|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Дисциплина «Разработка интернет-приложений»

Расчетно-пояснительная записка

Тема: «Сервис для работодателей»

Студент: Перевощиков Н.Д.

Группа ИУ5Ц-74Б

Преподаватель: Канев А.И.

2023г.

# **ВВЕДЕНИЕ**

Разработка сервиса подбора персонала для работодателя является на данный момент актуальной темой в связи с тем, что в прошедшем 2023 году одной из главных проблем для бизнеса и экономики России в целом стал дефицит кадров. Спусковым крючком дефицита рабочей силы стал период пандемии коронавируса в 2019—2021 годы: резко сократилось число мигрантов, занимающих рабочие позиции. По данным Росстата за 2020 год фактическое выбывание трудовых мигрантов составило 43%. В феврале 2022 года началась специальная военная операция. За 2022 год число мигрантов сократилось на 194,6 тысячи человек и 5,9% по сравнению с 2021 годом. Сентябрь 2022 года принес мобилизацию и массовый отток уже российских граждан в страны СНГ. Россию покинуло около 700 тысяч человек, не считая тех, кто попал под мобилизацию или заключил контракт на добровольных началах. Последствия кадрового голода не заставили себя долго ждать. Бизнес терпит серьезные потери и издержки, связанные с неукомплектованностью персонала: растет стоимость привлечения сотрудников, резко падает качество сервиса и объем продаж.

Целью работы является создание джобборда – сервиса для работодателей, помогающего быстро и качественно подобрать работников. Автоматизированная платформа рекрутинга будет состоять из веб-приложения, веб-сервиса, удаленного сервиса и мобильного приложения.

Основное назначения разработанной системы заключается в автоматизации и оптимизации процесса поиска работников, по средствам создания вакансий. Джобборд позволит работодателям просматривать всю необходимую информацию о городах, в которых располагаются филиалы их компаний. Также она позволит работодателям упростить процесс поиска новых работников – быстро и легко выкладывать новые вакансии, а администраторам рекрутинговой платформы – принимать / отклонять их. Автоматизированная платформа найма персонала освободит рекрутеров от рутины, а бизнес – от растущих затрат на подбор сотрудников.

Нефункциональные требования к разрабатываемой системе:

1. Система должна поддерживать кроссплатформенность.
2. Система должна поддерживать адаптивность.
3. Интерфейс системы и текст ошибок должны быть русифицируемы.
4. Должны поддерживаться последние версии ПО.

В ходе работы необходимо выполнить следующие задачи:

1. Разработать дизайн системы.
2. Создать базу данных в PostgreSQL на Django.
3. Реализовать веб-сервис на Django REST Framework.
4. Создать интерфейс гостя на React.
5. Развернуть веб-приложение на Gtihub Pages для предварительной демонстрации работы.
6. Добавить авторизацию и аутентификацию, основанную на JWT и Redis.
7. Реализовать интерфейс работодателя на React, с использованием Redux Toolkit.
8. Создать нативное приложение на React Native.
9. Создать удаленный сервис на Go, вычисляющий банкротство для вакансий.
10. Реализовать интерфейс администратора рекрутинговой платформы на React.
11. Подготовить набор документации, включающий РПЗ, ТЗ и набор диаграмм.

# **БИЗНЕС-ПРОЦЕСС**

Процесс поиска и найма новых работников в компаниях довольно сложен и затратен по времени [1]. Кроме этого, он имеет в себе еще кучу подводных камней. Разработанный джобборд позволяет работодателям [2] быстро и удобно выкладывать новые вакансии для поиска работников в выбранных городах. Каждая вакансия вначале создается как черновик, в который можно добавлять и из которого можно удалять города. Одна вакансия может содержать сразу несколько городов.

Города, помимо названия имеют различные характеристики, такие как: описание, фото, дату основания, население, климат и площадь. Администраторы рекрутинговой платформы системы имеют возможность отредактировать информацию о каждом городе. Также система предусматривает возможность добавление новых городов, а также удаление и редактирование уже существующих городов администратором рекрутинговой платформы.

Работодатели могут быстро и удобно создавать новые вакансии, выбирая города, в которых они ищут сотрудников. Когда работодатель определится с городами, в которых нужны работники, он должен указать название вакансии. После этого у него есть возможность либо удалить свою черновую вакансию, либо сформировать ее. Это обеспечивает гибкость в планировании и расширении поиска.

Сформированные вакансии отправляются на рассмотрение администратором рекрутинговой платформы. Перед этим для вакансии автоматически вычисляется банкротство работодателя. После этого вакансии отправляются на рассмотрение администратору рекрутинговой платформы, которые принимают решение об одобрении или отклонении на основе установленных критериев.

Работодатели, могут следить за статусами своих сформированных вакансий в режиме реального времени, на отдельной странице. Функции пользователей с разными ролями описаны на диаграмме прецедентов (рис. 1).



Рисунок 1 - Диаграмма прецедентов

Гостям доступен просмотр городов. Зарегистрированные гости – работодатели. Они могут добавлять города в вакансии, формировать новые вакансии либо же просматривать уже сформированные. Вакансии обрабатываются администраторам рекрутинговой платформы, и они могут принять или отклонить вакансию. Также администраторам рекрутинговой платформы доступно редактирование уже существующих городов, а также добавление новых. Процесс оформления вакансии отражен на диаграмме деятельности (рис. 2).

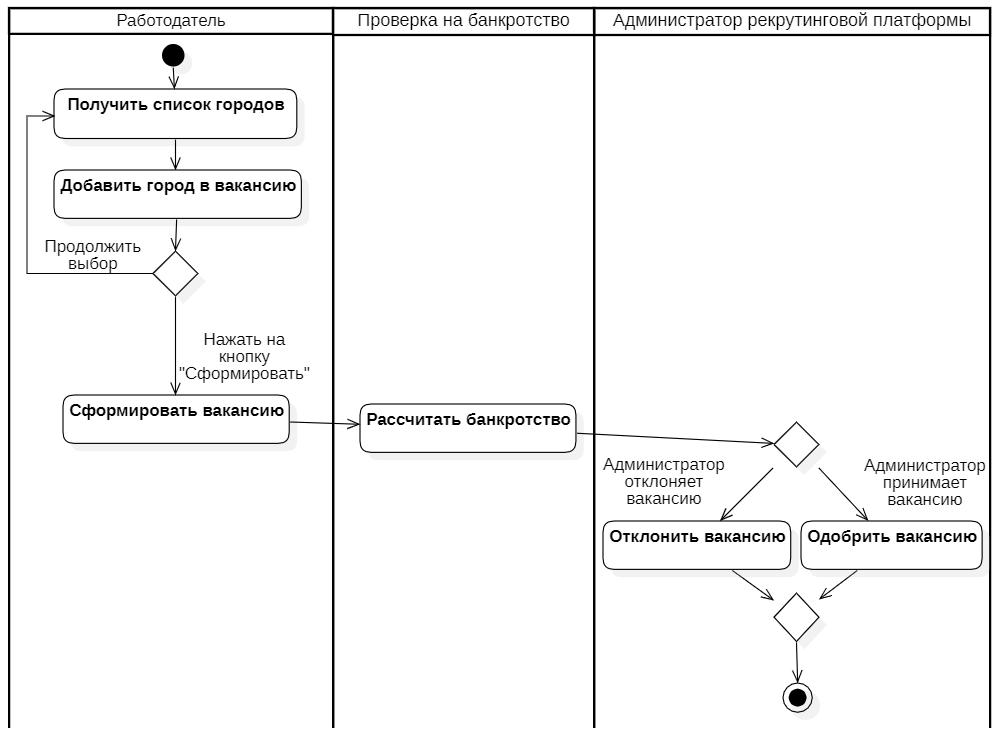


Рисунок 2 - Диаграмма деятельности

Работодатель выбирает города и формирует из них вакансию. После этого асинхронный сервис «Проверка на банкротство» определяет финансовую устойчивость работодателя. Затем администратор рекрутинговой платформы системы решает, принять или отклонить вакансию. Возможные состояния вакансии отражены на диаграмме состояний (рис. 3).

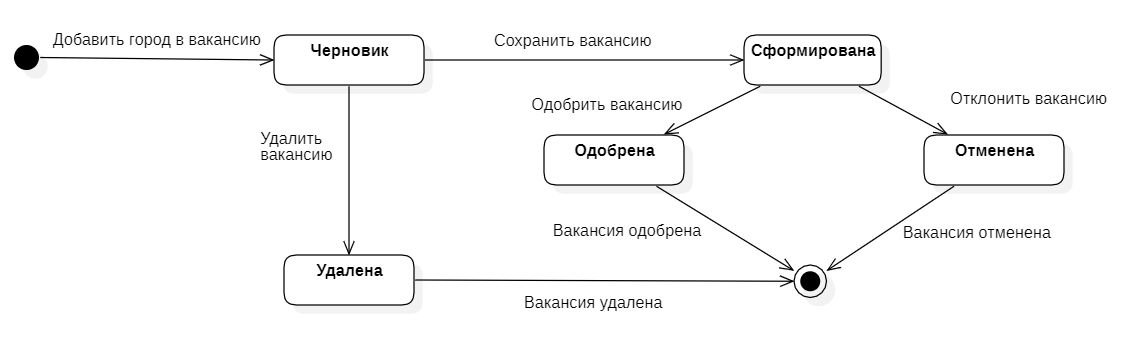


Рисунок 3 - Диаграмма состояний заявки

При выборе первого города у работодателя формируется черновик вакансии. Последующие выбранные города добавляются в этот черновик. Затем работодатель формирует вакансию и удаляет её. Сформированные вакансии обрабатывают администраторы рекрутинговой платформы системы. Они могут, как одобрить, так и отклонить их.

