



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

---

Институт информационных технологий (ИИТ)

Кафедра практической и прикладной информатики(ППИ)

## **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Микросервисная архитектура»

### **Практическое задание № 9**

Студент группы *ИКБО-25-20, Валиков Кирилл*  
*Юрьевич*

\_\_\_\_\_  
подпись)

Преподаватель *Лантев И.А.*

\_\_\_\_\_  
подпись)

Отчет представлен \_\_\_\_\_ 202\_\_ г.

# 1 Keycloak

```
1  server:
2    port: 9000
3  spring:
4    application:
5      name: api_gateway
6    cloud:
7      gateway:
8        discovery:
9          locator:
10             enabled: true
11             lowerCaseServiceId: true
12        routes:
13          - id: users
14            uri: lb://USERSERVICE
15            predicates:
16              - Path=/users/**
17          - id: appeals
18            uri: lb://APPEALSERVICE
19            predicates:
20              - Path=/appeals/**
21    security:
22      oauth2:
23        resourceserver:
24          jwt:
25            issuer-uri: http://localhost:8080/realms/spring-microservice-realm
26            jwk-set-uri: http://localhost:8080/realms/spring-microservice-realm/protocol/openid-connect/certs
27  eureka:
28    client:
29      enabled: true
30    service-url:
31      defaultZone: http://localhost:8761/eureka/
32
33  logging:
34    level:
35      org.springframework.security: TRACE
```

Рисунок 1. Изменения в application.yml

```
1  package com.example.apigateway.config;
2
3  import org.springframework.context.annotation.Bean;
4  import org.springframework.context.annotation.Configuration;
5  import org.springframework.security.config.Customizer;
6  import org.springframework.security.config.annotation.web.reactive.EnableWebFluxSecurity;
7  import org.springframework.security.config.web.server.ServerHttpSecurity;
8  import org.springframework.security.web.server.SecurityWebFilterChain;
9
10  no usages
11  @Configuration
12  @EnableWebFluxSecurity
13  public class SecurityConfig {
14
15    no usages
16    @Bean
17    @~ public SecurityWebFilterChain securityWebFilterChain(ServerHttpSecurity serverHttpSecurity) {
18      return serverHttpSecurity.csrf(ServerHttpSecurity.CsrfSpec::disable)
19        .authorizeExchange(exchange -> exchange.pathMatchers(...antPatterns: "/eureka/**") Access
20          .permitAll() AuthorizeExchangeSpec
21          .anyExchange().authenticated()
22        ).oauth2ResourceServer((oauth) -> oauth
23          .jwt(Customizer.withDefaults()))
24        .build();
25    }
```

Рисунок 2. SecurityConfig

```
45 api-gateway:
46   container_name: api-gateway
47   image: windflaw8/api-gateway
48   build:
49     context: ./api-gateway
50     dockerfile: Dockerfile
51   depends_on:
52     - eureka-server
53   ports:
54     - 9000:9000
55   environment:
56     - SPRING_APPLICATION_NAME=api_gateway
57     - SPRING_SECURITY_OAUTH2_RESOURCESERVER_JWT_ISSUER-URI=http://keycloak:8080/realms/spring-microservice-realm
58     - SPRING_SECURITY_OAUTH2_RESOURCESERVER_JWT_JWK-SET-URI=http://keycloak:8080/realms/spring-microservice-realm/protocol/openid-connect/certs
59     - eureka.client.serviceUrl.defaultZone=http://eureka-server:8761/eureka/
60
61 keycloak:
62   container_name: keycloak
63   image: quay.io/keycloak/keycloak:latest
64   hostname: keycloak
65   ports:
66     - 8080:8080
67   environment:
68     - KEYCLOAK_ADMIN=admin
69     - KEYCLOAK_ADMIN_PASSWORD=admin
70   command:
71     - start-dev
72     - --import-realm
73   volumes:
74     - ./keycloak-realm/microservice-auth.json:/opt/keycloak/data/import/microservice-auth.json
```

