
Vježba: 4 – Zadaća 1. FOLT - Dostava hrane i pića na bazi više poslužitelja na mrežnoj utičnici

Opća pravila

Nazivi klasa, nazivi atributa, nazivi metoda, nazivi varijabli, komentari i sl. pišu se na hrvatskom jeziku u skladu s preporukama za programski jezik Java. Metode u klasama NE smiju imati više od 35 linija programskog koda, u što se ne broji definiranje metode, njenih argumenata i lokalnih varijabli, prazna linija, linija samo s { ili }. U jednoj liniji može biti jedna instrukcija. U jednoj liniji može biti najviše 100 znakova. Pisanje programskog koda mora biti u skladu s preporukama Google Java Code Style. Ne smiju se koristiti klase ili metode koje su označene kao „deprecated". Klase i metode trebaju biti dokumentirane u Javadoc formatu.

Naziv projekta: **{LDAP_korisnik}_vjezba_04_dz_1**

Korijenski direktorij treba biti **{LDAP_korisnik}__vjezba_04_dz_1**

Sve nove klase trebaju biti u paketu **edu.unizg.foi.nwtis.{LDAP_korisnik}.vjezba_04_dz_1**. Za rad s postavkama treba koristiti Java biblioteku iz {LDAP_korisnik}_vjezba_02_1 verzije 1.1.0 (nastala na temelju {LDAP_korisnik}_vjezba_02_1 verzije 1.0.0). **Projekt se isključivo treba predati u formatu Eclipse IDE projekta s maven upravljanjem.** Prije predavanja projekta potrebno je **napraviti Clean na roditeljskom projektu** (i svim njegovim Maven modulima). Provjerite postoje li target direktoriji! Zatim cijeli roditeljski projekt (i sve njegove Maven module) sažeti u **.zip** (NE .rar) format s nazivom **{LDAP_korisnik}_vjezba_04_dz_1.zip** i predati u Moodle. Uključiti izvorni kod i popunjeni obrazac za zadaću pod nazivom **{LDAP_korisnik}_vjezba_04_dz_1.pdf** (u korijenskom direktoriju projekta). U radu programa datoteka konfiguracijskih podataka i ostale datoteke smještene su na direktoriju s kojeg se pokreće program. Program se može pokretati s različitih direktorija kako bi se mogao izvršavati s različitim datotekama konfiguracijskih podataka i ostalih datoteka. **Za određeni skup klasa (definirano u donjem dijelu) potrebno je napraviti klase za jedinično testiranje primjenom JUnit okvira.** U tim klasama sve metode moraju imati pridruženu public vidljivost ili vidljivost unutar paketa. NE smiju se koristiti druge biblioteke klase osim onih koje se nalazu u opisu zadaće.

Boduju se dijelovi koji su rađeni nakon vježbi!

Struktura .zip datoteke predane zadaće treba biti sljedeća:

```
{LDAP_korisnik}_vjezba_04_dz_1
	{LDAP_korisnik}_vjezba_04_dz_1.pdf

	{LDAP_korisnik}_vjezba_04_dz_1_kupac
		pom.xml
		src
			main
				...
			test
				...
	{LDAP_korisnik}_vjezba_04_dz_1_lib
		pom.xml
		src
			main
				...
			test
				...
	{LDAP_korisnik}_vjezba_04_dz_1_partner
		pom.xml
		src
			main
				...
			test
				...
	{LDAP_korisnik}_vjezba_04_dz_1_podaci
		pom.xml
		src
			main
				...
			test
				...
	{LDAP_korisnik}_vjezba_04_dz_1_tvrtka
		pom.xml
		src
			main
				...
			test
				...
```

Važne informacije:

1. Sintaksa komandi pojedinih poslužitelja NE može se mijenjati
2. Između riječi u komandama nalazi se jedna praznina
3. Svaka komanda završava s \n
4. Odgovori su tipa:
    - OK – znači da je u redu. Kod nekih komandi u nastavku su određeni podaci.
    - ERROR – znači da nije u redu. U nastavku je status i opis problema.
5. Svaki odgovor završava s \n
6. Kod bodovanja pojedini vaši poslužitelji i klijenti izvršavat će se u „društvu" s nastavničkim poslužiteljima i klijentima. Svi oni moraju raditi tj. komunicirati bez problema jer se moraju pridržati definirane sintakse komandi i odgovora. Npr. pokrene se studentski PosluziteljTvrtka, nastavnički PosluziteljPartner i studentski KorisnikKupac. Čak i ako „sve" dobro radi u „društvu" studentskih poslužitelja i klijenata, ako ne radi u „društvu" s nastavničkim poslužiteljima i klijentima, znači da nije postignuta interoperabilnost komponenti sustava (poslužitelja i klijenata). A to znači da bodovi za zadaću neće biti „izdašni".

Opis rada sustava:

Sustav je zamišljen da simulira tvrtku FOLT koja posjeduje franšizu za dostavu hrane i pića iz restorana. Tvrtka svoju franšizu iznajmljuje drugim partnerima. Franšiza uključuje nekoliko vrsta kuhinja (npr. mediteranska, kontinentalna, vegetarijanska, talijanska, kineska i sl.), a pojedini partner može odabrati samu jednu od ponuđenih kuhinja prilikom svoje registracije. Ponuda pića je jedinstvena bez obzira na odabranu kuhinju. Tvrtka za svaku vrstu kuhinje priprema jelovnik te zajedničku kartu pića. Svaki partner koji s tvrtkom FOLT sklopi ugovor za franšizu dobije svoju identifikacijsku oznaku i sigurnosni kod koji treba koristiti u kasnijem radu.

PosluziteljTvrtka

Pokretanje programa **PosluziteljTvrtka** sadrži jedan argument:

```
datoteka.(txt | xml | bin | json)
```

Npr.

```
NWTiS_04_tvrtka.txt
```

Tvrtku predstavlja poslužitelj koji sadrži jelovnike za sve kuhinje koje su pokrivene franšizom. Pojedini jelovnik predstavlja datoteku u kojoj su podaci o raspoloživim jelima i njihovim cijenama. Za jelovnike se koristi kolekcija sa zapisom Jelovnik. Uz jelovnike postoji i karta pića koja predstavlja datoteku u kojoj su podaci o raspoloživim pićima i njihovim cijenama. Za kartu pića se koristi kolekcija sa zapisom KartaPica. Obje vrste datoteka su u JSON formatu. Tvrtka posjeduje sigurnosne kodove za sve partnere, koje su upisane u datoteku u JSON formatu. Za partnere se koristi kolekcija sa zapisom Partner. Tvrtka uz ulogu poslužitelja za partnere i poslužitelja za kupce ima i ulogu posebnog klijenta za partnere. O tome će biti riječ kasnije.

Poslužitelj „Tvrtka" se pokreće i na početku učita konfiguracijske podatke. Konfiguracijski podaci za tvrtku su npr. sljedeći (ako je u tekstualnom/Properties formatu):

kuhinja_1=MK;Mediteranska kuhinja
kuhinja_2=KK;Kontinentalna kuhinja
…
kuhinja_9=ZK;Zagorska kuhinja

Nije nužno da svi brojevi od 1 do 9 budu iskorišteni, tako da neki mogu biti preskočeni. Vrijednost koja je pridružena pojedinoj kuhinji sadrži dva podatka razdvojena znakom „;", a to su vrsta kuhinje i naziv kuhinje. Npr. kod kuhinje s oznakom (ključem) „kuhinja_1", vrijednost je „MK;Mediteranska kuhinja" pa je onda njena vrsta „MK", a njen naziv „Mediteranska kuhinja".

Za svaku kuhinju postoji jelovnik koji je pohranjen u datoteci čiji naziv odgovara oznaci kuhinje. Npr. za Mediteransku kuhinju koja ima oznaku kuhinja_1 naziv datoteke je kuhinja_1.json. Datoteka s kartom pića ima naziv koji je određen postavkom „datotekaKartaPica".

Broj kuhinja može biti od 1 do 9, a to se provjerava prilikom pokretanja poslužitelja tvrtka. Polazi se od konfiguracijske datoteke i za svaku kuhinju se provjerava postoji li pridružene datoteka. Ako ne postoji datoteka tada se preskače ta kuhinja.

Naziv datoteke s podacima o partnerima uključujući njihove sigurnosne kodove određen postavkom „datotekaPartnera".

Prvo se učita datoteku s podacima o partnerima, zatim datoteke s jelovnicima i na kraju datoteka s kartom pića.

Slijedi pokretanje triju virtualnih dretvi s Future kojima se pokreću tri poslužitelja za:

1. kraj rada
2. registraciju partnera
3. rad s partnerima.

Postoji zastavicu za kraj rada tipa AtomicBoolean koja se inicijalno postavi na false. Dok ne završe svoj rad sva tri poslužitelja provjerava se stanje zastavice za kraj. Ako je true, tada se svim radnim virtualnim dretvama (ne tri gore spomenute) koje nisu završile svoj rad šalje metoda cancel(true) kao i dretvama za registraciju partnera i rad s partnerima ako još nisu završile svoj rad. Ako zastavica za kraj nije true onda se spava prema broju milisekundi koji je određen postavkom „pauzaDretve".

Poslužitelj za kraj rada

Njegova mrežna vrata određena su postavkom „mreznaVrataKraj". Poslužitelj radi u jednodretvenom tj. slijednom radu. Sigurnosni kod za kraj rada određen je postavkom „kodZaKraj". Jedina komanda koju on podržava je:

- **`KRAJ kodZaKraj`**
  - npr. `KRAJ ABBACABA`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaKraj odgovara onom iz postavki. Ako odgovara, provjerava da li je IP adresa s kojeg dolazi zahtjev jednaka IP adresi poslužitelja na kojem se izvršava. Ako je jednaka, postavlja zastavicu za kraj rada na true i vraća OK.
  - Npr. `OK`

Kodovi pogrešaka za poslužitelja za kraj:

- ERROR 10 - Format komande nije ispravan ili nije ispravan kod za kraj
- ERROR 11 - Adresa računala s kojeg je poslan zahtjev nije lokalna adresa
- ERROR 19 - Nešto drugo nije u redu.

Poslužitelj za registraciju partnera

Njegova mrežna vrata određena su postavkom „mreznaVrataRegistracija“. Poslužitelj radi u višedretvenom tj. paralelnom radu koristeći virtualne dretve s Future.

Komande koju on podržava su:

- **PARTNER id "Naziv partnera" vrstaKuhinje adresa mreznaVrata gpsSirina gpsDuzina**
  - npr. `PARTNER 1 "Roštilj Pero" MK localhost 8010 46.29950 16.33001`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li su podaci tog partnera spremljeni u memoriji u kolekciji partnera. Ako nisu, generira sigurnosni kod na sljedeći način. Prvo se spaja njegov naziv i adresa te se na dobivenu vrijednost pozove funkcija hashCode(). Vraćena vrijednost je argument kod poziva funkcije Integer.toHexString. Vraćena vrijednost predstavlja sigurnosni kod partnera. Kreira objekt zapisa Partner i dodaje ga u kolekciju partnera (tj. registrira partnera), sprema podatke kolekcije partnera u datoteku i vraća OK sigurnosniKod.
  - Npr. `OK 4958583733`

- **OBRIŠI id sigurnosniKod**
  - npr. `OBRIŠI 1 4958583733`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li su podaci tog partnera spremljeni u memoriji u kolekciji partnera. Ako jesu, provjerava da li sigurnosni kod odgovara njegovom iz kolekcije partnera. Ako odgovara, briše ga iz kolekcije partnera (tj. deregistracija partnera), sprema podatke kolekcije partnera u datoteku i vraća OK.
  - Npr. `OK`

- **POPIS**
  - npr. `POPIS`
  - Provjera da li ispravni podaci. Ako su ispravni vraća OK i u sljedećem retku vraća u json formatu kolekciju partnera primjenom zapisa PartnerPopis, koju kreira na temelju kolekcije partnera.
  - Npr. `OK`
    [{....}]

Kodovi pogrešaka za poslužitelja za registraciju:

- ERROR 20 - Format komande nije ispravan
- ERROR 21 - Već postoji partner s id u kolekciji partnera
- ERROR 22 - Neispravan sigurnosni kod partnera
- ERROR 23 - Ne postoji partner s id u kolekciji partnera i/ili neispravan sigurnosni kod partnera
- ERROR 29 - Nešto drugo nije u redu.