# **АРХИТЕКТУРА**

Архитектура системы отображена на диаграмме развертывания (рис. 4). На этой диаграмме показано, как компоненты системы взаимодействуют друг с другом на самом верхнем уровне. Данные хранятся в СУБД PostgreSQL [3]. Веб-сервис [4, 5] обращается к S3 хранилищу статических файлов Minio и к резидентной базе данных Redis. Браузерное [6] и Нативное [7] приложения обращаются к веб-сервису на базе технологии Django REST Framework через Web-API. Веб-сервис и асинхронный сервис [8] развернуты на 8000 и 5000 портах соответствено. Веб-сервис обращается к удаленному асинхронному сервису через API.

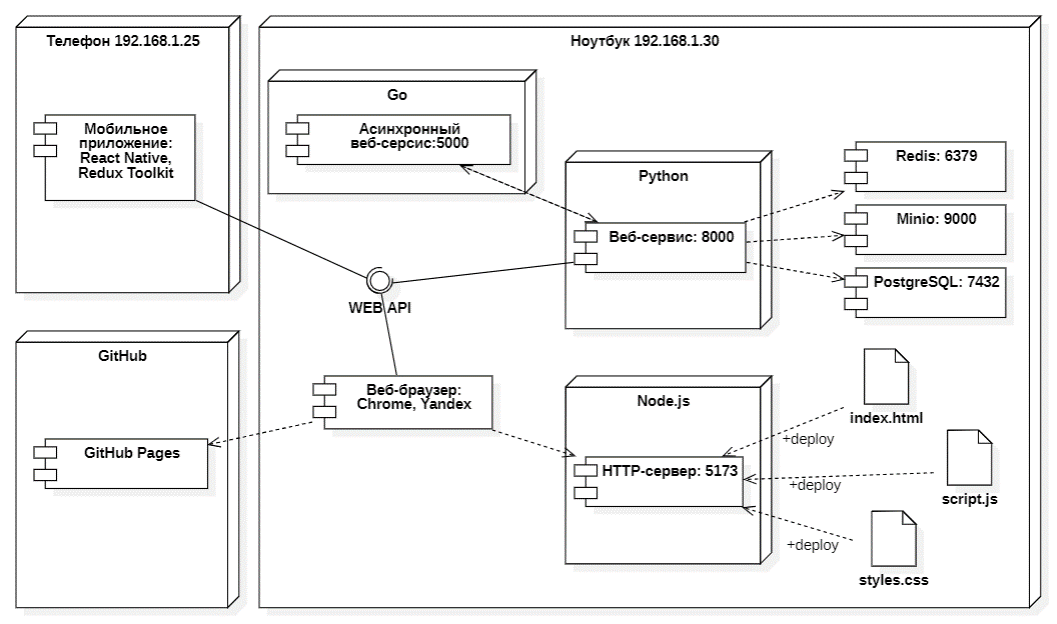


Рисунок 4 - Диаграмма развертывания

Данные хранятся в СУБД PostgreSQL, их структура отражена на ER диаграмме (рис. 5). СУБД PostgreSQL является одним из стандартов индустрии, поэтому было решено использовать её. В разработанной системе было решено использовать логическое удаление городов, за счёт хранения в таблице дополнительного поля «статус» и дальнейшей фильтрации по этому полю при попытке получения записей. Для хранения в одной вакансии нескольких городов используется промежуточную таблица, реализующая связь М-М.

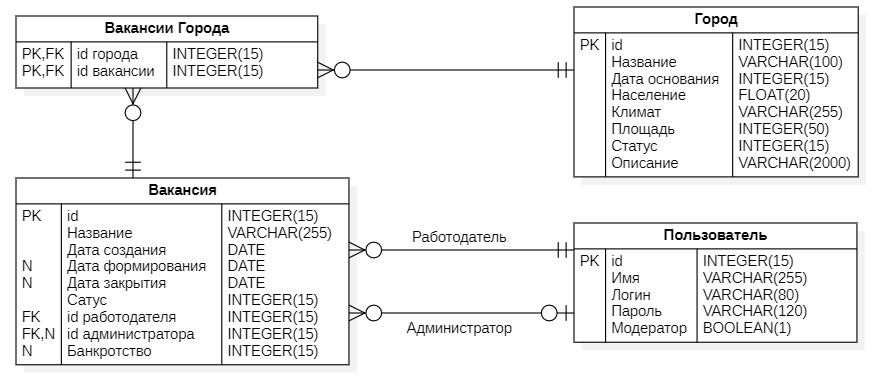


Рисунок 5 - ER диаграмма

На диаграмме классов отображена архитектура веб-сервиса (рис. 6). Указаны методы API, которые он предоставляет, а также его взаимодействие с базой данных. Веб-сервис предоставляет Web-API, которое делится на домен городов, домен вакансий и домен пользователей. Каждый домен содержит методы, которые обращаются к соответствующим таблицам в базе данных с помощью ORM моделей, предоставляемых Django. Также некоторые модели имеют связи с внешними сервисами. В частности, города имеют связь с сервером статических файлов, т.к. в городах хранится ссылка на их изображение, хранимое на сервере статических файлов.

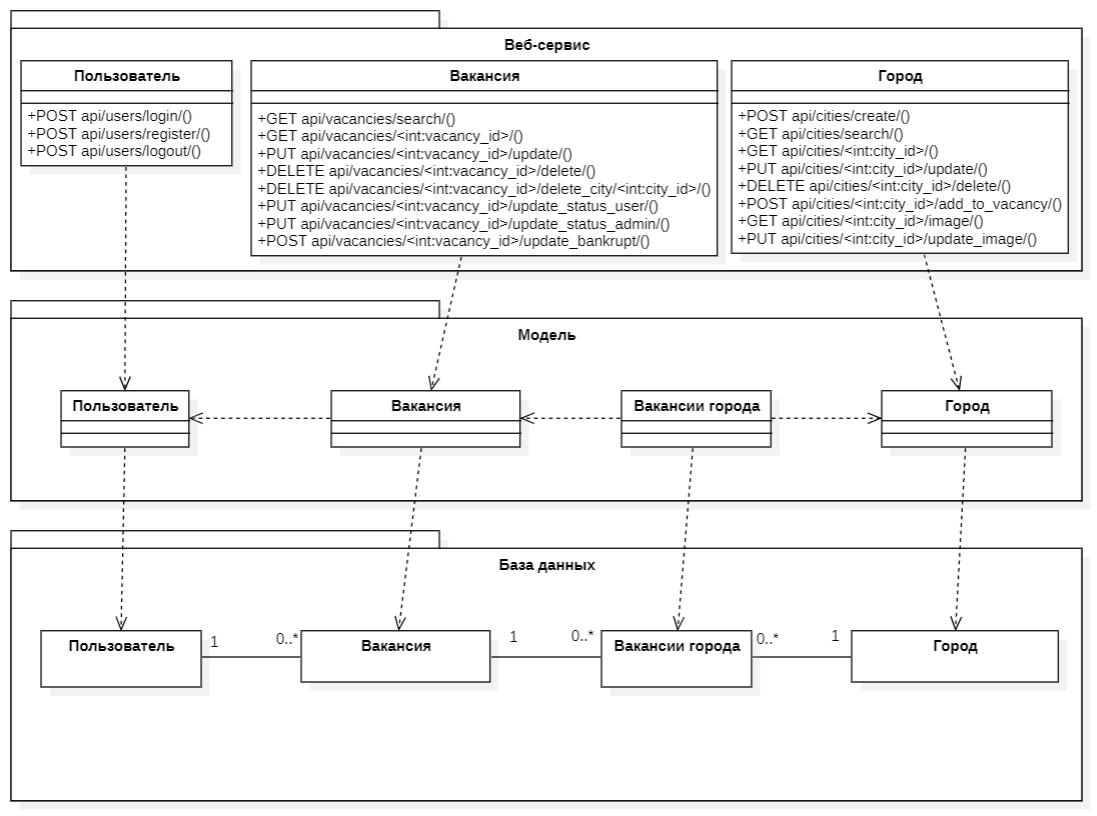


Рисунок 6 *-* Диаграмма классов бэкенда

Связь между фронтендом и бэкендом отражена на диаграмме фронтенда (рис. 7). Страницы со списком городов и с одним городом обращаются к API городов. Страницы со списком вакансий и добавлением новой вакансии обращаются к API вакансий. Страница с аутентификацией и регистрацией обращаются к API пользователей. Ключевые страницы имеют связь с API пользователей, т.к. доступ к ним осуществляется только для авторизированных пользователей с определенными правами (ролями).



Рисунок 7 - Диаграмма классов фронтенда

# **АЛГОРИТМЫ**

Алгоритм работы системы отображен на диаграмме последовательностей (рис. 8). В основе системы лежит веб-сервис, реализующий внутри себя всю бизнес-логику. Он предоставляет доступ к методам из следующих доменов: города, вакансии и пользователи. Методы следуют правилам REST API.

