
Vježba: 4 – Zadaća 1. Telemetrija e-vozila: Sustav za praćenje i analizu vožnje e-vozilom s 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). 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_app
              pom.xml
              src
                     main
                            ...
                     test
                            ...
       {LDAP_korisnik}_vjezba_04_dz_1_lib
              pom.xml
              src
                     main
                            ...
                     test
                            ...
```

Opis rada sustava:

Sustav je zamišljen na simulira vožnju e-vozilom po nekom gradu. Simulacija vožnje e-vozila temelji se na stvarnim podacima koji su zapisani u csv datotekama. Datoteka podataka za vožnju e-vozila sadrži podatke u redcima, a nazivi njihovih stupaca nalaze se u 1. retku u datoteci:

1. Milliseconds since Epoch
2. Speed
3. Watt
4. Ampere
5. Altitude
6. GPS Speed
7. Vehicle Body Temperature
8. Battery Percentage
9. Battery Voltage
10. Battery Capacity
11. Battery Temperature
12. Remaining Mileage
13. Total Mileage
14. Latitude
15. Longitude

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Milliseconds since Epoch	Speed	Watt	Ampere	Altitude	GPS Speed	Vehicle Body Temperature	Battery Percentage	Battery Voltage	Battery Capacity	Battery Temperature	Remaining Mileage	Total Mileage	Latitude	Longitude
2	1708073749078	0.02	0.8086	0.02	214.2	1.337297	19	93	40.43	7314	20	27.9	816.458	46.286644	16.35285
3	1708073749394	0.358	0.8086	0.02	214.2	1.337297	19	93	40.43	7314	20	27.9	816.458	46.286644	16.35285
4	1708073749756	2.149	0.8086	0.02	214.2	1.337297	19	93	40.43	7314	20	27.9	816.458	46.286644	16.35285
5	1708073750092	0.947	0.8086	0.02	214.2	1.337297	19	93	40.43	7314	20	27.9	816.458	46.286644	16.35285
6	1708073750654	0.575	0.8086	0.02	214.2	1.337297	19	93	40.43	7314	20	27.9	816.458	46.286644	16.35285
7	1708073750969	0	0.8086	0.02	214.2	1.337297	19	93	40.43	7314	20	27.9	816.458	46.286644	16.35285

Slika 1. Podaci vožnje e-vozila u tabličnom kalkulatoru

Podaci u datoteci odgovaraju zapisu PodaciVozila uz dva dodana podatka (id i broj) . Potrebno je obraditi pažnju na podatke tipa double jer se mogu javiti cjelobrojne vrijednosti u tim stupcima. Npr. u 7. retku u B stupcu pod nazivom Speed nalazi se vrijednosti 0 (nema decimalnu točku). To se posebno odnosi na poslužiteljske klase (i njihove radnike) koje primaju podatke o vozilima te trebaju ispravnost primljenih podataka. Treba predvidjeti da podatak kod tipa double može biti sa ali i bez decimalne točke i da treba u oba slučaja ispravno preuzeti vrijednost.

Podaci u datoteci bilježe se u vrlo kratkim intervalima (manje od 1 sek) tako da se može pratili vožnja e-vozilom pa određenom putanji. Podaci u datoteci tretiraju se kao „živi" podaci koje e-vozilo (tj. simulator vožnje) šalje podatke svakog pojedinog retka, jedan po jedan, prema sustavu za telemetriju i kontrolu brzine. Za svaki redak utvrđuje se razlika između njegovog vremena i vremena prethodnog retka. Ta razlika vremena množi se sa korekcijom vremena, koja se dobije dijeljenjem postavke trajanjeSek s 1000.0. Postavka trajanjeSek sadrži broj milisekundi. Dobivena vrijednost je argument za funkciju Thread.sleep kojom dretva spava određeno vrijeme i nakon toga nastavlja s radom na podacima iz retka i zatim spava koliko iznosi postavka trajanjePauze. Ako je vrijednost za trajanjeSek manje od 1000 to znači da će simulacija raditi brže nego u stvarnosti. Npr. ako trajanjeSek ima vrijednost 10 to znači da će simulacija raditi gotovo 100 puta brže od stvarnosti. Npr. ako trajanjeSek ima vrijednost 10000 to znači da će simulacija raditi gotovo 10 puta sporije od stvarnosti.

