Технології реалізації застосунку з використанням стеку фреймворків Spring на базі мікросервісної архітектури

Виконав студент: Вавдійчик В. О.

Науковий керівник: Борозенний С. О.

- **Актуальність теми:**

- Мікросервісна архітектура призначена для розробки високонавантажених розподілених систем, актуальність яких зараз зростає.

- Дослідження стеку фреймворків Spring, включаючи Spring Boot та Spring Cloud, який надає інструменти для побудови мікросервісних додатків на Java.

- Використання новітніх технологій та підходів для реалізації мікросервісного застосунку з використанням інструментів Spring Cloud

Що таке Spring?

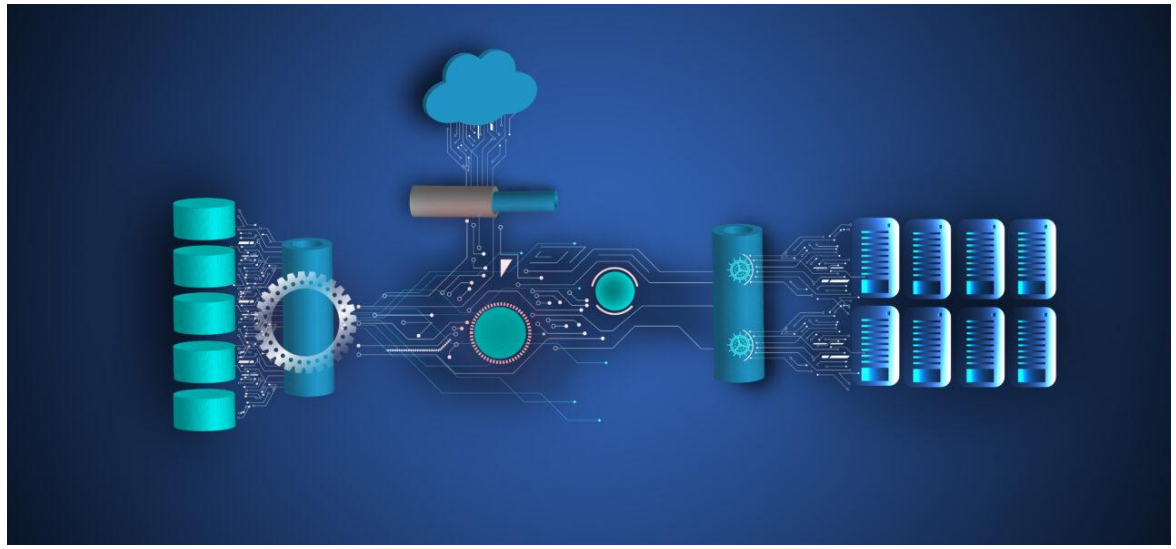
Основні компоненти Spring

- Spring Core
- Spring Data
- Spring Web
- Spring Security
- Spring Boot

Що таке Spring Cloud?