Poslužitelj za rad s partnerima

Njegova mrežna vrata određena su postavkom „mreznaVrataRad“. Poslužitelj radi u višedretvenom tj. paralelnom radu koristeći virtualne dretve s Future. Komande koju on podržava su:

- **`JELOVNIK id sigurnosniKod`**
  - npr. `JELOVNIK 1 4958583733`
  - Provjera da li su ispravni podaci. Ako su ispravni provjerava da li su podaci tog partnera spremljeni u memoriji u kolekciji partnera. Ako jesu, provjerava da li sigurnosni kod odgovara njegovom iz kolekcije partnera. Ako odgovara, vraća OK i u sljedećem retku vraća u json formatu kolekciju jelovnika za vrstu kuhinje koja je pridružena tom partneru.
  - Npr. `OK`
    `[ {....} ]`

- **`KARTAPIĆA id sigurnosniKod`**
  - npr. `KARTAPIĆA 1 4958583733`
  - Provjera da li su ispravni podaci. Ako su ispravni provjerava da li su podaci tog partnera spremljeni u memoriji u kolekciji partnera. Ako jesu, provjerava da li sigurnosni kod odgovara njegovom iz kolekcije partnera. Ako odgovara, vraća OK i u sljedećem retku vraća u json formatu kolekciju pića.
  - Npr. `OK`
    `[ {....} ]`

- **`OBRAČUN id sigurnosniKod jsonPodaciObračuna`**
  - npr. `OBRAČUN 1 4958583733`
    `[ {...} ]`
  - Provjera da li su ispravni podaci u prvom redu. Ako su ispravni vraća, čita ostale retke dok u retku ne pronađe znak „]“, koji označava kraj podataka. Pretvara preuzete podatke putem Gson u niz objekata zapisa Obracun. Učita prethodne obračune iz datoteke koja ima naziv koji je određen postavkom „datotekaObracuna“. Dodaje im nove obračune i sprema sve obračune u datoteku. Ako je sve u redu vraća OK.
  - Npr. `OK`

Kodovi pogrešaka za poslužitelja za rad s partnerima:

- ERROR 30 - Format komande nije ispravan
- ERROR 31 - Ne postoji partner s id u kolekciji partnera i/ili neispravan sigurnosni kod partnera
- ERROR 32 - Ne postoji jelovnik s vrstom kuhinje koju partner ima ugovorenu
- ERROR 33 - Neispravan jelovnik
- ERROR 34 - Neispravna karta pića
- ERROR 35 - Neispravan obračun
- ERROR 39 - Nešto drugo nije u redu.

PoslužiteljPartner

Poslužitelj „Partner" može se izvršiti na tri načina što ovisi o broju i sadržaju argumenata, od kojih je prvi uvijek naziv datoteke konfiguracije:

1. nazivDatotekeKonfiguracije – predstavlja registraciju partnera
2. nazivDatotekeKonfiguracije KRAJ – predstavlja slanje zahtjeva za kraj rada poslužitelja „Tvrtka"
3. nazivDatotekeKonfiguracije PARTNER – predstavlja poslužitelja za prijem zahtjeva kupaca.

Poslužitelj „Partner" na početku učita konfiguracijske podatke.

Registracija partnera

Pokretanje programa **PoslužiteljPartner**:

```
datoteka.(txt | xml | bin | json)
```

Npr.

```
NWTiS_04_partner_1.txt
```

Poslužitelj „Partner" se spaja na mrežnu utičnicu koja je određena postavkama „adresa" i „mreznaVrataRegistracija". Podaci za registraciju partnera čitaju se iz konfiguracijskih podataka, priprema se komanda prema formatu za registraciju partnera. Komanda se šalje. Ako je vraćen odgovor kojemu je prva riječ „OK", onda se u drugoj riječi nalazi sigurnosniKod partnera. Taj kod se sprema u postavku „sigKod" . Konfiguracija se sprema. Kraj programa.

Slanje zahtjeva za kraj rada poslužitelja „Tvrtka".

Pokretanje programa **PoslužiteljPartner**:

```
datoteka.(txt | xml | bin | json) KRAJ
```

Npr.

```
NWTiS_04_partner_1.txt KRAJ
```

Poslužitelj „Partner" se spaja na mrežnu utičnicu koja je određena postavkama „adresa" i „mreznaVrataKraj". Kod za kraj preuzima se iz postavke „kodZaKraj", priprema se komanda prema formatu za kraj rada. Komanda se šalje. Ako je vraćen odgovor „OK", onda se ispiše na ekranu „Uspješan kraj poslužitelja.". Kraj programa.

Poslužitelj za prijem zahtjeva kupaca

Pokretanje programa **PosluziteljPartner**:

```
datoteka.(txt | xml | bin | json) PARTNER
```

Npr.

```
NWTiS_04_partner_1.txt PARTNER
```

Poslužitelj „Partner" se spaja na mrežnu utičnicu koja je određena postavkama „adresa" i „mreznaVrata-Rad". Podatke o sebe preuzima iz postavki „id" i „sigKod". Šalje zahtjev za jelovnikom koji odgovara njegovoj kuhinji i dobivene podatke učitava u kolekciju jelovnika. Šalje zahtjev za kartom pića i dobivene podatke učitava u kolekciju karta pića. Ako nije uspješno punjenje obiju kolekcija, ispisuje poruku i kraj rada. Pokreće poslužitelj na mrežnim vratima koja su određena postavkom „mreznaVrata". Poslužitelj radi u višedretvenom tj. paralelnom radu koristeći virtualne dretve s Future. Čeka dok svaka virtualna dretva na završi svoj rad koristeći u petlji spavanje prema broju milisekundi koji je određen postavkom „pauzaDretve". Komande koju on podržava su

- **`JELOVNIK korisnik`**
  - npr. `JELOVNIK pero`
  - Provjera da li su ispravni podaci. Ako su ispravni, vraća OK i u sljedećem retku vraća u json formatu kolekciju jelovnika za vrstu kuhinje koja je pridružena tom partneru.
  - Npr. `OK`
    `[ {....} ]`

- **`KARTAPIĆA korisnik`**
  - npr. `KARTAPIĆA pero`
  - Provjera da li su ispravni podaci. Ako su ispravni, vraća OK i u sljedećem retku vraća u json formatu kolekciju pića.
  - Npr. `OK`
    `[ {....} ]`

- **`NARUDŽBA korisnik`**
  - npr. `NARUDŽBA pero`
  - Provjera da li su ispravni podaci. Ako su ispravni, provjerava da li korisnik ima otvorenu narudžbu. Ako nema, kreira novu narudžbu s njegovim nazivom, vraća OK.
  - Npr. `OK`

- **`JELO korisnik idJela količina`**
  - npr. `JELO pero MK1 1.0`
  - Provjera da li su ispravni podaci. Ako su ispravni, provjerava da li korisnik ima otvorenu narudžbu. Ako ima, provjerava da li postoji jelo s idJela u jelovniku partnera. Ako postoji, narudžbi korisnika dodaje novu stavku, vraća OK. Za stavke narudžbe se koristi kolekcija sa zapisom Narudzba.
  - Npr. `OK`

- **`PIĆE korisnik idPića količina`**
  - npr. `PIĆE pero p1 4.0`
  - Provjera da li su ispravni podaci. Ako su ispravni, provjerava da li korisnik ima otvorenu narudžbu. Ako ima, provjerava da li postoji piće s idPića u karti pića partnera. Ako postoji, narudžbi korisnika dodaje novu stavku, vraća OK. Za stavke narudžbe se koristi kolekcija sa zapisom Narudzba.
  - Npr. `OK`

- **`RAČUN korisnik`**
  - npr. `RAČUN pero`
  - Provjera da li su ispravni podaci. Ako su ispravni, provjerava da li korisnik ima otvorenu narudžbu. Ako ima, sve stavke iz narudžbe prebacuju se u plaćene narudžbe, a narudžba korisnika se briše. Povećava se broj naplaćenih narudžbi. Ako je modulo za broj narudžbi s brojem narudžbi (postavka „kvotaNarudzbi") jednak 0 tada se priprema obračun. To se radi tako da se zbraja količina prema idJela odnosno idPića za sve stavke naplaćenih narudžbi. Komandom OBRAČUN šalje obračun u json formatu na PoslužiteljTvrtka. Ako je sve u redu, briše sve stavke u plaćene narudžbe, vraća OK. Ako je modulo različih od 0, vraća OK.
  - Npr. `OK`

Kodovi pogrešaka za poslužitelja za prijem zahtjeva kupaca:

- ERROR 40 - Format komande nije ispravan
- ERROR 41 - Ne postoji jelo s id u kolekciji jelovnika kod partnera
- ERROR 42 - Ne postoji piće s id u kolekciji karte pića kod partnera
- ERROR 43 - Ne postoji otvorena narudžba za korisnika/kupca
- ERROR 44 - Već postoji otvorena narudžba za korisnika/kupca
- ERROR 45 - Neuspješno slanje obračuna
- ERROR 46 - Neuspješno preuzimanje jelovnika
- ERROR 47 - Neuspješno preuzimanje karte pića
- ERROR 49 - Nešto drugo nije u redu.

KorisnikKupac

Pokretanje programa **KorisnikKupac**:

```
datoteka.(txt | xml | bin | json) datotekaPodataka.csv
```

Npr.

```
NWTiS_04_kupac_1.txt komande_1.csv
```

Korisnik „Kupac" na početku učita konfiguracijske podatke.

