**Национальный исследовательский ядерный университет «МИФИ»**

|  |  |
| --- | --- |
|  | **Институт интеллектуальных и кибернетических систем**  **Кафедра «Компьютерные системы и технологии»** |

Пояснительная записка

к проекту по курсу «Программирование

сетевых приложений»

на тему «Агрегатор IT-вакансий»

Студент гр. Б20-523 /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Колойденко С. Д.

Руководитель / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Овчаренко Е. С.

Москва, 2023 г.

АННОТАЦИЯ

Пояснительная записка 19 с., 6 рис., 6 табл., 15 ист.

Объектом разработки является приложение для агрегации IT-вакансий.

Целью работы является разработка серверной и клиентской части приложения, с помощью которого можно отслеживать, фильтровать и добавлять в избранное IT-вакансии.

В рамках анализа предметной области были рассмотрены приложения агрегации IT-вакансий.

Было произведено ручное тестирование разработанного приложения.

Серверная часть приложения реализована на языке программирования Java с использованием Spring Framework. В качестве клиентской части выступает приложение, написанное на языке программирования TypeScript с использованием фреймворка React. Для работы с базой данных была выбрана СУБД MySQL.

Содержание

[**Введение** 4](#_Toc154436886)

[**1.** **Теоретическая часть** 5](#_Toc154436887)

[**1.1.** **Краткий анализ существующих решений** 5](#_Toc154436888)

[**1.2.** **Постановка задачи** 5](#_Toc154436889)

[**1.3.** **Схема приложения** 5](#_Toc154436890)

[**1.4.** **Формализация данных** 6](#_Toc154436891)

[**2.** **Практическая часть** 6](#_Toc154436892)

[**2.1.** **Используемые технологии** 6](#_Toc154436893)

[**2.2.** **Описание таблиц базы данных** 7](#_Toc154436894)

[**2.3.** **Уровень доступа к данным** 7](#_Toc154436895)

[**2.4.** **Уровень бизнес-логики** 8](#_Toc154436896)

[**2.5.** **Уровень взаимодействия** 9](#_Toc154436897)

[**2.5.1.** **Контроллер аутентификации** 10](#_Toc154436898)

[**2.5.2.** **Контроллер пользователей** 10](#_Toc154436899)

[**2.5.3.** **Публичный контроллер вакансий** 10](#_Toc154436900)

[**2.5.4.** **Контроллер вакансий для зарегистрированных пользователей** 10](#_Toc154436901)

[**2.5.5** **Контроллер вакансий для администратора** 11](#_Toc154436902)

[**2.6.** **Аутентификация пользователей** 11](#_Toc154436903)

[**2.7.** **Веб-приложение** 12](#_Toc154436904)

[**2.7.1.** **Интерфейс веб-приложения** 12](#_Toc154436905)

[**2.7.2.** **Взаимодействие с сервером** 15](#_Toc154436906)

[**3.** **Тестирование** 16](#_Toc154436907)

[**3.1.** **Тестирование серверной части приложения** 16](#_Toc154436908)

[**3.1.1.** **Ручное тестирование** 16](#_Toc154436909)

[**Заключение** 17](#_Toc154436910)

[**Список используемых источников** 18](#_Toc154436911)

# **Введение**

Для студентов старших курсов и выпускников IT направлений актуален вопрос поиска работы. Этот процесс подразумевает просмотр различных порталов с вакансиями, таких как hh.ru, career.habr.com и superjob.ru. Возникает потребность в централизованном сайте, на котором можно посмотреть интересующие активные вакансии или требования к уже закрытым вакансиям.

В процессе работы была поставлена задача выполнения проекта, который заключается в создании веб-приложения в качестве клиентской части и API для этого приложения в качестве серверной части.

# **Теоретическая часть**

* 1. **Краткий анализ существующих решений**

В настоящее время существуют агрегаторы IT-вакансий, такие как «Remote Ok» [1] или «Remote.io» [2]. Однако они не нацелены на людей, живущих в России. Также, как правило, выборка вакансий на этих сайтах не самая актуальная.