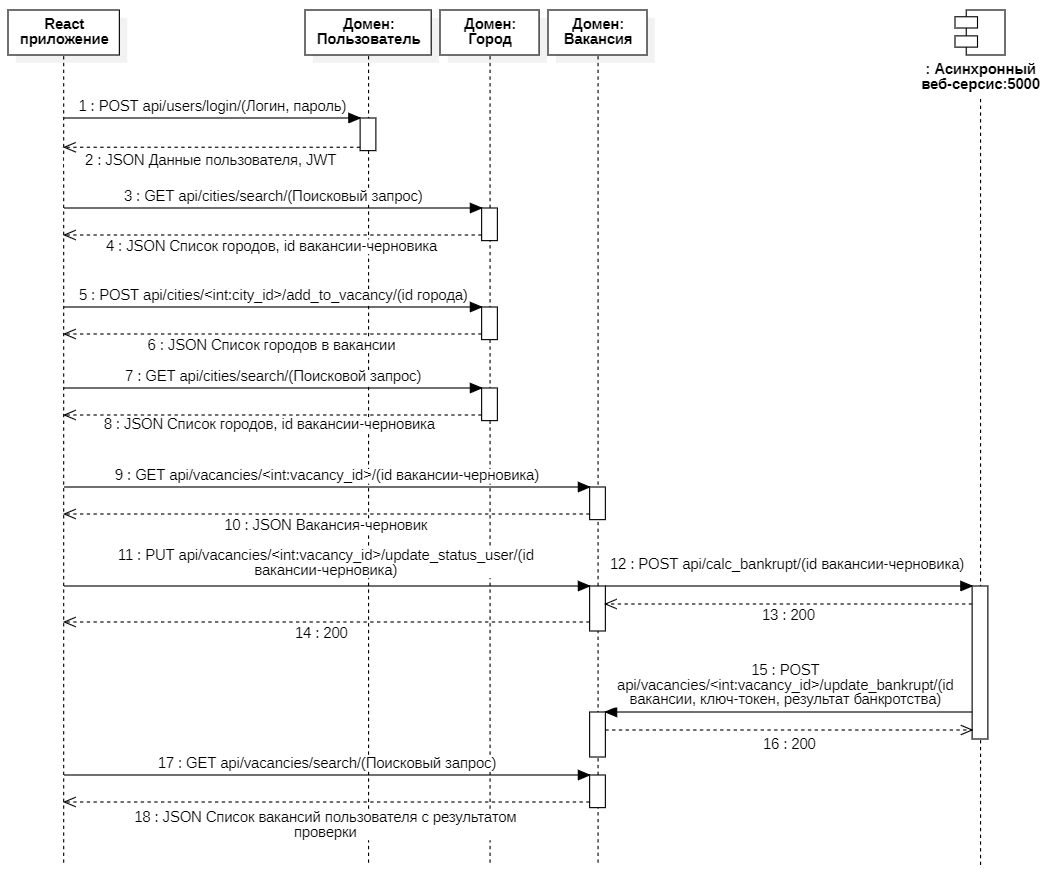


Рисунок 8 - Диаграмма последовательностей

В начале бизнес-процесса происходит аутентификация пользователя. Для этого он отправляет через графический интерфейс запрос, передавая в нем логин и пароль. Если аккаунт с такими данными существует, то пользователь получает JWT токен в ответном запросе и происходит переадресация на главную страницу сайта. Если же такого аккаунта не существует, или данные введены неверно, пользователь получит ошибку. В таком случае ему надо либо пройти регистрацию, либо ввести верный пароль. Когда пользователь переходит на страницу со списком городов, отправляется запрос на веб-сервис, который возвращает список городов в JSON формате, а также id черновой вакансии, если такое имеется. Пользователь выбирает город, который хочет добавить в вакансию, и нажимает на кнопку «добавить», после чего отправляется запрос на добавление города в черновую вакансию. Этот процесс может продолжаться несколько раз, пока пользователь не добавит все интересующие его города.

Когда пользователь определится с выбором городов, он нажимает на кнопку «сформировать». После этого приложение запрашивает id черновой вакансии пользователя и затем отправляет запрос на формирование этой вакансии. В этот момент основной веб-сервис отправляет асинхронный запрос к удаленному сервису на определение банкротства. После формирования вакансии, пользователь будет перенаправлен на страницу с его списком вакансий, где сразу же появится его сформированная вакансия, а через несколько секунд у вакансии отобразится вычисленный результат проверки на банкротство.

Процесс рассмотрения вакансий администратором рекрутинговой платформы происходит через графический интерфейс. Администраторы рекрутинговой платформы системы могут просматривать списки всех вакансий и, нажимая на соответствующие кнопки, отправлять запросы на одобрение или отклонение вакансий в основной веб-сервис. Также администраторам рекрутинговой платформы доступна фильтрация вакансий по статусам, дате формирования и пользователям. Помимо этого, администраторы рекрутинговой платформы могут редактировать, создавать или удалять города. Для каждой из этих функций присутствует свой метод, отправляемый на основной веб-сервис.

# **ОПИСАНИЕ ИНТЕРФЕЙСА**

Страница с формой авторизации (рис. 9)

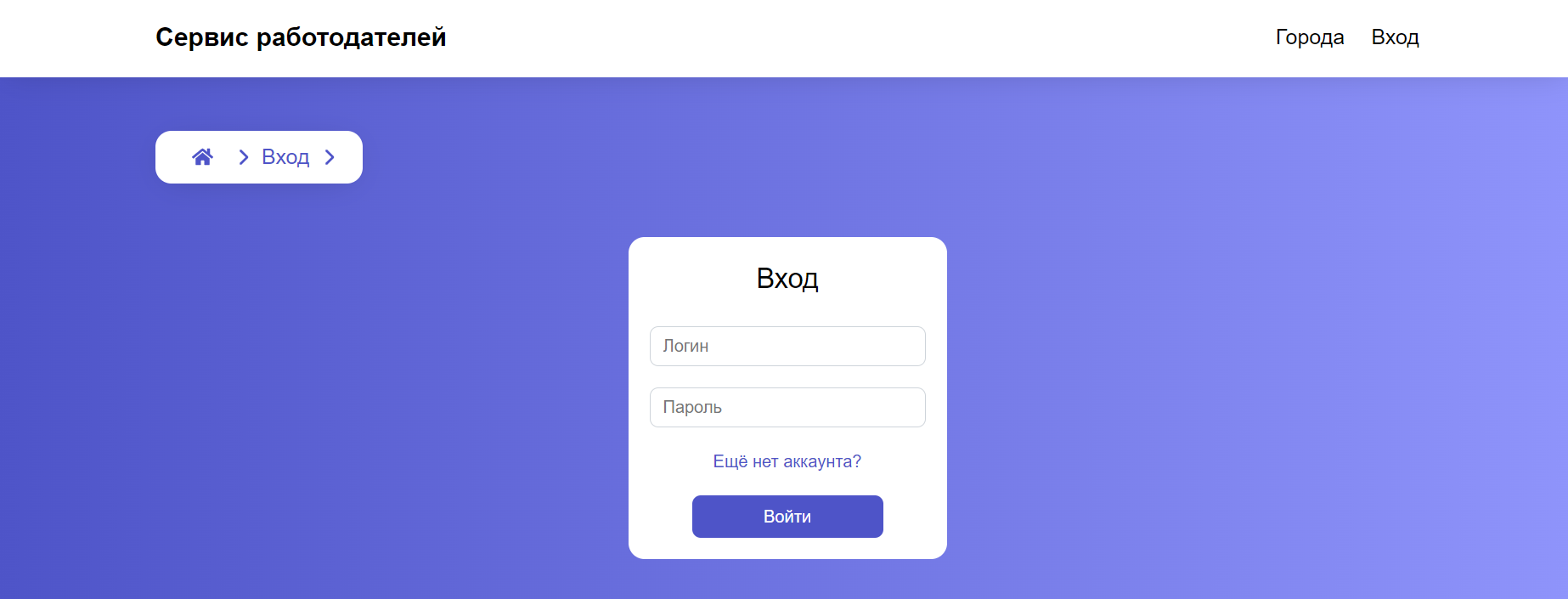


Рисунок 9 - Страница авторизации

Для авторизации на сайте пользователь должен ввести свой логин и пароль. При успешном вводе логина и пароля гость получает JWT токен, который сохраняется в cookies и используется в дальнейшем для авторизации при отправлении запросов.

Страница с формой регистрации (рис. 10)

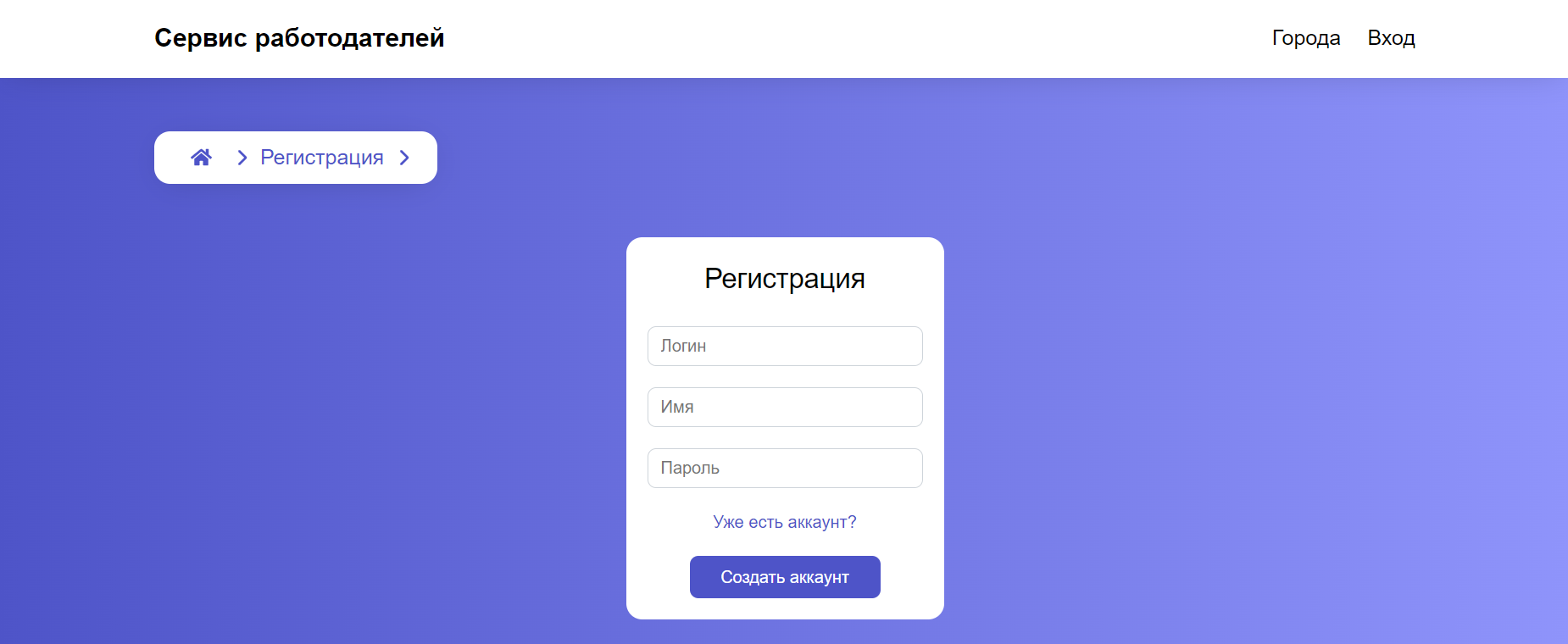


Рисунок 10 - Страница регистрации

Для регистрации на сайте пользователь должен ввести своё имя, логин и пароль. После успешной регистрации гость получает JWT токен, который сохраняется в cookies и используется в дальнейшем для авторизации при отправлении запросов.

