1. Laboratorijska vježba iz "Web aplikacija u Javi"

1.1. Jednostavna web aplikacija

Svrha laboratorijske vježbe je upoznavanje s razvojnim okruženjem IntelliJ IDEA Ultimate, autokonfiguracijom Spring Boot frameworka te osnovnom konfiguracijom Spring Web aplikacije koja se sastoji od jednog REST *controllera*, servisa i repozitorija. Kroz ovaj semestar radit ćemo na online sustavu za naručivanje hrane. Sustav će biti u mogućnosti prezentirati ponudu, kreirati narudžbe, prikazivati status narudžbe, omogućavati drugim davateljima usluga integraciju sa našim sustavom, višejezičnost…

1.2. Zadatak

Potrebno je kreirati Java web aplikaciju koja će omogućavati dohvat podataka o osnovnim podacima našeg sustava za online naručivanje hrane.

1. Unutar razvojnog okruženja IntelliJ IDEA Ultimate potrebno je kreirati Spring Boot projekt baziran na Maven strukturi. Odabrati „New Project", te u kategoriji Generators s lijeve strane odabrati Spring Boot. Na desnom dijelu ekranu odabrati Java verziju 24. Pod "Group" unijeti "**hr.tvz.prezime**" gdje prezime treba zamijeniti s vlastitim prezimenom. Pod "artifact" unijeti „njamApp". Pritisnuti "Next". Na sljedećem ekranu potrebno je odabrati samo "Spring Web" *dependency* te pritisnuti „Create". Nakon što se projekt otvori u novom ekranu, u donjem desnom kutu potrebno je dozvoliti Microsoft Defender konfiguraciju i/ili odabrati "Enable Auto-Import".

   Napomena : Nazive klasa / filedlova pisati ili na engleskom ili na hrvatskom, ali očekuje se konzistentnost. Kroz pripremu će se koristiti hrvatski jezik.

2. Napisati klasu Restoran sa fieldovima : šifra restorana, ime restorana, adresa, broj telefona, email, radno vrijeme (implementacija na izbor, mapa, enumeracija,..), trenutno otvoren (boolean), prosječno vrijeme dostave, prosječna ocjena kupaca, maksimalan broj narudžbi.

3. Napisati klasu RestoranDTO. Ova klasa predstavlja kratki prikaz informacija o restoranu na početnoj stranici naše aplikacije. Klasa RestoranDTO treba imati fieldove šifra restorana, ime restorana, adresa, trenutno otvoren i postotak koji predstavlja opterećenost restorana a koji se računa kao broj aktivnih narudžbi dijeljeno sa maksimalan broj narudžbi. (broj narudžbi će biti dostupan kasnije, trenutno ovo može biti random broj u intervalu [10,100])

4. Napisati repozitorij klasu koja implementira sljedeći *interface*:

```
1. interface RestoranRepository {
2.
3.        List<Restoran> findAll();
4.        Optional<Restoran> findRestoranByID(Long id);
5.
6. }
```

   Klasa treba funkcionirati kao odgovarajući Spring *bean* te se unutar nje treba kreirati lista s barem dva Restoran objekta iz koje će se vraćati povratne vrijednosti.

5. Napisati servisnu klasu koja implementira sljedeći *interface*:

```
1. public interface RestoranService {
2.
3.        List<RestoranDTO> findAll();
4.        RestoranDTO findRestoranByID(Longi id);
5.        List<RestoranDTO> findNajblizi(String adresa);
6.        List<RestoranDTO> findNajbolji(Double ocjena);
7.
8. }
```

   Klasa treba funkcionirati kao odgovarajući Spring *bean* te se u nju korištenjem *dependency injection* mehanizma treba ubaciti repozitorij klasa iz 4. koraka.

6. Napisati REST *controller* klasu s dvije GET *request handler* metode. Jednu za dohvat svih podataka o restoranima i drugu za dohvat podataka jednog restorana prema šifri. Klasa treba funkcionirati kao odgovarajući Spring *bean* te se u nju korištenjem *dependency injection* mehanizma treba ubaciti servisna klasa iz 5. koraka. Podaci koje REST *controller* vrati web pregledniku trebaju biti u JSON reprezentaciji.

7. Kroz Spring Boot postavku uključiti logiranje *requestova* i *responsova* za REST *controller* u konzoli. (opcionalno Logback ili log4j2)

8. Demonstrirati rad aplikacije kroz *web* preglednik ili Postman aplikaciju ili kao što je prikazano na predavanju.