* 1. **Постановка задачи**

Веб-приложение будет предоставлять следующий функционал:

1. Поиск и фильтрация активных вакансий по языкам программирования, фреймворкам, грейдам, опыту работы, зарплате и месту работы
2. Просмотр закрытых вакансий
3. Регистрация новых пользователей
4. Аутентификация пользователей по логину и паролю
5. Добавление вакансий в избранное
6. Просмотр, управление конфигурацией выгрузки вакансий с порталов и их редактирование администратором
   1. **Схема приложения**

Клиент-серверное приложение состоит из нескольких связанных частей. Схема взаимодействия между частями приложения представлена на рисунке 1.

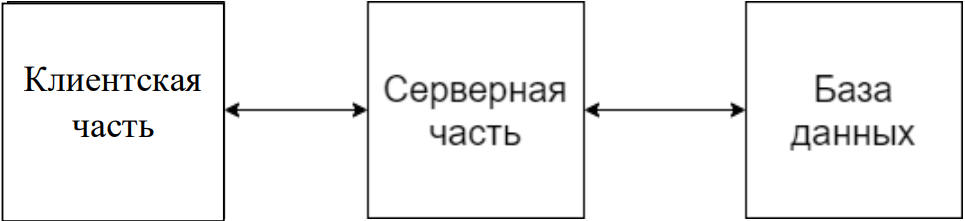


Рисунок 1. Схема взаимодействия между частями приложения

Пользователь через веб-браузер приложение может совершать действия, которые будут обрабатываться сервером и сохраняться в базе данных.

* 1. **Формализация данных**

База данных должна хранить данные о следующих сущностях, представленных в таблице 1.

Таблица 1. Сущности базы данных

|  |  |
| --- | --- |
| **Сущности** | **Сведения** |
| Пользователь | Почта, пароль, роль |
| Токен сброса | Хэш, дата инвалидации, id пользователя |
| Вакансия | Язык программирования, название, минимальная зарплата, максимальная зарплата, работодатель, ссылка, внешний id, тип агрегатора |
| Избранная вакансия | Id пользователя, id вакансии |

1. **Практическая часть**
   1. **Используемые технологии**

В качестве языка программирования для разработки серверной части приложения был выбран язык программирования Java, поскольку он поддерживает парадигму объектно-ориентированного программирования, а также поддерживает возможность работы с фреймворками, такими как Spring Framework [3]. Данный фреймворк упрощает разработку, используя такой паттерн как инверсия зависимостей [4] и конкретную реализацию паттерна – инверсию контроля.

В качестве базы данных используется СУБД MySQL [5], поскольку она надежна и быстро работает.

Также было принято решение разрабатывать веб-клиент на TypeScript [6], надстройкой над обычным JavaScript со статической типизацией, что позволяет улучшить качество кода и продуктивность разработки. В качестве фреймворка был выбран React [7], поскольку он прост и гибок.

