|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**КУРСОВАЯ РАБОТА**

по дисциплине: Разработка серверных частей интернет-ресурсов

по профилю: Разработка программных продуктов и проектирование информационных систем

направления профессиональной подготовки: 09.03.04 «Программная инженерия»

Тема: Веб-сервис доставки продуктов питания

Студент: Быченков Александр Константинович

Группа: ИКБО-32-21

Работа представлена к защите\_\_\_\_\_\_\_\_\_\_\_(дата)\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись и ф.и.о. студента)

Руководитель: ст. преподаватель Волков Михаил Юрьевич

Работа допущена к защите\_\_\_\_\_\_\_\_\_\_\_(дата)\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись и ф.и.о. рук-ля)

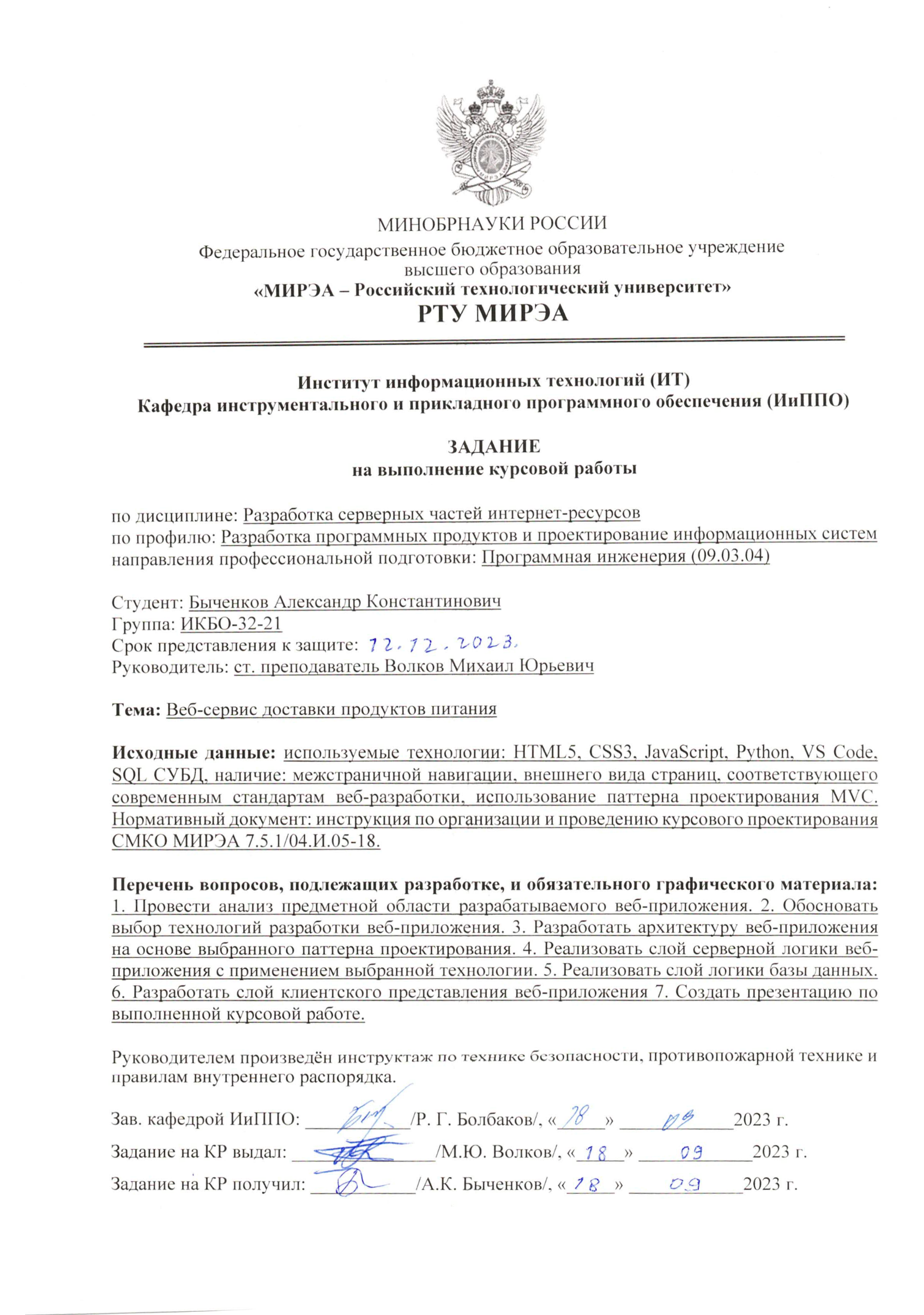
Оценка по итогам защиты: \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /

(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. РТУ МИРЭА 2023 г.



УДК 004.4

Руководитель курсовой работы: старший преподаватель М.Ю. Волков

Быченков А.К., Курсовая работа направления подготовки «Программная инженерия» на тему «Веб-сервис доставки продуктов питания»: М. 2023 г., МИРЭА – Российский технологический университет (РТУ МИРЭА), Институт информационных технологий (ИИТ), кафедра инструментального и прикладного программного обеспечения (ИиППО) – 50 стр., 15 рис., 21 табл., 25 источн. (в т.ч. 22 на английском яз.), 7 прил.

Ключевые слова: Уникальные термины или словосочетания от 5 до 15 шт. (не подходит: “проектирование”, “БД”, “сервер”, “ИС” и т.д.)

Целью работы является проектирование специализированной системы семиотического управления информационными репликаторами. Спроектирована информационная система, разработан алгоритм управления знаниевыми информационными репликаторами, разработано спроектированное решение для клиентской стороны, разработана база данных и настроен локальный сервер.

Ivanov I.I., Final qualification work for the educational program "Software development and information system design" of the training program "Software engineering" on the topic "Cognitive information system of semiotic management of knowledge with information replicators": M. 2023, MIREA - Russian Technological University (RTU MIREA), Institute of Information Technologies (IIT), Department of the Tool and Applied Software (Department of TAS) - 50 p., 15 ill., 21 tabl., 25 ref. (inc. 22 in English), 7 apps.

Keywords: Unique terms or phrases from 5 to 15 pcs. (not suitable: "design", "DB", "server", "IS", etc.)

The aim of the work is to design a specialized system for semiotic control of information replicators. An information system was designed, an algorithm for managing knowledge information replicators was developed, a designed solution for the client side was developed, a database was developed and a local server was configured.

РТУ МИРЭА: 119454, Москва, пр-т Вернадского, д. 78

кафедра инструментального и прикладного программного обеспечения (ИиППО)

Тираж: 1 экз. (на правах рукописи)

Файл: «ПЗ\_РСЧИР\_ИКБО-32-21\_БыченковАК.pdf», исполнитель Быченков А.К.

© А.К. Быченков

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем отчете применяются следующие сокращения и обозначения:

OIDC,

ПЕРЕПИСАТЬ

|  |  |  |
| --- | --- | --- |
| OAuth | – | Open Authorization (открытый протокол авторизации) |
| SSO | – | Single Sign-On (технология единого входа) |
| JWT | – | JSON Web Token (открытый стандарт для создания токенов доступа, основанный на формате JSON) |
| REST | – | Representational State Transfer (передача репрезентативного состояния) |
| CRUD | – | Create, Read, Update, Delete (базовые функции, используемые при работе с базами данных: создание, чтение, модификация, удаление) |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc146766955)

ОСНОВНАЯ ЧАСТЬ

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФОРМИРОВАНИЕ ОСНОВНЫХ ТРЕБОВАНИЙ К ПРИЛОЖЕНИЮ 8](#_Toc146766956)

[2 ВЫБОР СРЕДСТВ ВЕДЕНИЯ РАЗРАБОТКИ 13](#_Toc146766957)

[2.1 Прикладное программное обеспечение, необходимое для разработки и функционирования веб-приложения 13](#_Toc146766958)

[2.2 Языки и технологии реализации веб-приложения 13](#_Toc146766959)

[3 Разработка ВЕб-приложения 14](#_Toc146766960)

[3.1 Конфигурация окружения 14](#_Toc146766961)

[3.2 Логическая структура веб-приложения 15](#_Toc146766962)

[3.3 Реализация авторизации пользователей 16](#_Toc146766963)

[3.4 Реализация бизнес-логики 20](#_Toc146766964)

[4 Тестирование 23](#_Toc146766965)

[ЗАКЛЮЧЕНИЕ 29](#_Toc146766966)

[СПИСОК ЛИТЕРАТУРЫ 30](#_Toc146766967)

ПРИЛОЖЕНИЯ

**ВВЕДЕНИЕ**

Развитие сферы услуг доставки продуктов питания непрерывно нарастает в контексте современных тенденций образа жизни. В условиях изменяющихся предпочтений потребителей и активного использования технологий возникает стремительная необходимость в создании эффективных веб-сервисов, обеспечивающих удобство, скорость и качество доставки.