Struktura datotekaPodataka.csv je sljedeća (znak odvajanja „;"):

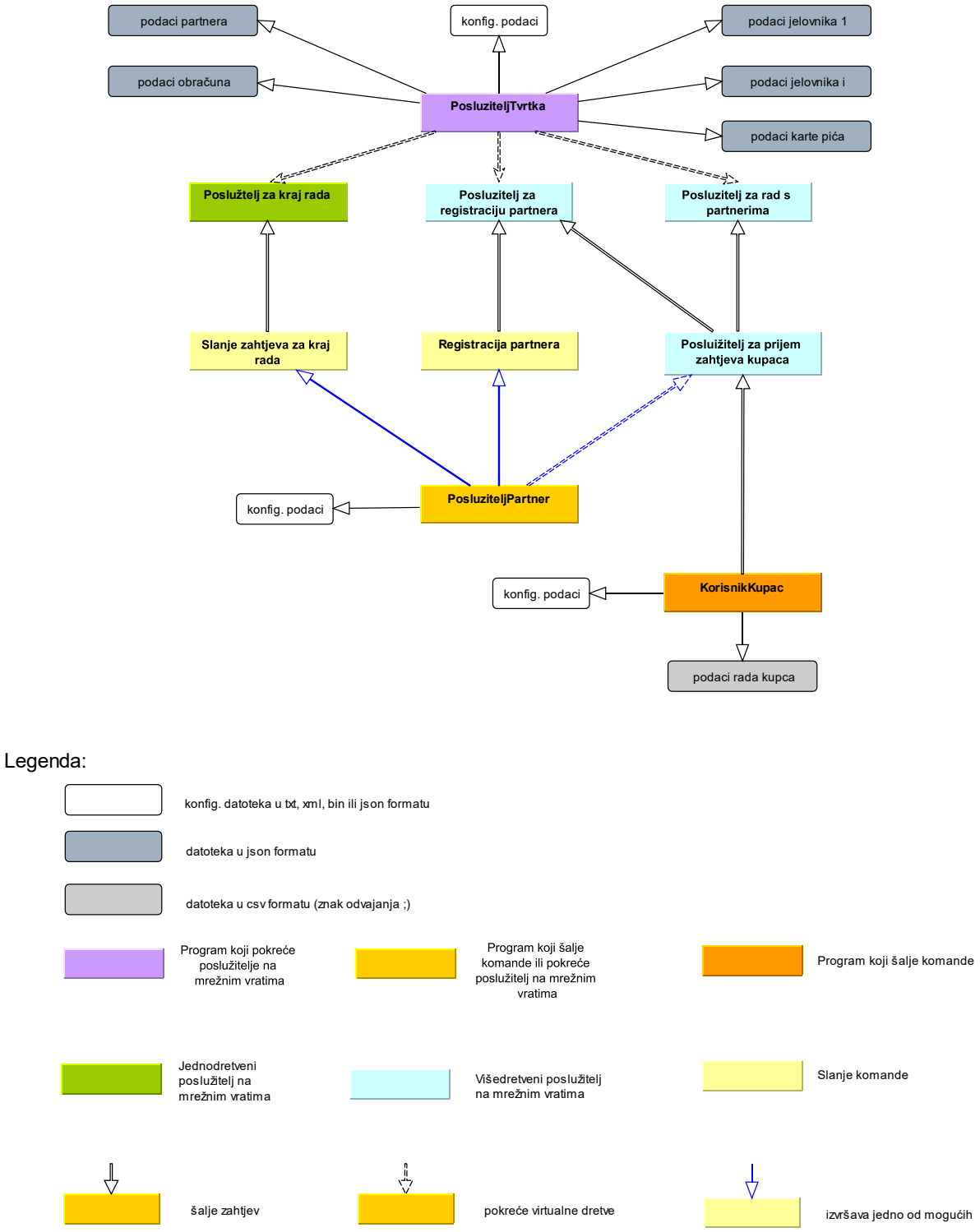
- Korisnik
- Adresa
- Mrežna vrata
- Spavanje
- Komanda.

Npr:

```
pero;localhost;8001;10;POPIS
pero;localhost;8010;10;JELOVNIK pero
pero;localhost;8010;10;JELOVNIK pero
pero;localhost;8010;10;KARTAPIĆA pero
pero;localhost;8010;100;NARUDŽBA pero
pero;localhost;8010;100;JELO pero MK1 1.0
pero;localhost;8010;100;PIĆE pero p1 4.0
mato;localhost;8010;10;NARUDŽBA mato
mato;localhost;8010;20;PIĆE mato p13 3.0
mato;localhost;8010;10;RAČUN mato
pero;localhost;8010;10;PIĆE pero p14 3.0
pero;localhost;8010;10;PIĆE pero p13 1.0
pero;localhost;8010;100;RAČUN pero
```

Korisnik „Kupac" otvara datotekuPodataka.csv, čita redak po redak, se spaja na mrežnu utičnicu koja je određena s „Adresa" i „Mrežna vrata". Odrađuje spavanje i zatim šalje komandu.

Shema sustava prikazana je na slici. U sustavu može biti samo jedan PosluziteljTvrtka, najmanje jedan PosluziteljPartner i najmanje jedan KorisnikKupac. Broj PosluziteljPartner i KorisnikKupac je proizvoljan. Kod testiranja će biti najmanje dva PosluziteljPartner i najmanje četiri KorisnikKupac.



Slika 1. Shema sustava

Klase PosluziteljTvrtka, PosluziteljPartner i KorisnikKupac realiziraju svoje funkcionalnosti bez dodavanja novih klasa (unutarnjih ili vanjskih).

Jedinično testiranje treba biti realizirano za klasu PosluziteljTvrtka. Poželjno je napraviti jedinično testiranje i za klase PosluziteljPartner i KorisnikKupac kako bi se otkrile eventualne pogreške i nakon njihovog ispravljanja postigla viša kvaliteta koda.

Programi PosluziteljTvrtka i PosluziteljPartner imaju kritične dijelove (metode ili dijelovi metoda) u kojima je potrebno osigurati da samo jedna virtualna dretva ima pristup. Kod PosluziteljTvrtka to je kod prijema komande OBRAČUN od učitavanja datoteke s postojećim obračunima preko dodavanja preuzetih podataka obračuna do spremanja ažuriranih podataka obračuna u datoteku. Kod PosluziteljPartner to je kod prijema:

- komande NARUDŽBA od provjere postojanja otvorene narudžbe do kreiranja nove narudžbe
- komandi JELO, PIĆE od provjere postojanja otvorene narudžbe do upisa stavke narudžbe
- komande RAČUN od provjere postojanja otvorene narudžbe preko prebacivanja stavki narudžbe u plaćene stavke narudžbi do slanja obračuna (ako je potrebno slati obračun).

Programi PosluziteljTvrtka i PosluziteljPartner moraju preuzeti kontrolu u slučaju pokušaja nasilnog prekida programa (Ctrl-C i sl.) te provjeriti sve aktivne virtualne dretve. To se radi tako da se za svaku od njih pojedinačno zatražiti prekid što povlači da virtualna dretva zatvori sve svoje otvorene veze na mrežnoj utičnici. I na kraju ispiše na ekranu za svaku virtualnu dretvu broj veza na mrežnoj utičnici koje je bilo potrebno zatvoriti. Na kraju prije izlaza iz programa ispiše se na ekranu broj virtualnih dretvi koje je bilo potrebno prekinuti.

Vježba: 7 – Zadaća 2. FOLT - Dostava hrane i pića na bazi više poslužitelja na mrežnoj utičnici i RESTful web servisima primjenom MicroProfile

Opća pravila

Nazivi klasa, nazivi atributa, nazivi metoda, nazivi varijabli, komentari i sl. pišu se na hrvatskom jeziku u skladu s preporukama za programski jezik Java. Metode u klasama NE smiju imati više od 35 linija programskog koda, u što se ne broji definiranje metode, njenih argumenata i lokalnih varijabli, prazna linija, linija samo s { ili }. U jednoj liniji može biti jedna instrukcija. U jednoj liniji može biti najviše 100 znakova. Pisanje programskog koda mora biti u skladu s preporukama Google Java Code Style. Ne smiju se koristiti klase ili metode koje su označene kao „deprecated". Klase i metode trebaju biti dokumentirane u Javadoc formatu.

Naziv projekta: **{LDAP_korisnik}_vjezba_07_dz_2**

Korijenski direktorij treba biti **{LDAP_korisnik}_vjezba_07_dz_2**

Vježba 7 – zadaća 2 nastavlja se na vježbu 4 – zadaću 1.

Sve nove klase trebaju biti u paketu **edu.unizg.foi.nwtis.{LDAP_korisnik}.vjezba_07_dz_2**. Sve projekte/Maven module iz roditeljskog projekta {LDAP_korisnik}_vjezba_04_dz_1 treba kopirati u roditeljski projekt {LDAP_korisnik}_vjezba_07_dz_2 i promijeniti im naziv direktorija i u njihovom pom.xml da bude usklađen s nazivom novog roditeljskog projekta. Isto vrijedi i za pakete. Verzija svakom Maven modulu povećava se za jedan u dijelu manje verzije (srednji broj) s obzirom na zadnju verziju.

Obavezno na svakom Maven modulu obrisati direktorije .settings i target te datoteke .classpath i .project. Učitati Maven module u Eclipse IDE. Takav postupak proveden je nekoliko puta na vježbama. **Projekt se isključivo treba predati u formatu Eclipse IDE projekta s maven upravljanjem.** Prije predavanja projekta potrebno je **napraviti Clean na roditeljskom projektu** (i svim njegovim Maven modulima). Zatim cijeli roditeljski projekt (i sve njegove Maven module) sažeti u **.zip** (NE .rar) format s nazivom **{LDAP_korisnik}_vjezba_07_dz_2.zip** i predati u Moodle. Uključiti izvorni kod i popunjeni obrazac za zadaću pod nazivom **{LDAP_korisnik}_vjezba_07_dz_2.pdf** (u korijenskom direktoriju projekta). U radu programa datoteka konfiguracijskih podataka i ostale datoteke smještene su na direktoriju s kojeg se pokreće program. Program se može pokretati s različitih direktorija kako bi se mogao izvršavati s različitim datotekama konfiguracijskih podataka i ostalih datoteka. **NE smiju se koristiti druge biblioteke klase osim onih koje se nalazu u opisu zadaće ili su dogovorene tijekom nastave i objavljene u forumu za zadaću da se smiju koristiti.**

Boduju se dijelovi koji su rađeni nakon vježbi!

Struktura .zip datoteke predane zadaće treba biti sljedeća:

```
{LDAP_korisnik}_vjezba_07_dz_2
	{LDAP_korisnik}_vjezba_07_dz_2.pdf

				...
	{LDAP_korisnik}_vjezba_07_dz_2_kupac
		pom.xml
		src
			main
				...
			test
				...
	{LDAP_korisnik}_vjezba_07_dz_2_lib_konfig
		pom.xml
		src
			main
				...
			test
	{LDAP_korisnik}_vjezba_07_dz_2_lib_podaci
		pom.xml
		src
			main
				...
			test
				...
	{LDAP_korisnik}_vjezba_07_dz_2_partner
		pom.xml
		src
			main
				...
			test
				...
	{LDAP_korisnik}_vjezba_07_dz_2_servis
		pom.xml
		src
			main
				...
			test
				...
				...
	{LDAP_korisnik}_vjezba_07_dz_2_rest
		pom.xml
		src
			main
				...
			test
				...

	{LDAP_korisnik}_vjezba_07_dz_2_tvrtka
		pom.xml
		src
			main
				...
			test
				...
```

