сообщение об успешном удалении вакансии |
| /promotion-request | Да | PUT | Проверяет существование запроса на подтверждение текущей компании. Если такой запрос уже существует, возвращает сообщение об ошибке. Создаёт запрос с идентификатором текущего работодателя |
| /drop-request | Да | PUT | Проверяет существование запроса на удаление учётной записи текущей компании. Если такой запрос уже существует, возвращает сообщение об ошибке. Создаёт запрос с идентификатором текущего работодателя и комментарием, полученным из тела запроса |

Роутер adminRouter предназначен для обработки запросов администратора. Для доступа ко всем обработчикам данного роутера требуется авторизация и роль администратора. Список представляемых обработчиков представлен в таблице 2.14.

Таблица 2.14 ­ список представляемых обработчиков роутера adminRouter

|  |  |  |
| --- | --- | --- |
| Адрес | Метод | Описание |
| /promotion-requests | GET | Возвращает список запросов на подтверждение компаний, каждый элемент которого содержит идентификатор компании, название компании и прикреплённое доказательство, и общее количество запросов |
| /drop-requests | GET | Возвращает список запросов на удаление учётных записей, и общее количество запросов |
| /promote | PATCH | Проверяет существование запроса на подтверждение компании с заданным идентификатором. Если такого запроса не существует, возвращает сообщение об ошибке. Получает объект компании из таблицы базы данных, устанавливает статус компании в подтверждённое значение и удаляет запрос на подтверждение. Если у компании установлен адрес электронной почты, отправляет на него письмо с сообщением о том, что запрос на подтверждение был удовлетворён. Возвращает сообщение об успешном изменении статуса компании |
| /promote | DELETE | Проверяет существование запроса на подтверждение компании с заданным идентификатором. Если такого запроса не существует, возвращает сообщение об ошибке. Удаляет запрос о повышении по его идентификатору. Получает объект компании из таблицы базы данных и, если у компании установлен адрес электронной почты, отправляет на него письмо с сообщением о том, что запрос на подтверждение был отклонён. Возвращает сообщение об успешном удалении запроса |
| /drop-user | DELETE | Проверяет существование запроса на удаление учётной записи с заданным идентификатором. Если такого запроса не существует, возвращает сообщение об ошибке. Получает тип пользователя по объекту запроса. Получает объект пользователя из таблицы базы данных и удаляет строку из соответствующей таблицы по идентификатору пользователя. Если у пользователя установлен адрес электронной почты, отправляет на него письмо с сообщением о том, что запрос на удаление учётной записи был удовлетворён. Удаляет запрос на удаление учётной записи по его идентификатору. Возвращает сообщение об успешном удалении пользователя |
| /drop-user | POST | Проверяет существование запроса на удаление учётной записи с заданным идентификатором. Если такого запроса не существует, возвращает сообщение об ошибке. Получает тип пользователя по объекту запроса. Получает объект пользователя из таблицы базы данных и удаляет строку из соответствующей таблицы по идентификатору пользователя. Если у пользователя установлен адрес электронной почты, отправляет на него письмо с сообщением о том, что запрос на удаление учётной записи был отклонён. Удаляет запрос на удаление учётной записи по его идентификатору. Возвращает сообщение об успешном отклонении запроса |
| /reported-reviews | GET | Возвращает список отзывов, на которые поступила жалоба, каждый элемент которого содержит идентификатор отзыва, тип пользователя автора, идентификатор автора, имя автора, идентификатор получателя и имя получателя, и общее число отзывов, на которые была оставлена жалоба |
| /review | POST | Проверяет существование отзыва по его идентификатору и существование жалобы по значению столбца в таблице базы данных. Если отзыва или жалобы не существует, возвращает сообщение об ошибке. Обновляет значение столбца reported строки таблицы REVIEWS по идентификатору и возвращает сообщение об успешном удалении жалобы |
| /review | DELETE | Проверяет существование отзыва по его идентификатору и существование жалобы по значению столбца в таблице базы данных. Если отзыва или жалобы не существует, возвращает сообщение об ошибке. Удаляет отзыв по его идентификатору и возвращает сообщение об успешном удалении жалобы |
| /ban | GET | Возвращает список пользователей, которые не имеют право оставлять отзывы хотя бы на одного другого пользователя, каждый элемент которого содержит идентификатор записи, идентификатор заблокированной компании, идентификатор заблокированного соискателя, название компании и имя соискателя, и общее число записей в чёрном списке |
| /ban | POST | Проверяет наличие записи в чёрном списке по идентификатору блокируемого и идентификатору того, у кого пользователь блокируется. Если запись существует, возвращает сообщение об ошибке. Создаёт новую запись в чёрном списке с параметрами, полученными из тела запроса, и получает объект заблокированного пользователя. Если у пользователя установлен адрес электронной почты, отправляет ему электронное письмо с сообщением о блокировке у другого пользователя. Возвращает сообщение о создании записи в чёрном списке |
| /ban | DELETE | Проверяет наличие записи в чёрном списке по идентификатору блокируемого и идентификатору того, у кого пользователь блокируется. Если запись существует, возвращает сообщение об ошибке. Получает объект заблокированного пользователя. Если у пользователя установлен адрес электронной почты, отправляет ему электронное письмо с сообщением о разблокировке у другого пользователя. Удаляет запись из таблицы. Возвращает сообщение об удалении записи из чёрного списка |

