

SPRAWOZDANIE

Imię Nazwisko: Klaudia Kromołowska

Temat zajęć laboratoryjnych: Java Micoservices

1. Zwięzły opis aplikacji realizowanej podczas zajęć laboratoryjnych.

W ramach zajęć wykonano prostą aplikację opartą o architekturę mikroserwiów przy wykorzystaniu Spring Boot i języka Java

2. Użyte narzędzia.

(język programowania, najistotniejsze biblioteki, moduły baza danych itp. Nie opisujemy, tylko wypisujemy w postaci listy)

Java, Spring Boot, H2, Spring Eureka, Spring Web, Eureka Discovery Client, Flyway, JDBC API, Spring Security, Spring Reactive Web

3. Sprawozdanie

(Podajemy nazwę endpointu, opis, parametry w zapytaniu oraz opis zwracanych kodów oraz

1. W ramach wstępu do sprawozdania wyjaśnij zalety i wady aplikacji opartych o mikroserwisy w porównaniu do aplikacji monolitycznych.

ZALETY MIKROSERWISÓW:

- możliwości rozwijania różnych usług niezależnie przez różne zespoły
- większa swoboda wyboru technologii i języka programowania
- szybsze wdrażanie
- większe zautomatyzowanie integracji
- minimalizacja ryzyka problemów z całym systemem na skutek błędu

WADY:

- dużo większa złożoność
- trudności w zarządzaniu dużą ilością mikroserwisów
- większa trudność w komunikacji między usługami
- więcej konfiguracji w testach integracyjnych

2. W ramach sprawozdania wyjaśnij znaczenie poszczególnych opcji konfiguracyjnych oraz załącz zrzut ekranu przeglądarki z wyjaśnieniem poszczególnych sekcji widocznych pod adresem <http://localhost:8761/>

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
logging.level.com.netflix.eureka=OFF
logging.level.com.netflix.discovery=OFF
```

server port wskazuje na którym serwerze ma być dostępny ten serwis

register-with-eureka i fetch-registry – umożliwia „rejonizację” rejestrów, żeby dało przechodzić się między regionami w zależności od wymagań, w tym projekcie wyłączone

netflix.eureka i netflix.discovery – wyłączają logi które nie będą potrzebne

localhost:8761

Gmail

Facebook

Netflix

STUDIA

KORKI

Politechnika Krako...

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

Name	Value
total-avail-memory	60mb
num-of-cpus	4
current-memory-usage	39mb (64%)
server-uptime	00:00
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/
available-replicas	

Instance Info

Name	Value
ipAddr	10.71.107.30
status	UP

Kolejne opcje reprezentują: dostępną pamięć, ilość cpu, obecne zużycie pamięci, czas użycia serwera, adresy dostępności serwerów i repliki (brak),

3. W ramach sprawozdania wyjaśnij znaczenie nowych opcji konfiguracyjnych oraz adnotacji `@EnableDiscoveryClient`

`spring.application.name` – nazwa aplikacji

`server.port` – wskazanie portu

`@EnableDiscoveryClient` – automatycznie wykrywa wystąpienia usług z serwera Eureka – dzięki temu nowa usługa pojawi się w rejestrze

4. W ramach sprawozdania omów poszczególne etapy pobierania adresu usługi oraz sposobu budowania zapytań HTTP w oparciu o klasę `WebClient` i obsługę błędów

```
private WebClient getLibraryDbServiceWebClient() {
    InstanceInfo service = client.getApplication("library-db-service").getInstances().get(0);

    return WebClient.builder()
        .baseUrl(String.format("http://%s:%s", service.getHostName(), service.getPort()))
        .defaultHeader(HttpHeaders.CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE)
        .build();
}
```

Na początek tworzona jest instancja – wstrzykiwany jest klient, za pomocą którego pobieramy informacje z db-service. Następnie `WebClient` jest budowany – podawany jest adres url i headery.

```

@GetMapping("/{id}")
public Mono<String> getBookRentals(@PathVariable Integer id) {
    return getLibraryDbServiceWebClient().get() WebClientRequestHeadersUriSpec<...>
        .uri( uri: "/internal/books/" + id) capture of ?
        .retrieve() WebClientResponseSpec
        .onStatus(HttpStatus::isError, response -> Mono.error(new ResponseStatusException(response.statusCode(), "DB error")))
        .bodyToMono(String.class);
}

```

W przykładowym zapytaniu wykorzystujemy poprzednio omówioną funkcję. Informujemy jaki rodzaj działania chcemy wykonać – w tym przypadku jest to get. Dodajemy informację na jakim zasobie będzie to wykonywane. „Retrieve” rozpoczyna „nasłuchiwanie” zdarzeń. Następnie onStatus przechwytuje i zwraca błędy.

5. W ramach sprawozdania szczegółowo udokumentuj poszczególne elementy wykorzystane do konfiguracji procesu autoryzacji aplikacji

