



ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»  
Институт компьютерных наук и кибербезопасности  
Высшая школа управления кибер-физическими системами

по дисциплине

Выполнил студент

группы

Руководитель

Санкт-Петербург  
2024

# Содержание

<b>1 Введение</b>	<b>2</b>
<b>2 Постановка задачи</b>	<b>3</b>
<b>3 Проектирование микросервисной архитектуры</b>	<b>4</b>
<b>4 Микросервис авторизации</b>	<b>5</b>
4.1 Метод регистрации . . . . .	5
4.2 Метод авторизации . . . . .	5
4.3 Метод проверки токена . . . . .	5
<b>5 Микросервис рекордов</b>	<b>7</b>
<b>6 Прокси-сервер</b>	<b>8</b>
<b>7 Фронтэнд</b>	<b>11</b>
<b>8 Заключение</b>	<b>13</b>

# 1 Введение

Микросервисная архитектура становится все более популярной среди разработчиков программного обеспечения благодаря своей гибкости, масштабируемости и возможности быстрой разработки и развертывания приложений. Одной из ключевых особенностей микросервисов является их способность работать в виде независимых сервисов, каждый из которых отвечает за конкретную функциональность.

В настоящей курсовой работе рассматривается создание сервера для многопользовательской онлайн игры с использованием микросервисной архитектуры. Использование микросервисной архитектуры позволяет абстрагировать логику авторизации от основной игровой логики. Целью данной работы является изучение принципов построения приложения с использованием микросервисной архитектуры.

Данная курсовая работа имеет актуальное значение в контексте развития современных информационных технологий и представляет интерес для специалистов в области программной инженерии, системного анализа и разработки прикладных решений на основе микросервисов.

## 2 Постановка задачи

В рамках данной работы необходимо разработать веб-сервер возможностью авторизации пользователя в свой аккаунт, возможности создания нового аккаунта, а также возможности опубликовать новый рекорд и получить актуальный список рекордов с игрового сервера. Полученное приложение необходимо контейнеризировать, и предоставить пример использования и запуска.

### 3 Проектирование микросервисной архитектуры

Логика приложения разделена на четыре составляющие: два микросервиса, nginx проху, и пример фронтенда приложения.

В проекте использованы следующие микросервисы:

- Микросервис авторизации, предоставляющий API эндпоинты для создания нового пользователя и авторизации с помощью BASIC Auth. Микросервис лидерборд, предоставляющий endpoint для получения списка рекордов и публикации нового рекорда.
- Для поддержки авторизации в приложении был развернут прокси-сервер с использованием engines для перенаправления запросов на сервер через сервис авторизации.

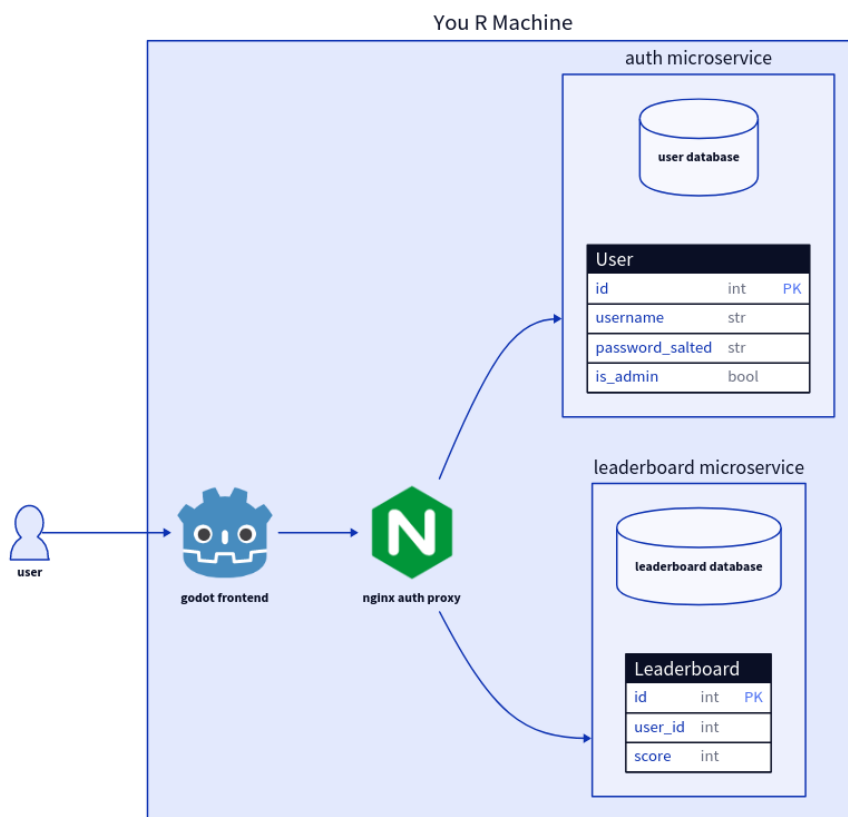


Рис. 1: Архитектура приложения

## 4 Микросервис авторизации

Микросервис авторизации имеет три основных метода: регистрацию, авторизацию и проверку токена (рис. 2).

