Poročilo projekta:

**Quick Wheels**

**Računalniške storitve v oblaku**

**SKUPINA 47**

**Nikolay Vasilev**

**63190338**

Januar, 2023

1. **Opis projekta**

Aplikacija **Quick Wheels** omogoča skrbnikom vseh vrst dostavnih podjetij izračunati optimalno dostavno pot za nabor naročil v določenem času na podlagi različnih kriterijev - vrste naročila, prometa, lokacije naročnika, lokacije ponudnika ter prevoznega sredstva dostavljavca.

Da bi lahko vse to omogočala, aplikacija uporablja dve mikrostoritvi, ki smo jih sami razvili - **mikrostoritev za naročila** in **mikrostoritev za dostave**, od katerih ima vsaka svoj REST API in podatkovno bazo. Ti dve mikrostoritvi sodelujeta med seboj ter z zunanjo mikrostoritvijo za zemljevide **Amazon Location Service**, da bi lahko uporabniki izkoristili glavne primere uporabe aplikacije.

1. **Uporabljena tehnologija**

Aplikacijo in mikrostoritve smo razvili v razvojnem okolju **IntelliJ IDEA**. Za razvoj aplikacije (vmesnika) smo uporabili **Java Swing** - komplet orodij za grafični uporabniški vmesnik (GUI), ki ponuja bogat nabor pripomočkov in paketov za izdelavo kompleksnih komponent vmesnika za Java aplikacije. Za razvoj obeh mikrostoritev pa smo uporabili mikrostoritveno ogrodje **KumuluzEE** v **Javi**, za katerega smo potrebovali tudi **Apache Maven** za izdelavo našega projekta. Vsaka mikrostoritev ima svoj **REST API** in uporablja lastno bazo podatkov **SQL**. Prav tako smo vzpostavili cikel zvezne integracije CI/CD z uporabo **Github Actions** za ustvarjanje novo Docker sliko vsake mikrostoritve in objavo na portalu **Docker Hub** (v oblaku). V razvojnem procesu smo uporabljali **Postman**, da smo zagotovili, da zahteve in odgovori mikrostoritev delujejo tako, kot bi morali. Ustrezno smo konfigurirali **Kubernetes storitve**, kjer smo na javnem IP naslovu objavili naše mikrostoritve. Naše mikrostoritve podpirajo tudi:

* **KumuluzEE Config**, ki nam omogoča implementirati upravljanje s konfiguracijo
* **KumuluzEE Health**, ki nam omogoča implementirati kontrola zdravja ter preverjati vitalnosti
* **KumuluzEE Metrics** za zbiranje metrik Kubernetes okolja
* **Log4j2** za implementacijo centralnega beleženja dnevnikov

1. **Seznam povezav do GitHub repozitorijev**

Mikrostoritve so obljavljene na GitHub repozitorijih v organizaciji **rso-2022-2023**:

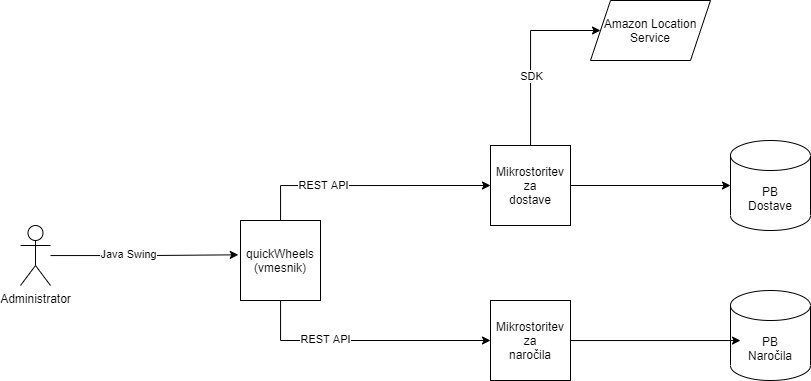
* **RSO\_S47** – Na tem GitHub repozitoriju, ki se nahaja na povezavi <https://github.com/rso-2022-2023/RSO_S47>, je dokumentacija, ki opisuje trenutno verzijo projekta.
* **RSO\_S47\_Order** – Na tem GitHub repozitoriju, ki se nahaja na povezavi <https://github.com/rso-2022-2023/RSO_S47_Order>, je mikrostoritev za naročila.
* **RSO\_S47\_Delivery** – Na tem GitHub repozitoriju, ki se nahaja na povezavi <https://github.com/rso-2022-2023/RSO_S47_Delivery>, je mikrostoritev za dostave.

1. **Seznam povezav do DockerHub repozitorijev**

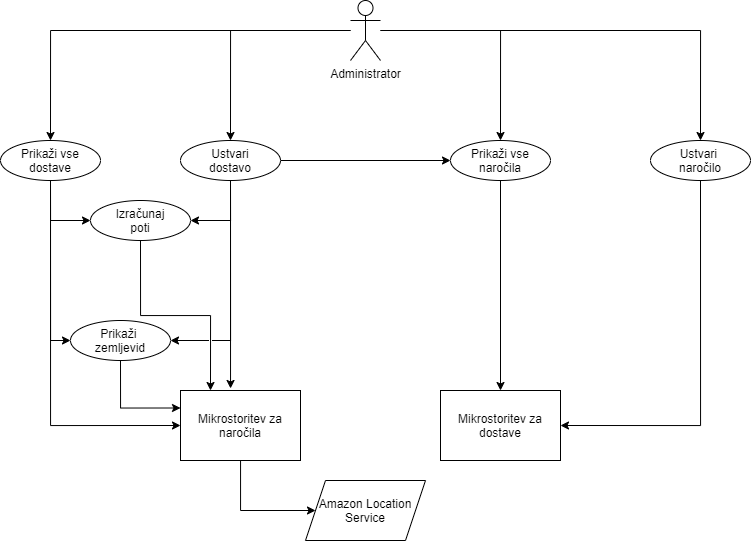
Vsaka mikrostoritev uporablja zvezno integracijo s pomočjo GitHub Actions, ki vso kodo iz GitHub veje main prevede in zapakira, ustvari novo sliko Docker z ustreznim tag-om in jo objavi na portalu DockerHub. Zato ima vsaka mikrostoritev svoj repozitorij na DockerHub-u:

* **repozitorij mikrostoritve za naročila**, ki se nahaja na povezavi <https://hub.docker.com/repository/docker/feniksa18/orders> in
* **repozitorij mikrostoritve za dostave**, ki se nahaja na povezavi <https://hub.docker.com/repository/docker/feniksa18/deliveries>.

1. **Sheme**
   1. **Shema arhitekture**

****

* 1. **Shema interakcij**

****

1. **Glavne funkcionalnosti**
   1. **Mikrostoritev za naročila**
      * statusih naročil - ustvarjanje novega statusa naročila, prikaz seznam statusov naročil, posodabljanje obstoječega statusa naročila;
      * izdelkov - ustvarjanje novega izdelka, prikaz seznam izdelkov, posodabljanje obstoječega izdelka;
      * naročil - ustvarjanje novega naročila, prikaz seznam naročil, posodobitev obstoječega naročila;
   2. **Mikrostoritev za dostave**

* naročnikih - ustvarjanje novega naročnika, prikaz seznam vseh naročnikov, posodabljanje obstoječega naročnika;
* dostave - ustvarjanje novo dostavo, prikaz seznam vseh dostav, posodabljanje obstoječo dostavo;
* naslov - ustvarjanje novega naslova, prikaz seznam vseh naslovov, posodabljanje obstoječega naslov;
* vrst dostave - ustvarjanje nove vrste dostave, prikaz seznam vseh vrst dostave, posodabljanje obstoječo vrsto dostave;
* statusih dostave - ustvarjanje novega statusa dostave, prikaz seznam statusov dostave, posodabljanje obstoječega statusa dostave;
* dostavnih prevozov - ustvarjanje novega dostavnega prevoza, prikaz seznam vseh dostavnih prevozov, posodabljanje obsteječega dostavnega prevoza;

1. **Seznam primerov uporabe**

V aplikaciji imamo le eno vlogo - administrator, ki upravlja z naročili in njihovo dostavo. Primeri uporabe aplikacije lahko v grobem razdelimo na funkcionalnosti o:

* Izdelava novega naročila, ki vsebuje določeno količino izbranega izdelka iz baze.
* Ogled seznama obstoječih naročil in podrobnosti o njih.
* Urejanje obstoječih naročil.
* Kreiranje nove dostave za eno ali več naročil za določeno stranko iz baze.
* Ogled seznama obstoječih dostav in podrobnosti o njih.
* Urejanje obstoječih dostav.
* Prikaz zemljevida.
* Prikaz poti od prodajalca do kupca na zemljevidu.
* Izračunan čas dostave glede na vrste prevoza in dostave.

1. **Opis namestitve na Kubernetes**

Obe mikrostoritvi sta na javnem IP naslovu v storitvi **Amazon Elastic Kubernetes**. Ker integracija ingress ni bila uspešna, je vsaka mikrostoritev na svojem naslovu IP (url).

1. **Opis različnih virov konfiguracije**

V obeh mikrostoritvah smo implementirali rest resources, kot sta maintainance mode in broken, katere vrednosti lahko s pomočjo **Consula** spreminjamo med delovanjem storitev.

1. **Seznam kontrol zdravja in metrik**

Obe mikrostoritvi podpirata kontrol zdravja, ki preverja delovanja in pripravljenosti, vendar samo mikrostoritev za dostave podpira metrik. Njihove sonde lahko najdemo na končnih točkah **/health/live** in **/health/ready**. Meritve iz mikrostoritve za dostave se zbirajo glede na čas, ki je potreben, da se izbrana dostave vzame iz podatkovne baze.

1. **Opis dokumentiranja storitev z OpenAPI**

Obe mikrostoritvi sta dokumentirani z uporabo specifikacij OpenAPI. Orodje **Kumuluzee OpenApi** nam omogoča pisanje dokumentacije s pomočjo Java funkcij. Do uporabniškega vmesnika Swagger lahko dostopamo na končni točki **/api-specs/ui**, do specifikacije uporabniškega vmesnika Swagger pa na **/openapi**.

1. **Opis uporabe zunanjih API-jev**

Kot smo že omenili, smo zunanji API, ki smo uporabili, implementirali s pomočo **Amazon Location Service**. Implementiran je v mikrostoritev za dostave in se uporablja za prikaz zemljevida in izračun poti dostave glede na vrsto transporta in dostave.

1. **Opis centraliziranega beleženja**

Vsaka od našjih mikrostoritev uporablja orodje **KumuluzEE Logs** za beleženje dnevnikov, ki dnevnike pošilja za centralizirano beleženje dnevnikov v **Logit.io**.

1. **Opis izolacijo in toleranco napak**

Implementirali smo toleranca napak samo pri mikrostoritvi za dostave, oziroma pri pridobivanju zemljevida iz **Amazon Location Service**. Uporabljamo asinhroni način klicanja funkcije in ji dodamo časovno omejitev. Na ta način funkcija deluje v drugi niti in če traja več časa, kot je potrebno, ali če pride do druge napake, obstaja funkcija povratnega klica, ki nam vrne drugo vrednost in ne prekinja mikrostoritve.