```

builder.inMemoryAuthentication() InMemoryUserDetailsManagerConfigurer<AuthenticationManagerBuilder>
    .withUser( username: "root") UserDetailsManagerConfigurer<...>.UserDetailsBuilder
    .password("{noop}secret")
    .roles("USER");

```

Podawana jest informacja o nazwie użytkownika, hasło i rodzaju uprawnień. Dzięki temu ograniczamy dostęp do usługi.

6. Zaznacz w sprawozdaniu zalety oraz typowe przypadki użycia dla programowania reaktywnego.

Programowanie reaktywne polega na asynchronicznym przetwarzaniu danych. Przykładem tego jest retrieve(), które działa jako subscriber – odpowiada za nasłuch i odczyt danych np. o błędach.

Zalety:

- efektywniejsze zarządzanie pulą wątków
- optymalizacja czasu obsługi żądań
- po przyjęciu żądania wątek deleguje je dalej nasłuchując kolejnych – nie jest blokowany

7. Testy:

GET	GET all books	http://localhost:58983/books	/ GET all books	200 OK	72 ms	712 B
	Pass	Status code is 200				
PATCH	PATCH return book	http://localhost:58983/books/return/3	/ PATCH return book	401 Unauthorized	50 ms	280 B
	Pass	Status code is 200				
PATCH	PATCH return book - missing book	http://localhost:58983/books/return/111	/ PATCH return book - missing book	400 Bad Request	31 ms	251 B
	Pass	Status code is 400				
PATCH	PATCH return book - missing user	http://localhost:58983/books/return/1	/ PATCH return book - missing user	401 Unauthorized	51 ms	280 B
	Pass	Status code is 401				
PATCH	PATCH rent book	http://localhost:58983/books/rent/4	/ PATCH rent book	200 OK	36 ms	461 B
	Pass	Status code is 200				
PATCH	PATCH rent unavailable book	http://localhost:58983/books/rent/1	/ PATCH rent unavailable book	409 Conflict	38 ms	270 B
	Pass	Status code is 409				
PATCH	PATCH rent book - no user	http://localhost:58983/books/rent/1	/ PATCH rent book - no user	401 Unauthorized	14 ms	278 B
	Pass	Status code is 401				
GET	GET book rentals	http://localhost:58983/books/3	/ GET book rentals	200 OK	27 ms	458 B
	Pass	Status code is 200				
GET	GET available books	http://localhost:58983/books?available=true	/ GET available books	200 OK	43 ms	712 B
	Pass	Status code is 200				

GET GET users http://localhost:54670/users / GET users

Pass Status code is 200

GET GET users with id http://localhost:54670/users/10 / GET users with id

Pass Status code is 200

Pass ID present

GET GET users with broken id http://localhost:54670/users/aaa / GET users with broken id

Pass Status code is 404

GET GET users with missing id http://localhost:57541/users/11 / GET users with missing id

Pass Status code is 404

POST

http://localhost:58006/users

ParamsAuthorization ●Headers (10)Body ●Pre-request ScriptTests ●Settings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON

1

{

2

"username": "Alicja Kowalczyk",

3

"roles": [

4

"user"

5

]

6

}

BodyCookies (1)Headers (11)Test Results (1/2)

AllPassedSkippedFailed

PASSStatus code is 200

FAILID present | AssertionError: expected 12 to be above 39

Drugi test nie przechodzi, ponieważ z powodu usuniętej dużej ilości użytkowników dodaję brakujące indeksy (w tym przypadku user nr 12)

DELETE

http://localhost:54670/users/7

ParamsAuthorization ●Headers (8)BodyPre-request ScriptTests ●Settings



Query Params


KEY	VALUE
Key	Value

BodyCookies (1)Headers (11)Test Results (1/1)

AllPassedSkippedFailed

PASSStatus code is 200

Users / remove user with broken id  

DELETE  http://localhost:58006/users/aaa

Params Authorization Headers (7) Body Pre-request Script Tests  Setting

Query Params

	KEY	VALUE
	Key	Value

Body Cookies (1) Headers (10) Test Results (1/1)

All Passed Skipped Failed

PASS Status code is 400

Users / remove user with missing id

DELETE  http://localhost:58006/users/11

Params Authorization Headers (7) Body Pre-request Script Tests  Settings

Query Params

	KEY	VALUE
	Key	Value

Body Cookies (1) Headers (10) Test Results (1/1)

All Passed Skipped Failed

PASS Status code is 400

DELETE  http://localhost:58006/users/10

Params Authorization Headers (7) Body Pre-request Script Tests ● Settings

Query Params

	KEY	VALUE
	Key	Value

body Cookies (1) Headers (11) Test Results (1/1)

All Passed Skipped Failed

PASS Status code is 403