Страница со списком городов (рис. 11)

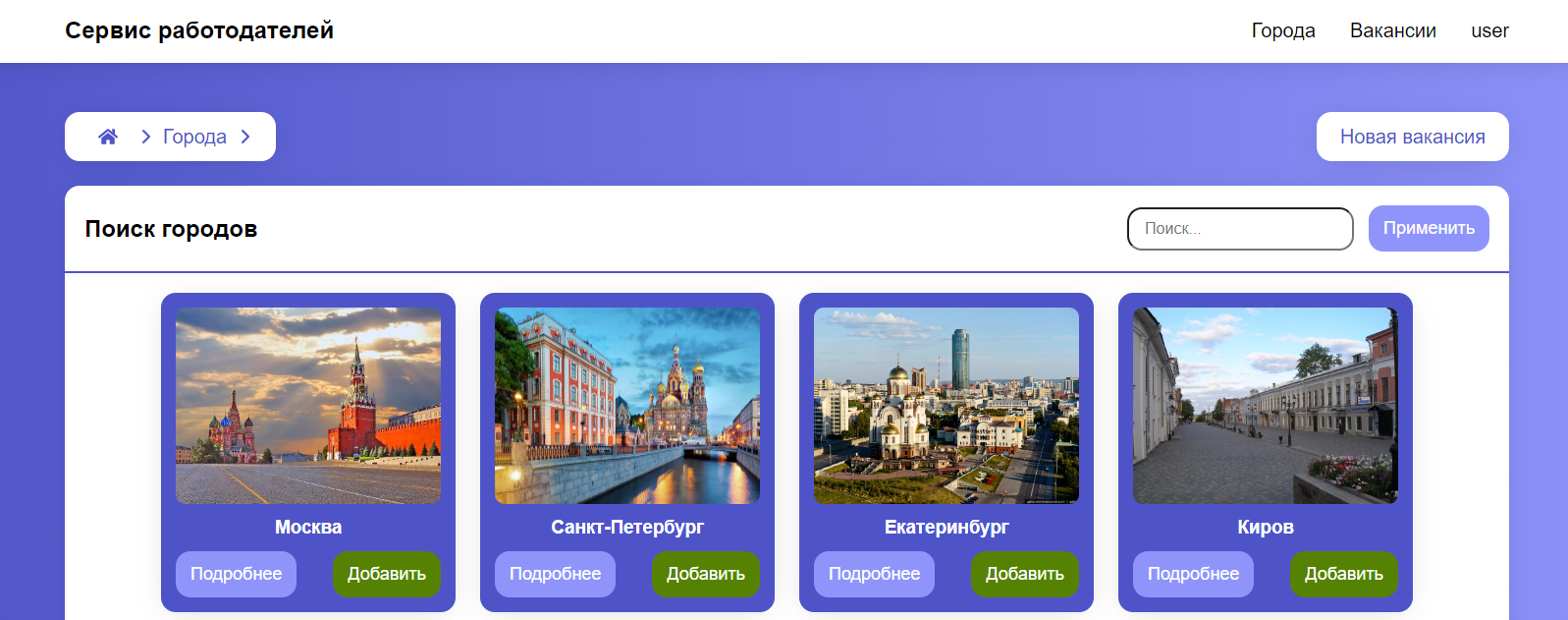


Рисунок 11 - Страница со списком городов

Данная страница содержит список городов в виде карточек. Доступна фильтрация городов по названию. Каждая карточка имеет две кнопки. Кнопка «Подробнее» перенаправляет пользователя на страницу с подробной информацией об города. А с помощью кнопки «Добавить» пользователь может добавить город в черновую вакансию.

Вид страницы со списком городов на экране телефона (рис. 12)

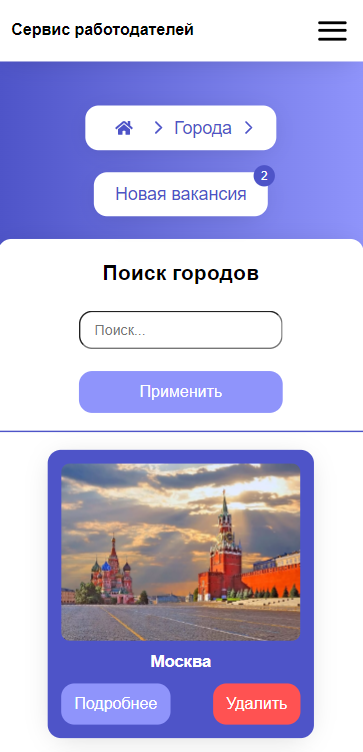


Рисунок 12 - Страница со списком городов (адаптивность)

Страница с подробным описанием города (рис. 13)

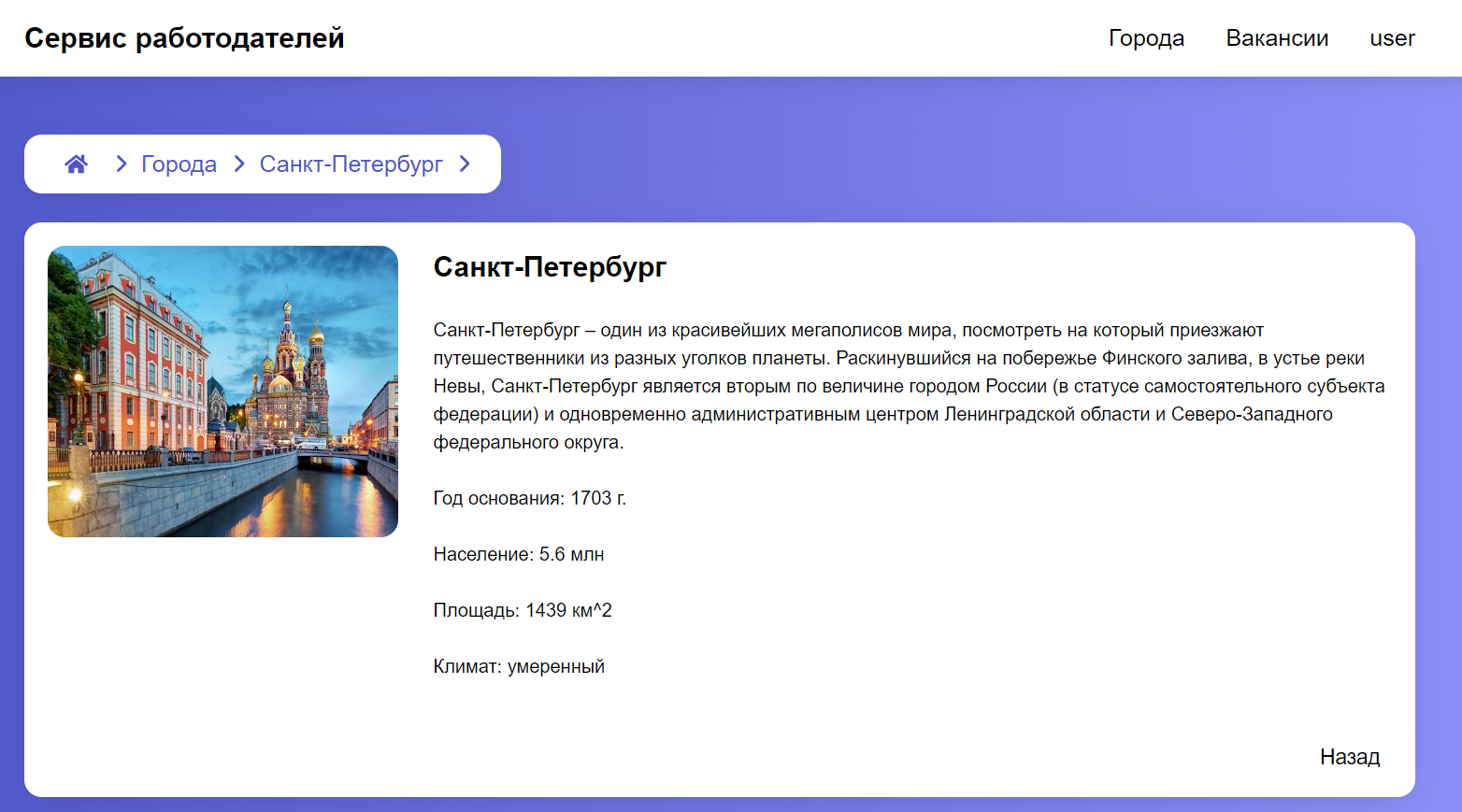


Рисунок 13 - Страница с подробным описанием города

На этой странице находится подробная информация о городе, содержащая название, описание, фото, год основания, население, площадь, а также климат.