* 1. **Описание таблиц базы данных**

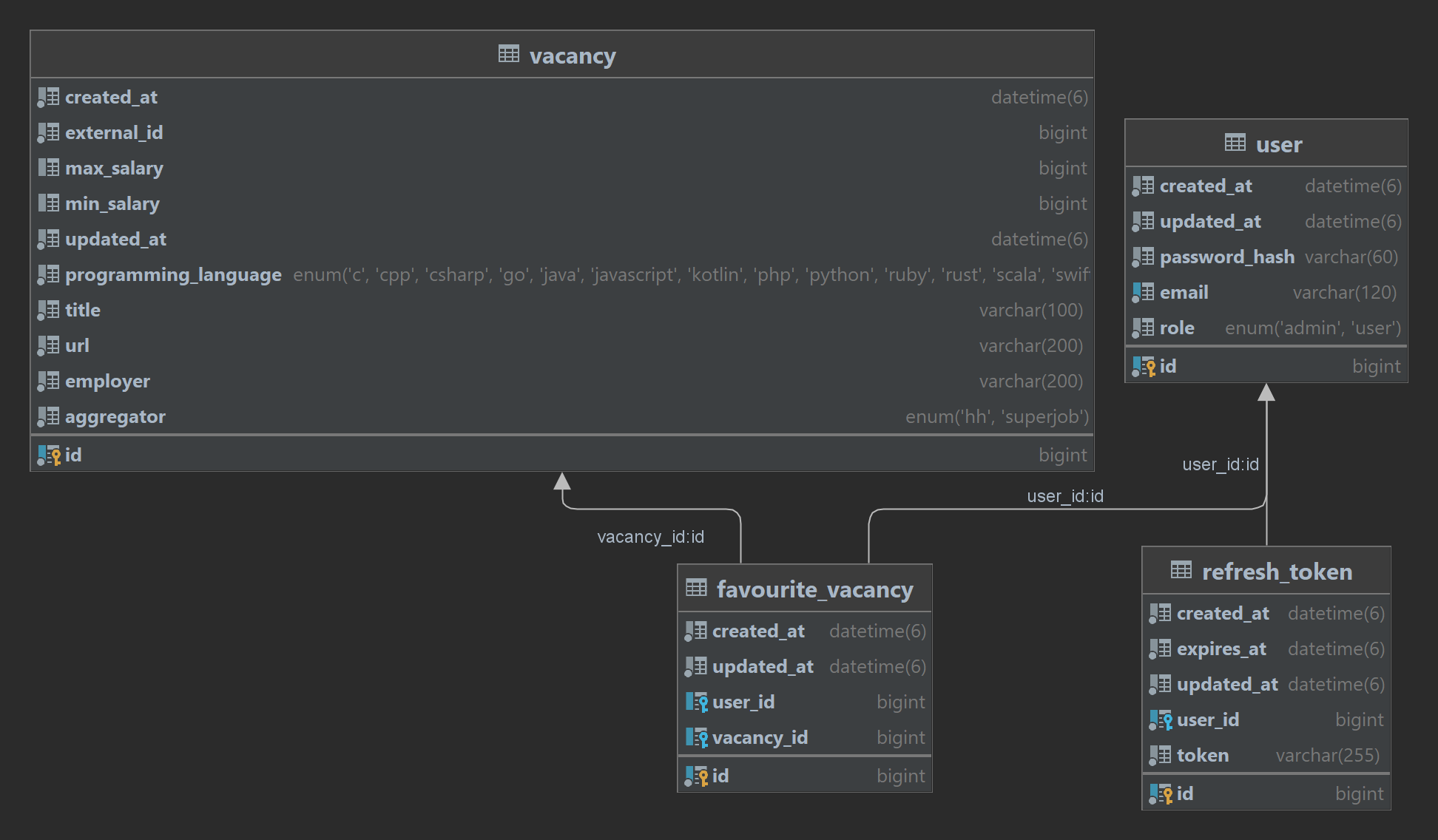
Чтобы организовать хранение информации о сущностях используется база данных MySQL. Структура базы данных представлена на рисунке 2. 

Рисунок 2. Структура базы данных

Сущность User – в ней хранятся id, даты создания и изменения, почта, хэш пароля и роль пользователя.

Сущность RefreshToken – в ней хранятся id, даты создания и изменения, дата инвалидации, токен и id пользователя.

Сущность Vacancy – в ней хранится id, даты создания и изменения, язык программирования, название, минимальная зарплата, максимальная зарплата, работодатель, ссылка, внешний id, тип агрегатора.

Сущность FavouriteVacancy – в ней хранится id, даты создания и изменения, id пользователя и id вакансии.

* 1. **Уровень доступа к данным**

Data Access Object (Объект доступа к данным) – структурный паттерн, который позволяет изолировать бизнес-логику от уровня базы данных [8]. Это помогает изменять функциональность сервиса, не затрагивая схемы в БД – лишь изменять уровень доступа к данным.

Чтобы создавать объекты доступа к данным было решено использовать реализацию Hibernate для спецификации Java Persistence API (JPA) [9]. Hibernate – это библиотека объектно-реляционного отображения для Java, которая позволяет конвертировать Java классы в сущности для реляционной базы данных и наоборот. [10]

Чтобы создать отображения Java классов в объекты реляционной БД нам понадобится репозиторий, который предоставляет Spring Data, часть фреймворка Spring [11].