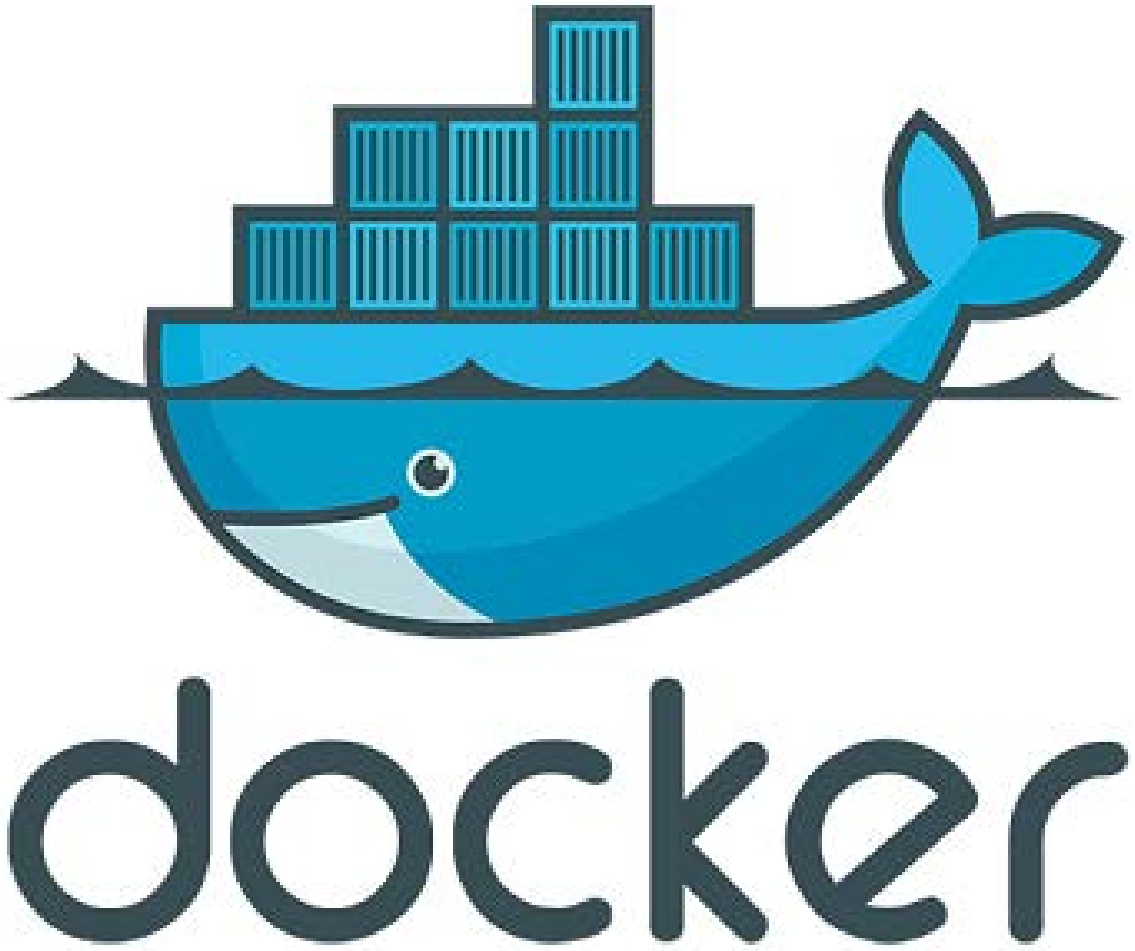
- Spring Cloud Service Discovery.
- Spring Cloud Gateway
- Spring Cloud LoadBalancer

Мікросервісна архітектура



- Масштабованість.
- Незалежність.
- Гнучкість.
- Легкість розгортання
- Легка заміна.