Для проверки авторизации используется middleware, которое проверяет, находится ли запрашиваемый ресурс в списке защищаемых, и в случае, если для доступа к данному ресурсу необходима роль соискателя, работодателя или администратора, производит авторизацию и аутентификацию.

Все запросы, для которых не был зарегистрирован обработчик, сначала обрабатываются middleware, обрабатывающим запросы на статические файлы. Если и это middleware не вернуло ответ, возвращается страница React-приложения.

3 Разработка web-приложения

3.1 Разработка бэкэнда

Для разработки бэкэнда был использован фреймворк h3. В нём для обработки запросов могут применяться простые обработчики запросов. Обработчики запросов, добавляющие к ответу заголовки CORS и возвращающие html-страницу на любой запрос, представлены в листинге 3.1.

|  |
| --- |
| app.use(defineEventHandler(event => {  appendHeaders(event, {  'Access-Control-Allow-Origin': '\*',  'Access-Control-Expose-Headers': '\*'  })  }));  app.use(defineEventHandler(async event => {  setResponseHeader(event, 'Content-Type', 'text/html');  return fs.readFileSync('./views/react-front/dist/index.html');  })); |

Листинг 3.1 – Простые обработчики запросов

Также для обработки запросов могут использоваться более продвинутые роутеры. Фрагмент роутера masterRouter представлен в листинге 3.2.

|  |
| --- |
| const masterRouter = createRouter()  .get('/public-companies', defineEventHandler(async event => {  const query = getQuery(event);  const companies = await BOURGEOISIE.findAndCountAll({ attributes: ['id', 'name'] })  companies.rows = companies.rows.map(e => e.dataValues);  if (!query.skipRating) {  for (let company of companies.rows) {  company.rating = await GetRating('C', company.id);  }  }  return { companies: companies.rows, totalElements: companies.count };  }))  .get('/logout', defineEventHandler(async event => {  await logout(event);  return { message: 'what kind of message did you expect?' };  })); |

Листинг 3.2 – Фрагмент роутера masterRouter

Объекты запроса и ответа объединены в объект event, над которым можно производить различные операции, такие как получение и установка cookie, чтение тела запроса, установка тела ответа при помощи оператора return и так далее.

Для запуска сервера необходимо создать объект приложения, определить обработчики запросов и роутеры, преобразовать объект приложения к слушателю событий Node.js и запустить сервер при помощи метода listen. Код запуска сервера представлен в листинге 3.3.

|  |
| --- |
| import { createServer } from "node:https";  import { createApp, toNodeListener } from "h3";  import { masterRouter } from './routers/master.mjs';  export const app = createApp();  app.use(masterRouter);  const nodeApp = toNodeListener(app);  const credentials = {  key: fs.readFileSync('../server.key'),  cert: fs.readFileSync('../server.crt')  }  const httpsServer = createServer(credentials, nodeApp);  httpsServer.listen(process.env.PORT || 3000); |

Листинг 3.3 – Код запуска сервера

Для обработки запросов на WebSocket-соединение используется специальный адаптер для Node.js от Crossws. Функция обработки WebSocker-соединений представлена в листинге 3.4.

|  |
| --- |
| import wsAdapter from "crossws/adapters/node";  const { handleUpgrade } = wsAdapter({      hooks: {        async open(peer) {          console.log("[ws] open", peer);          vacancyEmitter.on('changed', vacancy => { peer.send(vacancy); console.log('sending') });        },          message(peer, message) {          console.log("[ws] message", peer, message);          if (message.text().includes("ping")) {            peer.send("pong");          }        },        close(peer, event) {console.log("[ws] close", peer, event);        },        error(peer, error) {console.log("[ws] error", peer, error);        },      },    }); |

Листинг 3.4 – Функция обработки WebSocker-соединений

Файлы, требующиеся для работы фронтэнда, такие как таблицы стилей и скрипты, запрашиваются браузером и должны обрабатываться отдельно. Для этого предусмотрена функция handleStatic, которая проверяет запрашиваемый ресурс на соответствие шаблону и в случае, если путь к ресурсу оканчивается на .css, .js или .jpg, ищет нужный файл в специальной папке на сервере. Код данной функции представлен в листинге 3.5.