Целью данной курсовой работы является разработка веб-сервиса по доставке продуктов питания, ориентированного на обеспечение клиентов возможностью удобного и оперативного заказа.

Актуальность данной работы определяется растущим спросом на услуги доставки продуктов питания среди потребителей, особенно в условиях изменений образа жизни, связанных с увеличением удаленной работы и ограничениями мобильности.

Объектом исследования является процесс разработки веб-сервиса доставки продуктов питания, а предметом - функциональность, архитектура и оптимизация процессов заказа и доставки через данный сервис.

Ожидаемый результат работы - создание работоспособного прототипа веб-сервиса, который будет демонстрировать эффективную работу механизмов заказа и доставки, удовлетворяя требованиям современных потребителей и обеспечивая оперативное выполнение заказов.

Для достижения поставленной цели будут рассмотрены современные технологии в сфере разработки веб-сервисов, а также анализированы основные аспекты, определяющие удовлетворенность клиентов услугами доставки продуктов питания.

**1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФОРМИРОВАНИЕ ОСНОВНЫХ ТРЕБОВАНИЙ К ПРИЛОЖЕНИЮ**

Для анализа предметной области было проведено сравнение двух популярных российских веб-сервисов доставки продуктов питания: СберМаркет (sbermarket.ru), Самокат (samokat.ru). Были выделены общие черты, функционал, который присутствует во всех сервисах.

Для возможности оформления заказа на каждом из сервисов клиентам необходимо выполнить вход: либо по номеру телефона с подтверждением по СМС, либо, используя Сбер ID OIDC, рисунки 1.1, 1.2. Так как отправка СМС сообщений платная, в разрабатываемом приложении будет использоваться вариант с OIDC, а именно identity-провайдер KeyCloak.

Клиентами сервиса являются либо наниматели (работодатели), либо соискатели (потенциальные сотрудники). На каждом сервисе им необходимо авторизоваться, используя телефон, электронную почту или сторонние сервисы (VK ID, Yandex ID или другие) для аутентификации по протоколу открытой авторизации OAuth 2, рис 1.1.

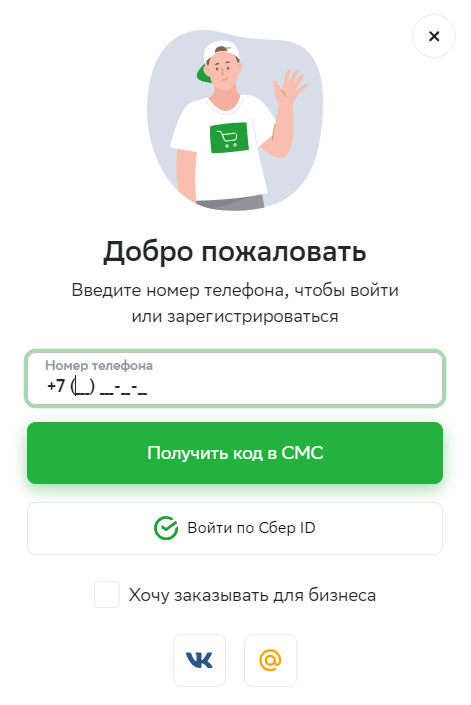


Рисунок 1.1 – Окно авторизации на sbermarket.ru



Рисунок 1.2 – Окно авторизации на samokat.ru

После авторизации клиентам показывается каталог с продуктами, разбитыми на категории, рисунок 1.3.

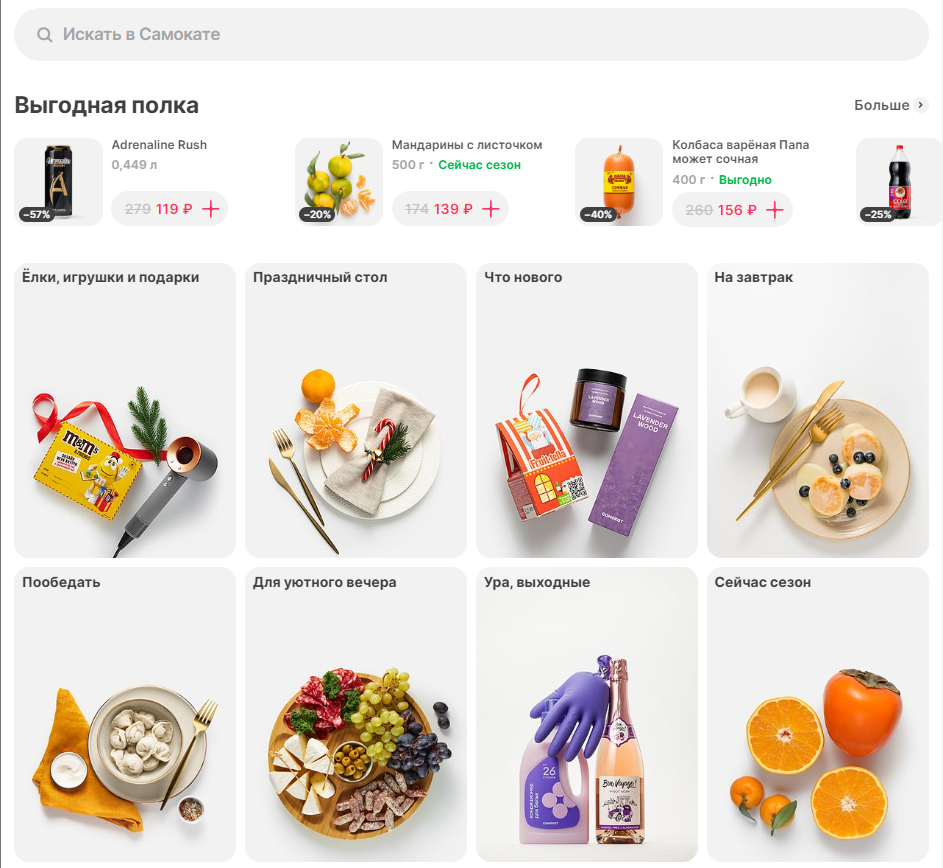
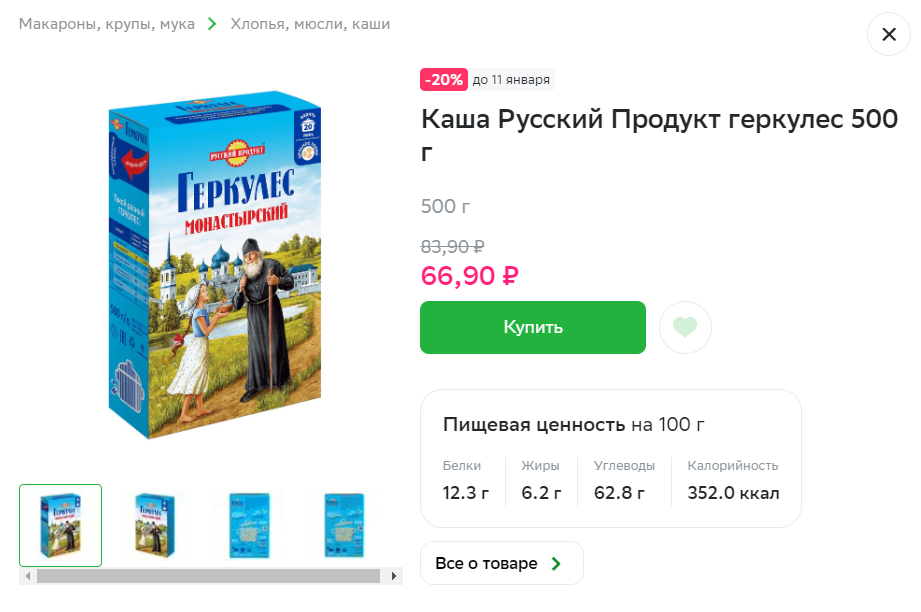