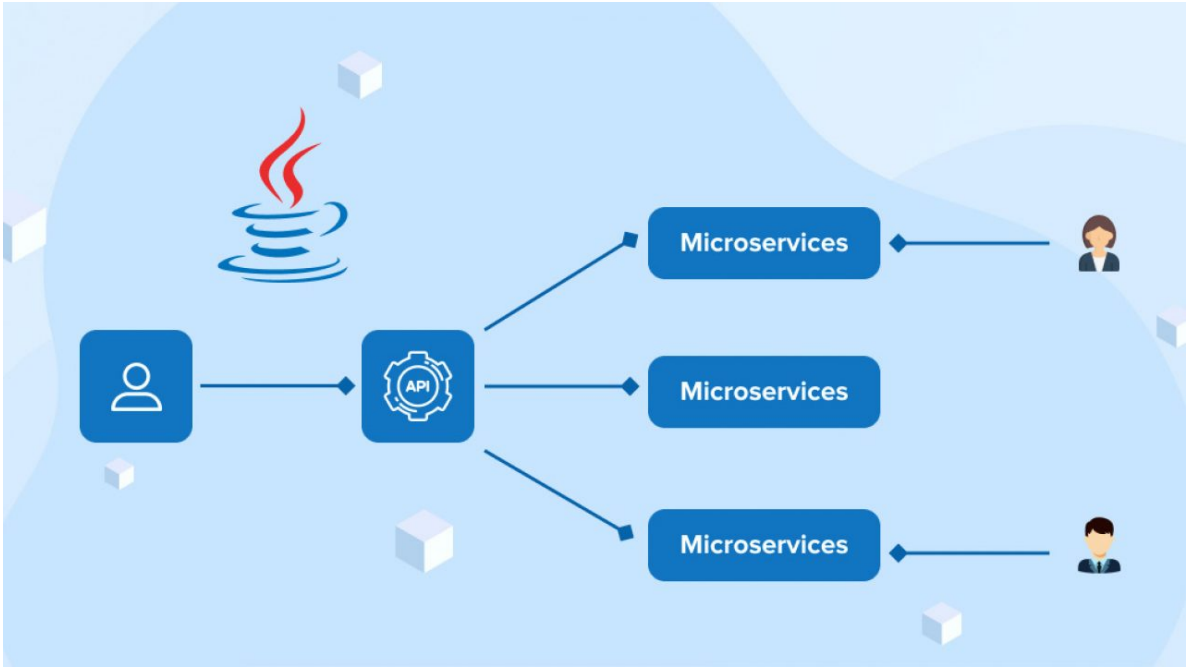
Що таке Docker?



Scheduler API

Сервіси :

- admin-panel
- register-service
- login-service
- schedule_service



spring

WebClient vs RestTempla

EUREKA

SERVICE DISCOVERY

SERVICE REGISTRY

spring Eureka

HOME LAST 1000 SINCE STARTUP

System Status

Environment	test	Current time	2024-05-13T16:08:05 +0300
Data center	default	Uptime	00:54
		Lease expiration enabled	false
		Renews threshold	6
		Renews (last min)	6

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

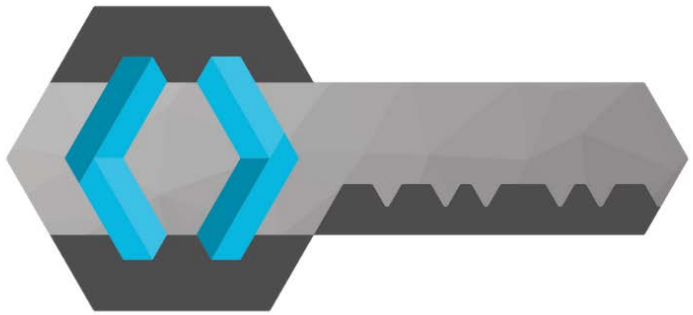
Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
LOGIN_SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-23MTV4Q:login_service:8081
REGISTER_SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-23MTV4Q:register_service:8082
SCHEDULE_SERVICE	n/a (1)	(1)	UP (1) - DESKTOP-23MTV4Q:schedule_service:8181