Рисунок 3. KeyCloak в docker-compose

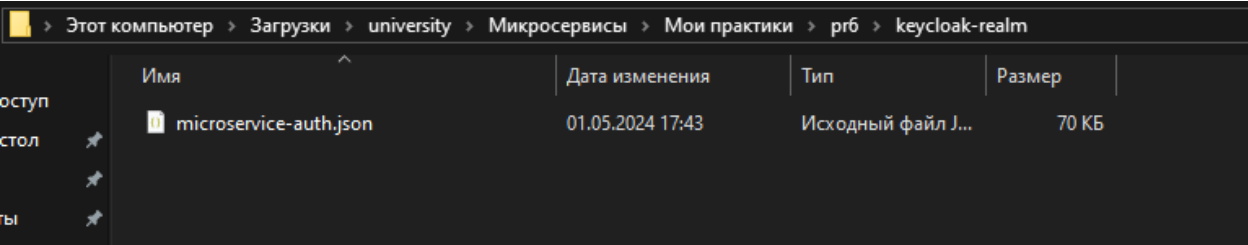


Рисунок 4. Настроенный realm для импорта

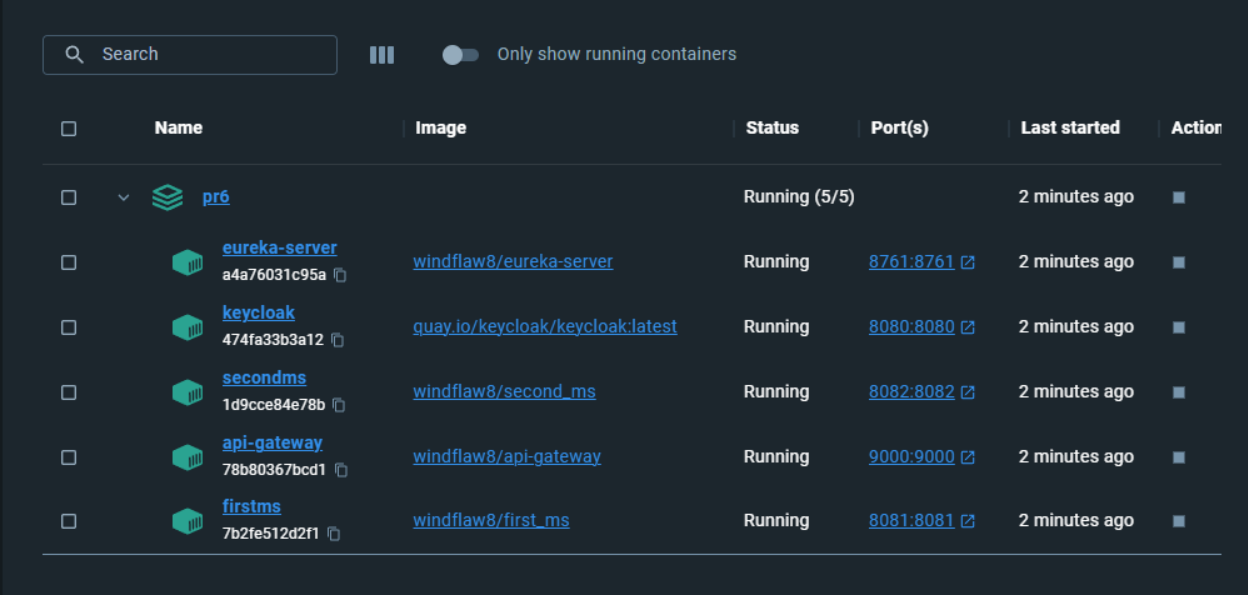


Рисунок 5. Запущенные контейнеры

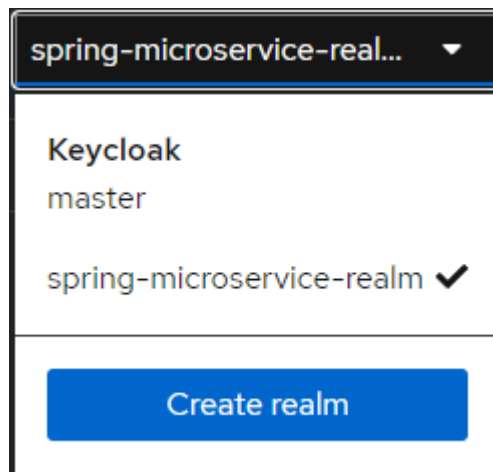


Рисунок 6. Созданный realm

**Clients**  
Clients are applications and services that can request authentication of a user. [Learn more](#)

Clients list Initial access token Client registration

Search for client → Create client Import client Refresh 1 - 7 < >

Client ID	Name	Type	Description	Home URL
account	\${client_account}	OpenID Connect	—	<a href="http://localhost:8080/realms/spring-microservice-realm/account/">http://localhost:8080/realms/spring-microservice-realm/account/</a>
account-console	\${client_account-console}	OpenID Connect	—	<a href="http://localhost:8080/realms/spring-microservice-realm/account/">http://localhost:8080/realms/spring-microservice-realm/account/</a>
admin-cli	\${client_admin-cli}	OpenID Connect	—	—
broker	\${client_broker}	OpenID Connect	—	—
microservice-auth	microservice-auth	OpenID Connect	microservice-auth	<a href="http://localhost:9000/">http://localhost:9000/</a>
realm-management	\${client_realm-management}	OpenID Connect	—	—
security-admin-console	\${client_security-admin-console}	OpenID Connect	—	<a href="http://localhost:8080/admin/spring-microservice-realm/console/">http://localhost:8080/admin/spring-microservice-realm/console/</a>