Рисунок 1.3 – Каталог продуктов на utkonos.ru

При выборе определенного продукта отображается его краткое описание



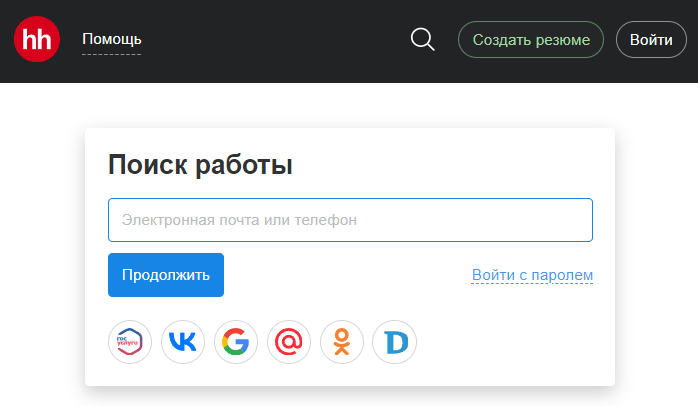


Рисунок 1.1 – Авторизация на сервисе HeadHunter

Затем нанимателям необходимо зарегистрировать свою компанию и разместить вакансии в необходимых регионах. Соискателям же необходимо создать резюме, указав желаемую позицию (должность), заработную плату, имеющееся профессиональные навыки, образование, предыдущие места работы. Для каждого соискателя сервисы предоставляют самые релевантные вакансии, рис. 1.2.

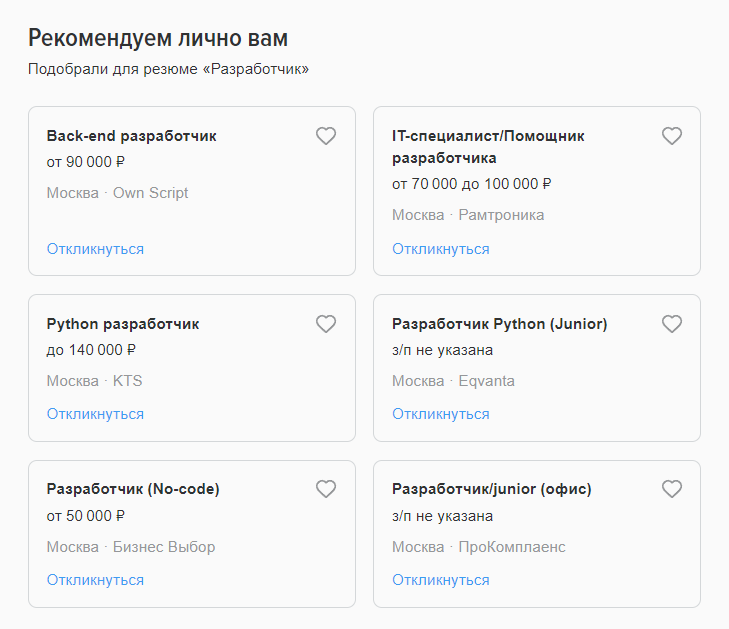


Рисунок 1.2 – Рекомендуемые вакансии

На главной странице сервисов для соискателей отображается список вакансий в их регионах. Вакансии можно отфильтровать по многим параметрам, включая: ключевые слова, уровень дохода, требуемый опыт, график работы, рис. 1.3.

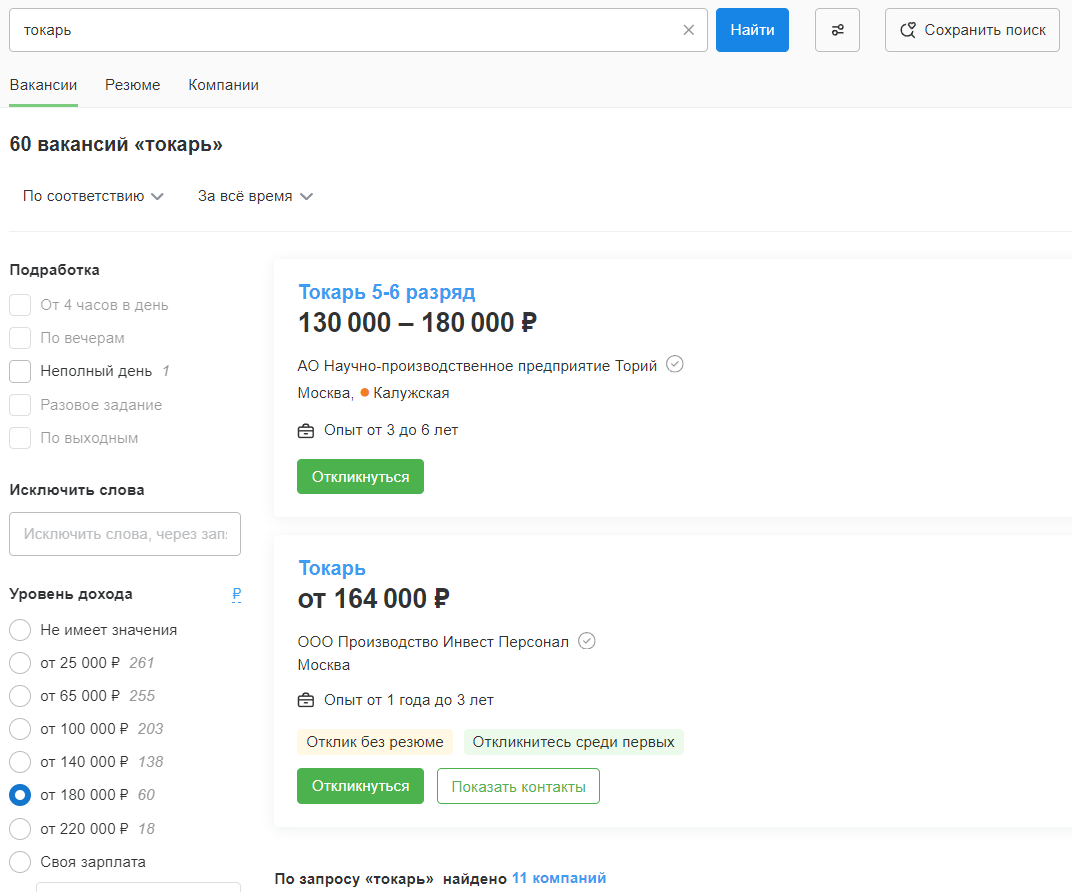


Рисунок 1.3 – Отфильтрованные вакансии

Для работодателей же сервисы предоставляют резюме сотрудников, которые находятся в поиске работы, рис. 1.4.