|  |
| --- |
| export function handleStatic(event) {  const regex = /.(css|js|jpg)$/;  if (!regex.test(event.path)) return;  return serveStatic(event, {  getContents: async id => {  return await readFile(join(publicDir, id))  },  getMeta: async id => {  const stats = await stat(join(publicDir, id)).catch(() => {});  if (!stats || !stats.isFile()) {  return;  }  const extension = id.split('.').pop();  return {  type: extension === 'js' ? 'text/javascript'  : extension === 'css' ? 'text/css'  : extension === 'html' ? 'text/html'  : 'image/jpeg',  size: stats.size,  mtime: stats.mtimeMs  };}});} |

Листинг 3.5 – Функция обработки запросов на статические файлы

Для получения данных из базы данных использовалась ORM Sequelize. Применялся подход model-first, при котором в первую очередь разрабатывались модели данных, а затем в соответствии с ними создавались таблицы и обработчики запросов. Были определены модели для соискателей, резюме, работодателей, вакансий, откликов, отзывов, запросов на подтверждение, запросов на удаление учётных записей, токенов и записей чёрного списка. Объявление модели запросов на подтверждение представлено в листинге 3.6.

|  |
| --- |
| class PROMOTION\_REQUESTS extends Model {}  PROMOTION\_REQUESTS.init({  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true  }, company\_id: {type: DataTypes.INTEGER,allowNull: false,  references: {model: 'BOURGEOISIE',key: 'id'}},  proof: {type: DataTypes.STRING(125000)}}, {  sequelize,  timestamps: false,  modelName: 'PROMOTION\_REQUESTS',  tableName: 'PROMOTION\_REQUESTS'  }); |

Листинг 3.6 – Функция обработки запросов на статические файлы

Для поиска, создания, обновления и удаления записей в базе данных применялись методы моделей findOne (findByPk, findAndCountAll, findAll), create, update и destroy соответственно.

Для отправки почты использовался пакет nodemailer и сервис mail.ru. Код функции для отправки почты представлен в листинге 3.7.

|  |
| --- |
| export function sendMail(to, subject, text) {  let options = {  service: 'mail.ru',  auth: {  user: config.mailUsername,  pass: config.mailPassword  }  }  nodemailer.createTransport(smtpTransport(options)).sendMail({  from: config.mailUsername, to, subject, text  }, function(error){  if (error) {  log('email sending error: ' + JSON.stringify(error));  }  });  } |

Листинг 3.7 – Функция отправки электронных писем

Данные для авторизации в почтовом сервисе, а также секретные строки для access-токена и refresh-токена и соль для хеширования паролей хранятся в конфигурационном файле в формате JSON.

3.2 Разработка фронтэнда

Для разработки сайта использовалась библиотека react в качестве базы, пакет react-router-dom для создания нескольких страниц с разными URI и библиотека mui-material с готовыми компонентами для упрощения разработки. Фрагмент роутера фронтэнд-приложения представлен в листинге 3.8.

|  |
| --- |
| const router = createBrowserRouter([  {  path: '/',  element: <Root />,  errorElement: <ErrorPage/>,  children: [  { index: true,  element: <Index />,  loader: indexLoader,  },  {  path: 'cv',  element: <CVs/>,  loader: cvsLoader,  }, |

Листинг 3.8 – Фрагмент роутера фронтэнд-приложения

За каждую страницу отвечает свой компонент, который запрашивает данные с сервера и отображает их на странице. Для запроса данных с сервера была разработана специальная функция-декоратор, которая запрашивает данные с определённого адреса и в зависимости от возвращённого статуса ответа либо возвращает данные, либо перенаправляет пользователя на главную страницу или на страницу выхода из учётной записи в случае ошибки. Код данной функции представлен в листинге 3.9.

|  |
| --- |
| export async function fetchForLoader(path) {  return fetch(path).then(r => {  if (r.ok) return r.json();  else throw r.json();  })  .catch(async err => {  err = await err;  console.log(err);  if (err.code === 401) {  location.href = '/signout';  } else if (err.code === 403) {  location.href = '/';  }  })  .then(d => {  return d;  });} |

Листинг 3.9 – Функция запроса данных с сервера для загрузчика

Для отправки данных на сервер была разработана другая функция-декоратор fetchWithResult. Её код представлен в листинге 3.10.

|  |
| --- |
| export async function fetchWithResult(path, options, showAlert, onSuccess, onError) {  fetch(path, options)  .then(r => {  if (r.ok) return r.json();  else throw r.json();  })  .then(d => {  showAlert(d.message, 'success');  if (onSuccess) onSuccess(d);  })  .catch(async err => {  err = await err;  console.log(err);  showAlert(err.message, 'error');  if (onError) onError(err);  });  } |

Листинг 3.10 – Функция для отправки данных на сервер

Данная функция дополнительно принимает параметры showAlert (функция отображения на странице всплывающего сообщения) и функции onSuccess и onError, вызывающиеся при успешном и неудачном получении данных с сервера соответственно. Сообщение, показываемое при помощи showAlert, отображает сообщение, которое вернул сервер. Также оно цветом отображает, был ли запрос выполнен успешно, или произошла ошибка.