Страница формирования вакансии (рис. 14)

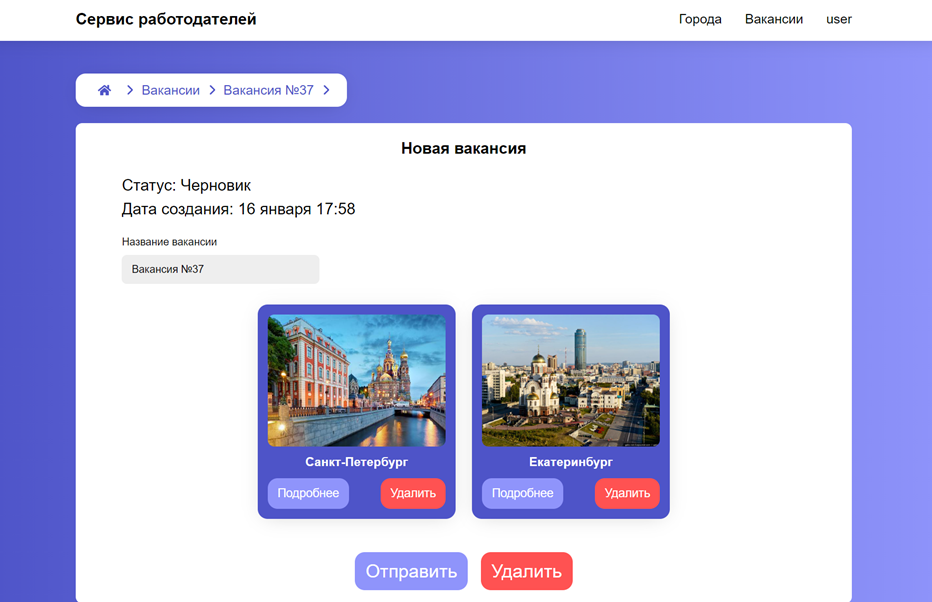


Рисунок 14 - Страница формирования вакансии

На данной странице находится форма добавления новой вакансии. С помощью кнопок «Отправить» и «Удалить» пользователь может сформировать черновую вакансию, либо же удалить его.

Страница со списком вакансий работодателя (рис. 15)

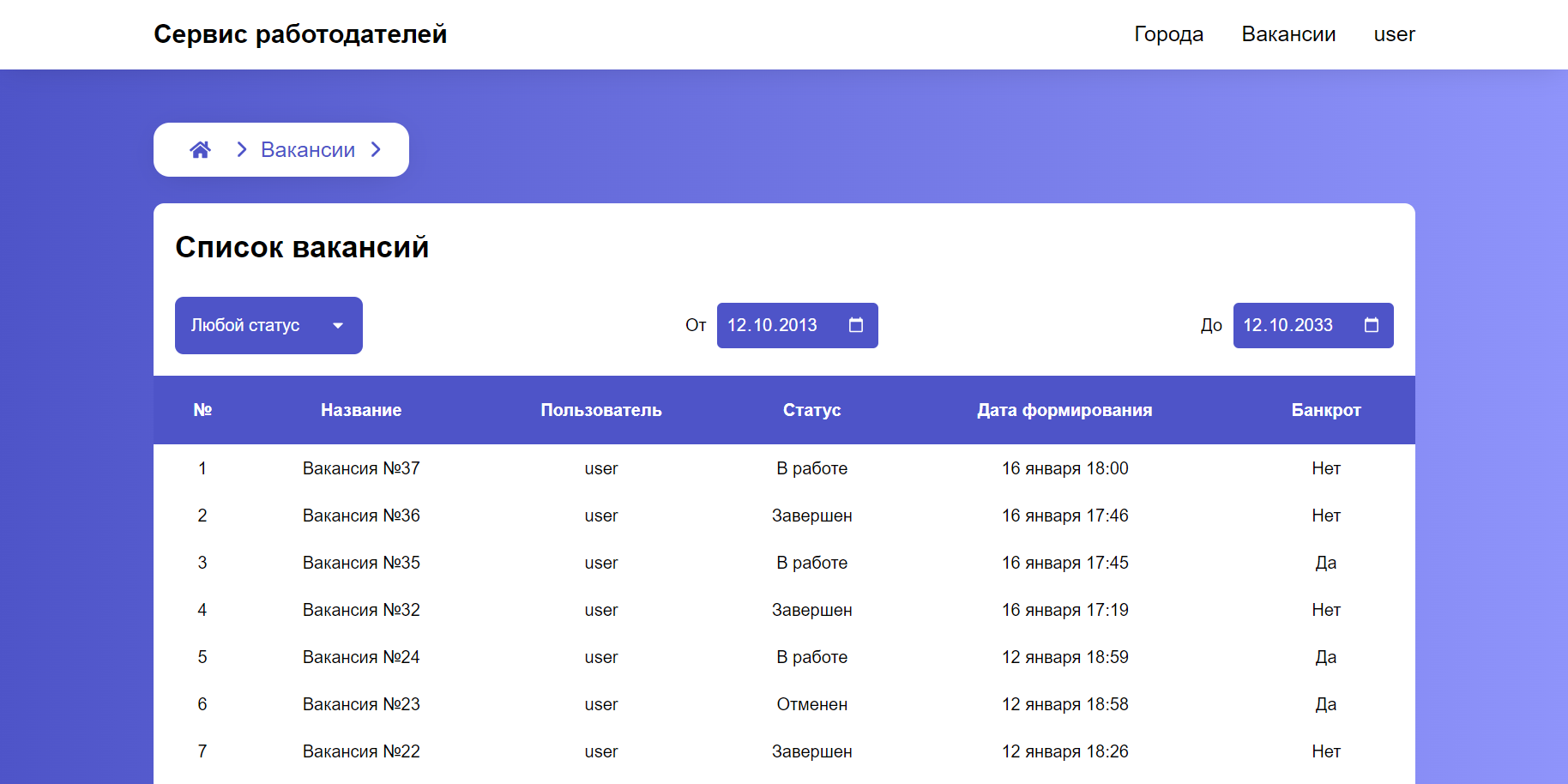


Рисунок 15 – Страница со списком вакансий (работодатель)

На этой странице пользователь может просмотреть список своих вакансий. Доступна фильтрация вакансий по дате формирования и по статусу.

Страница со списком вакансий администратора рекрутинговой платформы (рис. 16)

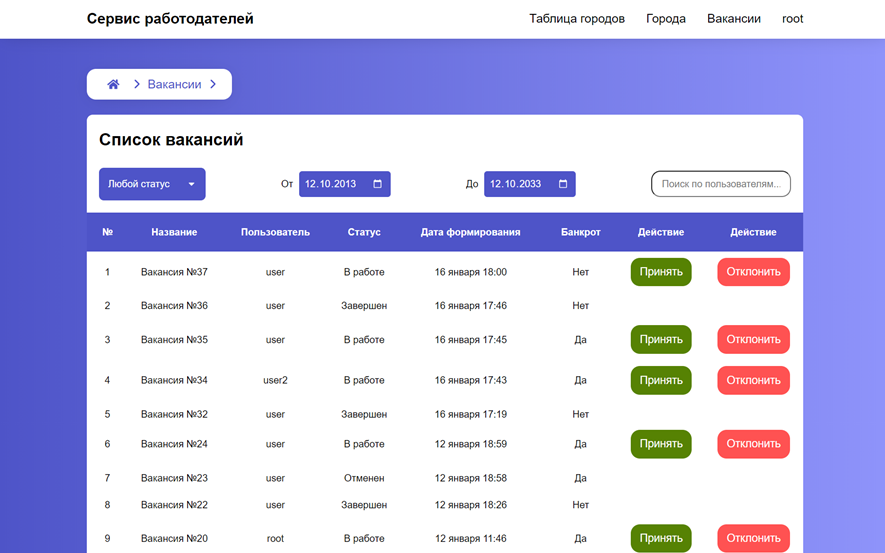


Рисунок 16 - Страница со списком вакансий (администратор рекрутинговой платформы)

На этой странице администратор рекрутинговой платформы может просмотреть вакансии пользователей. Доступна фильтрация вакансий по дате формирования, статусу и пользователю. Администратор рекрутинговой платформы может принять или отклонить любое вакансию, нажав на соответствующие кнопки.

Страница с подробным описанием вакансии (рис. 17)

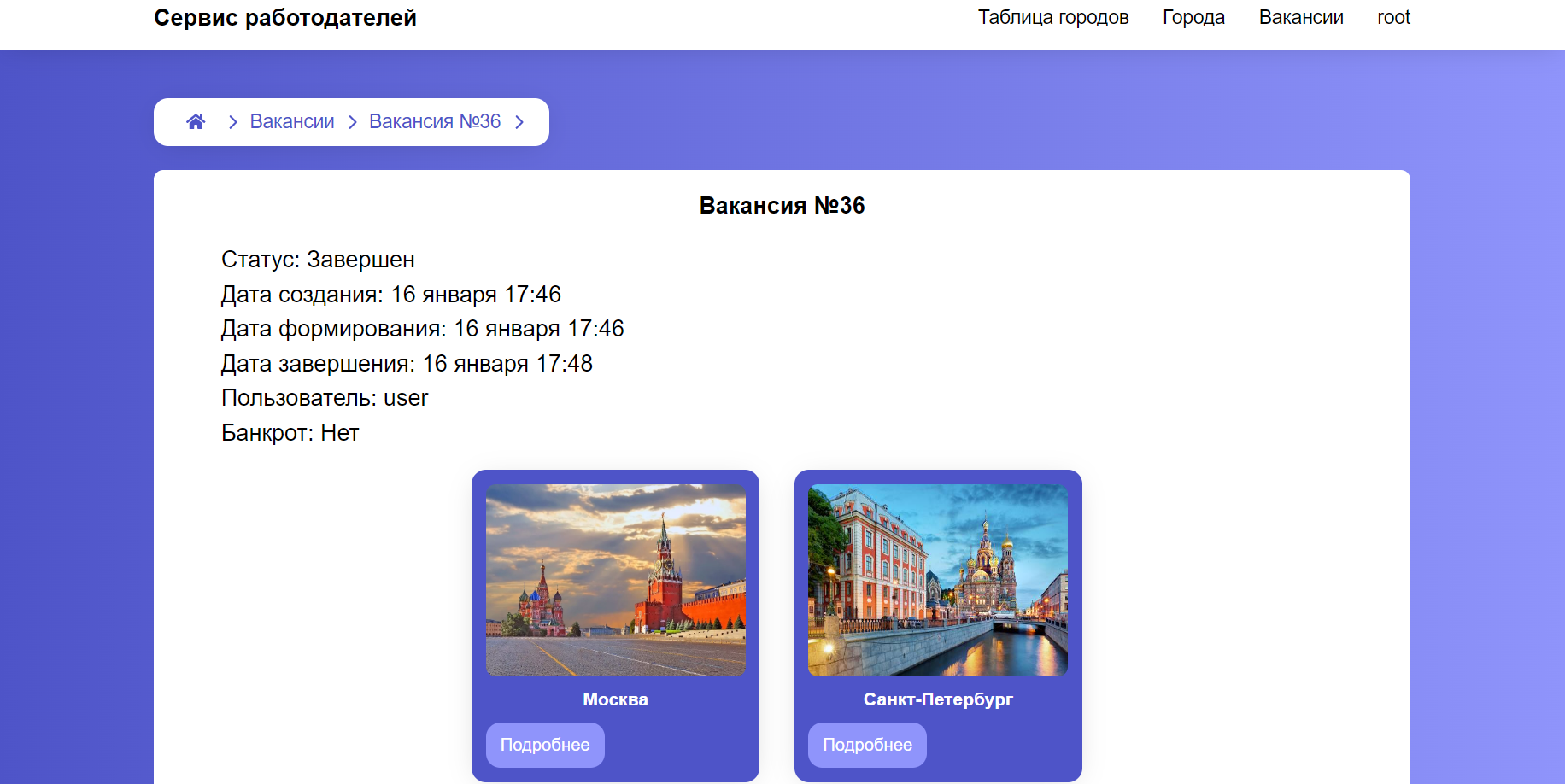


Рисунок 17 - Страница с подробным описанием вакансии

На странице с подробным описанием вакансии отображается подробная информация о вакансии. Список выбранных городов в виде карточек, а также статус вакансии, даты, пользователь и банкротство.