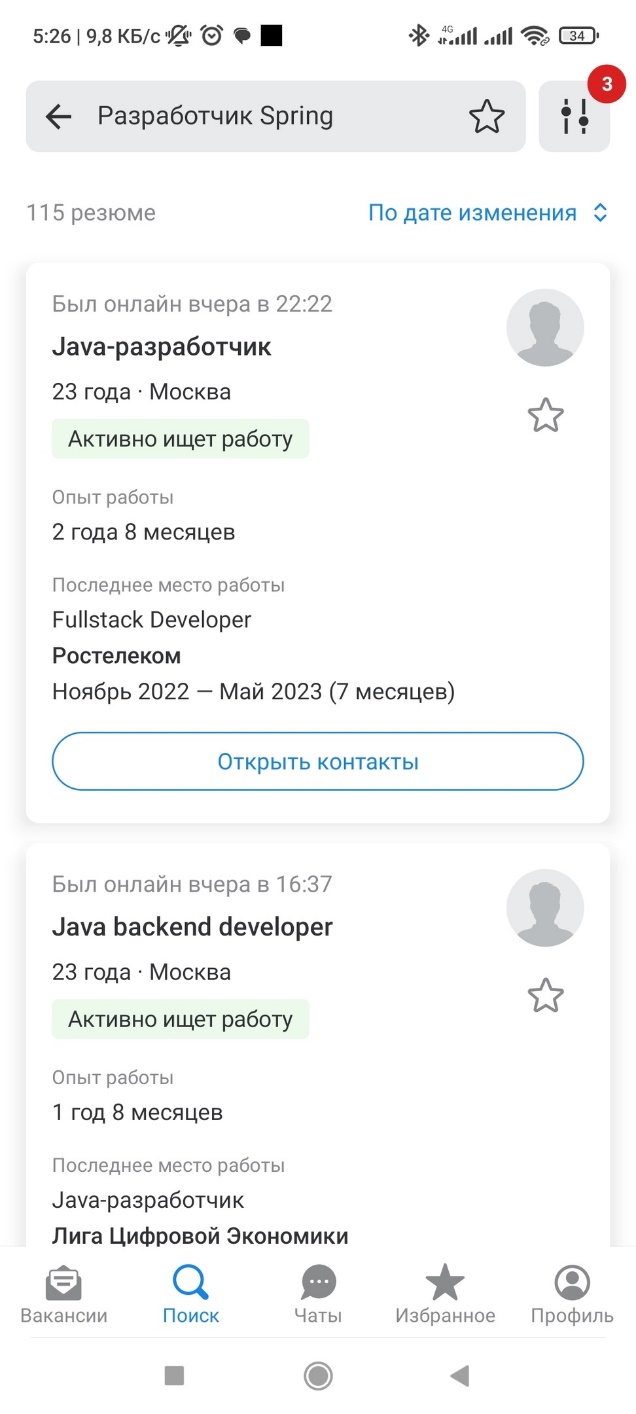


Рисунок 1.4 – Поиск потенциальных сотрудников

Кандидатов на требуемую должность можно найти как самостоятельно написав им, используя поиск по выложенным резюме, так и кандидаты могут откликнуться на вакансии, приложив резюме и написав сопроводительное письмо, рис. 1.5.

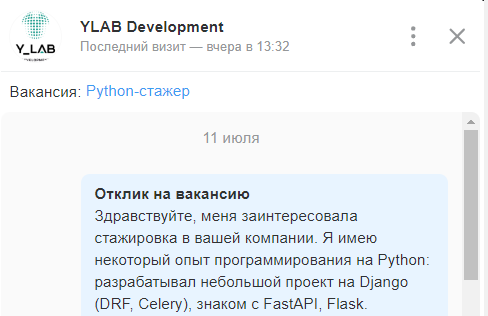


Рисунок 1.5 – Сопроводительное письмо для отклика на вакансию

Изучив вышеперечисленные сервисы, были сформированы основные требования к разрабатываемому приложению:

1. Надежная аутентификация пользователей в системе и их авторизация, используя различные роли: наниматель, соискатель и сервисные роли (модератор, администратор). Пользователи в зависимости от их роли должны иметь доступ только к определенным функциям системы;
2. Реализация сущностей системы: пользователь, регион, компания, вакансия, резюме, отклик;
3. Управление каждой сущностью и их фильтрация по различным параметрам.

**2 ВЫБОР И ОБОСНОВАНИЕ ТЕХНОЛОГИЙ РАЗРАБОТКИ**

**2.1 Прикладное программное обеспечение, необходимое для разработки и функционирования веб-приложения**

Для написания программного кода использовалась среда разработки VS Code, для функционирования приложения необходима программная платформа контейнеризации приложений Docker, которая в настоящее время является стандартом развертывания веб-приложений.

**2.2 Языки и технологии реализации веб-приложения**

Используется язык программирования Java 17 версии в связке с фреймворком Spring и системой сборки Gradle.

Данные приложения хранятся внутри СУБД PostgreSQL, запущенной внутри Docker-контейнера. СУБД PostgreSQL широко используется в настоящее время и является промышленным стандартом хранения данных. Взаимодействие с базой данных реализовано, используя модуль Spring ORM.

Для аутентификации пользователей используется KeyСloak – готовое SSO (Single Sign-On) решение, имеющее встроенную реализацию протокола OpenID Connect. Интегрировано с модулем Spring Security.

**3 Разработка ВЕб-приложения**

## **3.1 Конфигурация окружения**

Для функционирования необходимо подключение к базе данных PostgreSQL, серверу KeyCloak SSO. Приложение и окружение развертывается в Docker-контейнерах. Для этого был написан файл Docker Compose, рис. 3.1.

services:

postgres:

build: containers/postgres

container\_name: work-in-postgres

restart: always

ports:

- 5432:5432

environment:

POSTGRES\_USER: ${POSTGRES\_USER}

POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD}

POSTGRES\_DB: workin

volumes:

- postgres-data:/var/lib/postgresql/data

keycloak:

build: containers/keycloak

container\_name: work-in-keycloak

restart: always

command: start-dev --db postgres

ports:

- 8443:8443

environment:

KC\_DB\_URL: jdbc:postgresql://postgres:5432/keycloak

KC\_DB\_USERNAME: ${POSTGRES\_USER}

KC\_DB\_PASSWORD: ${POSTGRES\_PASSWORD}

KEYCLOAK\_ADMIN: ${KEYCLOAK\_ADMIN}

KEYCLOAK\_ADMIN\_PASSWORD: ${KEYCLOAK\_ADMIN\_PASSWORD}

KC\_HTTPS\_CERTIFICATE\_FILE: /opt/keycloak/conf/certs/tls.crt

KC\_HTTPS\_CERTIFICATE\_KEY\_FILE: /opt/keycloak/conf/certs/tls.key

healthcheck:

test:

["CMD", "curl", "--head", "fsS", "https://keycloak:8443/health/ready"]

interval: 5s

timeout: 2s

retries: 15

depends\_on:

- postgres

app:

image: descenty/work\_in\_spring

pull\_policy: always

container\_name: work-in-app

restart: always

ports:

- 8001:8001

environment:

POSTGRES\_HOST: postgres

POSTGRES\_PORT: 5432

POSTGRES\_DB: workin

KC\_PORT: 8443

env\_file:

- .env

depends\_on:

- postgres

- keycloak

volumes:

postgres-data:

Рисунок 3.1 – Листинг docker-compose.yaml

**3.2 Логическая структура веб-приложения**

Все Java-классы разбиты по модулям и слоям, следуя по архитектурному паттерну HMVC (Hierarchical Model–View–Controller), который позволяет решить некоторые проблемы масштабируемости приложений, имеющих классическую MVC-архитектуру. Благодаря разделению по модулям, в проекте легко найти Java-файлы для любого модуля приложения, рисунок 3.1.



Рисунок 3.1 – Логическая структура веб-приложения

**3.3 Реализация авторизации пользователей**

Для авторизации пользователей используется готовое SSO (Single Sign-On) решение KeyCloak. Keycloak — это полноценная реализация Identity Provider для OpenID Connect (а значит, Authorization Server для OAuth2). Для работы приложения был создан отдельный Realm с названием workin, в него добавлен клиент под названием backend. Этот клиент будет использоваться приложением для доступа к пользователям, их аутентификации и авторизации, получения ролей.