### 4.1 Метод регистрации

Метод регистрации создаёт запись в базу данных с данными о пользователе включая имя пользователя и зашифрованный пароль. Сохранение пароля в зашифрованном виде используется с помощью библиотеки bcrypt.

### 4.2 Метод авторизации

Метод авторизации: авторизация осуществляется по протоколу basic auth точка В результате успешной авторизации API возвращает username пользователя и его ID.

### 4.3 Метод проверки токена




Метод проверки токена: в настоящий момент проверка токена заключается в проверке хедера basic auth. Данный метод используется прокси-сервером для определения прав доступа конкретного пользователя. В случае получения корректных логина и пароля метод проверки токена возвращает сообщение о валидности токена точка в дальнейшем планируется реализовать выдачу пользователю JWT токена и его проверку.

Authorize 

## Users ^

GET	/api/v1/oauth/users/	Get Users	▼
POST	/api/v1/oauth/users/	Create User	▼
GET	/api/v1/oauth/users/{user_id}/	Get User	▼
PATCH	/api/v1/oauth/users/{user_id}/	Update User Partial	▼
DELETE	/api/v1/oauth/users/{user_id}/	Delete User	▼

## Demo Auth ^

GET	/api/v1/oauth/basic-auth-username/	Demo Basic Auth Username	 ▼
GET	/api/v1/oauth/basic-auth/	Demo Basic Auth Credentials	 ▼
GET	/api/v1/oauth/token-introspection/	Demo Basic Token Introspection	 ▼

## default ^

GET	/	Hello Index	▼
GET	/hello/	Hello	▼
GET	/calc/add/	Add	▼

## Schemas ^

HTTPValidationError > Expand all object

UserCreate > Expand all object

UserSchema > Expand all object

UserUpdatePartial > Expand all object

ValidationError > Expand all object

Рис. 2: API эндпоинты метода авторизации

## 5 Микросервис рекордов

Микросервис рекордов отвечает за управление записями в базе данных соответствующими лучшим очкам игроков в процессе игры. Для осуществления этой задачи были реализованы следующие методы: создание новой записи в таблице, получение всех записей с именами пользователя и соответствующими очками получение списка лучших очков по запрошенному количеству.