Spring Cloud Gateway

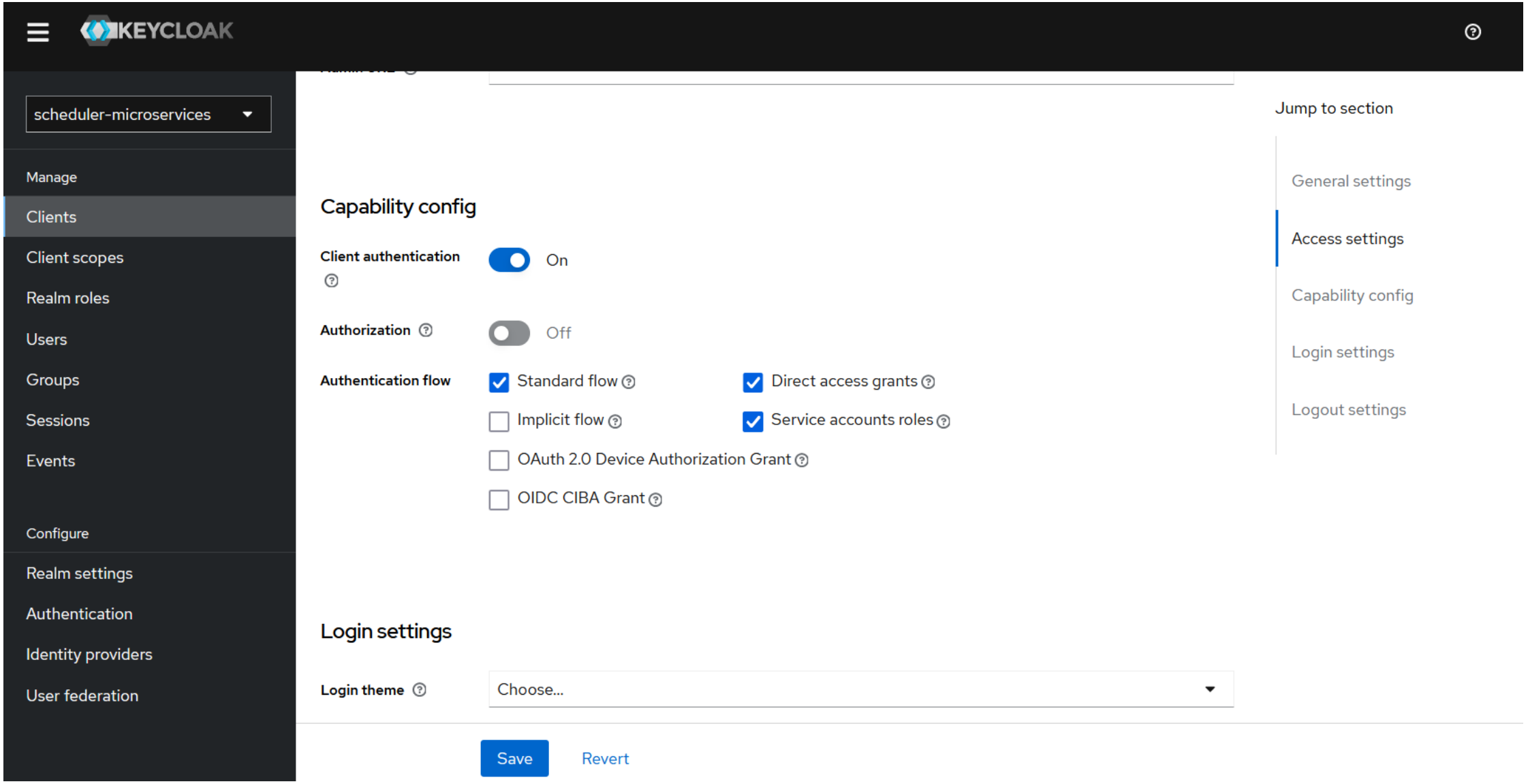
```
        - Path=/api/register
    - id: login-service
      uri: lb://login-service/
      predicates:
        - Path=/api/login
    - id: schedule-service
      uri: lb://schedule-service/
      predicates:
        - Path=/schedule/**
```