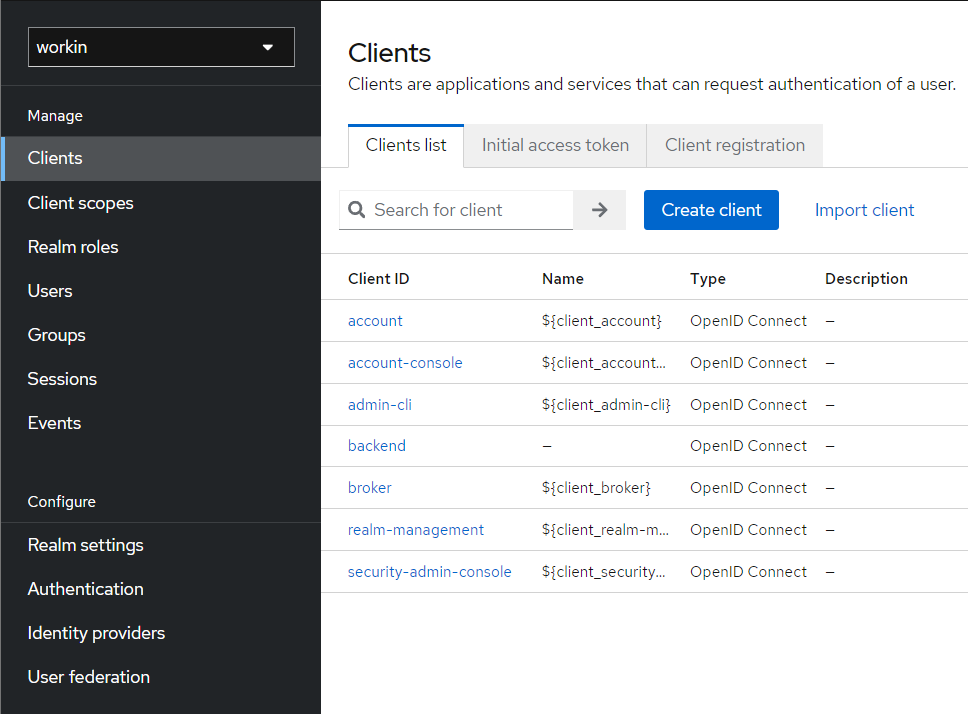


Рисунок 3.2 – Экран клиентов KeyCloak

Для работы с KeyCloak в конфигурации приложения application.yml написаны URL для KeyCloak Resource Server, ID и имя созданного клиента backend, и ключ доступа для него.

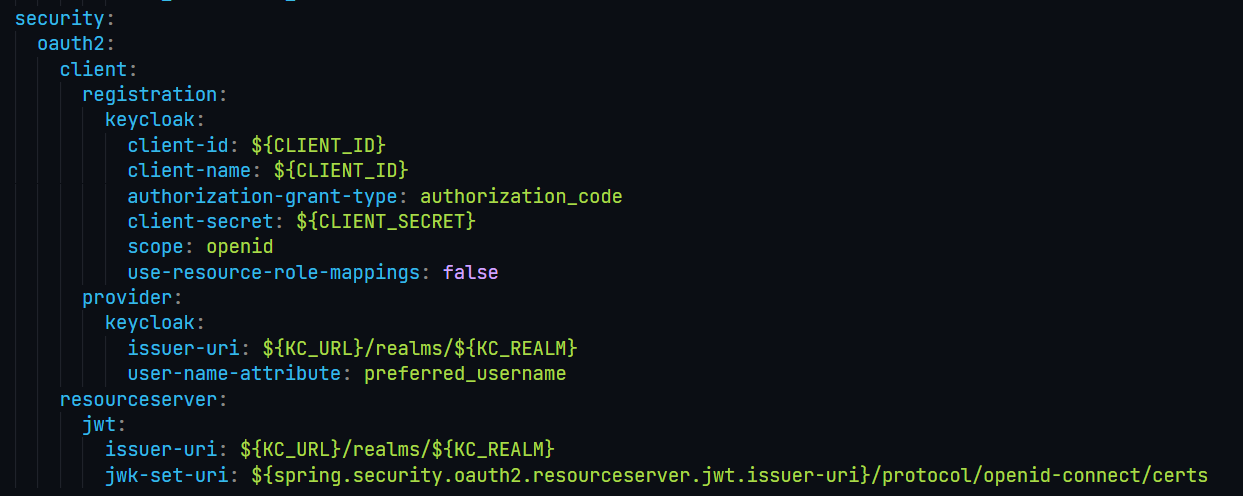


Рисунок 3.3 – Конфигурация Spring Security в application.yml

Был также написан Java-класс для конфигурации Spring Security, рис. 3.4



Рисунок 3.4 – Java-класс конфигурации Spring Security

Аутентификация и авторизация пользователей производится по JWT-токенам, которые выдаются KeyCloak. В токены также записываются роли пользователя в системы (модератор, администратор).

Для взаимодействия с KeyCloak написан UserController, отвечающий за регистрацию, аутентификацию. Через него администратор также может добавить, удалить роли или удалить пользователя, рисунок 3.5.

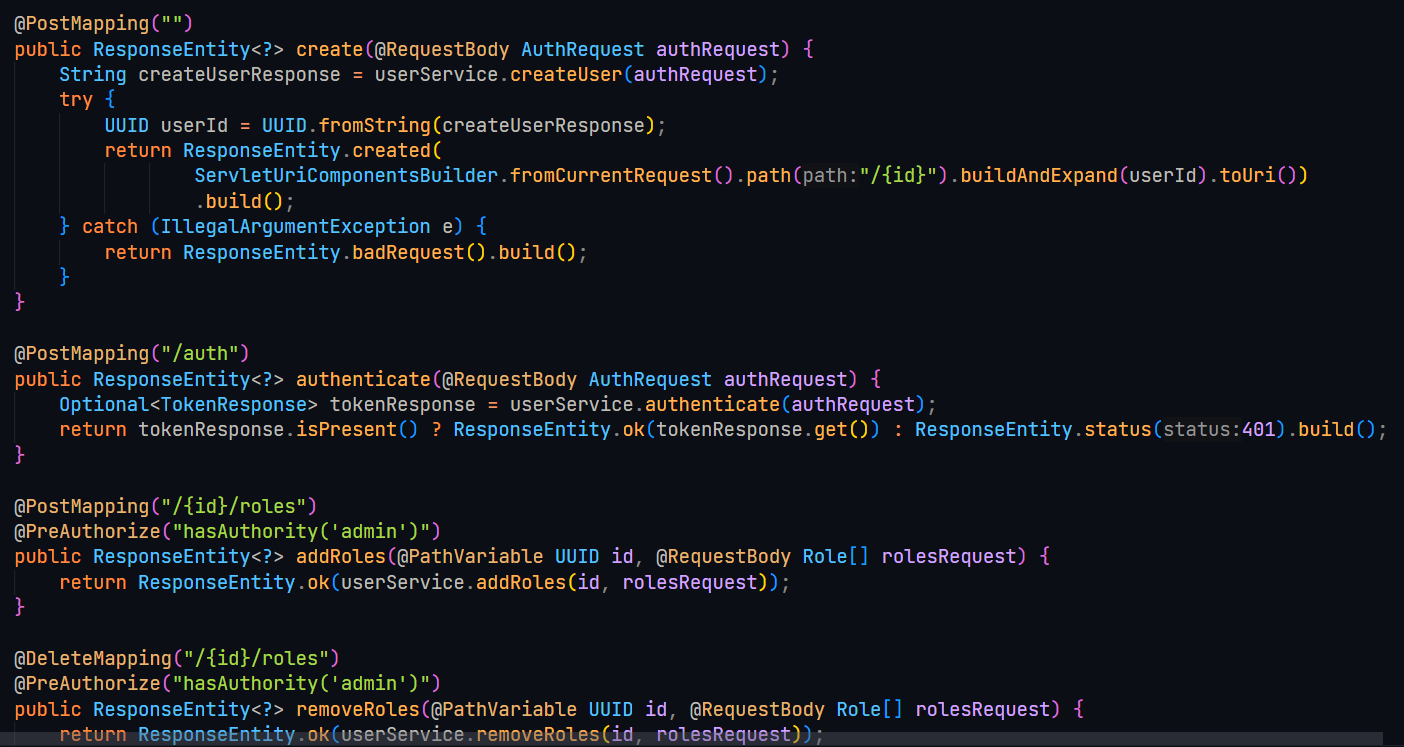


Рисунок 3.5 – Листинг UserController

Этот контроллер является обёрткой над KeyCloak. В сервисном слое лишь отправляются запросы на KeyCloak-сервер, рисунок 3.6.