Также для отображения номеров страниц в случаях, когда содержимое не помещается на одну страницу, применяется функция CustomPagination. Она использует компонент Pagination из библиотеки mui-material. Её код представлен в листинге 3.11.

|  |
| --- |
| export function CustomPagination(  query,  totalElements,  callback) {  return (<Box sx={{  display: 'flex',  justifyContent: 'center',  width: '100%'}}>  <Pagination count={Math.ceil(totalElements / 20 || 1)}  page={Math.floor((query.offset ?? 0) / 20) + 1}  onChange={callback}/>  </Box>);  } |

Листинг 3.11 – Функция для отправки данных на сервер

В качестве параметров данная функция принимает параметры строки запроса текущей страницы, общее количество элементов и функцию обратного вызова, которая вызывается при изменении состояния блока Pagination. Так, при выборе любого номера страницы будет произведён переход на данную страницу, но с параметром offset, равным произведению разности номера страницы и единицы и двадцати. Сервер, получив запрос с параметром URI offset, запросит из базы данных строки с заданным смещением.

4 Тестирование web-приложения

Для тестирования web-приложения использовалось ручное и автоматическое тестирование. Были проверены все обработчики запросов всеми HTTP-методами с различными параметрами. Фрагмент приложения для тестирования представлен в листинге 4.1.

|  |
| --- |
| process.env.NODE\_TLS\_REJECT\_UNAUTHORIZED = "0";  (async () => {  for (let method of ['GET', 'PUT', 'POST', 'DELETE']) {  for (let uri of uris) {  for (let param of params) {  const response = await fetch(`https://localhost:4433/${uri}?${params}`, { method });  const contentType = response.headers.get('content-type');  let responseBody;  if (contentType === 'application/json') {  responseBody = await response.json();  } else {  responseBody = await response.text();  }  console.log(method, uri, response.status, responseBody);  }  }  }  })() |

Листинг 4.1 – Фрагмент приложения для тестирования

В ходе тестирования были выявлены и исправлены несоответствия запрашиваемых клиентом и возвращаемых сервером данных, а также ошибки в исходном коде.

5 Руководство пользователя

При первом открытии сайта пользователь видит страницу вакансий, представленную на рисунке 5.1.