spring
SECURITY

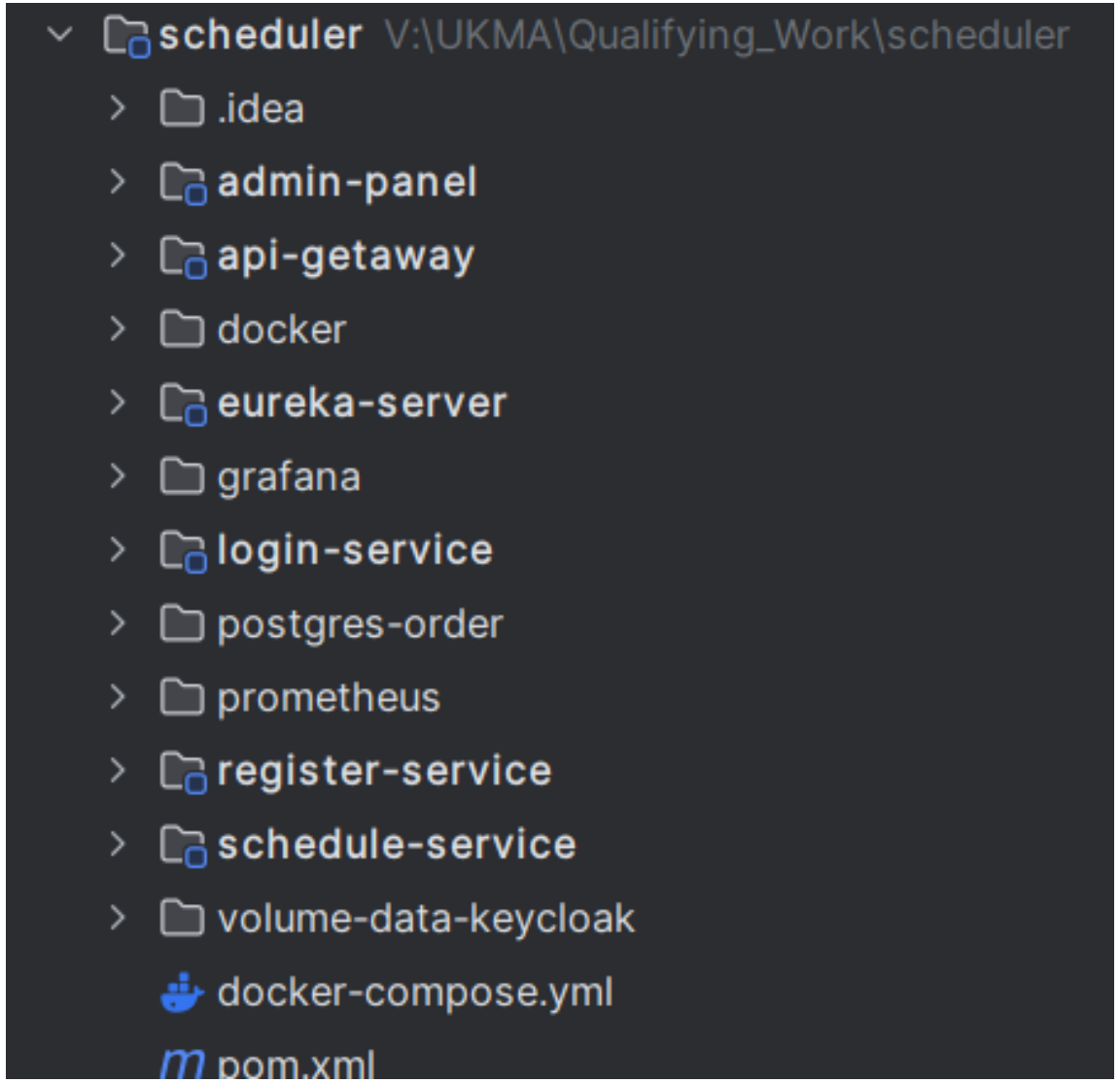


KEYCLOAK

Забезпечення безпеки мікросервісів використовуючи Spring Security та Keycloak


KEYCLOAK
scheduler-microservices
Manage
Clients
Client scopes
Realm roles
Users
Groups
Sessions
Events
Configure
Realm settings
Authentication
Identity providers
User federation
Jump to section
General settings
Access settings
Capability config
Login settings
Logout settings
Capability config
Client authentication
On
Authorization
Off
Authentication flow
Standard flow
Direct access grants
Implicit flow
Service accounts roles
OAuth 2.0 Device Authorization Grant
OIDC CIBA Grant
Login settings
Login theme
Choose...
Save
Revert

