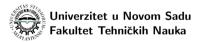
# Alati za razvoj softvera

#### Projektni zadatak



#### Uvod

- ▶ Broj članova u timu 3
- I pored timskog rada, bodovi su individualni
- Pravilno rasporediti posao
- Raditi domaće zadatke na vežbama mogućnost završetka predmeta pre roka
- Druga opcija je da branite projekat u definisanom terminu odbrane

## Projektni zadatak

- Vaš zadatak je da implementirate sistem za centralizovanu konfiguraciju servisa
- Sistem se sastoji od dve glavne komponente:
  - 1. Web servis koji prihvata korisničke zahteve i vrši obradu
  - 2. Baza podataka koja čuva statnje sistema
- ▶ I dve pomoćne komponente koje održavaju sistem:
  - Komponente za čuvanje i pregled logova i trace-a
  - Komponente za čuvanje i pregled metrika

## Komponenta — Web service

- ▶ Web service treba da bude implementiran koristeći programski jezik Go (golang)
- Servis treba da omogući sledeće operacije:
  - Dodavanje konfiguracije u sistem, konfiguracija se prihvata kao JSON podatak
  - Dodavanje konfiguracione grupe, gde grupa može da ima 1 ili više konfiguracija, konfiguraciona grupa se prihvata kao JSON podatak
  - Pregled konfiguracije, konfiguacija se vraća po identifikatoru
  - Pregled konfiguracione grupe, grupa se vraća po identifikatoru
  - ▶ Brisanje konfiguracije, konfiguracija se briše identifikatoru
  - Brisanje konfiguracione grupa, grupa se briše po identifilatoru
  - ► Proširenje konfiguracione grupe, dodavanje nove konfiguracije unutar konfiguracione grupe
  - Naprednije operacije nad konfiguracionom grupom koristeći sistem labela

#### Sistem labela

- Svaka konfiguracija unutar konfiguracione grupe treba da ima skup labela koje će biti korišćene za filter(pretragu)
- ▶ Više konfiguracija unutar grupe mogu da imaju isti set labela
- Labele su teksutalni parovi u formatu *ključ:vrednost* razdvojeni ;  $(l_1 : v_1; l_2 : v_2, ...)$
- ► Kada korisnik želi da vrati kofiguracije unutar konfiguracione grupe koristeći labele, sve labele iz upita moraju se poklapati sa onima koje su pridružene konfiguraciji
- Podržati brisanje koristeći sistem labela, ista pravila važe kao i za pretragu

- Omogućiti imutabilnost, tj. nema delimičnog menjanja konfiguracije konfiguracija se može zameniti samo u potpunosti
- ► Podržati da su zahtevi idempotentni
- Kao jedinstvene identifikatore koristiti UUID
- Omogućiti verzioniranje, tako da konfiguracije možemo čuvati u različitim verzijama
- Kada klijent traži konfiguraciju, mora navesti i verziju konfiguracije koju želi da dobije nazad
- Kada klijent traži konfiguracionu grupu, mora navesti i verziju grupe koju želi da dobije nazad

## Dokumentovanje endpoint-a

- Omogućiti dokumentovanje implementiranog API-ja
- Dokumentovati svaki endpoint koristeći Swagger

## Baza podataka

- Konfiguacije čuvati u NoSQL bazi Consul
- Konfiguacione grupe čuvati u NoSQL bazi Consul
- ▶ Čuvati informacije o idempotentnosti zahteva u NoSQL bazi Consul

#### Ostali zahtevi

- Servis treba da bude kontejnerzovan koristeći Docker mutli stage build
- Baza podataka treba da bude kontejnerzovan koristeći Docker
- Podržati traceing u vašem servisu
- Podržati skupljanje metrika u vašem servisu (dovoljne su metrike radjene na vežbama)
- Sve elemente pokrenuti unutar docker compose-a
- ► Trigerovati CI sistem (Git Actions) kada spojite izmene na master (main) granu
- Kao sistem za kontrolu verzija koristiti Git i pridržavati se GitFlow principa
- Testirati servis koristeći Postman ili cURL

## Projektni zadatak

- Projektni zadatak vam je zadatak sa nekog drugog predmeta gde pravite servisno-orijentisanu aplikaciju (backend)
- Koristite programski jezik koji se radi na tom predmetu
- Koristite bazu podataka sa kojom radite na tom predmetu
- Zahtevi treba da budu idempotentni
- Imutabilnost primeniti shodno zahtevima predmetnog projekta (ako podržvate delimičnu izmenu podataka, imutabilnost nema smisla )

#### Ostali zahtevi

- Servis treba da bude kontejnerzovan koristeći Docker
- ▶ Baza podataka treba da bude kontejnerzovan koristeći Docker
- Podržati traceing u vašem servisu
- Podržati metrike u vašem servisu (dovoljne su metrike radjene na vežbama)
- Sve elemente pokrenuti unutar docker compose-a
- ► Trigerovati CI sistem (Git Actions) kada spojite izmene na master (main) granu
- Kao sistem za kontrolu verzija koristiti Git i pridržavati se GitFlow principa
- Testirati servis koristeći mehanizme sa predmeta sa kojim spajate projekat
- Dokumentovati svaki endpoint koristeći Swagger
- ► NAPOMENA: Moraćete sami malo da uložite napor da omogućet traceing u odgovarajućem jeziku i da podržite skupljanje metrika