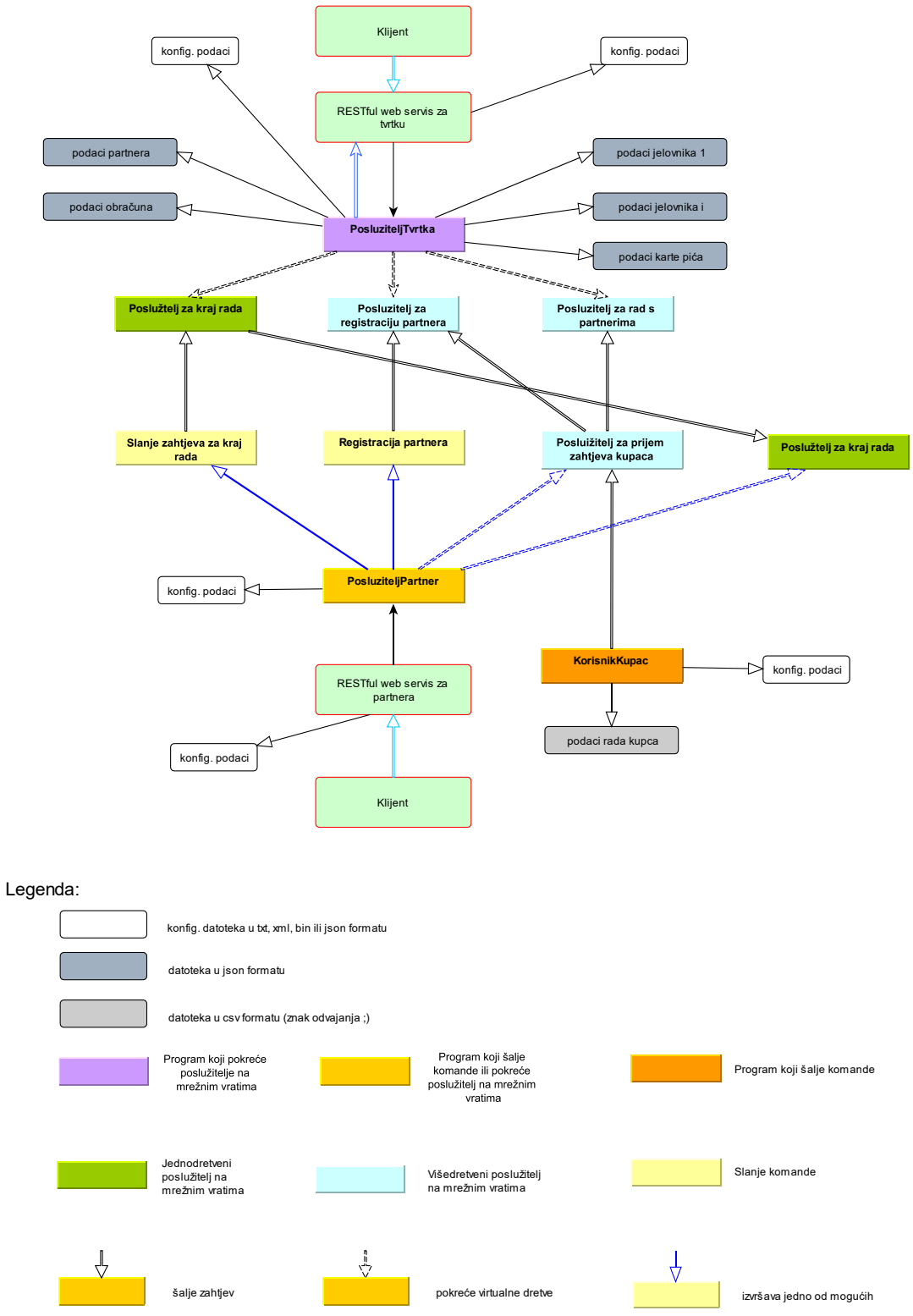
Tablica 1. Popis Maven modula

Naziv Maven modula	Uloga Maven modula i kako je nastao	Verzija
`{LDAP_korisnik}_vjezba_07_dz_2_lib_konfig`	Knjižnica klasa za rad s konfiguracijskom podacima. Nastala iz `{LDAP_korisnik}_vjezba_04_dz_1_lib`	1.4.0
`{LDAP_korisnik}_vjezba_07_dz_2_lib_podaci`	Knjižnica klasa za rad podacima. Nastala izdvajanjem paketa podaci i pomocnici iz `{LDAP_korisnik}_vjezba_04_dz_1_podaci`	1.2.0
`{LDAP_korisnik}_vjezba_07_dz_2_tvrtka`	Aplikacija za rad na mrežnoj utičnici. Nastala iz `{LDAP_korisnik}_vjezba_04_dz_1_tvrtka.`	1.2.0
`{LDAP_korisnik}_vjezba_07_dz_2_partner`	Aplikacija za rad na mrežnoj utičnici. Nastala iz `{LDAP_korisnik}_vjezba_04_dz_1_partner.`	1.1.0
`{LDAP_korisnik}_vjezba_07_dz_2_kupac`	Korisnik aplikacije za rad na mrežnoj utičnici. Nastala iz `{LDAP_korisnik}_vjezba_04_dz_1_kupac.`	1.1.0
`{LDAP_korisnik}_vjezba_07_dz_2_lib_rest`	Knjižnica klasa za podršku RESTful web servisa u radu s bazom podataka. Slična nwtis.rest.lib	1.0.0
`{LDAP_korisnik}_vjezba_07_dz_2_servisi`	Aplikacija s poslužiteljem RESTful web servisa. Dijelovi preuzeti iz nwtis.rest.cdi.korisnici	1.0.0

Opis rada sustava:

Sve što se tražilo u opisu rada sustava u vježbi 4 – zadaći 1 i dalje je potrebno za rad vježbe 7 – zadaće 2. Određeni poslužitelji dobit će dodatne funkcionalnosti putem kojih će se integrirati u rješenje.

Shema sustava prikazana je na slici.



Slika 1. Shema sustava

PosluziteljTvrtka ima dodatne komande (Poslužitelj za kraj rada):

- **STATUS kodZaAdminTvrtke [1, 2]**
  - npr. `STATUS FEBADACE 1`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminTvrtke odgovara onom iz postavki. Ako je u redu vraća status za odabrani dio poslužitelja: 1 – registracija, 2 – partneri. Vraća OK [0, 1]. 0 – pauza, 1 – aktivni rad
  - Npr. `OK 0`

- **PAUZA kodZaAdminTvrtke [1, 2]**
  - npr. `PAUZA FEBADACE 1`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminTvrtke odgovara onom iz postavki. Ako je u redu i ako odabrani dio poslužitelja nije u pauzi postavlja odabrani dio poslužitelja u pauzu i vraća OK.
  - Npr. `OK`

- **START kodZaAdminTvrtke [1, 2]**
  - npr. `START FEBADACE 1`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminTvrtke odgovara onom iz postavki. Ako je u redu i ako odabrani dio poslužitelja je u pauzi postavlja odabrani dio poslužitelja u aktivni rad i vraća OK.
  - Npr. `OK`

- **SPAVA kodZaAdminTvrtke n**
  - npr. `SPAVA FEBADACE 1500`
  - Provjera da li ispravni podaci. Ako su ispravni dretva spava n milisekundi i vraća OK.
  - Npr. `OK`

- **KRAJWS kodZaKraj**
  - npr. `KRAJWS ABBACABA`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaKraj odgovara onom iz postavki. Šalje komandu KRAJ kodZaKraj svim aktivnim PosluziteljPartner. Ako su svi vratili OK, vraća OK i prestaje s radom. Ako barem jedan ne javi OK, vraća kod pogreške i ne prestaje s radom.
  - Npr. `OK`

- **OSVJEŽI kodZaAdminTvrtke**
  - npr. `OSVJEŽI FEBADACE`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminTvrtke odgovara onom iz postavki. Ako je u redu i ako dio poslužitelja za partnere nije u pauzi, ponovno učitava kartu pića i jelovnike, i vraća OK.
  - Npr. `OK`

PosluziteljTvrtka ima izmjenu/dopunu komande (Poslužitelj za kraj rada):

- **`KRAJ kodZaKraj`**
  - npr. `KRAJ ABBACABA`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaKraj odgovara onom iz postavki. ~~Ako odgovara, provjerava da li je IP adresa s kojeg dolazi zahtjev jednaka IP adresi poslužitelja na kojem se izvršava. Ako je jednaka, postavlja zastavicu za kraj rada na true i vraća OK.~~ Šalje komandu KRAJ kodZaKraj svim aktivnim PosluziteljPartner. Ako su svi vratili OK, šalje zahtjev HEAD/kraj/info na krajnju točku api/tvrtka RESTful web servisa, vraća OK i prestaje s radom. Ako barem jedan ne javi OK, vraća kod pogreške i ne prestaje s radom.
  - Npr. `OK`

Dodatni kodovi pogrešaka su:

- ERROR 12 – Pogrešan kodZaAdminTvrtke
- ERROR 13 – Pogrešna promjena pauze ili starta
- ERROR 14 – Barem jedan partner nije završio rad
- ERROR 15 – Poslužitelj za partnere u pauzi
- ERROR 16 – Prekid spavanja dretve
- ERROR 17 – RESTful zahtjev nije uspješan

PosluziteljTvrtka ima izmjenu/dopunu komande (Poslužitelj za registraciju partnera):

- **`PARTNER id "Naziv partnera" vrstaKuhinje adresa mreznaVrata gpsSirina gpsDuzina mreznaVrataKraj adminKod`**
  - npr. `PARTNER 1 "Roštilj Pero" MK localhost 8010 46.29950 16.33001 8011 FEBADACE`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li su podaci tog partnera spremljeni u memoriji u kolekciji partnera. Ako nisu, generira sigurnosni kod na sljedeći način. Prvo se spaja njegov naziv i adresa te se na dobivenu vrijednost pozove funkcija hashCode(). Vraćena vrijednost je argument kod poziva funkcije Integer.toHexString. Vraćena vrijednost predstavlja sigurnosni kod partnera. Kreira objekt zapisa Partner i dodaje ga u kolekciju partnera (tj. registrira partnera), sprema podatke kolekcije partnera u datoteku i vraća OK sigurnosniKod.
  - Npr. `OK 4958583733`

Dodatni kodovi pogrešaka su:

- ERROR 24 – Poslužitelj za registraciju partnera u pauzi

PosluziteljTvrtka ima dodatne komande (Poslužitelj za rad s partnerima):

- **`OBRAČUNWS id sigurnosniKod jsonPodaciObračuna`**
  - npr. `OBRAČUNWS 1 4958583733 [ {...} ]`
  - Provjera da li su ispravni podaci u prvom redu. Ako su ispravni vraća, čita ostale retke dok u retku ne pronađe znak „]“, koji označava kraj podataka. Pretvara preuzete podatke putem Gson u niz objekata zapisa Obracun. Učita prethodne obračune iz datoteke koja ima naziv koji je određen postavkom „datotekaObracuna“. Dodaje im nove obračune i sprema sve obračune u datoteku. Ako je sve u redu vraća OK.
  - Npr. `OK`

PosluziteljTvrtka ima izmjenu/dopunu komande (Poslužitelj za rad s partnerima):

- **`OBRAČUN id sigurnosniKod jsonPodaciObračuna`**
  - npr. `OBRAČUN 1 4958583733 [ {...} ]`
  - Provjera da li su ispravni podaci u prvom redu. Ako su ispravni vraća, čita ostale retke dok u retku ne pronađe znak „]“, koji označava kraj podataka. Pretvara preuzete podatke putem Gson u niz objekata zapisa Obracun. Učita prethodne obračune iz datoteke koja ima naziv koji je određen postavkom „datotekaObracuna“. Dodaje im nove obračune i sprema sve obračune u datoteku. Šalje šalje zahtjev POST/obracun na krajnju točku api/tvrtka RESTful web servisa. Ako je sve u redu vraća OK.
  - Npr. `OK`

Dodatni kodovi pogrešaka su:

- ERROR 36 – Poslužitelj za partnere u pauzi
- ERROR 37 – RESTful zahtjev nije uspješan

PosluziteljPartner ima izmjenu/dopunu komande (Poslužitelj za prijem zahtjeva kupaca):

- **`STANJE korisnik`**
  - npr. `STANJE pero`
  - Provjera da li su ispravni podaci. Ako su ispravni, provjerava da li korisnik ima otvorenu narudžbu. Ako ima, vraća OK i u sljedećem retku vraća u json formatu stavke narudžbe.
  - Npr. `OK`
    [{....}]

Dodatni kodovi pogrešaka su:

- ERROR 48 - Poslužitelj za prijem zahtjeva kupaca u pauzi

PosluziteljPartner ima dodatne komande (**novi Poslužitelj za kraj rada**):

- **KRAJ kodZaKraj**
  - npr. `KRAJ ABBACABA`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaKraj odgovara onom iz postavki. Ako odgovara, postavlja zastavicu za kraj rada na true i vraća OK.
  - Npr. `OK`

- **OSVJEŽI kodZaAdminPartnera**
  - npr. `OSVJEŽI FEBADACE`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminPartnera odgovara onom iz postavki. Ako je u redu ponovno preuzima kartu pića i jelovnika, i vraća OK.
  - Npr. `OK`

- **STATUS kodZaAdminPartnera [1]**
  - npr. `STATUS FEBADACE 1`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminPartnera odgovara onom iz postavki. Ako je u redu vraća status za odabrani dio poslužitelja: 1 – kupci. Vraća OK [0, 1]. 0 – pauza, 1 – aktivni rad
  - Npr. `OK 0`

- **PAUZA kodZaAdminPartnera [1]**
  - npr. `PAUZA FEBADACE 1`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminPartnera odgovara onom iz postavki. Ako je u redu i ako odabrani dio poslužitelja nije u pauzi postavlja odabrani dio poslužitelja u pauzu i vraća OK.
  - Npr. `OK`

- **START kodZaAdminPartnera [1]**
  - npr. `START FEBADACE 1`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li kodZaAdminPartnera odgovara onom iz postavki. Ako je u redu i ako odabrani dio poslužitelja je u pauzi postavlja odabrani dio poslužitelja u aktivni rad i vraća OK.
  - Npr. `OK`

- **SPAVA kodZaAdminPartnera n**
  - npr. `SPAVA FEBADACE 1500`
  - Provjera da li ispravni podaci. Ako su ispravni dretva spava n milisekundi i vraća OK.
  - Npr. `OK`

Kodovi pogrešaka su:

- ERROR 60 - Format komande nije ispravan ili nije ispravan kod za kraj
- ERROR 61 – Pogrešan kodZaAdminPartnera
- ERROR 62 – Pogrešna promjena pauze ili starta
- ERROR 63 – Prekid spavanja dretve
- ERROR 69 - Nešto drugo nije u redu.