JpaRepository – интерфейс, который позволяет выполнять базовые операции (create, read, update, delete), а также пагинировать ответ. Был создан интерфейс AbstractRepository, который наследуется от JpaRepository и используется для инкапсуляции типа первичного ключа. Все созданные впоследствии репозитории наследовались от этого интерфейса с указанием типа сущности.

В итоге, на уровне доступа к данным были созданы репозитории и Java классы, в которые отображаются сущности из базы данных.

* 1. **Уровень бизнес-логики**

После того, как появилась возможность взаимодействовать с базой данных посредствам Java классов, можно приступать к уровню бизнес-логики. На данном этапе будут проводится все вычисления, которые нужны для функционирования самого сервиса.

Всего существует 4 сервиса, которые разделены функционально:

* сервис аутентификации нужен для регистрации, входа в систему, обновления токенов и выхода из системы;
* сервис пользователей используется для получения информации о зарегистрированных пользователях;
* сервис авторизации нужен для ограничения доступа пользователей к определенному функционалу;
* сервис вакансий нужен для сбора данных с помощью внешнего API и отображения вакансий.
  1. **Уровень взаимодействия**

Следующий уровень в цепочке – уровень отображения данных в бизнес-логике пользователю или другим сервисам, которые хотят взаимодействовать с API. На этом уровне определяется способ взаимодействия сервиса с остальными пользователями сети. Было принято решение выбрать REST (Representation State Transfer) [12] – это набор правил и ограничений для проектирования сервиса, чтобы все системы легко обменивались данными и приложение можно было масштабировать.

На данном этапе важно определить, что именно будет отдаваться для внешнего мира. Сущности, которые отдаются по какому-либо запросу извне называются DTO (data transfer object) [13]. Их создание необходимо по нескольким причинам. Во-первых, не нужно показывать данные, которые должны храниться только на сервере, например, пароли пользователей. Во-вторых, принцип REST советует отдавать объекты как JSON (JavaScript Object Notation) – текстовый формат обмена данными. Сущности в репозиториях могут иметь ссылки сами на себя. При представлении JSON объектов такого быть не должно, поскольку будет бесконечный JSON. Именно поэтому нужно предусмотреть, чтобы возвращаемый DTO не содержал рекурсии. Это реализуется путём исключения полей в DTO, которые могут содержать ссылки на родительскую сущность.

Чтобы пользователи сети посылали запрос на сервис, нужны конечные точки, по запросу на которые сервис будет отдавать ответ. Для этой цели были созданы контроллеры, которые управляют запросами приходящие на сервер. Всего было разработано 5 контроллера, которые отвечают за запросы аутентификации, пользователя и вакансий (публичный, для зарегистрированных пользователей и для администрации).

* + 1. **Контроллер аутентификации**

Общий URL для всех запросов будет содержать путь /public/auth.

Информацию о контроллере аутентификации можно увидеть в таблице ниже.

Таблица 2. Конечных точек контроллера аутентификации

|  |  |
| --- | --- |
| Запрос | Функция конечной точки |
| POST /public/auth/register | Регистрация нового пользователя |
| POST /public/auth/login | Вход в систему |
| POST /public/auth/refresh | Обновление токенов |
| POST /public/auth/logout | Выход из системы |

* + 1. **Контроллер пользователей**

Общий URL для всех запросов будет содержать путь /user/profile.

Всю информацию о контроллере пользователей можно увидеть в таблице ниже.

Таблица 3. Конечных точек контроллера пользователей

|  |  |
| --- | --- |
| Запрос | Функция конечной точки |
| GET /user/profile | Получение своего профиля |

* + 1. **Публичный контроллер вакансий**

Общий URL для всех запросов будет содержать путь /public/vacancy.

Всю информацию о публичном контроллере вакансий можно увидеть в таблице ниже.