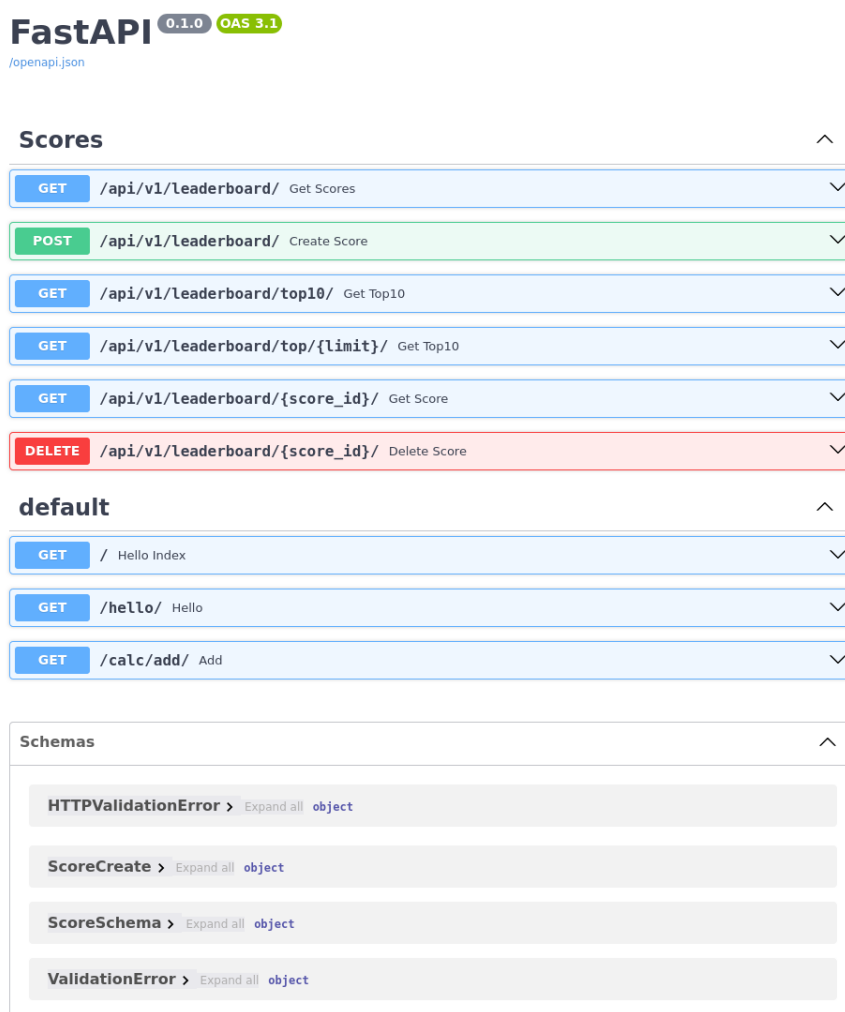


Рис. 3: API endpoints метода рекордов



## 6 Прокси-сервер

Для реализации авторизации в приложении был использован прокси-сервер реализованные с помощью nginx.

Данный сервер позволяет перенаправить любой запрос на сервис рекордов на сервис авторизации для проверки прав доступа. Для этого в файле конфигурации `nginx.conf` были прописаны маршруты сервиса авторизации и сервиса рекордов для перенаправления. Также реализована функция на JavaScript для проверки текущего токена полученного от пользователя. При получении web запроса производится запрос в метод интроспекции токена на сервисе авторизации. В случае подтверждения валидности токена запрос пробрасывается по назначению. В случае некорректности токена возвращается ошибка 401.

```
1 worker_processes 1;
2 load_module modules/nginx_http_js_module.so;
3
4 events {
5     worker_connections 1024;
6 }
7
8 http {
9     js_import /etc/nginx/conf.d/oauth2.js; # Location of JavaScript code
10
11     server {
12         listen 8000;
13         server_name _;
14
15         location * {
16             if ($request_method = 'OPTIONS' ) {
17                 # add_header Content-Length 0;
18                 # add_header Content-Type text/plain;
19                 # add_header Content-Type: application/json;
20                 # add_header Content-Length: 33;
21                 add_header Allow: "*";
22                 add_header Access-Control-Allow-Origin: "*";
23                 add_header Access-Control-Allow-Credentials: true;
24                 return 200;
25             }
26         }
27
28         location / {
29             return 200;
30         }
31     }
32 }
```

```

31
32     location /api/v1/oauth/ {
33         auth_request off;
34         proxy_set_header Host $host;
35         proxy_pass http://ms.auth:8000;
36     }
37
38     location /api/v1/ {
39         auth_request /_oauth2_token_introspection;
40         # auth_request off;
41         error_page 403 /403.json;
42         error_page 401 /401.json;
43         error_page 500 /401.json;
44         error_page 404 /404.json;
45
46         location ~ leaderboard/ {
47             proxy_set_header Host $host;
48             proxy_pass http://ms.leaderboard:8000;
49         }
50     }
51
52     location /404.json {
53         return 404 '{"error": "Requested resource not found"}';
54     }
55
56     location /401.json {
57         return 401 '{"error": "Unauthenticated"}';
58     }
59
60     location /403.json {
61         return 403 '{"error": "Forbidden"}';
62     }
63
64     location = /_oauth2_token_introspection {
65         if ($request_method = 'OPTIONS' ) {
66             # add_header Content-Length 0;
67             # add_header Content-Type text/plain;
68             # add_header Content-Type: application/json;
69             # add_header Content-Length: 33;
70             add_header Allow: "*";
71             add_header Access-Control-Allow-Origin: "*";
72             add_header Access-Control-Allow-Credentials: true;
73             return 200;
74         }
75         internal;
76         js_content oauth2.introspectAccessToken;
77     }
78

```

```

79     location /_oauth2_send_request {
80         internal;
81         proxy_method      GET;
82         proxy_set_header  Host $host;
83         proxy_set_header  Authorization $http_authorization;
84         proxy_pass_header  Authorization;
85         proxy_pass         http://ms.auth:8000/api/v1/oauth/token-
introspection/;
86         proxy_set_header  Content-Length "";
87
88         proxy_ignore_headers  Cache-Control Expires Set-Cookie;
89     }
90 }
91 }

```

## 7 Фронтэнд

Для реализации фронтенда приложение было решено использовать игровой движок godot. Данный игровой движок достаточно прост в освоении и позволяет с небольшим количеством трудозатрат получить графический интерфейс и осуществлять веб-запросы.

Приложение предоставляет возможность авторизоваться пользователю, возможность создать новый аккаунт и взаимодействовать с таблицей рекордов точка есть возможность получить топ n рекордов с сервера или же загрузить на сервер новый рекорд из-под своего логина.