Рисунок 7. Созданный client

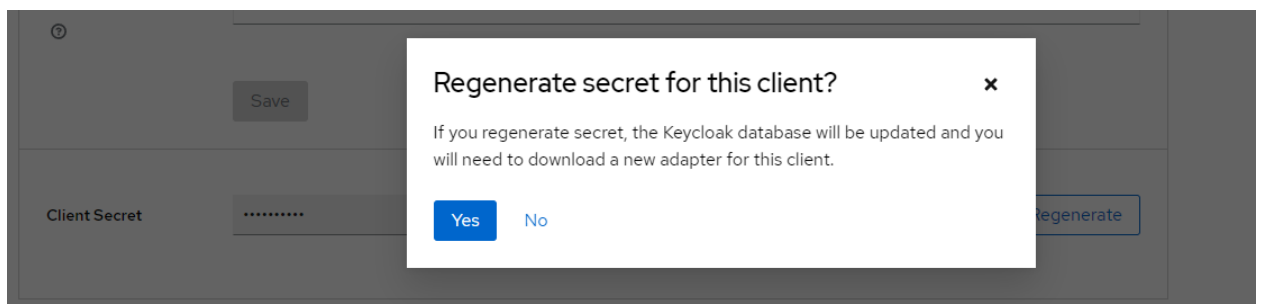


Рисунок 8. Генерируем client secret

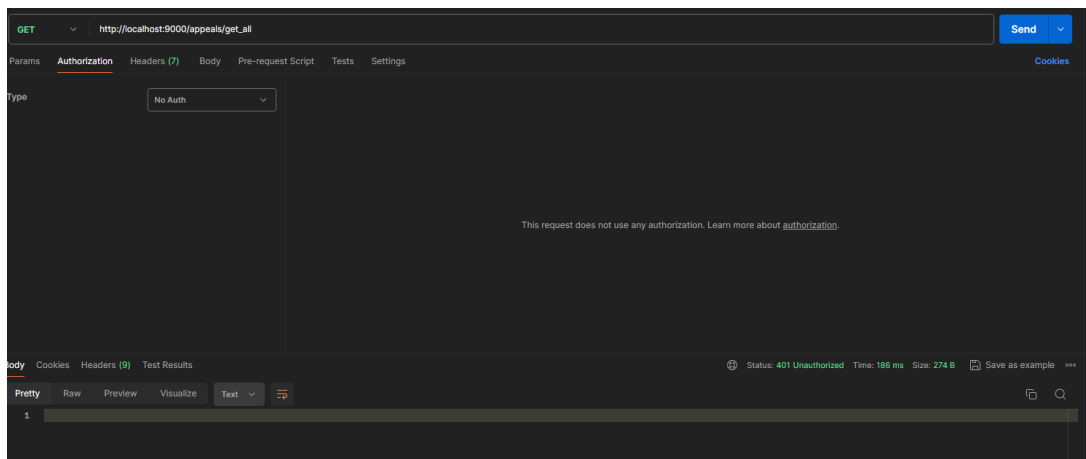


Рисунок 9. Запрос без токена к api-gateway

