**РУКОВОДСТВО ПРОГРАММИСТА**

**«Централизованный сервис для учёта книг библиотеки»**

Киров, 2023 г.

# Назначение проекта

«Централизованный сервис для учёта книг библиотеки» предназначена для ведения учёта книг и их выдачи читателям и отслеживания статистики по выдаче.

# Основной функционал

* Возможность получения статистики по выдаче книг - самые популярные книги за период,
* Возможность внесения/изменения данных о читателях (ФИО, телефон, номер читательского билета), книгах (номер, название, автор, год издания) и выдаче (книга, читатель, дата выдачи, срок и статус возврата),
* Просмотр списка невозвращенных книг (если срок выдачи вышел/ещё не вышел),
* Синхронизованная работа с различных устройств (соответствующих системным требованиям).

# Требования к техническим средствам

Для корректной работы приложения необходимо: ПК, клавиатура, мышь, монитор с разрешением не менее 1280x1024, доступ в Интернет.

Минимальные характеристики ПК:

* ОС Windows 10 x64;
* Процессор - AMD Ryzen 3 2200G;
* не менее 4 ГБ оперативной памяти;
* не менее 1 ГБ на жёстком диске для данного ПО;

Для работы backend-сервиса необходим веб-сервер с аналогичными характеристиками и доступом в Интернет.

# Требования к программным средствам

Для серверной части приложения необходимо установить: JDK 17, PostgreSQL 14, система сборки Maven.

Для клиентского приложения необходимо утановить Python 3 и систему управления пакетами Pip.

# Установка, настройка и запуск программы

В репозитории, доступном по ссылке <https://github.com/Mihail-Ko/rest-library-yp>, размещены исходные файлы проекта для backend и клиент-приложения в соответствующих директориях, которые необходимо клонировать.

Для работы программы потребуется установить СУБД Postgres 15 версии, создать базу, назначив ей название (в проекте используется название “library\_db”), имя пользователя и пароль.

Вариант с установкой Postgres с помощью Docker:

Создать Dockerfile со следующим содержимым, при этом указав нужные параметры логина и пароля:

FROM postgres:15.4-alpine3.18

WORKDIR /app

COPY . .

ENV POSTGRES\_USER=postgres

ENV POSTGRES\_PASSWORD=postgres

ENV POSTGRES\_DB=library\_db

EXPOSE $PORT

VOLUME ["pgdata:/var/lib/postgresql/data/"]

Создать образ из Dockerfile с помощью команды:

docker build -t library\_pg\_image .

Создание и запуск контейнера:

docker run --name library\_pg\_cont -p 5432:5432 -v pgdata:/var/lib/postgresql/data/ -d library\_pg\_image

Настройка backend-сервиса.

На рисунке 1 представлена схема устройства микросервисов в проекте. APIGateway является входным шлюзом, discovery-service управляет микросервисами, само приложение rest-service может быть запущено в нескольких экземплярах, адреса и порты будут выбраны автоматически.

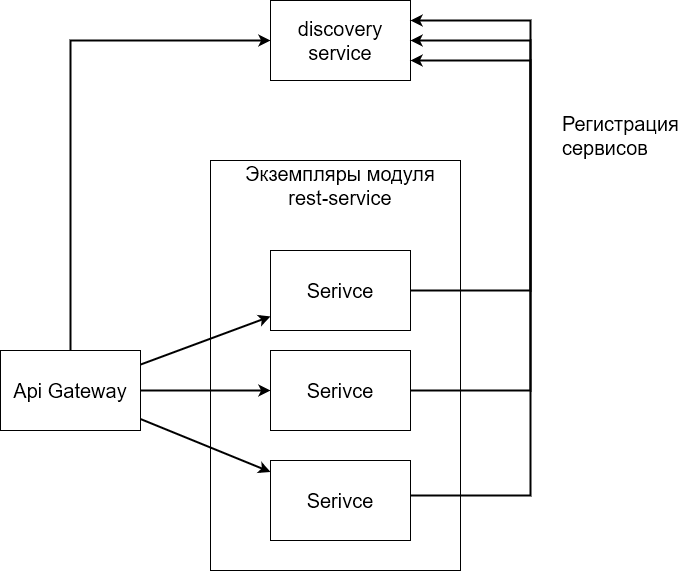


Рисунок 1 – Схема микросервисов

Файл cache-rest-service\api-gateway\src\main\resources\application.yml содержит параматры адреса хоста и порт.

Файл cache-rest-service\discovery-service\src\main\resources\application.yml содержит настройки адреса расположения Spring Cloud eureka, в остальных файлах необходимо ссылаться на этот адрес.

Файл cache-rest-service\rest-service\src\main\resources\application.yml содержит настройки подключения к базе, необходимо указать её адрес и порт, название, имя пользователя и пароль. Также в файле заданы настройки Spring Security: логин и пароль. Также в файле задан параметр порта 0, т.е. динамический, т.к. обращаться к нему будет Api Gateway, который принимает запросы извне.

Файл cache-rest-service\rest-service\src\main\resources\hazelcast.yml содержит настройки кэша, в данном случаи указано его время жизни в секундах и распределённый режим его работы.

Сборка backend части программы, если установлен Maven осуществляется командой:

mvn clean package

Для Windows в репозитории проекта есть пакетный файл в случаи отсутствия Maven, команда будет иметь следующий вид:

mvnw clean package

Для запуска необходимо выполнить три команды из трёх терминалов:

java -jar discovery-service\target\discovery-service-0.0.1-SNAPSHOT.jar

java -jar api-gateway\target\api-gateway-0.0.1-SNAPSHOT.jar

java -jar rest-service\target\rest-service-0.0.1-SNAPSHOT.jar

После чего необходимо дождаться регистрации сервисов, о статусе запуска можно узнать из логов.

Swagger интерфейс доступен по адресу 127.0.0.2/swagger-ui/index.html (для изначальной конфигурации).

Для запуска клиентского приложения необходимо установить виртуальную среду и пакеты: requests и PyQT6, а также указать URL-адрес для API в файле settings.py

# Возможности модернизации и настройки системы

Для API backend-сервиса есть возможность создания любого клиента, а также отправки запросов напрямую.

Данное web-приложение имеет широкие возможности для модернизации и настройки, в том числе добавление оптимизация кэширования (или его отключение при необходимости), добавление новых API методов и сущностей для базы данных, улучшения графического интерфейса программы, изменение типа авторизации (в данном случаи базовая), изменение способа хранения пароля в клиентской программе.

Дополнительные настройки, добавленные в yml-файлах:

* Для сервисов настроено логирование: отображение sql-запросов к базе со скрытием данных, отключено логирование для discovery-service.
* CSRF-токен включён
* Настройка маршрутизации для сокращения URL находится в файле cache-rest-service\api-gateway\src\main\resources\application.yml после параметра predicates.

Файл rest-service\src\main\resources\database\data.sql содержит тестовые данные для заполнения базы данных.