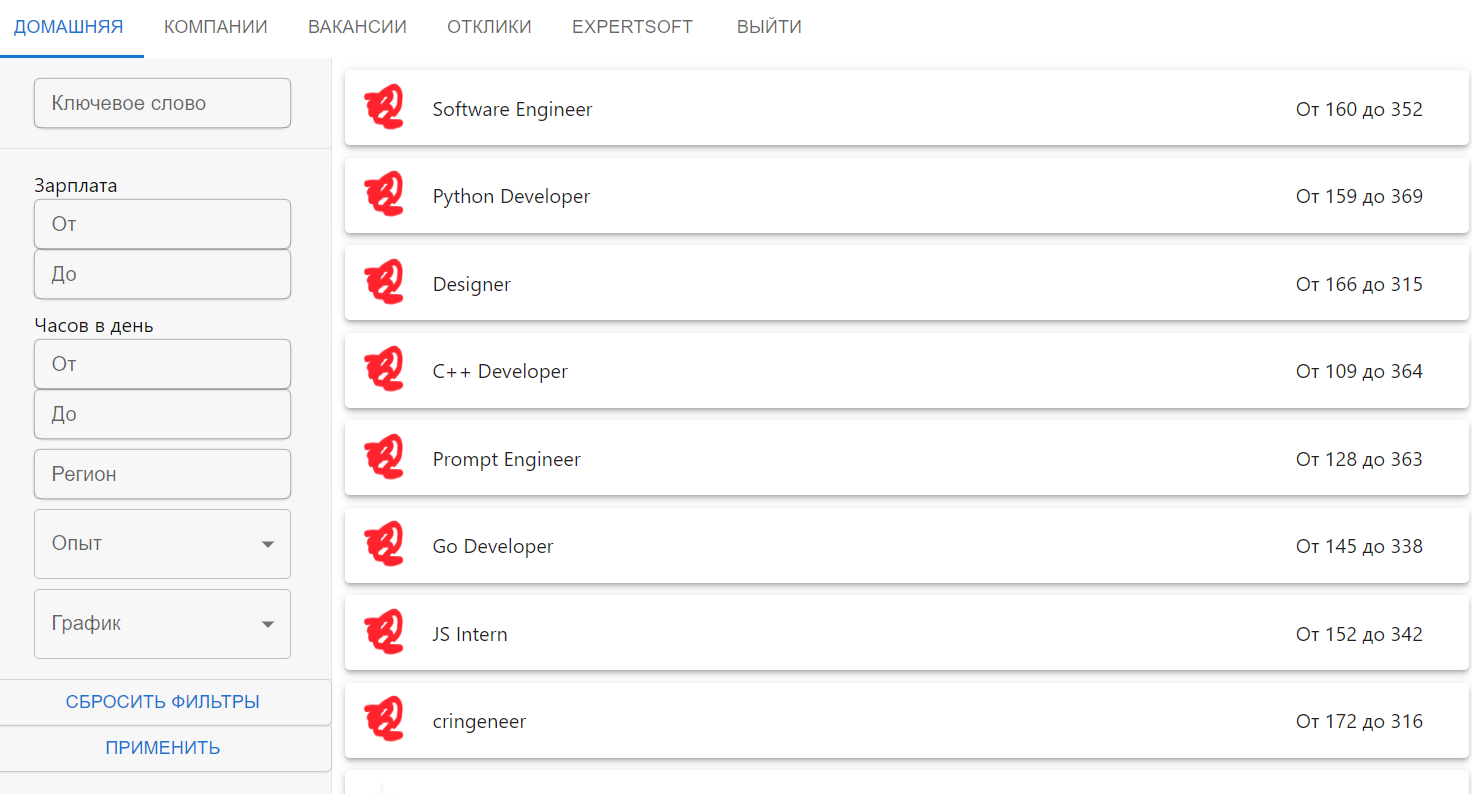


Рисунок 5.1 – Страница публичных вакансий

Чтобы перейти на страницу нужно вакансии, пользователь должен нажать на её карточку. Чтобы отфильтровать вакансии по нужным признакам, пользователь должен ввести желаемые значения в нужные поля блока фильтрации в левой части страницы и нажать на кнопку «Применить». Для сброса фильтров пользователь может нажать на соответствующую кнопку.

На странице вакансии, представленной на рисунке 5.2, пользователь может ознакомиться со всеми данными вакансии и перейти на страницу описания работодателя.

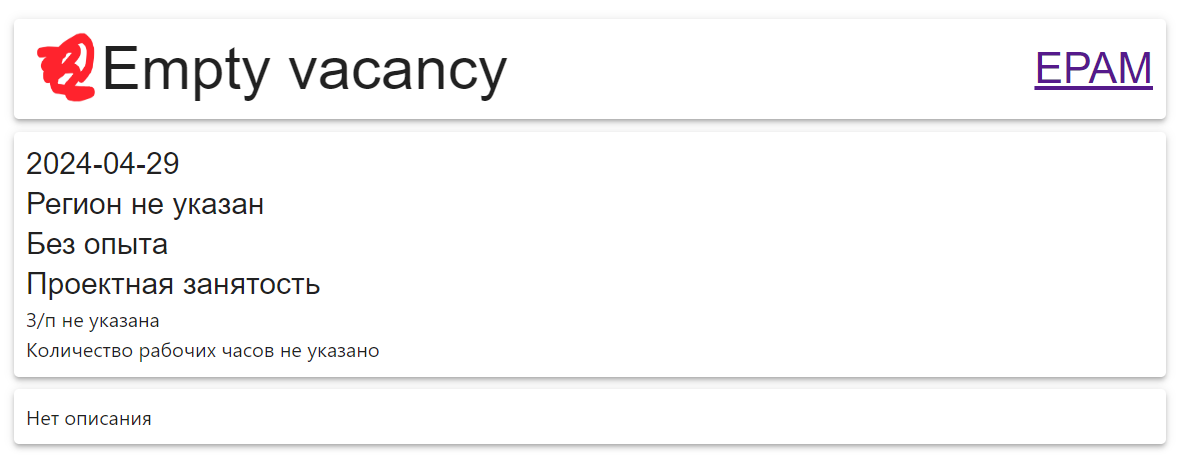


Рисунок 5.2 – Страница вакансии

На странице работодателя пользователь может ознакомиться с его описанием. Для ознакомления с отзывами о работодателе пользователь должен перейти на страницу со списком всех компаний, представленную на рисунке 5.3.

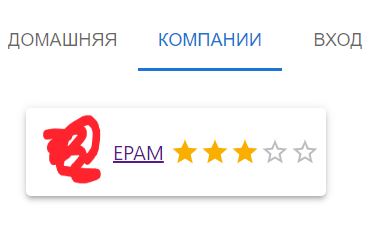


Рисунок 5.3 – Страница всех компаний

По ссылке в карточке пользователь может перейти на страницу с отзывами о компании, представленную на рисунке 5.4.