Таблица 4. Конечных точек публичного контроллера вакансий.

|  |  |
| --- | --- |
| Запрос | Функция конечной точки |
| GET /public/vacancy | Получение списка вакансий |

* + 1. **Контроллер вакансий для зарегистрированных пользователей**

Общий URL для всех запросов будет содержать путь /user/vacancy.

Всю информацию о контроллере вакансий для зарегистрированных пользователей можно увидеть в таблице ниже.

Таблица 5. Конечных точек контроллера вакансий для зарегистрированных пользователей

|  |  |
| --- | --- |
| Запрос | Функция конечной точки |
| GET /user/vacancy/favourite | Просмотр избранных вакансий |
| POST /user/vacancy/favourite | Добавление вакансии в избранное |
| DELETE /user/vacancy/favourite | Удаление вакансии из избранного |

* + 1. **Контроллер вакансий для администратора**

Общий URL для всех запросов будет содержать путь /admin/vacancy.

Всю информацию о контроллере вакансий для администратора можно увидеть в таблице ниже.

Таблица 6. Конечных точек контроллера вакансий для администратора

|  |  |
| --- | --- |
| Запрос | Функция конечной точки |
| PUT /admin/vacancy | Обновление вакансии |

* 1. **Аутентификация пользователей**

В сервисе предусмотрена аутентификация пользователей и последующее взаимодействие используя JWT (JSON Web Token) [14]. Для этого был создан отдельный контроллер PublicAuthController, который позволяет пользователю выполнять вход в аккаунт и выходить из него. После того, как пользователь посылает POST запрос на вход /public/auth/login с логином и паролем в качестве параметров, генерируется access token, который отдаётся пользователю в теле ответа, и refresh token, который записывается в cookie. Пользователь должен использовать этот токен в заголовке Authentication последующих запросах. При истечении времени жизни access token на сервер отправляется запрос на /public/auth/refresh с refresh token для получения новой пары токенов. После аутентификации пользователя, он получает доступ к ограниченному функционалу в зависимости от его ролей. Если пользователь захотел прекратить работу, он может выйти из аккаунта, послав POST запрос /public/auth/logout. Без аутентификации можно использовать только публичное API.

* 1. **Веб-приложение**
     1. **Интерфейс веб-приложения**

Веб-приложение состоит из нескольких окон, с помощью которых пользователь может выполнять нужные ему действия.

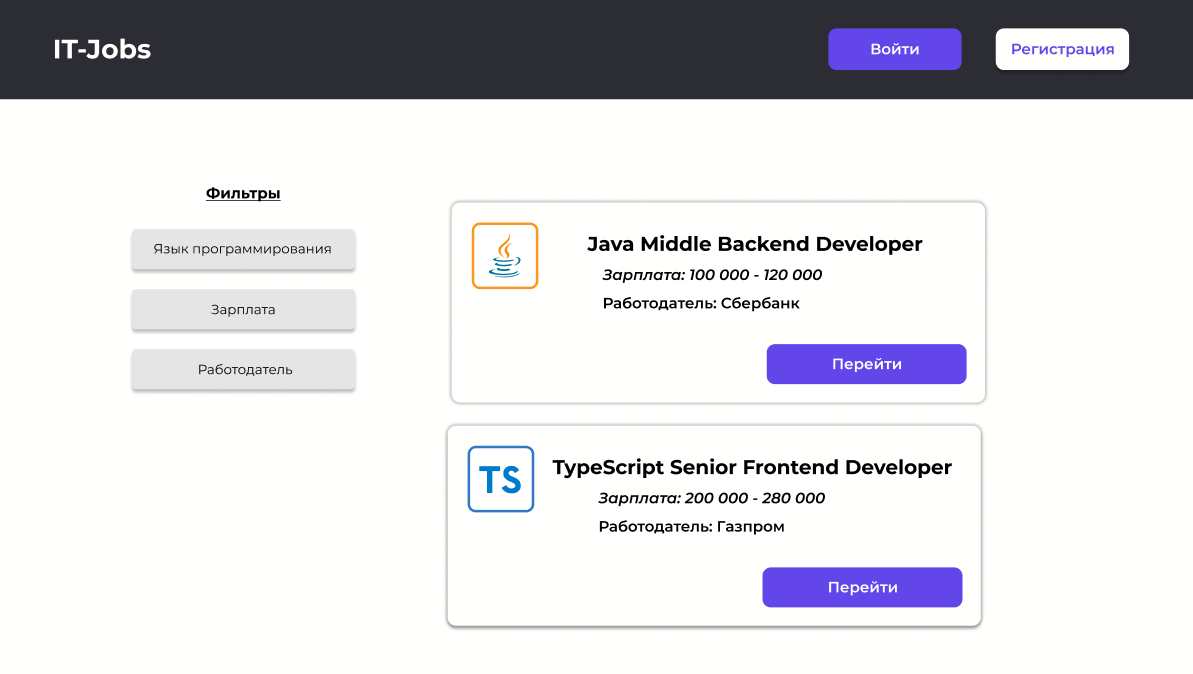
**Без аутентификации пользователь может просматривать список вакансий (рис 3.).