RESTful web servisi smješteni su u Maven modul **{LDAP_korisnik}_vjezba_07_dz_2_servisi**. Definirane su 2 krajnje točke/putanje koje slijede osnovnu putanju „nwtis/v1/“:

- api/tvrtka – pokriva područje rada poslužitelja PoslužiteljTvrtka s kojim dvosmjerno komunicira. PoslužiteljTvrtka ima sljedeće dijelove: kontrola (kraj), registracija ili partneri. Dio podataka koji se vraćaju potrebno je dohvatiti/filtrirati iz tablice u bazi podataka.
    - HEAD – provjera da li radi poslužitelj PoslužiteljTvrtka
    - HEAD/status/{id} – šalje informacije o radu određenog dijela poslužitelja PoslužiteljTvrtka osim kontrola
    - HEAD/pauza/{id} – šalje zahtjev za pauziranje određenog dijela poslužitelja PoslužiteljTvrtka osim kontrola
    - HEAD/start/{id} – šalje zahtjev za izlazom iz pauze određenog dijela poslužitelja PoslužiteljTvrtka osim kontrola
    - HEAD/kraj – šalje zahtjev za kraj rada poslužitelja PoslužiteljTvrtka i svih njegovih dijelova
    - HEAD/kraj/info – prima informaciju za kraj rada poslužitelja PoslužiteljTvrtka i svih njegovih dijelova
    - GET/jelovnik – vraća informacije o jelovnicima
    - GET/jelovnik/{id} – vraća jelovnik s id
    - GET/kartapica – vraća kartu pića
    - GET/partner – vraća listu partnera
    - GET/partner/provjera – vraća listu samo onih partnera koji se nalaze u tablici i na poslužitelj PoslužiteljTvrtka (komanda POPIS)
    - GET/partner/{id} – vraća partnera s id
    - POST/partner – dodaje novog partnera (ne registrira ga na poslužitelju PoslužiteljTvrtka)
    - GET/obracun[?od=vrijemeOd&do=vrijemeDo] – vraća listu obračuna, a ako su upisani parametri od i/ili do tada one koji zadovoljavaju uvjet
    - GET/obracun/jelo[?od=vrijemeOd&do=vrijemeDo] – vraća listu obračuna za jela, a ako su upisani parametri od i/ili do tada one koji zadovoljavaju uvjet
    - GET/obracun/pice[?od=vrijemeOd&do=vrijemeDo] – vraća listu obračuna za piće, a ako su upisani parametri od i/ili do tada one koji zadovoljavaju uvjet
    - GET/obracun/{id}[?od=vrijemeOd&do=vrijemeDo] – vraća listu obračuna za partnera s id, a ako su upisani parametri od i/ili do tada one koji zadovoljavaju uvjet
    - POST/obracun – prima obračun od poslužitelja PoslužiteljTvrtka i dodaje novi obračun.
    - POST/obracun/ws – dodaje novi obračun i šalje komandu na poslužitelj PoslužiteljTvrtka.
    - GET/spava&vrijeme=trajanje – šalje zahtjev za spavanje dretve u trajanju milisekunda

Partneri se spremaju u tablicu „partneri“ i sve operacije s GET i POST metodama na putanji „partner“ temelje se na toj tablici.

Obračuni se spremaju u tablicu „obracuni“ i sve operacije s GET i POST metodama na putanjama „obracun“ i „obracun/ws“ temelje se na toj tablici.

VrijemeOd i vrijemeDo izraženi su u broju milisekundi od „00:00:00 UTC on Thursday, 1 January 1970“.

- api/partner - pokriva područje rada poslužitelja PosluziteljPartner s kojim jednosmjerno komunicira. PosluziteljPartner ima sljedeće dijelove: kontrola (kraj), kupci. Podatke sprema u memoriji.
    - HEAD – provjerava da li radi poslužitelj PosluziteljPartner
    - HEAD/status/{id} – šalje informacije o radu određenog dijela poslužitelja PosluziteljPartner osim kontrola
    - HEAD/pauza/{id} – šalje zahtjev za pauziranje određenog dijela poslužitelja PosluziteljPartner osim kontrola
    - HEAD/start/{id} – šalje zahtjev za izlazom iz pauze određenog dijela poslužitelja PosluziteljPartner osim kontrola
    - HEAD/kraj – šalje zahtjev za kraj rada poslužitelja i svih njegovih dijelova
    - GET/jelovnik – *vraća jelovnik
    - GET/kartapica – *vraća kartu pića
    - GET/narudzba – *vraća stavke otvorene narudžbe
    - POST/narudzba – *dodaje novu narudžbu
    - POST/jelo – *dodaje novo jelo u narudžbu
    - POST/pice – *dodaje novo piće u narudžbu
    - POST/racun – *zahtjeva račun za svoju otvorenu narudžbu
    - GET/korisnik – vraća listu korisnika
    - GET/korisnik/{id} – vraća korisnika s id
    - POST/korisnik – dodaje novog korisnika
    - GET/spava&vrijeme=trajanje – šalje zahtjev za spavanje dretve u trajanju milisekunda

Kod GET i POST metoda kod kojih se u opisu nalazi *, a odnose se na cjelokupni rad s narudžbama, korisničko ime i lozinka šalju se putem zaglavlja korisnik i lozinka. Za sada u čitljivom obliku. Ako autentikacija nije uspješna, javlja se pogreška s pripadajućim statusom i ne šalje se komanda prema PosluziteljPartner.

Korisnici se spremaju u tablicu „korisnici“ i sve operacije s GET i POST metodama na putanji „korisnik“ temelje se na toj tablici.

Shematski prikaz spomenutih krajnjih točaka RESTful web servisa nalazi se na slici s oznakom Slika 2.

Sve metode RESTful web servisa moraju:

- kod vraćanja koristiti klasu Response. Ako vraćaju podatke oni moraju biti u obliku application/json i moraju se temeljiti na jednom od raspoloživih zapisa
- kod statusa treba odgovarati HTTP statusima[1]
- kod POST metode primati podatke u obliku application/json i moraju se temeljiti na jednom od raspoloživih zapisa.

Tablica 2. RESTful metode za api/tvrtka i HTTP statusi koje mogu vratiti

Metoda	U redu	Nije u redu/pogreška
HEAD	200	500
HEAD/status/{id}	200	204
HEAD/pauza/{id}	200	204
HEAD/start/{id}	200	204
HEAD/kraj	200	204
HEAD/kraj/info	200	204
GET/jelovnik	200	500
GET/jelovnik/{id}	200	404, 500
GET/kartapica	200	500
GET/partner	200	500
GET/partner/provjera	200	500
GET/partner/{id}	200	404, 500
POST/partner	201	409, 500
GET/obracun[?od=vrijemeOd&do=vrijemeDo]	200	500
GET/obracun/jelo[?od=vrijemeOd&do=vrijemeDo]	200	500
GET/obracun/pice[?od=vrijemeOd&do=vrijemeDo]	200	500
GET/obracun/{id}[?od=vrijemeOd&do=vrijemeDo]	200	500
POST/obracun	201	500
POST/obracun/ws	201	500
GET/spava&vrijeme=trajanje	200	500

[1] https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

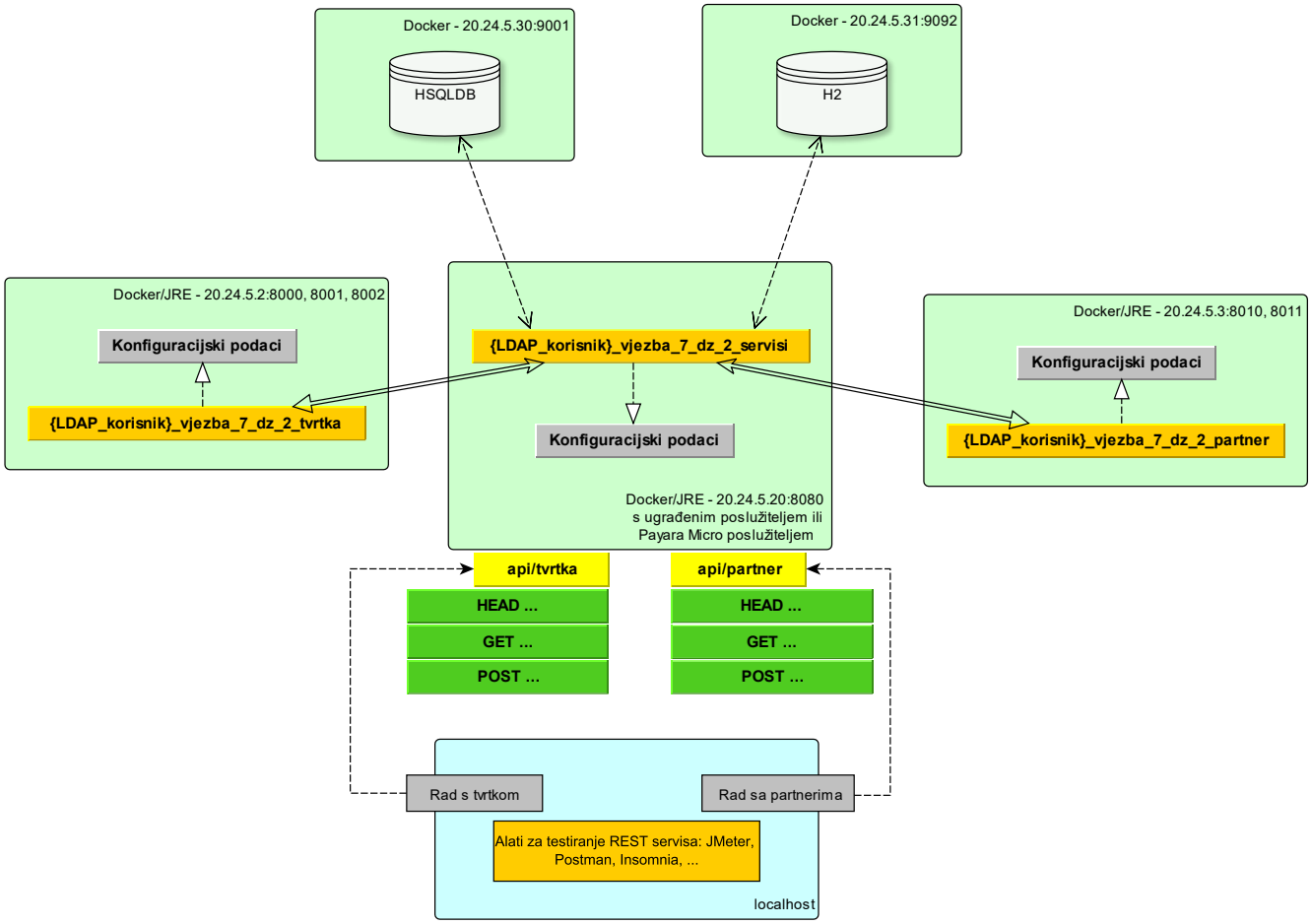
Tablica 3. RESTful metode za api/partner i HTTP statusi koje mogu vratiti