Страница с таблицей городов (рис. 18)

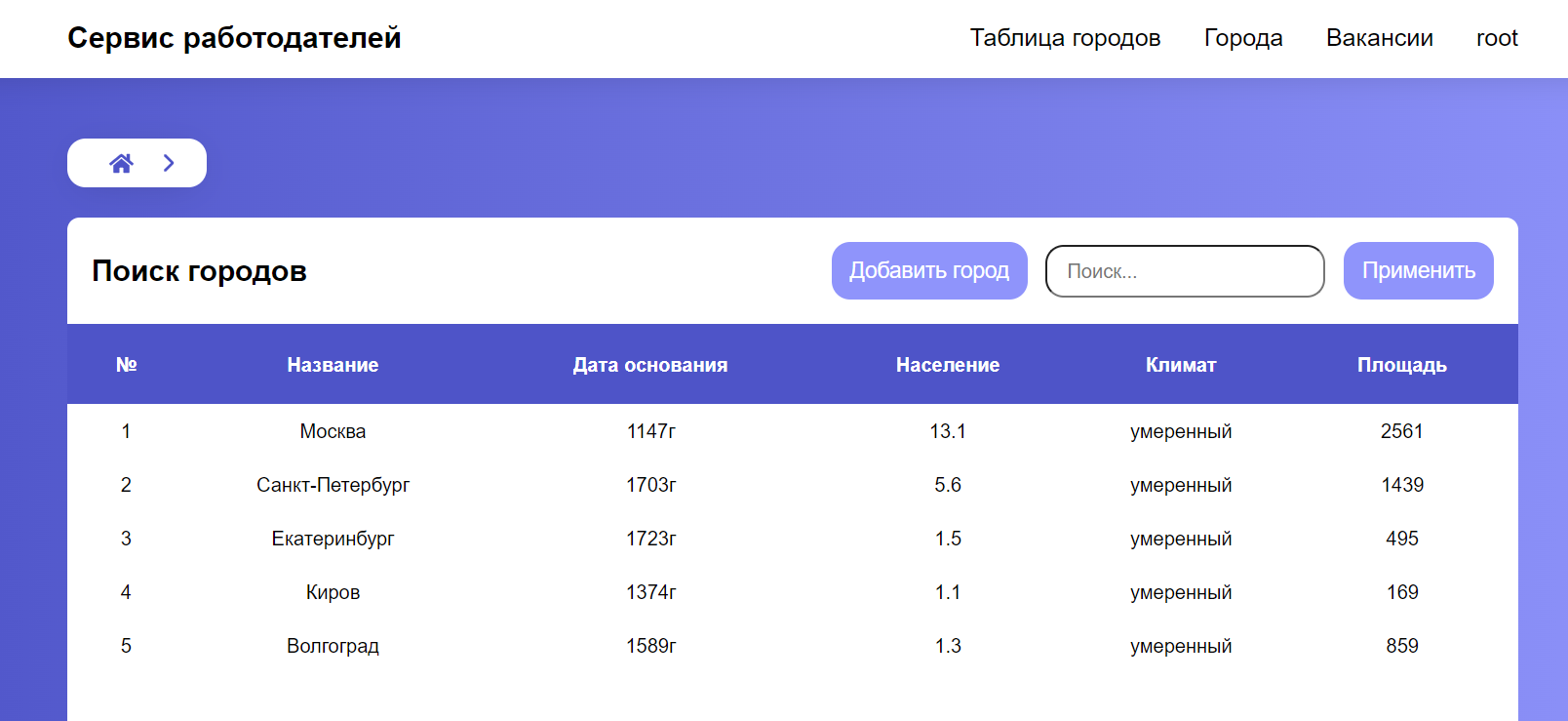


Рисунок 18 - Страница с таблицей городов

Администратору рекрутинговой платформы города представлены в виде таблицы. Доступна фильтрация городов по названию. Есть возможность редактирования информации о городах. Для этого нужно нажать на соответствующую строку таблицы. После этого произойдет переадресация на страницу редактирования выбранного города.

Страница добавления нового города (рис. 19)

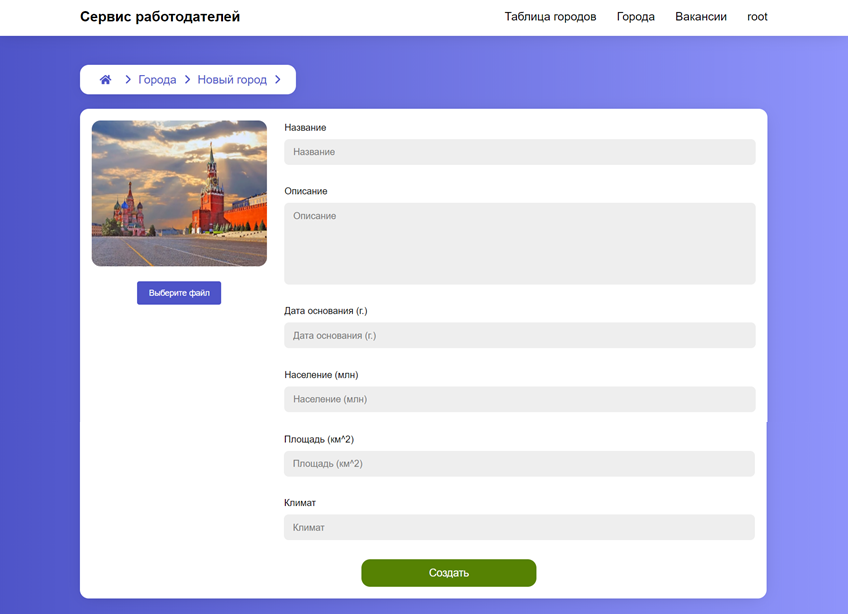


Рисунок 19 – Форма добавления города

На этой странице администратор рекрутинговой платформы может добавить новый город, заполнив соответствующие поля и прикрепив фото города, после чего нажав на кнопку «Создать».

Страница редактирования города (рис. 20)

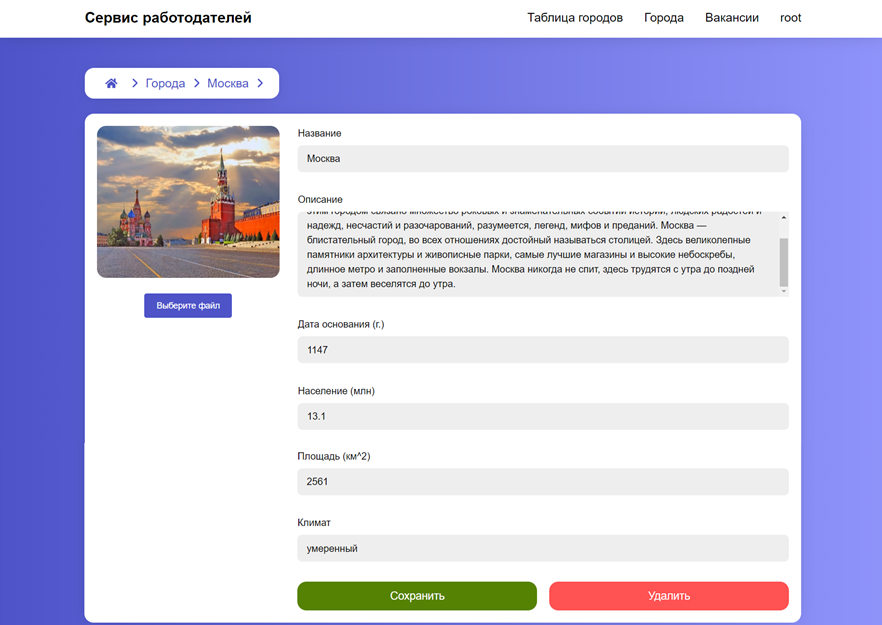


Рисунок 20 - Страница редактирования города

На этой странице администратор рекрутинговой платформы может отредактировать информацию о городе, либо полностью удалить её, нажав на соответствующие кнопки.