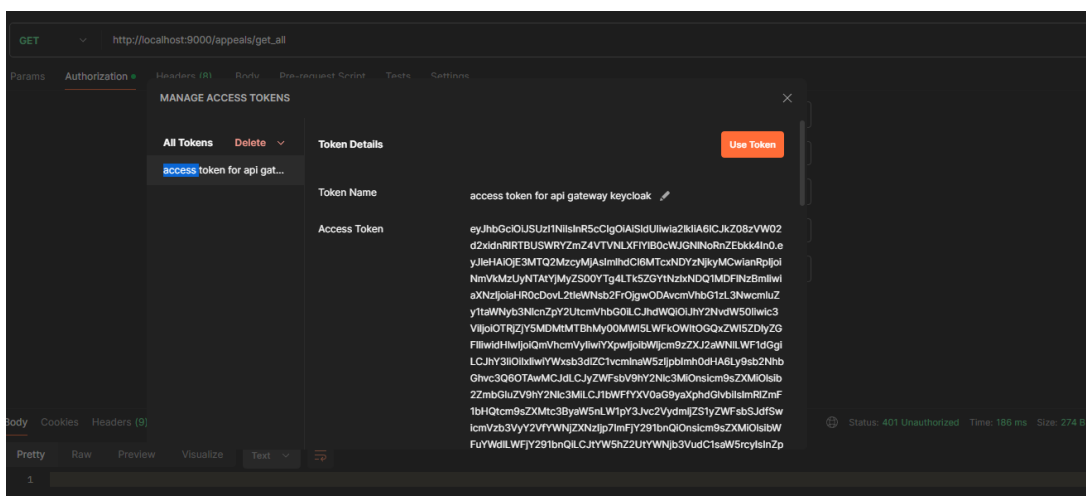


Рисунок 10. Созданный token в postman

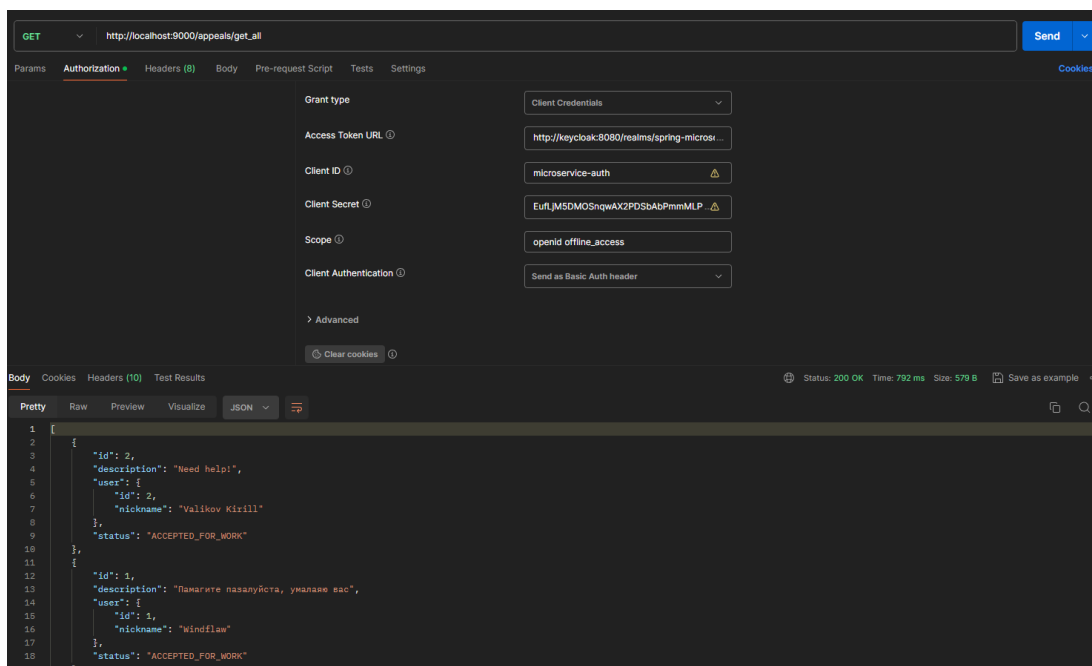


Рисунок 11. Успешный запрос с токеном