Metoda	U redu	Nije u redu/pogreška
HEAD	200	500
HEAD/status/{id}	200	204
HEAD/pauza/{id}	200	204
HEAD/start/{id}	200	204
HEAD/kraj	200	204
GET/jelovnik	200	401, 500
GET/kartapica	200	401, 500
GET/narudzba	200	401, 500
POST/narudzba	201	401, 409, 500
POST/jelo	201	401, 409, 500
POST/pice	201	401, 409, 500
POST/racun	201	401, 409, 500
GET/korisnik	200	500
GET/korisnik/{id}	200	404, 500
POST/korisnik	201	409, 500
GET/spava&vrijeme=trajanje	200	500

Svaka od komponenti sustava (poslužitelji baza podataka, poslužitelji na mrežnoj utičnici i poslužitelj na RESTful web servise) mora biti u vlastitom Docker kontejneru prema instalacijskoj arhitekturi sustava koja se nalazi se na slici s oznakom Slika 2. Svi Docker kontejneri moraju biti povezani na istu Docker mrežu pod nazivom mreza_{LDAP_korisnik} i integrirani u Docker compose. U compose.yaml treba podesiti da je naziv {LDAP_korisnik}_docker_compose, da se spaja na postojeću vanjsku mrežu i da se koristi postojeći vanjski svezak/volumen.

Maven moduli {LDAP_korisnik}_vjezba_07_dz_2_tvrtka i {LDAP_korisnik}_vjezba_07_dz_2_partner moraju biti podešeni da kreiraju prilagođenu Java sliku tj. da su pakirani kao jlink.

Docker kontejneri za Maven module {LDAP_korisnik}_vjezba_07_dz_2_tvrtka i {LDAP_korisnik}_vjezba_07_dz_2_partner moraju koristiti Docker svezak/volume pod nazivom svezak_{LDAP_korisnik} koji se povezuje na /usr/app/podaci. Pokretanje aplikacija u Docker kontejneru za ta dva modula obavlja se s direktorija /usr/app/podaci. Spajanje na poslužitelje u Docker kontejnerima mora se obavljati putem zadane statičke adrese pojedinog Docker kontejnera na prikazanoj slici.



Slika 2. Instalacijska arhitektura sustava i metode RESTful web servisa

Konfiguracijski podaci za PosluziteljTvrtka (NWTiS_04_tvrtka.txt):

```
kuhinja_1=MK;Mediteranska kuhinja
kuhinja_2=KK;Kontinentalna kuhinja
kuhinja_3=VK;Vegetarijanska kuhinja
mreznaVrataKraj=8000
mreznaVrataRegistracija=8001
mreznaVrataRad=8002
brojCekaca=10
pauzaDretve=100
datotekaPartnera=tvrtka.json
datotekaKartaPica=kartaPica.json
kodZaKraj=BABECABACC
kodZaAdminTvrtke=AB12CD23EF45
datotekaObracuna=obracun.json
restAdresa=http://20.24.5.20:8080/nwtis/v1/api/tvrtka
```

Konfiguracijski podaci za PosluziteljPartner (NWTiS_04_partner_1.txt):

```
adresa=20.24.5.2
brojCekaca=10
gpsDuzina=16.34009
gpsSirina=46.30803
id=1
kodZaKraj=BABECABACC
kodZaAdmin=CACEDACEAA
kuhinja=MK
kvotaNarudzbi=2
mreznaVrata=8010
mreznaVrataKrajPartner=8011
mreznaVrataKraj=8000
mreznaVrataRad=8002
mreznaVrataRegistracija=8001
naziv=FOLT 1 - Varaždin, Pavlinska 9
pauzaDretve=100
sigKod=da53e5e7
```

Konfiguracijski podaci za Servisi (microprofile-config.properties):

```
adresa=20.24.5.2
kodZaKraj=BABECABACC
kodZaAdminPartnera=FE98DC76BA54
kodZaAdminTvrtke=AB12CD23EF45
mreznaVrataKraj=8000
mreznaVrataRad=8002
mreznaVrataRegistracija=8001
korisnickoImeBazaPodataka=nwtis_1
lozinkaBazaPodataka=nwtis#1
upravljacBazaPodataka=org.hsqldb.jdbcDriver
urlBazaPodataka=jdbc:hsqldb:hsql://20.24.5.30:9001/nwtis_1
adresaPartner=20.24.5.3
mreznaVrataKrajPartner=8011
mreznaVrataRadPartner=8010
idPartner=1
```

Vježba: 8 – Zadaća 3. FOLT - Dostava hrane i pića na bazi više poslužitelja na mrežnoj utičnici i RESTful web servisima primjenom MicroProfile, uz korisničko sučelje, WebScoket, JPA

Opća pravila

Nazivi klasa, nazivi atributa, nazivi metoda, nazivi varijabli, komentari i sl. pišu se na hrvatskom jeziku u skladu s preporukama za programski jezik Java. Metode u klasama NE smiju imati više od 35 linija programskog koda, u što se ne broji definiranje metode, njenih argumenata i lokalnih varijabli, prazna linija, linija samo s { ili }. U jednoj liniji može biti jedna instrukcija. U jednoj liniji može biti najviše 100 znakova. Pisanje programskog koda mora biti u skladu s preporukama Google Java Code Style. Ne smiju se koristiti klase ili metode koje su označene kao „deprecated". Klase i metode trebaju biti dokumentirane u Javadoc formatu.

Naziv projekta: **{LDAP_korisnik}_vjezba_08_dz_3**

Korijenski direktorij treba biti **{LDAP_korisnik}_vjezba_08_dz_3**

Vježba 8 – zadaća 3 nastavlja se na vježbu 7 – zadaću 2, koja se nastavlja se na vježbu 4 – zadaću 1.

Sve nove klase trebaju biti u paketu **edu.unizg.foi.nwtis.{LDAP_korisnik}.vjezba_08_dz_8**. Sve projekte/Maven module iz roditeljskog projekta {LDAP_korisnik}_vjezba_07_dz_2 treba kopirati u roditeljski projekt {LDAP_korisnik}_vjezba_08_dz_3 i promijeniti im naziv direktorija i u njihovom pom.xml da bude usklađen s nazivom novog roditeljskog projekta. Isto vrijedi i za pakete. Verzija svakom Maven modulu povećava se za jedan u dijelu manje verzije (srednji broj) s obzirom na zadnju verziju.

Obavezno na svakom Maven modulu obrisati direktorije .settings i target te datoteke .classpath i .project. Učitati Maven module u Eclipse IDE. Takav postupak proveden je nekoliko puta na vježbama. **Projekt se isključivo treba predati u formatu Eclipse IDE projekta s maven upravljanjem.** Prije predavanja projekta potrebno je **napraviti Clean na roditeljskom projektu** (i svim njegovim Maven modulima). Zatim cijeli roditeljski projekt (i sve njegove Maven module) sažeti u **.zip** (NE .rar) format s nazivom **{LDAP_korisnik}_vjezba_08_dz_3.zip** i predati u Moodle. Uključiti izvorni kod i popunjeni obrazac za zadaću pod nazivom **{LDAP_korisnik}_vjezba_08_dz_3.pdf** (u korijenskom direktoriju projekta). U radu programa datoteka konfiguracijskih podataka i ostale datoteke smještene su na direktoriju s kojeg se pokreće program. Program se može pokretati s različitih direktorija kako bi se mogao izvršavati s različitim datotekama konfiguracijskih podataka i ostalih datoteka. **NE smiju se koristiti druge biblioteke klase osim onih koje se nalazu u opisu zadaće ili su dogovorene tijekom nastave i objavljene u forumu za zadaću da se smiju koristiti.**

Boduju se dijelovi koji su rađeni nakon vježbi!

Struktura .zip datoteke predane zadaće treba biti sljedeća:

```
{LDAP_korisnik}_vjezba_08_dz_3
	{LDAP_korisnik}_vjezba_08_dz_3.pdf

			...
	{LDAP_korisnik}_vjezba_08_dz_3_jpa
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_klijenti
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_kupac
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_lib_konfig
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_lib_podaci
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_partner
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_servis
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_rest
		pom.xml
		src
			main
			...
	{LDAP_korisnik}_vjezba_08_dz_3_tvrtka
		pom.xml
		src
			main
				...
```