Рисунок 3. Окно просмотра вакансий

При нажатии на кнопки «Войти» или «Регистрация» пользователь может перейти на окна входа в систему (рис. 4) и регистрации (рис. 5).

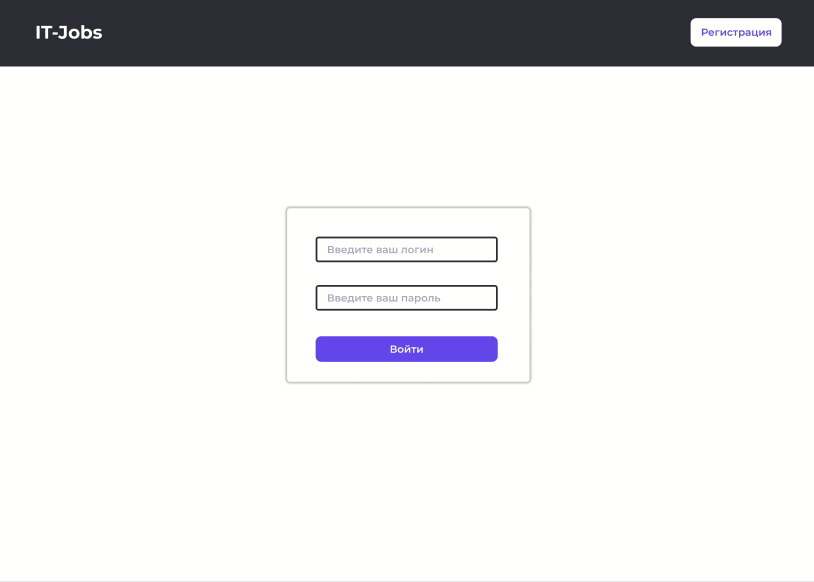