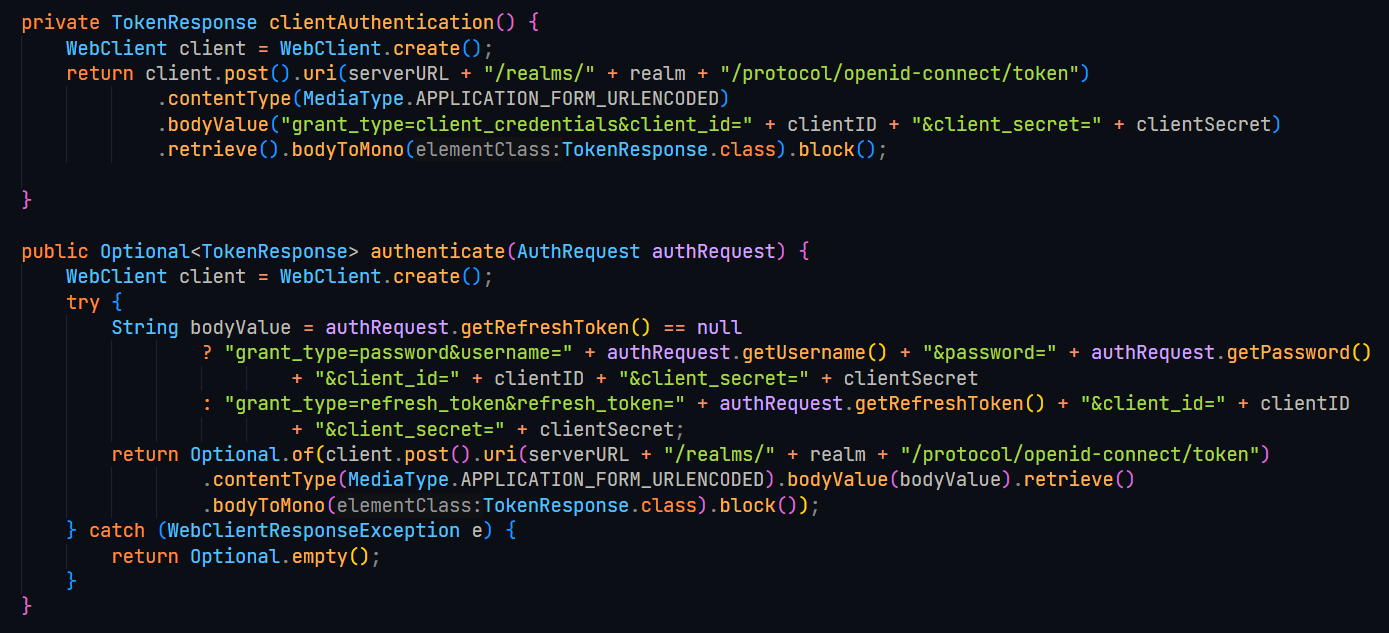


Рисунок 3.6 – Листинг UserService

Интеграция с KeyCloak настроена, JWT-токены могут передаваться в Authorization заголовке запроса. Контроллеры могут предоставлять доступ к эндпоинтам по различным условиям. К примеру, AreaController ограничивает возможность создания новых регионов для обычных пользователей, только для администраторов, рисунок 3.7.



Рисунок 3.7 – Листинг AreaController

## **3.4 Реализация бизнес-логики**

Сущности базы данных разработаны, используя Spring Data JPA. Все сущности связаны друг с другом внешними ключами и для них настроено каскадное удаление. Пример реализации сущности региона, рисунок 3.8.

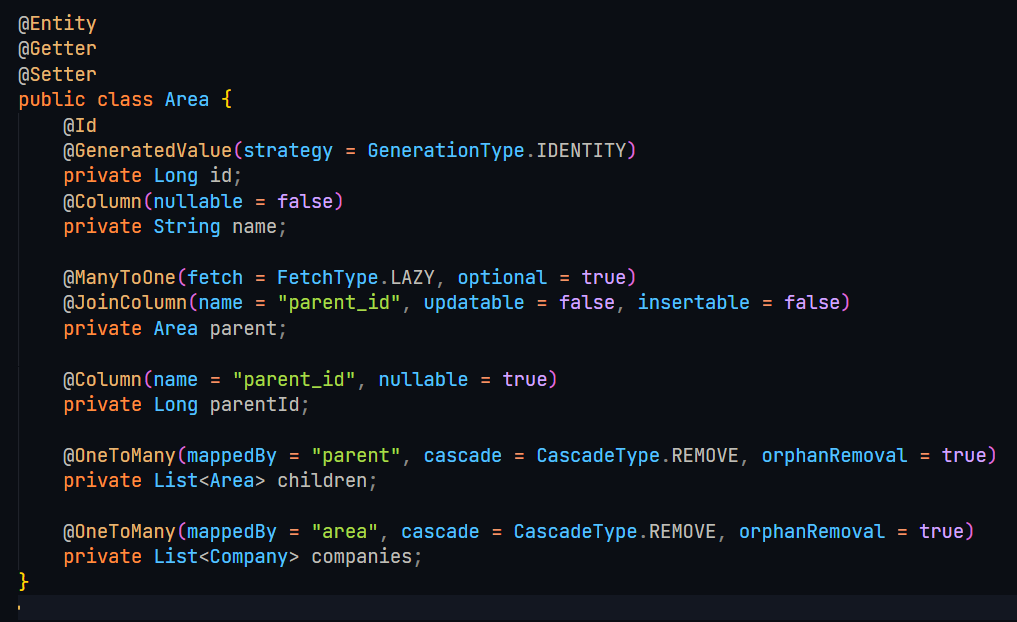


Рисунок 3.8 – Листинг Area

Взаимодействие с базой данных реализовано, используя Spring Data JpaRepository. Репозитории автоматически для каждой сущности создают CRUD-методы. Дополнительные методы для запросов к базе данных можно создать, написав соответствующие спецификации методов, используя предопределенные ключевые слова. Например, на рисунке 3.9 показаны пользовательские методы к сущности компании.

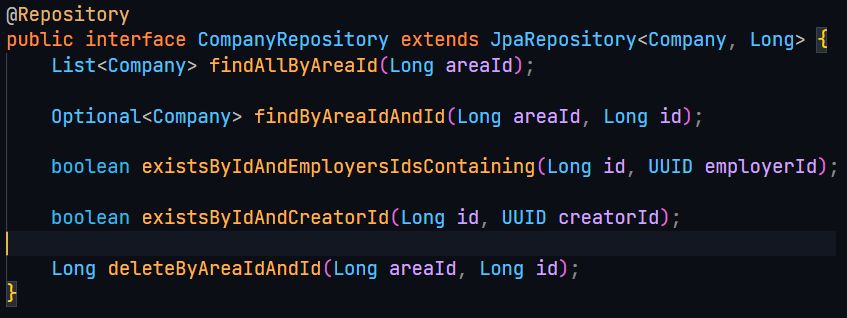


Рисунок 3.9 – Листинг CompanyRepository

Для работы с репозиториями для сущности региона написан сервисный слой, где вызываются методы репозитория, рисунок 3.9.