- Структура готового проєктку



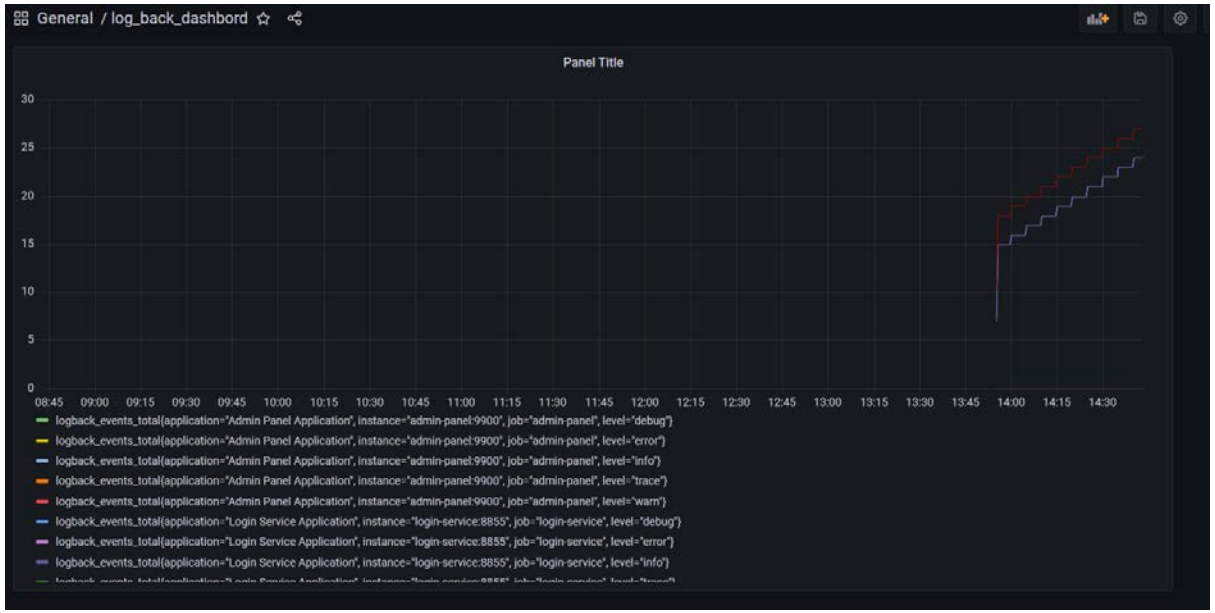

```yaml
postgres-db:
  container_name: postgres-db
  image: postgres
  environment:
    POSTGRES_DB: schedulerDB
    POSTGRES_USER: root
    POSTGRES_PASSWORD: 1111
    PGDATA: /data/postgres
  volumes:
    - ./postgres-order:/data/postgres
  ports:
    - "5432:5432"
  restart: always
```