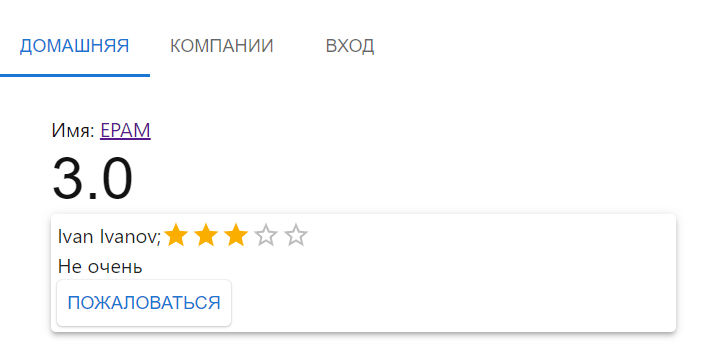


Рисунок 5.4 – Страница отзывов о компании

С данной страницы пользователь может попасть на страницу описания компании и пожаловаться на отзыв нажатием на соответствующую кнопку в карточке отзыва.

Для входа в учётную запись или регистрации пользователь должен перейти на страницу Вход, представленную на рисунке 5.5.

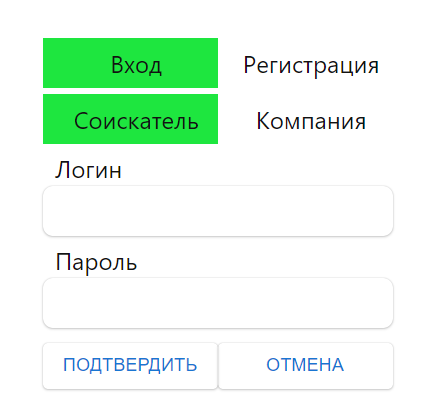


Рисунок 5.5 – Страница входа и регистрации

Пользователь должен нажатием на соответствующие блоки выбрать, что он хочет сделать: войти в учётную запись или зарегистрироваться – и в качестве кого: соискателя или работодателя – и нажать на кнопку «Подтвердить» для совершения действия или нажать на кнопку «Отмена» для перехода на главную страницу.

Работодателю доступны дополнительные страницы. Например, страница вакансий работодателя, представленная на рисунке 5.6.

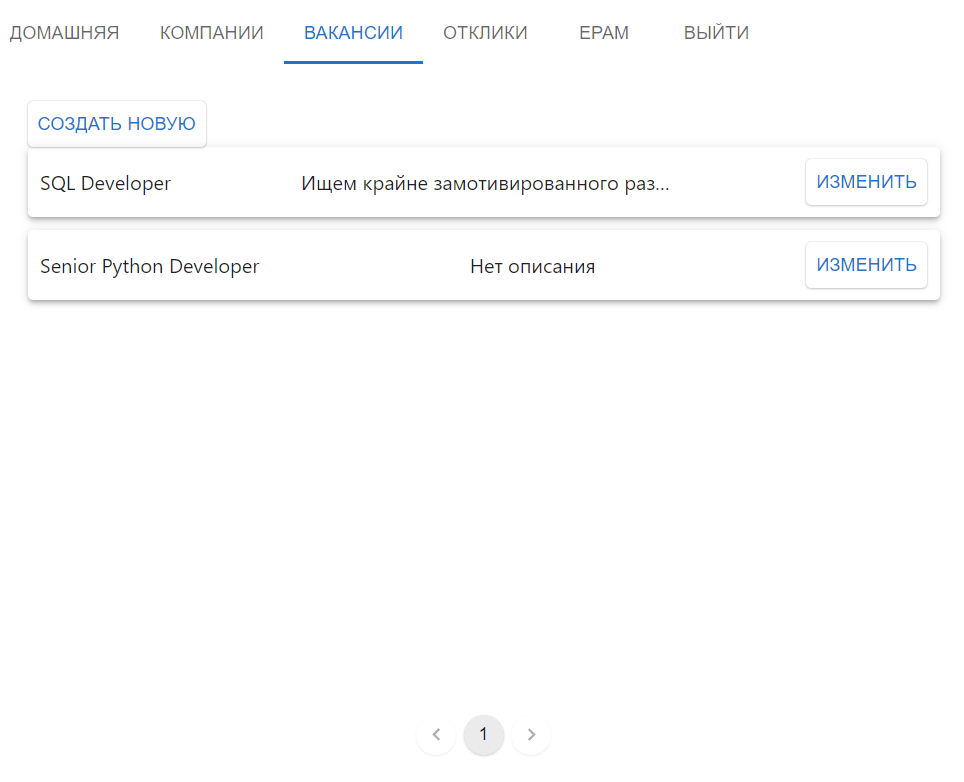


Рисунок 5.6 – Страница вакансий компании

На данной странице работодатель может просмотреть список своих вакансий, создать новую нажатием на соответствующую кнопку и перейти к изменению нужной вакансии.