Рисунок 3.10 – Листинг AreaService

В сервисный слой передается одно DTO для создания новых объектов, их изменения, другое DTO возвращается для конвертации объектов в JSON. Для преобразования DTO в сущности базы данных и наоборот используется мапперы из библиотеки Mapstruct. Пример маппера для сущности региона, рисунок 3.10.

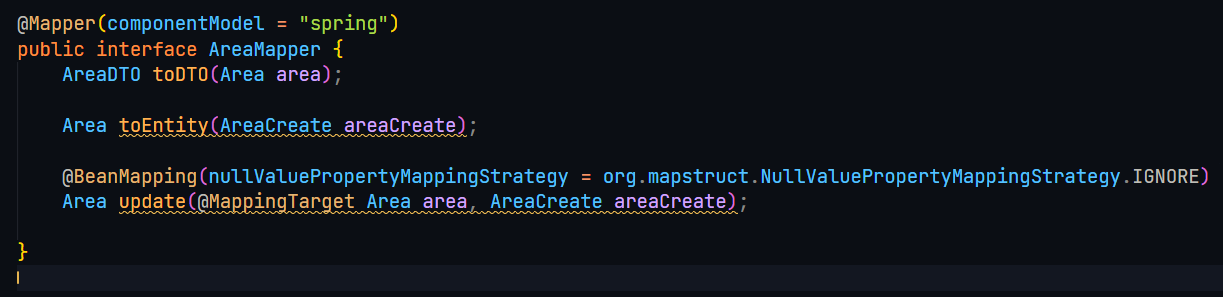


Рисунок 3.11 – Листинг AreaMapper

Например, для создания объекта региона необходимо передать объект класса AreaCreate, затем этот объект будет преобразован в сущность Area, сохранен в базе данных, клиенту вернется код 201, а в заголовке ответа Location будет записан URL для доступа к созданному объекту.

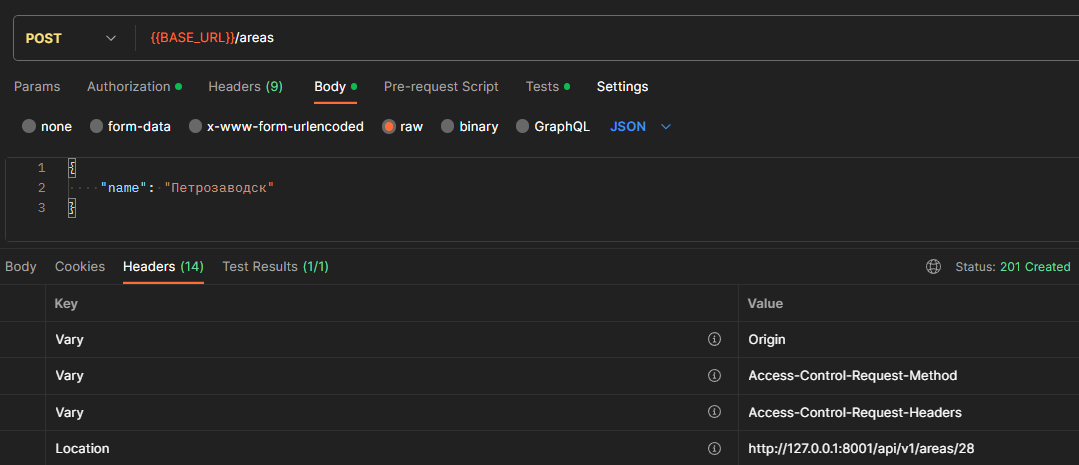


Рисунок 3.12 – Пример запроса в Postman

Для каждой сущности базы данных были реализованы CRUD-операции и фильтрация по различным параметрам. Например, для вывода всех вакансий компании в определенном регионе используется запрос, показанный в Postman на рисунке 3.12.

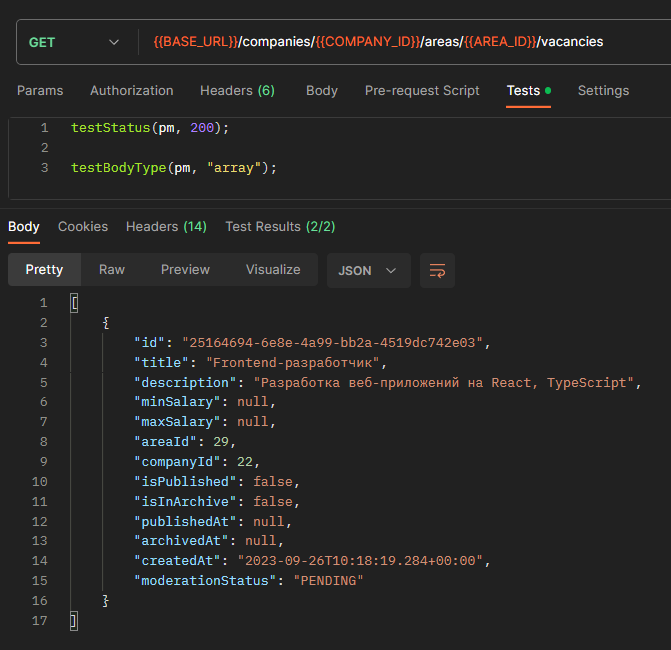


Рисунок 3.13 – Пример вывода всех вакансий компании

**4 Тестирование**

Веб-приложение было протестировано с помощью инструмента тестирования API – Postman. Всего было написано 197 тестов (проверка статусов ответа сервера, типа отправляемого объекта, значений полей, существования объектов в массиве), которые разбиты на 6 тестовых сценариев, рисунок 4.1, 4.2.

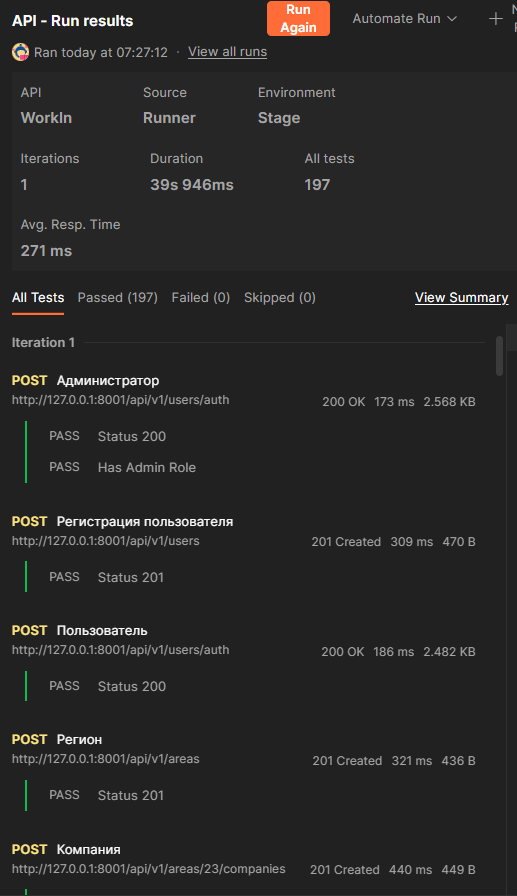


Рисунок 4.1 – Пройденные тесты

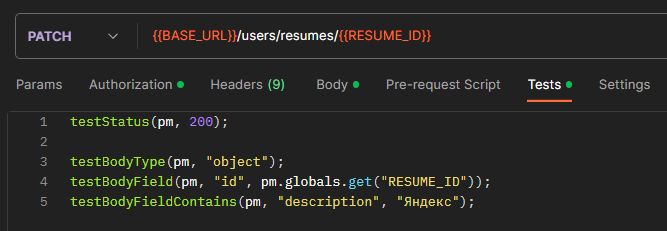


Рисунок 4.2 – Пример тестов для изменения резюме по ID

1. Проверка каскадного удаления всех сущностей

Происходит авторизация сервисного аккаунта администратора и регистрация нового пользователя. От имени администратора создаются Регион, Компания, Вакансия, пользователь создает Резюме и Отклик. Затем удаляется Регион и проверяется, что все связанные сущности (Компания, Вакансия, Отклик) теперь не существуют. Наконец, удаляется пользователь и проверяется на существование Резюме, рисунок 4.3.