# **ЗАКЛЮЧЕНИЕ**

В ходе работы были достигнуты следующие результаты:

1. Был разработан дизайн приложения с помощью набора стилей CSS и HTML тегов.
2. База данных была создана с помощью СУБД PostgreSQL.
3. Был создан веб-сервис на Django REST Framework.
4. Разработан интерфейс гостя с использованием технологии React Framework и подключен к веб-сервису.
5. Приложение интерфейса было развернуто на сервисе Github Pages по ссылке https://nikistor.github.io/Frontend/
6. В веб-сервис добавлена авторизация через JWT, а методы задокументированы через Swagger.
7. Реализован интерфейс работодателя. Доступ к нему имеют только авторизированные пользователи.
8. Реализован интерфейс администратора рекрутинговой платформы для подтверждения новых вакансий и редактирования городов.
9. Было реализовано нативное приложение на React Native.
10. Разработан выделенный сервис для определения банкротства работодателя.
11. Подготовлен набор документации, включающий РПЗ, ТЗ и набор UML диаграмм.
12. Исходный код проекта доступен в GitHub https://github.com/Nikistor/Vacancy

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* 1. Договор найма работника [Электронный ресурс] // glavkniga. URL: https://glavkniga.ru/forms/konsultation/19\_04\_17\_dogov\_trud\_nayma.pdf (дата обращения 02.10.2023)
  2. Трудовой кодекс РФ [Электронный ресурс] // superjob. URL: https://www.superjob.ru/trudovoj-kodeks/15-obschie-polozheniya.html (дата обращения 04.10.2023)
  3. Руководство по PostgreSQL [Электронный ресурс] // Metanit. URL: https://metanit.com/sql/postgresql/ (дата обращения 24.11.2023)
  4. Документация Django [Электронный ресурс] // Django. URL: https://django.fun/docs/django/4.2/ (дата обращения 13.11.2023)
  5. Документация Django REST Framework [Электронный ресурс] // Django-Rest-Framework. URL: https://www.django-rest-framework.org/ (дата обращения 22.11.2023)
  6. Руководство по React [Электронный ресурс] // Metanit. URL: https://metanit.com/web/react/ (дата обращения 24.10.2023)
  7. Quick Start – React Native [Электронный ресурс] // Redux. URL https://redux.js.org/introduction/getting-started (дата обращения 09.12.2023)
  8. Документация GO [Электронный ресурс] // GO. URL https://go.dev/doc/ (дата обращения 14.12.2023)

# **ПРИЛОЖЕНИЕ. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Дисциплина «Разработка интернет-приложений»

Техническое задание

Тема: «Сервис для работодателей»

Студент: Перевощиков Н.Д.

Группа ИУ5Ц-74Б

Преподаватель: Канев А.И.

2023г.

## **Цель**

Создание системы для автоматизации управления вакансиями, которая состоит из веб-сервиса, веб-приложения, мобильного приложения и асинхронного сервиса проверки работодателя на банкротство.

## **Назначение**

Назначение данной системы заключается в создании удобного инструмента для работодателей и администраторов рекрутинговой платформы, который позволит им эффективно управлять вакансиями в городе. Для работодателей приложение предоставляет возможность создания и управления вакансиями. Система позволяет работодателям быстро и удобно размещать заявки на новые вакансии, получать обратную связь от администраторов рекрутинговой платформы и отслеживать статус заявок. Администраторам рекрутинговой платформы предоставляется возможность проводить модерацию заявок, взаимодействовать с базой данных и обеспечивать точное управление услугами, включая выделенный сервис для проведения проверок работодателей, позволяя администраторам рекрутинговой платформы оценивать статус вакансий, таких как банкрот или не банкрот. Все это позволяет ускорить процесс размещения вакансий и повысить эффективность взаимодействия между работодателями и администраторам рекрутинговой платформы и. Кроме того, система предоставляет возможность управления данными в режиме реального времени, что помогает оперативно реагировать на изменения и обеспечивать актуальность информации.

## **Задачи**

* 1. Разработка базового дизайна системы.
  2. Создать структуры базы данных PostgreSQL с использованием Django ORM.
  3. Реализовать веб-сервиса на Django REST Framework.
  4. Создать интерфейс гостя на React.
  5. Развертывать систему на GitHub Pages для предварительной демонстрации работы.
  6. Добавить аутентификации и авторизации в методы веб-сервиса.
  7. Создать интерфейс работодателя на React, с использованием Redux Toolkit.
  8. Создание нативного приложения на React Native.
  9. Создать асинхронный веб-сервис для проверки на банкротство на Go.
  10. Создать интерфейс администратора рекрутинговой платформы на React.
  11. Подготовить набор документации, включающий ТЗ, РПЗ и диаграмм.

## **Методы веб-сервиса**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Метод** | **URL** | **Описание** | **Входные данные** | **Выходные данные** |
| * 1. Услуга - город | | | | | |
| 4.1.1. | GET | api/cities/search/ | Получить список городов, удовлетворяющих переданных критериям  Будут возвращены города со статусом: действует  Доступно всем пользователям | {  query: VARCHAR  } | {  draft\_vacancy\_id: INTEGER,  cities: [  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  },  ...  ]  } |
| 4.1.2. | GET | api/cities/<city\_id>/ | Получить информацию о городе  Доступно всем пользователям | {  city\_id: INTEGER  } | {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  } |
| 4.1.3. | GET | api/cities/<city\_id>/image/ | Получить фотографию города  Доступно всем пользователям | {  city\_id: INTEGER  } | {  image : Byte[],  } |
| 4.1.4. | PUT | api/cities/<city\_id>/update/ | Обновить город  Доступно только администраторам рекрутинговой платформы | {  city\_id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  access\_token: VARCHAR  } | {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  } |
| 4.1.5. | PUT | api/cities/<int:city\_id>/update\_image/ | Обновить фотографию города  Доступно только администраторам рекрутинговой платформы | {  city\_id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  access\_token: VARCHAR  } | {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  } |
| 4.1.6. | POST | api/cities/create/ | Создать новый город  Доступно только администраторам рекрутинговой платформы | {  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  access\_token: VARCHAR  } | {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  } |
| 4.1.7. | POST | api/cities/<int:city\_id>/add\_to\_vacancy/ | Добавить город в вакансию  Доступно только авторизованным пользователям  Доступно в статусе Введен | {  city\_id: INTEGER,  vacancy\_id: INTEGER,  access\_token: VARCHAR } | [      {          id: INTEGER,          name: VARCHAR,          status: INTEGER      }  ] |
| 4.1.8. | DELETE | api/cities/<int:city\_id>/delete/ | Удалить город  Доступно только администраторам рекрутинговой платформы  Доступно в статусе Введен | {  city\_id: INTEGER,  access\_token: VARCHAR  } | {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  } |
| * 1. Заявка - вакансия | | | | | |
| 4.2.1. | GET | api/vacancies/search/ | Получить список вакансий пользователя.  Возможна фильтрация по дате формирования и статусу.  Доступно только авторизованным пользователям | {  status : INTEGER,  date\_start: DATE,  date\_end: DATE,  access\_token: VARCHAR  } | [  {  id: INTEGER,  cities: [  {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  }  ],  employer:  {  email: VARCHAR,  },  moderator:  {  email: VARCHAR,  },  bankrupt: INTEGER,  name : VARCHAR,  status : INTEGER,  date\_created: DATE,  date\_of\_formation: DATE,  date\_complete: DATE,  },  ...  { ... }  ] |
| 4.2.2. | GET | api/vacancies/<vacancy\_id>/ | Получить информацию о вакансии  Доступно только авторизованным пользователям | {  vacancy\_id: INTEGER,  access\_token: VARCHAR  } | {  id: INTEGER,  cities: [  {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  }  ],  employer:  {  email: VARCHAR,  },  moderator:  {  email: VARCHAR,  },  bankrupt: INTEGER,  name : VARCHAR,  status : INTEGER,  date\_created: DATE,  date\_of\_formation: DATE,  date\_complete: DATE,  } |
| 4.2.3. | PUT | api/vacancies/<int:vacancy\_id>/update/ | Обновить вакансию  Доступно только авторизованным пользователям | {  vacancy\_id: INTEGER,  name : VARCHAR,  access\_token: VARCHAR  } | {  id: INTEGER,  cities: [  {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  }  ],  employer:  {  id: INTEGER,  name: VARCHAR,  email: VARCHAR,  is\_moderator: BOOLEAN,  },  moderator:  {  id: INTEGER,  name: VARCHAR,  email: VARCHAR,  is\_moderator: BOOLEAN,  },  bankrupt: INTEGER,  name : VARCHAR,  status : INTEGER,  date\_created: DATE,  date\_of\_formation: DATE,  date\_complete: DATE,  } |
| 4.2.4. | PUT | api/vacancies/<int:vacancy\_id>/update\_status\_user/ | Одобрить вакансию работодателем  Доступно только авторизованным пользователям  Доступно в статусе Введен | {  vacancy\_id: INTEGER,  access\_token: VARCHAR  } | {  id: INTEGER,  cities: [  {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  }  ],  employer:  {  id: INTEGER,  name: VARCHAR,  email: VARCHAR,  is\_moderator: BOOLEAN,  },  moderator:  {},  bankrupt: INTEGER,  name : VARCHAR,  status : INTEGER,  date\_created: DATE,  date\_of\_formation: DATE,  date\_complete: DATE,  } |
| 4.2.5. | PUT | api/vacancies/<int:vacancy\_id>/update\_status\_admin/ | Обновить статус вакансии администратором рекрутинговой платформы  Доступно только администраторам рекрутинговой платформы  Доступно в статусе В работе | {  vacancy\_id: INTEGER,  status: INTEGER,  access\_token: VARCHAR  } | {  id: INTEGER,  cities: [  {  id: INTEGER,  name: VARCHAR,  description: VARCHAR,  status: INTEGER,  foundation\_date: INTEGER,  grp: FLOAT,  climate: VARCHAR,  square : INTEGER,  image : VARCHAR,  }  ],  employer:  {  id: INTEGER,  name: VARCHAR,  email: VARCHAR,  is\_moderator: BOOLEAN,  },  moderator:  {  id: INTEGER,  name: VARCHAR,  email: VARCHAR,  is\_moderator: BOOLEAN,  },  bankrupt: INTEGER,  name: VARCHAR,  status: INTEGER,  date\_created: DATE,  date\_of\_formation: DATE,  date\_complete: DATE,  } |
| 4.2.6 | DELETE | api/vacancies/<int:vacancy\_id>/delete/ | Удалить вакансию  Доступно только авторизированным пользователям  Доступно в статусе Введен | {  vacancy\_id: INTEGER,  access\_token: VARCHAR  } | Отсутствует |
| 4.2.7. | DELETE | api/vacancies/<int:vacancy\_id>/delete\_city/<int:city\_id>/ | Удаление город из вакансии  Доступно только авторизированным пользователям  Доступно в статусе Введен | {  city\_id: INTEGER,  vacancy\_id: INTEGER,  access\_token: VARCHAR  } | Отсутствует |
| 4.2.8. | POST | api/vacancies/<int:vacancy\_id>/update\_bankrupt/ | Обновляет поле банкрот  Доступно только администраторам рекрутинговой платформы  Доступно в статусе В работе | {  vacancy\_id: INTEGER,  bankrupt: INTEGER,  access\_key: VARCHAR  } | Статус 200 / 403 |
| * 1. Пользователь | | | | | |
| 4.3.1. | POST | api/users/login/ | Выполняет аутентификацию пользователя  Доступно только неавторизованным пользователям | {  email : VARCHAR,  password: VARCHAR,  } | Отсутствует |
| 4.3.2. | POST | api/users/register/ | Регистрирует нового пользователя в базе данных  Доступно только неавторизованным пользователям | {  email: VARCHAR,  password: VARCHAR,  } | {  id: INTEGER,  email: VARCHAR,  password: VARCHAR, admin: VARCHAR,  } |
| 4.3.3. | POST | api/users/logout/ | Выполняет выход из аккаунта пользователя и сброс его токена  Доступно только авторизованным пользователям | {  access\_token: VARCHAR  } | Статус 200 |
| * 1. Асинхронный сервис | | | | | |
| 4.4.1. | POST | /api/calc\_bankrupt/ | Определяет, является ли работодатель банкротом  Действие запускается через 4 секунды. Вероятность успеха 70% | {  vacancy\_id: INTEGER } | Статус 200 |