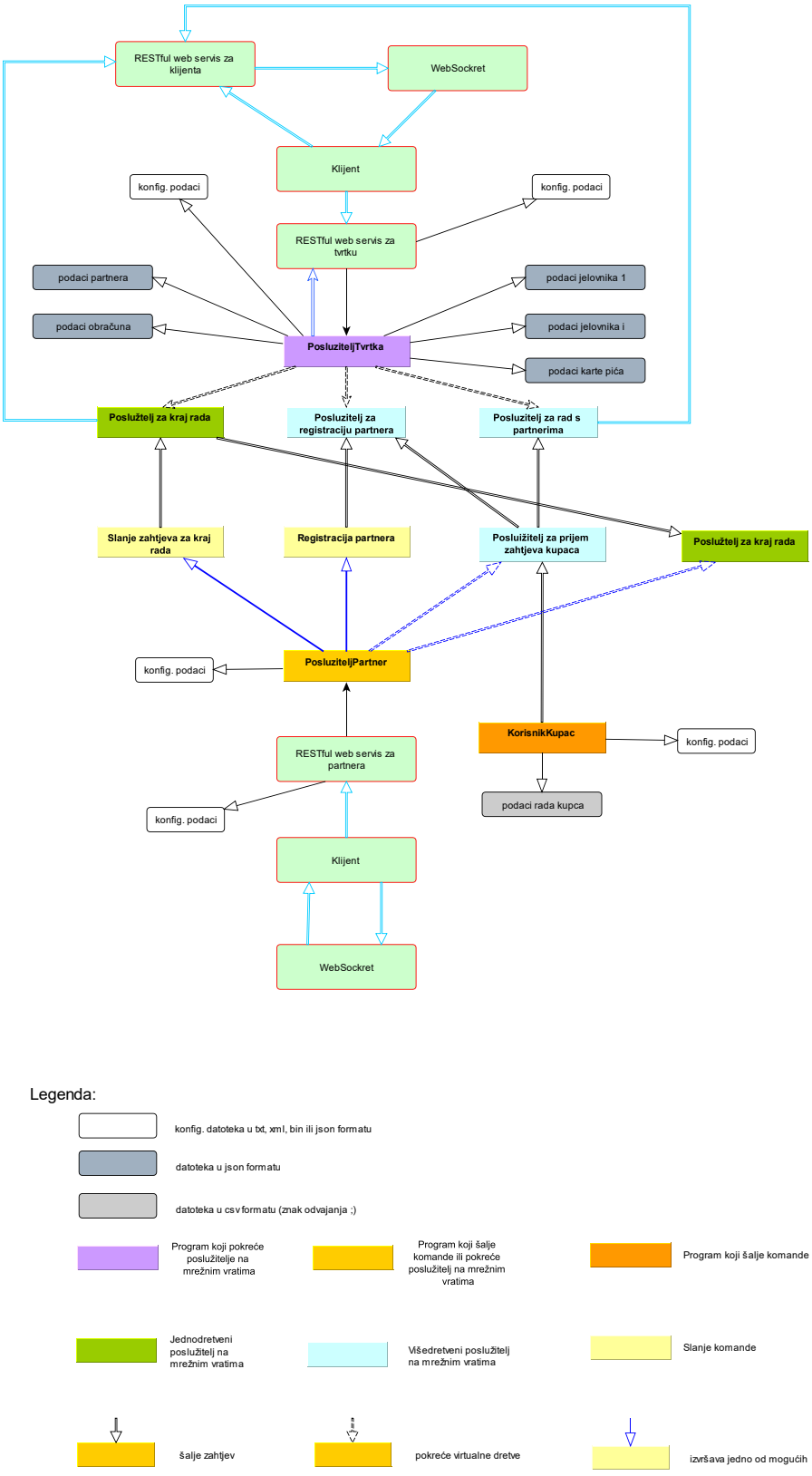
Tablica 1. Popis Maven modula

Naziv Maven modula	Uloga Maven modula i kako je nastao	Verzija
{LDAP_korisnik}_vjezba_08_dz_3_jpa	Knjižnica klasa za rad s Jakarta JPA – entiteti i fasade/pomoćnici.	1.0.0
{LDAP_korisnik}_vjezba_08_dz_3_lib_konfig	Knjižnica klasa za rad s konfiguracijskom podacima. Nastala iz **{LDAP_korisnik}_vjezba_04_dz_1_lib**	1.5.0
{LDAP_korisnik}_vjezba_08_dz_3_lib_podaci	Knjižnica klasa za rad podacima. Nastala izdvajanjem paketa podaci i pomocnici iz **{LDAP_korisnik}_vjezba_04_dz_1_podaci**	1.3.0
{LDAP_korisnik}_vjezba_08_dz_3_tvrtka	Aplikacija za rad na mrežnoj utičnici. Nastala iz **{LDAP_korisnik}_vjezba_04_dz_1_tvrtka.**	1.3.0
{LDAP_korisnik}_vjezba_08_dz_3_partner	Aplikacija za rad na mrežnoj utičnici. Nastala iz **{LDAP_korisnik}_vjezba_04_dz_1_partner.**	1.2.0
{LDAP_korisnik}_vjezba_08_dz_3_kupac	Korisnik aplikacije za rad na mrežnoj utičnici. Nastala iz **{LDAP_korisnik}_vjezba_04_dz_1_kupac.**	1.2.0
{LDAP_korisnik}_vjezba_08_dz_3_lib_rest	Knjižnica klasa za podršku RESTful web servisa u radu s bazom podataka. Nastala iz **{LDAP_korisnik}_vjezba_07_dz_2_lib_rest**	1.1.0
{LDAP_korisnik}_vjezba_08_dz_3_servisi	Aplikacija s poslužiteljem RESTful web servisa. Nastala iz **{LDAP_korisnik}_vjezba_07_dz_2_servisi**	1.1.0
{LDAP_korisnik}_vjezba_08_dz_3_klijenti	Aplikacija s klijentima RESTful seb servisa korištenjem Jakarta MVC Dijelovi preuzeti iz nwtis.rest.mvc.korisnici i nwtis.rest.mvc	1.0.0

Opis rada sustava:

Sve što se tražilo u opisu rada sustava u vježbi 4 – zadaći 1 i dalje je potrebno za rad vježbe 7 – zadaće 2 i za rad vježbe 8 – zadaće 3. Sve što se tražilo u opisu rada sustava u vježbi 7 – zadaća 2 i dalje je potrebno za rad vježbe 8 – zadaće 3. Određeni poslužitelji dobit će dodatne funkcionalnosti putem kojih će se integrirati u rješenje.

Shema sustava prikazana je na slici.



Slika 1. Shema sustava

Dopune na RESTful web servisa koji su smješteni u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_servisi**:

1. api/tvrtka:
    - HEAD/kraj/info – prima informaciju za kraj rada poslužitelja PosluziteljTvrtka i svih njegovih dijelova. Šalje GET metodom REST zahtjev na krajnju točku api/tvrtka/kraj/info koja je implementirana u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_klijenti**. Koristi se Microprofile REST Client s konfiguracijskim ključem „klijentTvrtkaInfo".
    - POST/obracun – prima obračun od poslužitelja PosluziteljTvrtka i dodaje novi obračun. Ako je bilo uspješno šalje GET metodom REST zahtjev na krajnju točku api/tvrtka/obracun/ws koja je implementirana u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_klijenti**. Koristi se Microprofile REST Client s konfiguracijskim ključem „klijentTvrtkaInfo".

Maven modul **{LDAP_korisnik}_vjezba_08_dz_3_klijenti** sastoji se od 3 dijela:

1. korisničko sučelje za rad s aplikacijom
2. RESTful servis za primalje informacija
3. WebSocket servis za komunikaciju prema korisničkom sučelju.

U korisničkom sučelju postoji klasa GlobalniPodaci anotirana s @ApplicationScoped u kojoj se pohranjuju sljedeći podaci/atributi i metode:

1. ukupan broj primljenih obračuna (private int brojObracuna)
2. broj otvorenih narudžbi pojedinog partnera. Kod kreiranje nove narudžbe povećava se broj za 1. Nakon plaćana računa smanjuje se broj za 1.
   (private Map<Integer, Integer> brojOtvorenihNarudzbi = new ConcurrentHashMap<>();
3. broj plaćenih računa pojedinog partnera. Nakon plaćana računa povećava se broj za 1.
   (private Map<Integer, Integer> brojRacuna = new ConcurrentHashMap<>();
4. get/set metode
5. metode za povećavanje/smanjivanje za 1.

Korisničko sučelje podijeljeno je u dva područja:

1. za komunikaciju s tvrtkom i realizira se s Jakarta MVC
2. za komunikaciju s partnerom i realizira se s Jakarta Faces.

U oba područja korisničkog sučelja postoje dijelovi koji su:

1. javni i nije potrebno da je korisnik prijavljen
2. privatni za koje je potrebno da je korisnik prijavljen, a oni mogu biti:
   - za sve prijavljene korisnike neovisno o grupi kojoj pripadaju (nwtis, admin)
   - samo za članove grupe admin.

Kontrola pristupa provodi se pomoću @FormAuthenticationMechanismDefinition. Prijavljivanje korisnika provodi se zajedničko u dijelu s Jakarta Faces. Privatni i administracijski dijelovi sa zaštićenim pristupom definiraju se u web.xml na sljedeći način:

1. dio s Jakarta MVC definira se putem putanje u klasi Kontroler:
   a. privatni dio: /mvc/tvrtka/privatno/*
   b. administracijski dio: /mvc/tvrtka/admin/*
2. dio s Jakarta Faces temelji se na smještaju pogleda u direktorije/mape:
   a. privatni dio: /privatno/
   b. administracijski dio: /admin/

U dijelu s Jakarta MVC potrebno je realizirati sljedeće poglede:

- javni dio:
  - provjera rada poslužitelja (koristi REST api/tvrtka metodu HEAD)
  - pregled naziva partnera/restorana (koristi REST api/tvrtka metodu GET/partner). Odabirom partnera poziva se pogled za pregled odabranog partnera/restorana (koristi REST api/tvrtka metodu GET/partner/{id}).
- privatni dio:
  - pregled obračuna u razdoblju uz odabir razdoblja od-do i odabir:
    - jelo i piće (koristi REST api/tvrtka metodu GET/obracun[?od=vrijemeOd&do=vrijemeDo])
    - jelo (koristi REST api/tvrtka metodu GET/obracun/jelo[?od=vrijemeOd&do=vrijemeDo])
    - piće (koristi REST api/tvrtka metodu GET/obracun/pice[?od=vrijemeOd&do=vrijemeDo])
  - pregled obračuna u razdoblju uz odabir partnera i razdoblja od-do (koristi REST api/tvrtka metodu GET/obracun//{id}[?od=vrijemeOd&do=vrijemeDo])
- administracijski dio:
  - dodaje novog partnera (koristi REST api/tvrtka metodu POST/partner). U obrascu se unose potrebni podaci za partnera (ne kao json).
  - aktivira spavanje (koristi REST api/tvrtka metodu GET/spava&vrijeme=trajanje). U obrascu se unos vrijeme spavanja.
  - konzola za upravljanje poslužiteljem Tvrtka. Ona sadrži:
    - informacije o statusu dijelova poslužitelja Tvrtka (koristi REST api/tvrtka metodu HEAD/status/{id})
    - gumbe za aktiviranje pauze odabranog dijela poslužitelja Tvrtka (koristi REST api/tvrtka metodu HEAD/pauza/{id})
    - gumbe za aktiviranje izlaza iz pauze odabranog dijela poslužitelja Tvrtka (koristi REST api/tvrtka metodu HEAD/start/{id})
    - gumb za aktiviranje kraja rada poslužitelja Tvrtka (koristi REST api/tvrtka metodu HEAD/kraj)
    - informaciju o statusu rada poslužitelja Tvrtka (osvježava se pomoći WebSocket poruke na /ws/tvrtka). Ako radi prikazuje se zelenom bojom, a ako ne radi onda crvenom bojom.
    - informaciju o broju primljenih obračuna kod poslužitelja Tvrtka (osvježava se pomoći WebSocket poruke na /ws/tvrtka)
    - prostor za objavu interne poruke (osvježava se pomoći WebSocket poruke na /ws/tvrtka)
    - obrazac za slanje interne poruke (pomoću WebSocket poruke na /ws/tvrtka).

WebSocket poruka na /ws/tvrtka je u tekstualnom obliku i sadrži 3 dijela odvojena znakom „;“:

1. status poslužitelja tvrtka (može biti „RADI“ ili „NE RADI“)
2. broj obračuna kod poslužitelja Tvrtka koji preuzima iz objekta klase GlobalniPodaci.
3. tekst interne poruke ako se šalje interna poruka inače ima vrijednost „“.