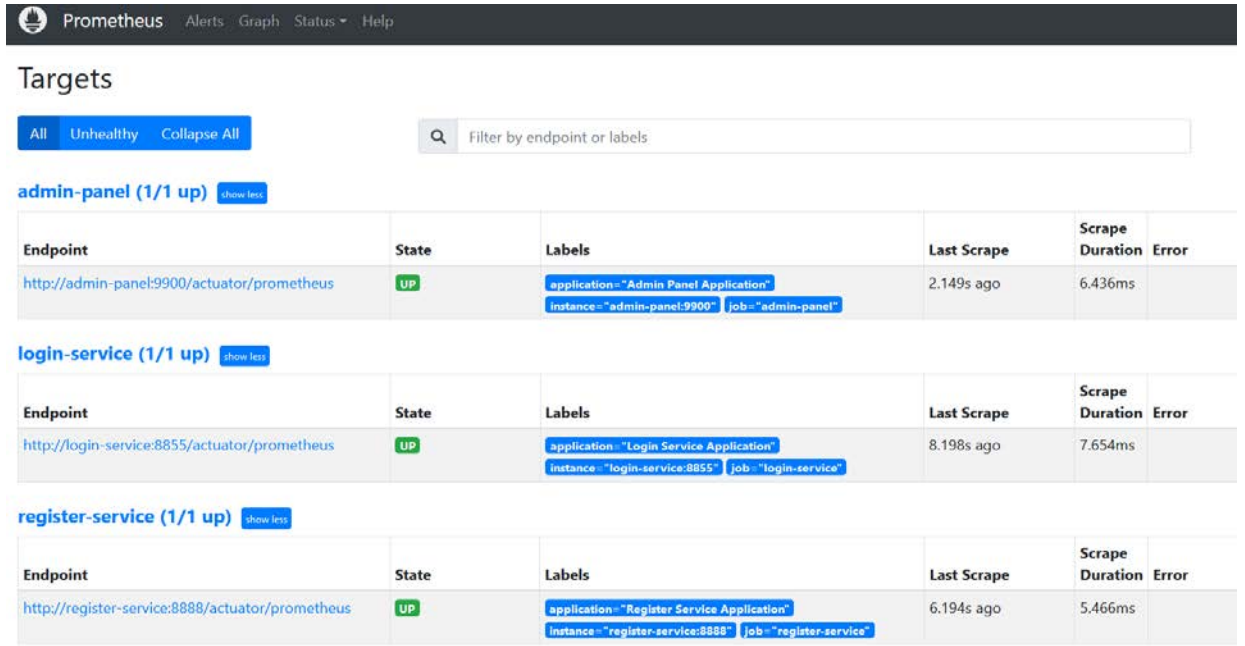
Контейнеризація застосунку

Моніторинг роботи мікросервісів

Grafana

Prometheus





Висновки

- Було розглянуто основні модулі Spring
- Було розглянуто опис задач та принципів мікросервісної архітектури, а також її недоліки
- Практична частина роботи демонструє створення мікросервісного додатку Scheduler API. Використовуються інструменти Spring Cloud.
- Було показано процес контейнеризації застосунку та моніторингу роботи його контейнерів

Література та список джерел:

-
  - Spring in Action, 6th edition. Craig Walls
  - Spring Microservices in Action. Second Edition . John Carnell
  - Using Docker: Developing and Deploying Software with Containers 1st Edition. Adrian Mouat
  - Docker in Action 2nd ED. Jeff Nickoloff
-
  - https://docs.spring.io/spring-boot/docs/current/reference/html/
  - https://www.baeldung.com/jib-dockerizing
  - https://docs.spring.io/spring-cloud-gateway/reference/index.html
  - https://docs.spring.io/spring-cloud-config/reference/
  - https://docs.spring.io/spring-cloud-netflix/docs/current/reference/html/#netflix-eureka-client-starter
  - https://docs.spring.io/spring-cloud-security/docs/current/reference/html/#_oauth2_protected_resource
  - https://cloud.google.com/java/getting-started/jib
  - https://www.keycloak.org/getting-started/getting-started-docker
  - https://docs.docker.com/compose/gettingstarted/

Дякую за увагу!

Джерела зображень взятих для презентації

- https://www.azilen.com/blog/everything-must-know-spring-boot-application-scratch/
- https://blog.stackademic.com/the-complete-guide-to-spring-data-jpa-building-a-bookstore-application-from-scratch-part-i-3f8ba9d13b10?gi=e9c8c1dcd53e
- https://www.javaguides.net/p/spring-framework-tutorial.html
- https://velog.io/@gundorit/Spring-Security-6.1.0-%EC%97%85%EB%8D%B0%EC%9D%B4%ED%8A%B8-%EA%B0%80%EC%9D%B4%EB%93%9C
- https://www.tatvasoft.com/blog/microservices-implementation-java/
- https://medium.com/@kmrpushkar09/spring-webclient-vs-resttemplate-whats-better-in-2023-99844f649b53
- https://medium.com/@malindudilshan389/service-discovery-with-spring-cloud-netflix-eureka-d0b8ece1446d
- https://pretius.com/blog/keycloak-sso/
- https://tcude.net/updating-container-with-docker-compose/
- https://medium.com/techieahead/why-we-need-prometheus-8cbf59ec6516