Рисунок 4. Окно входа в систему

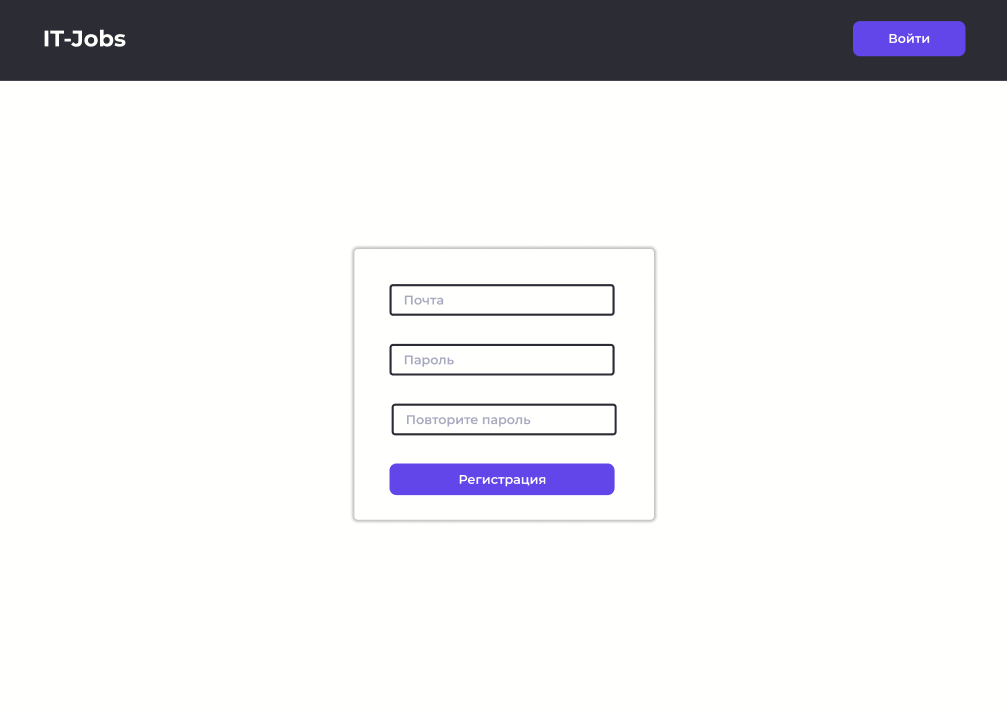


Рисунок 5. Окно регистрации

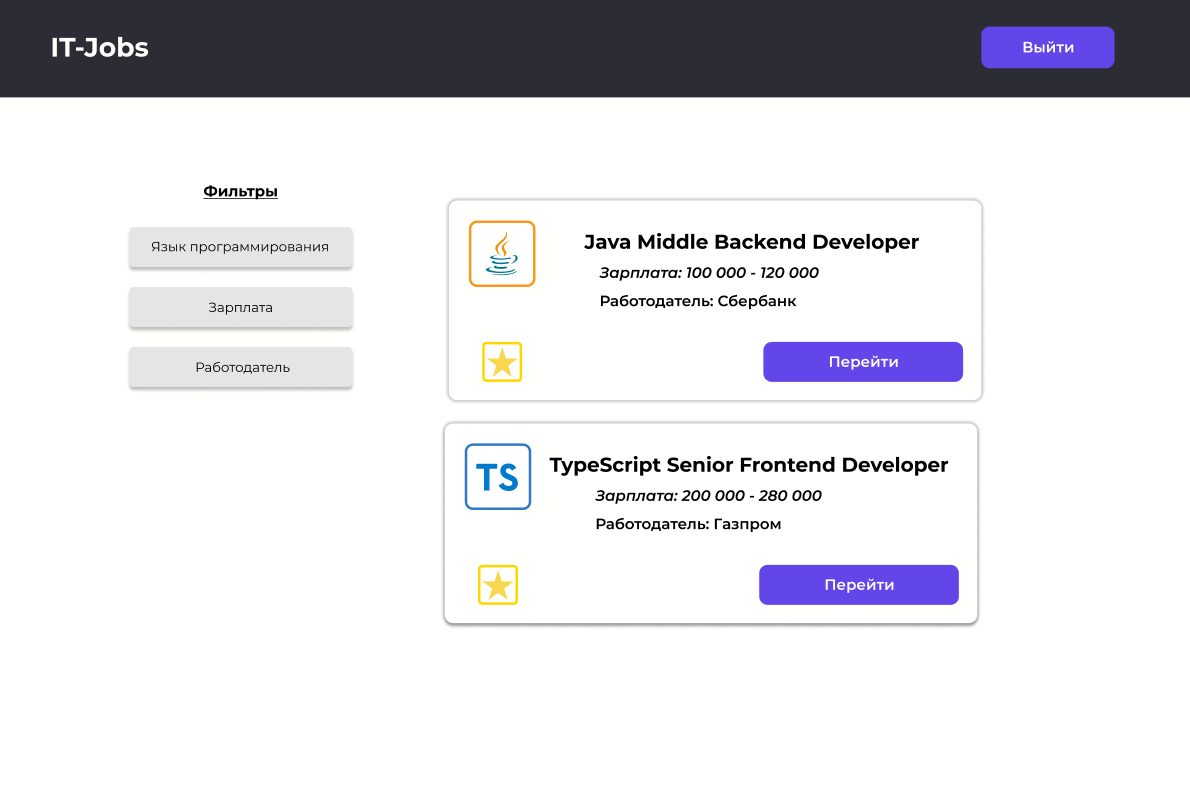
В случае успешной аутентификации пользователя он сможет добавлять вакансии в избранное (рис 6.).