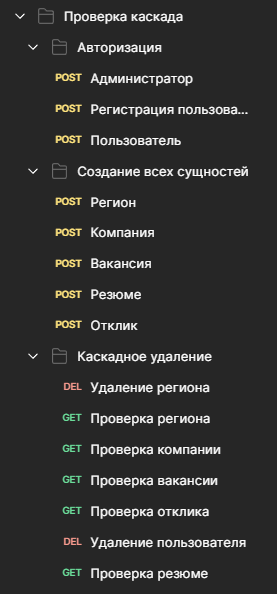


Рисунок 4.3 – Проверка каскадного удаления сущностей

1. CRUD-операции и проверка прав для Региона, рис 4.4.

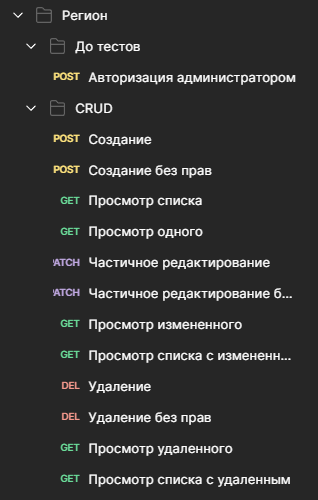


Рисунок 4.4 – Тестирование для Региона

1. CRUD-операции и проверка прав для Компании, рис 4.5.

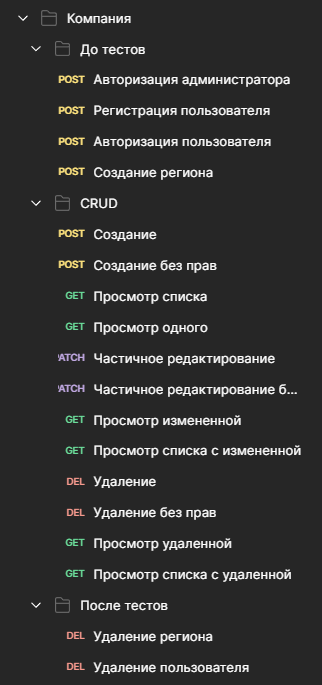


Рисунок 4.5 – Тестирование для Компании

1. CRUD-операции, проверка прав, проверка модератором для Вакансии, рисунок 4.6.

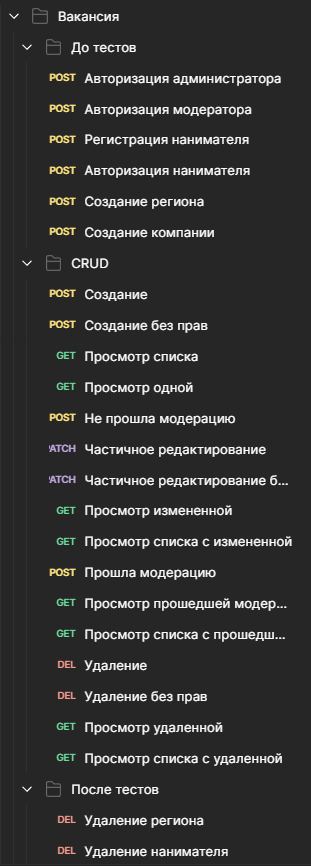


Рисунок 4.6 – Тестирование для Вакансии

1. CRUD-операции, проверка прав и модерация для Резюме, рисунок 4.7.

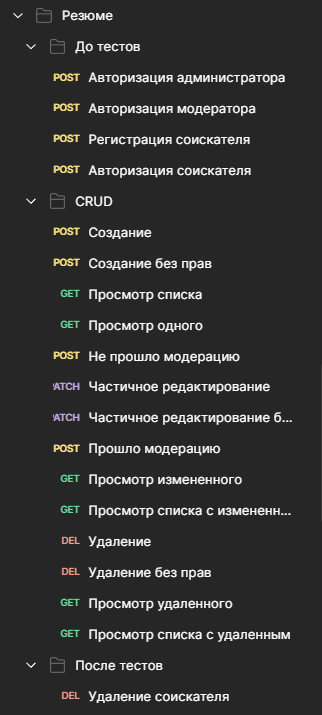


Рисунок 4.7 – Тестирование для Резюме

1. CRUD-операции, проверка прав, модерация для Отклика, рисунок 4.8.

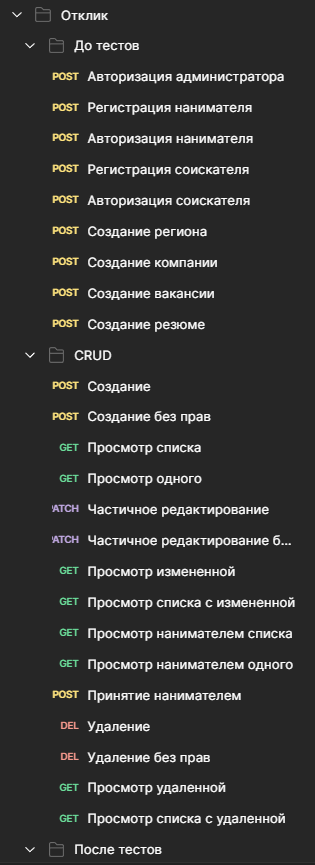


Рисунок 4.8 – Тестирование для Отклика

**ЗАКЛЮЧЕНИЕ**

В рамках данной курсовой работы, используя возможности языка Java совместно с фреймворком Spring, разработан RESTful веб-сервис для автоматизации поиска потенциальных сотрудников.

Были использованы актуальные подходы к разработке современные веб-сервисов, включая контейнеризацию, OAuth 2 аутентификацию.

Для основного функционала приложения (REST API) написаны тестовые сценарии, используя Postman как инструмент автоматизированного тестирования.

Курсовая работа позволила изучить на практике востребованные веб-технологии и применить в разработке современные подходы. В результате были выполнены поставленные задачи и достигнуты все цели курсовой работы.

С готовым проектом можно ознакомиться в GitHub-репозитории:

<https://github.com/descenty/work_in_spring>

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Spring Boot Documentation [Электронный ресурс]. – URL: <https://docs.spring.io/spring-boot/docs/current/reference/html/> (дата обращения 26.09.2023).
2. JPA @Entity Annotation [Электронный ресурс]. – URL: <https://www.baeldung.com/jpa-entity> (дата обращения 26.09.2023).
3. PostgreSQL Documentation [Электронный ресурс]. – URL: <https://www.postgresql.org/docs/> (дата обращения 26.09.2023).
4. KeyCloak Documentation [Электронный ресурс]. – URL: <https://www.keycloak.org/documentation> (дата обращения 26.09.2023).
5. Spring Boot Reference Documentation [Электронный ресурс]. – URL: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/> (дата обращения 26.09.2023).
6. Spring Data JPA [Электронный ресурс]. – URL: <https://spring.io/projects/spring-data-jpa> (дата обращения 06.05.2023).
7. Spring MVC [Электронный ресурс]. – URL: <https://docs.spring.io/spring-framework/docs/current/reference/html/web.html> (дата обращения 07.05.2023).
8. Postman Documentation [Электронный ресурс]. – URL: <https://learning.postman.com/docs/introduction/overview/> (дата обращения: 26.09.2023)