## **Функциональные требования**

* 1. Главное меню.
     1. Города – переход на страницы со списком / таблицей городов, только всем пользователям.
     2. Выполняется переход на страницу со списком городов (страница 5.4), для авторизованных пользователей.
     3. Выполняется переход на страницу с таблицей городов (страница 5.5), для администраторов рекрутинговой платформы.
     4. Вакансии – переход на страницу со списком вакансий (страница 5.9), только авторизированным пользователям.
     5. Вход – переход на страницу с формой авторизации (страница 5.2), только неавторизированным пользователям.
     6. Выход - при нажатии выполняется перенаправление на страницу с формой авторизации (страница 5.2), только авторизированным пользователям.
  2. Страница с формой авторизации.
     1. Доступна только неавторизованным пользователям.
     2. Содержит
        1. Текстовое поле для логина.
        2. Текстовое поле для пароля.
     3. Действия
        1. При нажатии на кнопку Войти отправляется запрос (метод 4.3.1), в котором передаются введенные данные формы. Если введённые данные верны, то в ответе возвращается access токен, который сохраняется в куки. В противном случае возвращается ошибка 405. Только неавторизованным пользователям.
        2. При нажатии на кнопку Еще нет аккаунта? происходит перенаправление на страницу регистрации (страница 5.3), только неавторизованным пользователям.
  3. Страница с формой регистрации.
     1. Доступна только неавторизованным пользователям.
     2. Содержит
        1. Текстовое поле для логина.
        2. Текстовое поле для имени.
        3. Текстовое поле для пароля.
     3. Действие
        1. При нажатии на кнопку Создать аккаунт отправляется запрос (метод 4.3.2), в котором передаются введенные данные формы. Если введённые данные верны, то возвращается access токен, который сохраняется в куки. В противном случае возвращается ошибка 405. Только неавторизованным пользователям.
        2. При нажатии на кнопку Уже есть аккаунт? перенаправляет пользователя на страницу авторизации (страница 5.2), только неавторизованным пользователям.
  4. Страница со списком городов.
     1. Доступна авторизованным пользователям.
     2. Содержит текстовое поле для фильтрации городов по названиям, города в виде карточек (метод 4.1.1) и неактивную кнопку Новая вакансия, доступна авторизированным пользователям. Каждая карточка состоит из:
        1. Картинки города.
        2. Названия города.
     3. Действия
        1. При наборе символов в поле поиска отправляется запрос на API (метод 4.1.1), который возвращает список городов, отфильтрованный по названию.
        2. При нажатии на кнопку Подробнее отправляется запрос на API (метод 4.1.2), происходит перенаправление пользователя на страницу с подробным описанием города (страница 5.6).
        3. При нажатии на кнопку Добавить отправляется запрос на API (метод 4.1.7), который помещает выбранный город в новую / существующую вакансию-черновик, статус Введен.
        4. При нажатии на кнопку Новая вакансия выполняется запрос на API (метод 4.2.2).
  5. Страница с таблицей городов.
     1. Доступна только администраторам рекрутинговой платформы.
     2. Содержит таблицу с городами, состоящую из столбцов (метод 4.1.1)
        + 1. № Города.
          2. Название города.
          3. Год основания города.
          4. Климат города.
          5. Площадь города.
          6. Численность населения города.
          7. Описание города.
     3. Действия
        1. При нажатии на кнопку Открыть происходит перенаправление на страницу с редактированием города (страница 5.8), только администраторам рекрутинговой платформы.
        2. При нажатии на кнопку Создать происходит перенаправление на страницу с созданием города (страница 5.8), только администраторам рекрутинговой платформы.
  6. Страница с подробным описанием города.
     1. Доступна всем пользователям.
     2. Содержит поля (метод 4.1.2)
        1. Название города.
        2. Год основания города.
        3. Численность населения города.
        4. Климат города.
        5. Площадь города.
        6. Статус
        7. Описание города.
  7. Страница создания вакансии.
     1. Доступна только авторизованным пользователям.
     2. Содержит (метод 4.2.1)
        1. Текстовое поле для названия вакансии.
        2. Блок, содержащий карточки городов, входящих в вакансию. Каждая карточка состоит из:
           1. Картинки города.
           2. Названия города.
     3. Действия
        1. При нажатии на кнопку Удалить в блоке, отправляется запрос на API (метод 4.2.7), который удаляет город из вакансии, только авторизованным пользователем, статус Введен.
        2. При нажатии на кнопку Удалить, отправляется запрос на API (метод 4.2.6), который удаляет вакансию. После чего происходит переадресация на страницу со списком городов пользователя (страница 5.4), только авторизованным пользователем, статус Введен.
        3. При нажатии на кнопку Сохранить, отправляется запрос на API (метод 4.2.3), который обновляет информацию о вакансии, только авторизованным пользователем, статус Введен.
        4. При нажатии на кнопку Отправить отправляется запрос на API (метод 4.2.4), который отправляет вакансию на рассмотрение модерации. После чего происходит переадресация на страницу со списком вакансий пользователя (страница 5.9), только авторизованным пользователем, статус из Введен на В работе.
  8. Страница с редактированием / созданием города.
     1. Доступна только администраторам рекрутинговой платформы.
     2. Содержит (метод 4.1.1)
        1. Текстового поля для названия города.
        2. Текстового поля для даты основания города.
        3. Текстового поля для населения города.
        4. Текстового поля для климата города.
        5. Текстового поля для площади города.
        6. Текстового поля для описания города.
     3. Действия
        1. При нажатии на кнопку Сохранить отправляется запрос на API (метод 4.1.4), который обновляет информацию о городе, только администраторам рекрутинговой платформы, статус В работе.
        2. При нажатии на кнопку Удалить отправляется запрос на API (метод 4.1.8), который удаляет город, только администраторам рекрутинговой платформы, статус В работе.
        3. При нажатии на кнопку Создать отправляется запрос на API (метод 4.1.6), который создает город, только администраторам рекрутинговой платформы.
  9. Страница со списком вакансий.
     1. Доступна только авторизованным пользователям.
     2. Содержит таблицу со списком вакансий пользователя, содержащую столбцы (метод 4.2.1)
        1. № Вакансии.
        2. Статус вакансии.
        3. Названия городов, входящих в вакансию.
        4. Дата формирования вакансии.
        5. Фильтр по дате формирования вакансии, по статусу и по пользователю.
     3. Действия
        1. При нажатии на кнопку Принять отправляется запрос на API (метод 4.2.5), который меняет статус вакансии: В работе на Завершён, только администраторам рекрутинговой платформы.
        2. При нажатии на кнопку Отклонить отправляется запрос на API (метод 4.2.5), который меняет статус вакансии: В работе на Отклонён, только администраторам рекрутинговой платформы.
        3. При изменении состояния фильтра по дате формирования отправляется запрос на API (метод 4.2.1), который возвращает все вакансии, удовлетворяющие переданным критериям, только авторизованным пользователям.
        4. При изменении состояния фильтра по статусам отправляется запрос на API (метод 4.2.1), который возвращает все вакансии, удовлетворяющие переданным критерии, только авторизованным пользователям.
        5. При изменении состояния фильтра по пользователям отправляется запрос на API (метод 4.2.1), который возвращает все вакансии, удовлетворяющие переданным критериям, только авторизованным пользователям.

## **Требования к программному обеспечению:**

* 1. Серверная часть
     1. Операционная система (любая из)
        1. Linux (6.4.12 и выше)
        2. Windows (22H2 и выше)
     2. Веб-сервер Nginx
     3. Python (2.7 и выше) с Фреймворком Django (3.2 и выше)
     4. СУБД PostgreSQL (16.1 и выше)
     5. Доступ к интернету
  2. Клиентская часть
     1. Операционная система (любая из)
        1. Windows (22H2 и выше)
        2. MacOS (11.7.10 и выше)
        3. Linux (6.4.12 и выше)
     2. Веб-браузер (любой из)
        1. Safari (16.5.2 и выше)
        2. Chrome (119.0.6045 и выше)
        3. Opera (105.0.4970.16 и выше)
     3. Доступ к интернету

## **Требования к аппаратному обеспечению:**

* 1. Серверная часть
     1. Процессор минимум 2-ядерный с частотой от 2 ГГц
     2. Оперативная память от 4 Гб
     3. Место на жестком диске от 2 Гб
  2. Клиентская часть
     1. Процессор с частотой от 1ГГц
     2. Оперативная память от 512 Мб