Npr:

- Šalje se interna poruka - RADI;201;Danas je petak
- Ne šalje se interna poruka - RADI;202;

U dijelu s Jakarta Faces potrebno je realizirati sljedeće poglede:

- javni dio:
  - provjera rada poslužitelja (koristi REST api/partner metodu HEAD)
  - pregled naziva partnera/restorana ((iz baze podataka). Odabirom partnera poziva se pogled za pregled odabranog partnera/restorana (iz baze podataka)
  - dodavanje novog korisnika (koristi REST api/partner metodu POST/korisnik). U obrascu se unose potrebni podaci za korisnika (ne kao json).
- privatni dio:
  - nevidljiva prijava korisnika na bazi kontejnera (dodaje zapis u tablicu „zapisi")
  - odjava korisnika (dodaje zapis u tablicu „zapisi")
  - odabir partnera/restorana (samo ako ne postoji aktivna narudžba) u kojem će se raditi narudžba
  - pregled jelovnika odabranog partnera/restorana (samo ako postoji odabrani partner) (koristi REST api/partner metodu GET/jelovnik).
  - pregled karte pića odabranog partnera/restorana (samo ako postoji odabrani partner) (koristi REST api/partner metodu GET/kartapica).
  - nova narudžba (samo ako ne postoji aktivna narudžba). (koristi REST api/partner metodu POST/narudzba i ako je u redu dodaje zapis u tablicu „zapisi"). Sadrži dva bloka. Jedan ispod drugog. Prvi blok sadrži obrazac u kojem postoje tri dijela jedan ispod drugog. Gornji dio ima gumb za plaćanje narudžbe (koristi REST api/partner metodu POST/racun i ako je u redu dodaje zapis u tablicu „zapisi"). Srednji dio ima izbornik s jelovnikom s odabirom jednog elementa, unos količine u rasponu 1-100 i gumb za naručivanje jela (koristi REST api/partner metodu POST/jelo). Donji dio ima izbornik s kartom pića s odabirom jednog elementa, unos količine u rasponu 1-100 i gumb za naručivanje pića (koristi REST api/partner metodu POST/pice). U drugom bloku postoji lijevi i desni dio. U lijevom se prikazuju naručena jela u obliku tablice (naziv, količina, cijena, iznos). U desnom se prikazuju naručena pića u obliku tablice (naziv, količina, cijena, iznos).
  - dopuna narudžbe (samo ako postoji aktivna narudžba). (za stavke otvorene narudžbe koristi REST api/partner metodu GET/narudzba). Isto kao i kod nove narudžbe.
  - plaćanje narudžbe (samo ako postoji aktivna narudžba). (koristi REST api/partner metodu POST/racun i ako je u redu dodaje zapis u tablicu „zapisi").
- administracijski dio:
  - aktivira spavanje (koristi REST api/partner metodu GET/spava&vrijeme=trajanje). U obrascu se unos vrijeme spavanja
  - pregled korisnika s pretraživanjem po prezimenu i imenu (iz baze podataka)
  - pregled ime i prezimena korisnika (koristi REST api/partner metodu GET/korisnik). Odabirom korisnika poziva se pogled za pregled odabranog korisnika (koristi REST api/ partner metodu GET/ korisnik/{id}).
  - pregled obračuna u razdoblju uz odabir partnera i razdoblja od-do (iz baze podataka)
  - pregled rada uz odabir korisnika i razdoblja od-do (iz baze podataka)

- konzola za upravljanje poslužiteljem Partner. Ona sadrži:
  - informacije o statusu dijelova poslužitelja Partner (koristi REST api/partner metodu HEAD/status/{id})
  - gumbe za aktiviranje pauze odabranog dijela poslužitelja Partner (koristi REST api/partner metodu HEAD/pauza/{id})
  - gumbe za aktiviranje izlaza iz pauze odabranog dijela poslužitelja Partner (koristi REST api/partner metodu HEAD/start/{id})
  - gumb za aktiviranje kraja rada poslužitelja Partner (koristi REST api/ partner metodu HEAD/kraj)
  - informaciju o statusu rada poslužitelja Partner (osvježava se pomoći WebSocket poruke na /ws/partneri). Ako radi prikazuje se zelenom bojom, a ako ne radi onda crvenom bojom.
  - informaciju o broju otvorenih narudžbi kod poslužitelja Partner (osvježava se pomoći WebSocket poruke na /ws/partneri)
  - informaciju o broju plaćenih računa kod poslužitelja Partner (osvježava se pomoći WebSocket poruke na /ws/partneri).

WebSocket poruka na /ws/partneri je u tekstualnom obliku i sadrži 3 dijela odvojena znakom „;“:

1. status poslužitelja tvrtka (može biti „RADI“ ili „NE RADI“)
2. broj otvorenih narudžbi kod poslužitelja Partner koji preuzima iz objekta klase GlobalniPodaci
3. broj plaćenih kod poslužitelja Partner koji preuzima iz objekta klase GlobalniPodaci.

Npr:

- RADI;1;0
- RADI;0;1
- …
- RADI;5;10
- RADI;4;11
- RADI;3;12
- RADI;2;13
- RADI;1;14
- RADI;0;15
- …
- RADI;1;100
- RADI;0;101

Rad s JPA smješten je u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_JPA** i temelji se na EclipseLink kao pružatelju perzistentnosti. Sastoji se od:

- klasa entiteta
- klasa fasada/pomoćnika za rad s entitetima primjenom Criteria API i za pretvaranje objekta u entitet i obratno.

U Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_klijenti** rad s bazom podataka provodi se putem JPA uz korištenje Maven modula **{LDAP_korisnik}_vjezba_08_dz_3_JPA**. To znači da će u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_klijenti** biti potrebno obrisati ovisnost na Maven modul **{LDAP_korisnik}_vjezba_08_dz_3_lib_rest** i dodati ovisnost na Maven modul **{LDAP_korisnik}_vjezba_08_dz_3_JPA**. A zatim se sve aktivnosti s bazom podataka realiziraju putem JPA.

Dodatni bodovi mogu se dobiti ako su realizirane sve tražene funkcionalnosti putem JPA u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_klijenti** i ako se i u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_servisi** rad s bazom podataka provodi putem JPA uz korištenje Maven modula **{LDAP_korisnik}_vjezba_08_dz_3_JPA**. To znači da će u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_servisi** biti potrebno obrisati ovisnost na Maven modul **{LDAP_korisnik}_vjezba_08_dz_3_lib_rest** i dodati ovisnost na Maven modul **{LDAP_korisnik}_vjezba_08_dz_3_JPA**. A zatim se sve aktivnosti s bazom podataka realiziraju putem JPA.

RESTful web servisi u Maven modulu **{LDAP_korisnik}_vjezba_08_dz_3_klijenti**:

- api/tvrtka – pokriva prijem poruka od drugih REST web servisa.
  - GET/kraj/info – prima informaciju za kraj rada poslužitelja PosluziteljTvrtka i svih njegovih dijelova. Šalje WebSocket poruku na /ws/tvrtka.
  - GET/obracun/ws – prima informaciju da je stigao novi obračun. U objektu klase GlobalniPodaci povećava broj primljenih obračuna. Šalje WebSocket poruku na /ws/tvrtka.

Shematski prikaz spomenutih krajnjih točaka RESTful web servisa i WebSocket nalazi se na slici s oznakom Slika 2.

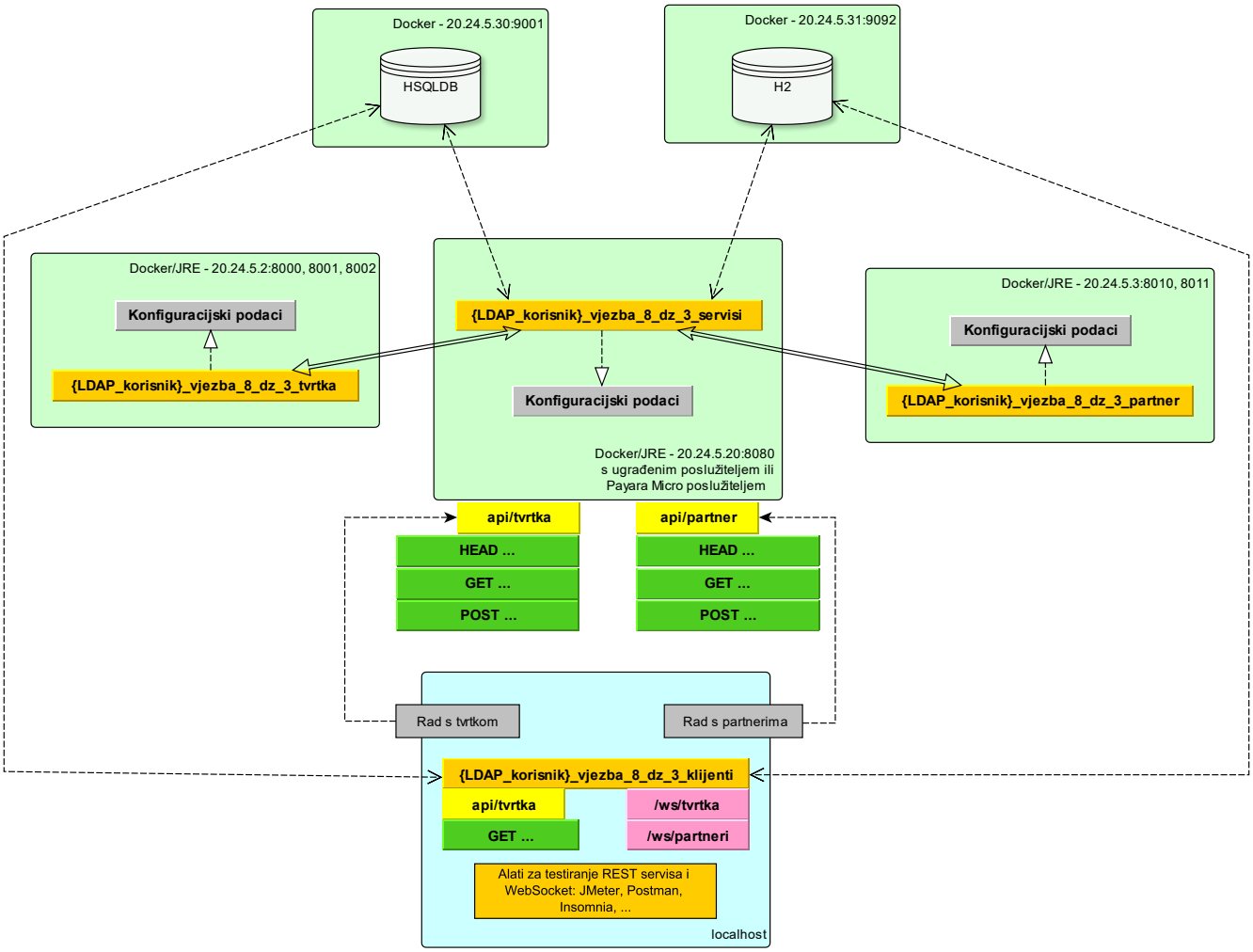
Sve metode RESTful web servisa moraju:

- kod vraćanja koristiti klasu Response. Ako vraćaju podatke oni moraju biti u obliku application/json i moraju se temeljiti na jednom od raspoloživih zapisa
- kod statusa treba odgovarati HTTP statusima
- kod POST metode primati podatke u obliku application/json i moraju se temeljiti na jednom od raspoloživih zapisa.

Svaka od komponenti sustava (poslužitelji baza podataka, poslužitelji na mrežnoj utičnici i poslužitelj na RESTful web servise) mora biti u vlastitom Docker kontejneru prema instalacijskoj arhitekturi sustava koja se nalazi se na slici s oznakom Slika 2. Svi Docker kontejneri moraju biti povezani na istu Docker mrežu pod nazivom mreza_{LDAP_korisnik} i integrirani u Docker compose. U compose.yaml treba podesiti da je naziv {LDAP_korisnik}_docker_compose, da se spaja na postojeću vanjsku mrežu i da se koristi postojeći vanjski svezak/volumen.

Maven moduli {LDAP_korisnik}_vjezba_08_dz_3_tvrtka i {LDAP_korisnik}_vjezba_08_dz_3_partner moraju biti podešeni da kreiraju prilagođenu Java sliku tj. da su pakirani kao jlink.

Docker kontejneri za Maven module {LDAP_korisnik}_vjezba_08_dz_3_tvrtka i {LDAP_korisnik}_vjezba_08_dz_3_partner moraju koristiti Docker svezak/volume pod nazivom svezak_{LDAP_korisnik} koji se povezuje na /usr/app/podaci. Pokretanje aplikacija u Docker kontejneru za ta dva modula obavlja se s direktorija /usr/app/podaci. Spajanje na poslužitelje u Docker kontejnerima mora se obavljati putem zadane statičke adrese pojedinog Docker kontejnera na prikazanoj slici.



Slika 2. Instalacijska arhitektura sustava, metoda RESTful web servisa i WebSocket