Также работодателю доступна страница откликов на его вакансии, представленная на рисунке 5.7, на которой работодатель может просмотреть список откликов.

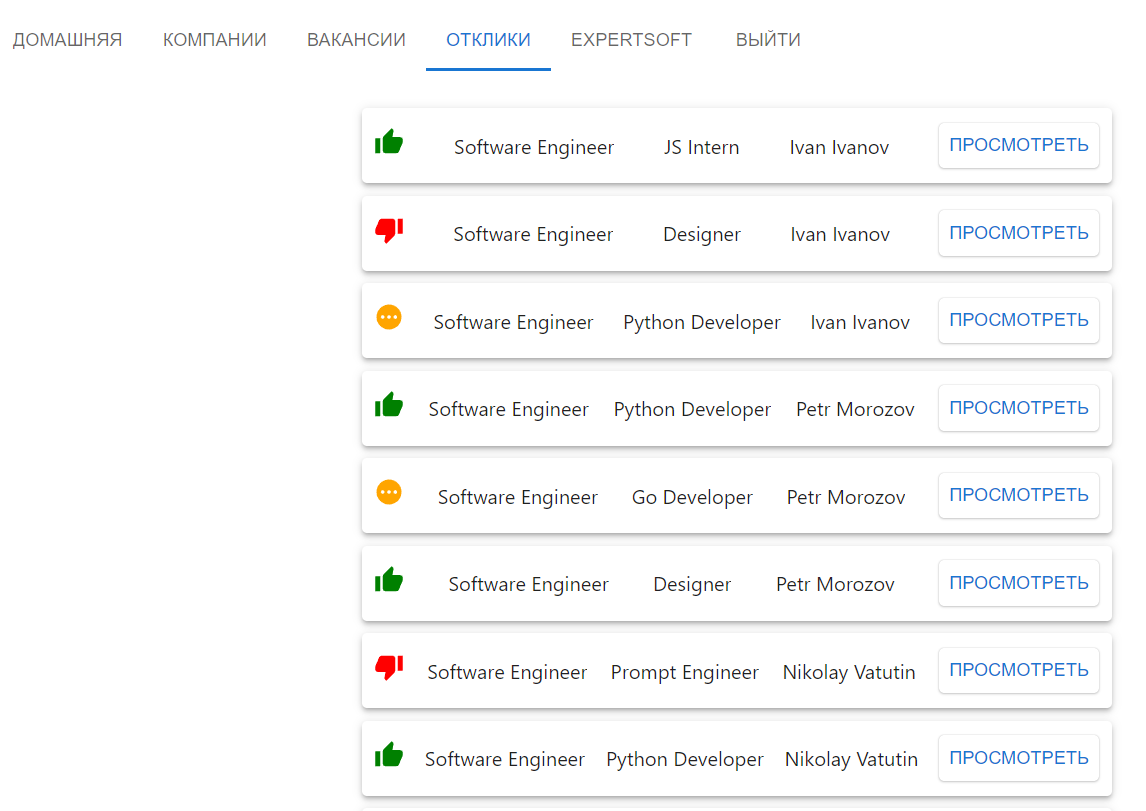


Рисунок 5.7 – Страница откликов компании

Пользователь может вызвать всплывающее окно с подробной информацией о каждом отклике по нажатию на кнопку «Просмотреть», пример которого представлен на рисунке 5.8.

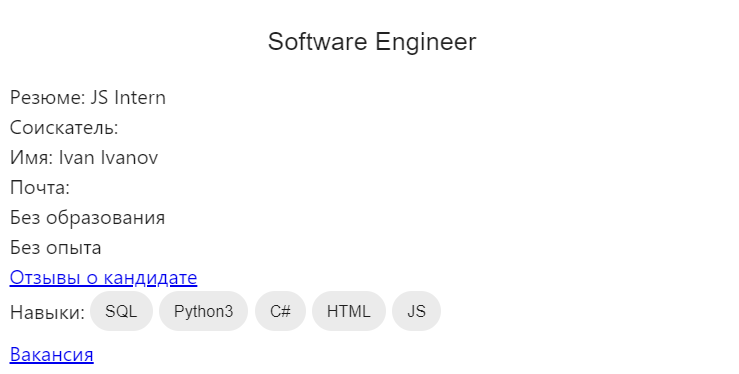


Рисунок 5.8 – Страница откликов компании

Из окна подробностей пользователь может перейти на страницу отзывов о кандидате и на страницу вакансии, на которую был отправлен отклик. Также, если отклик ещё ожидает рассмотрения, работодатель может принять или отклонить его нажатием на соответствующую кнопку.