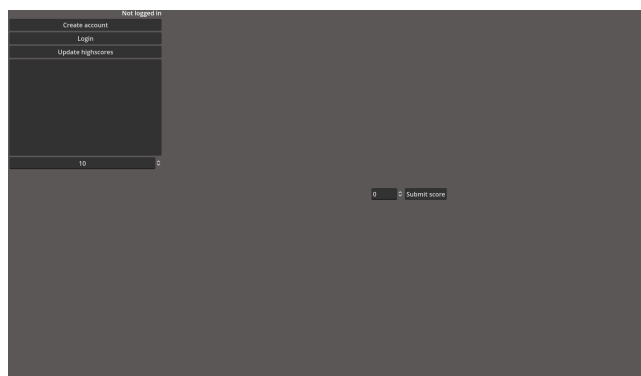


Рис. 4: Внешний вид фронтэнда приложения

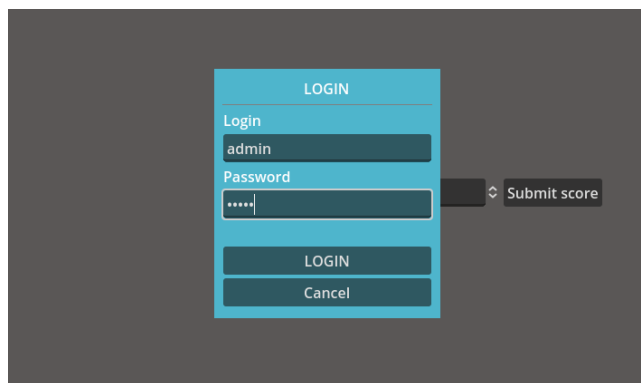


Рис. 5: Окно авторизации

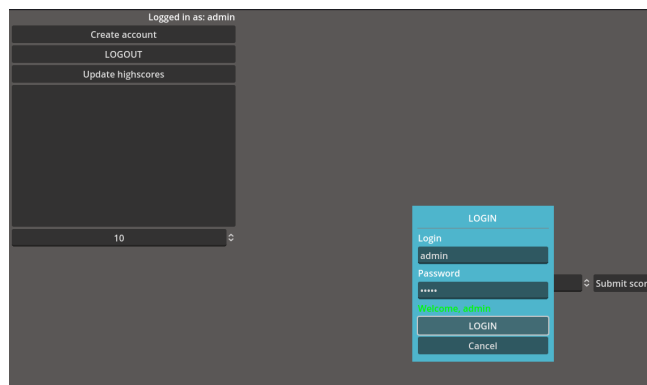


Рис. 6: Авторизация успешна

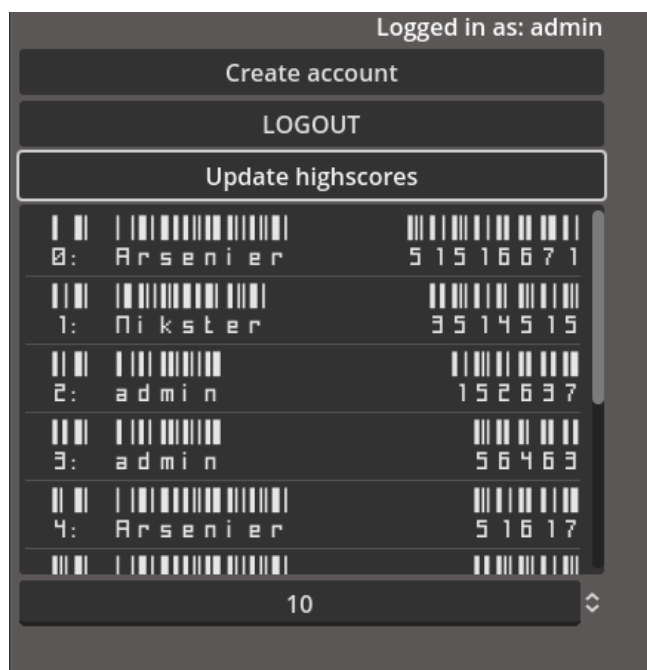


Рис. 7: Обновленная таблица рекордов с 10 лучшими результатами

## 8 Заключение

В ходе выполнения курсовой работы был разработан сервис заметок, состоящий из двух микросервисов, прокси-сервера и фронтэнда. Основные функции сервиса были успешно реализованы, включая возможность создания и нового пользователя, а также публикации новых записей в список рекордов. Реализовано автоматизированное развертывание сервиса, что позволило упростить процесс развертывания приложения. Кроме того, была проведена интеграция микросервисов и настроена их взаимодействие. Курсовая работа позволила познакомиться с архитектурой микросервисов, принципами их работы и основными этапами разработки и развертывания таких сервисов.