Рисунок 6. Окно просмотра курсов для аутентифицированного пользователя

* + 1. **Взаимодействие с сервером**

На данный момент сервер запускается на локальном компьютере в локальной сети. Существует возможность запуска на стороннем сервере.

1. **Тестирование**
   1. **Тестирование серверной части приложения**

При разработке серверной и клиентской части приложения было проведено только ручное тестирование. Были протестированы все возможности системы, включая некорректный ввод данных.

* + 1. **Ручное тестирование**

Ручное тестирование серверной части приложения проводилось с помощью инструмента Postman [15].

Postman это программа для отправки запросов на сервер и получения ответов. С помощью неё удобно проводить ручное тестирование, получать ответы от сервера, добавлять нужные поля в заголовки и выбирать тип авторизации.

# **Заключение**

В соответствии с техническим заданием было разработано приложение, которое состоит из серверной и клиентской частей.

В приложении представлены такие функции как:

* регистрация и вход в аккаунт пользователя;
* просмотр и фильтрация вакансий;
* управление конфигурацией выгрузки вакансий с порталов и их редактирование администратором;
* добавление вакансий в избранное.

Была спроектирована база данных, которая позволяет обращаться с информацией, которая нужна для функционирования приложения. Серверное приложение, которое позволяет производить действия по обработке данных и мобильное приложение, которое является клиентской частью. С помощью него пользователь может получать информацию и вносить изменения в удобном виде.

Произведено ручное тестирования серверной и клиентской части.

# **Список используемых источников**

1. Remote OK [Электронный ресурс]. 2023. URL: https://remoteok.com/ (дата обращения: 21.12.2023)
2. Remote.io [Электронный ресурс]. 2023. URL: https://www.remote.io/remote-jobs-to-work-from-home (дата обращения: 21.12.2023)
3. VMware, Spring Framework [Электронный ресурс]. 2023. URL: https://spring.io/projects/spring-framework (дата обращения: 21.12.2023)
4. Роберт Мартин. Чистая архитектура. Искусство разработки программного обеспечения. — СПб.: Питер, 2018 — 352 с.: ил. — (Серия «Библиотека программиста»).
5. MySQL [Электронный ресурс]. 1996-2023 URL: https://www.mysql.com/ (дата обращения: 21.12.2023)
6. TypeScript [Электронный ресурс]. 2023 URL: https://www.typescriptlang.org/ (дата обращения 21.12.2023)
7. React [Электронный ресурс]. 2023 URL: https://react.dev/ (дата обращения 21.12.2023)
8. Baeldung. The DAO Pattern in Java [Электронный ресурс]. 2023 URL: https://www.baeldung.com/java-dao-pattern (дата обращения 21.12.2023)
9. Oracle. Java Persistence API [Электронный ресурс]. 2023 URL: https://www.oracle.com/java/technologies/persistence-jsp.html (дата обращения 21.12.2023)
10. Hibernate. What is Object/Relational Mapping? [Электронный ресурс]. 2023 URL: https://hibernate.org/orm/what-is-an-orm/ (дата обращения 21.12.2023)
11. VMware, Spring Data [Электронный ресурс]. 2023 URL: https://spring.io/projects/spring-data (дата обращения 21.12.2023)
12. Lokesh Gupta. What is REST [Электронный ресурс]. 2023 URL: https://restfulapi.net/ (дата обращения 21.12.2023)
13. Baeldung. The DTO Pattern (Data Transfer Object) [Электронный ресурс]. 2023 URL: https://www.baeldung.com/java-dto-pattern (дата обращения 21.12.2023)
14. Okta. JWT [Электронный ресурс]. 2023 URL: https://jwt.io/ (дата обращения 21.12.2023)
15. Postman API [Электронный ресурс] 2023 URL: https://www.postman.com/api-documentation-tool/ (дата обращения 21.12.2023)