Соискатель имеет доступ к другим дополнительным страницам. Одной из них является резюме соискателя. На ней пользователь может ознакомиться со списком своих резюме, создать новое и перейти к редактированию нужного резюме по нажатию на ссылку нужного резюме.

Страница редактирования резюме содержит поле ввода названия, область отображения ключевых навыков, поле ввода названия нового навыка, кнопку добавления навыка в список и кнопки сохранения и удаления резюме. В случае, если пользователь введёт название уже присутствующего в резюме навыка, появится уведомляющее об этом сообщение.

Пользователь также может просматривать список своих откликов на странице Отклики. В карточке каждого отклика находится кнопка «Отозвать», нажатие на которую отправляет запрос на удаление отклика.

Соискателям и работодателям доступна страница создания отзывов друг о друге. На данной странице расположен выпадающий список доступных кандидатов для отзыва, поле ввода комментария, поле ввода оценки и кнопка отправки отзыва.

Другой доступной для всех авторизованных пользователей страницей является страница персональных данных. На ней пользователи могут изменить своё имя и адрес электронной почты, соискатели дополнительно могут изменить данные о своём образовании и опыте работы, а работодатели могут изменить описание и иконку и отправить запрос о подтверждении компании, по желанию приложив текстовый файл со своим комментарием. Также соискателям и работодателям доступна отправка запроса на удаление учётной записи.

Администратору доступен другой набор страниц. Он может просматривать список запросов на подтверждение компаний, получать комментарий по каждому из них и отклонять либо принимать их нажатиями на соответствующие кнопки. Точно так же администратор может взаимодействовать со списком запросов на удаление учётных записей.

Ещё одной доступной только администратору страницей является страница жалоб на отзывы. На ней расположены карточки жалоб, содержащие тип пользователя, оставившего отзыв, его имя, имя того, на кого был оставлен отзыв, и текст отзыва. Администратор может удалить комментарий, заблокировать возможность автору отзыва впредь оставлять отзывы о данном пользователе и удалить жалобу.

Последней доступной только администратору страницей является страница чёрного списка, на которой отображаются записи о заблокированных пользователях. Администратор может снять блокировку возможности нужному пользователю оставлять отзывы о другом указанном пользователе, нажав на кнопку «Разблокировать» в карточке нужной записи чёрного списка.

Заключение

При выполнении курсового проекта было создано приложение поиска и предложения работы. Сервер был создан при помощи платформы Node.js, языка программирования JavaScript и фреймворка h3. Web-сайт был реализован при помощи библиотеки React. База данных была реализована в СУБД PostgreSQL. Были реализованы все функциональные требования, а именно:

– обеспечивать возможность регистрации и авторизации;

– поддерживать роли администратора и обычного пользователя;

– позволять изменять образование, опыт работы и список ключевых навыков

соискателя;

– позволять работодателю принимать и отклонять отклики;

− позволять оставлять отзывы об исполнителе и работодателе;

− предоставлять возможность отслеживать статус предложения о работе в

реальном времени;

− позволять соискателю откликаться на предложения о работе;

− позволять создавать и удалять учётные записи;

− позволять фильтровать предложения по критериям (тип, необходимый опыт,

оплата и так далее);

− позволять размещать и удалять вакансии и заказы;

− поддерживать роли работодателя и соискателя.

Также были реализованы WebSocket-сервер и поддержка протокола HTTPS. Приложение было протестировано на наличие ошибок.

Список используемых источников

1. PostgreSQL Documentation [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.postgresql.org/docs/. Дата доступа: 20.04.2024.
2. Material UI components [Электронный ресурс]. – Электронные данные. – Режим доступа: https://mui.com/material-ui/all-components/. Дата доступа: 21.04.2024.
3. h3 – The Web Framework for Modern JavaScript Era [Электронный ресурс]. – Электронные данные. – Режим доступа: https://h3.unjs.io. Дата доступа: 23.04.2024.
4. Enabling HTTPS on express.js [Электронный ресурс]. – Электронные данные. – Режим доступа: https://stackoverflow.com/questions/11744975/enabling-https-on-express-js. Дата доступа: 15.04.2024.

Приложение А

Листинг создания таблиц в базе данных

|  |
| --- |
|  |

Приложение Б

Листинг создания процедур

|  |
| --- |
|  |

Приложение В

Листинг создания пользователей и выдачи им необходимых привилегий

|  |
| --- |
|  |

Приложение Г

Листинг функций импорта-экспорта приложения

|  |
| --- |
|  |