Osim e-vozila, kojim se vozi po gradu, postoje i radari (mjerači brzine e-vozila). Zbog sigurne vožnje e-vozilo ne smije voziti brže nego što je dopuštena brzina u području pojedinog radara (određeno u m) i to

dulje od dopuštenog vremena brže vožnje. Radar ima svoje GPS koordinate. U svakom retku u datoteci zapisane su GPS koordinate e-vozila u tom trenutku. Ako je e-vozilo udaljeno od radara manje nego što je njegovo područje tada radar obrađuje podatke o e-vozilu. Npr. primjer 1: kod radara dopuštena brzina iznosi 20 km/h i maksimalno 7 sekundi. Pretpostavimo da e-vozilo dolazi u područje radara u retku 100 s brzinom od 18 km/h, koja je manja od dopuštene brzine. Radar ne reagira na e-vozilo. U sljedećem retku 101 brzina je 22 km/h i to je veća brzina od dopuštene. Radar sprema podatke o e-vozilu (zapis BrzoVozilo u kojem postoji i vrijeme). U sljedećim redcima e-vozilo i dalje vozi brže od dopuštene brzine. Radar utvrđuje vrijeme brze vožnje tako da od vremena iz retka oduzme spremljeno vrijeme. Ako vrijeme brze vožnje traje 2 puta dulje od dopuštenog vremena brže vožnje, smatra se da je došlo do neočekivane situacije zbog čega treba poništiti spremljene podatke o e-vozilu. To se radi tako da se spremaju novi podaci o e-vozilu uz status false, što znači da to nisu podaci za brzu vožnju. Ako vrijeme brze vožnje traje dulje od dopuštenog vremena brže vožnje, smatra se da treba generirati kaznu zbog brze vožnje. Opet se poništavaju spremljeni podaci o e-vozilu. Npr. primjer 2: kod radara dopuštena brzina iznosi 20 km/h i maksimalno 7 sekundi. Pretpostavimo da e-vozilo dolazi u područje radara u retku 200 s brzinom od 18 km/h, koja je manja od dopuštene brzine. Radar ne reagira na e-vozilo. U sljedećem retku 201 brzina je 22 km/h i to je veća brzina od dopuštene. Radar sprema podatke o e-vozilu (zapis BrzoVozilo u kojem postoji i vrijeme). U sljedećim retku 202 brzina je 15 km/h i ona je manja od dopuštene brzine. Radar utvrđuje da se radi o dozvoljenoj vožnji jer se nije prekoračilo vrijeme brze vožnje. Poništavaju se spremljeni podaci o e-vozilu.

Sustav (slika 2) se sastoji od 3 poslužiteljska programa: CentralniSustav, PosluziteljRadara i PosluziteljKazni. Program CentralniSustav zadužen je za pokretanje dvaju poslužitelja na mrežnim vratima: PosluziteljZaRegistracijuRadara i PosluziteljZaVozila. PosluziteljZaRegistracijuRadara otvara mrežnu utičnicu na zadanim mrežnim vratima i radi u jednodretvenon načinu sa sinkronim slanjem/primanjem poruka rade te prima komande (registraciju ili brisanje) koje se odnose na pojedini radar. U sustavu može postojati više radara.

Program PosluziteljRadara započinje rad svojom registracijom. Ona se provodi slanjem komande na poslužitelj PosluziteljZaRegistracijuRadara. Nakon uspješne registracije otvara mrežnu utičnicu na zadanim mrežnim vratima i radi u višedretvenon načinu sa sinkronim slanjem/primanjem poruka. Za svakog klijenta (RadnikZaVozila) PosluziteljRadara kreira novu virtualnu dretvu koja izvršava objekt klase RadnikZaRadare. U njemu se prima komanda i ispituje se njena ispravnost. Ispravna komanda odlazi na obradu u kojoj se ispituje da li je e-vozilo napravilo prekršaj brze vožnje. Ako je došlo do prekršaja šalje se komanda poslužitelju PosluziteljKazni, koji sprema sve kazne koje su napravila e-vozila.

Program PosluziteljKazni otvara mrežnu utičnicu na zadanim mrežnim vratima i radi u jednodretvenon načinu sa sinkronim slanjem/primanjem poruka. Kada se primi poruka ispituje se njena ispravnost. Ispravna komanda odlazi na obradu u kojoj se spremaju podaci o kazni ili se dohvaća kazna određenog e-vozila u zadanom vremenskom intervalu ili se dohvaća statistika kazni u zadanom vremenskom intervalu.

PosluziteljZaVozila otvara mrežnu utičnicu na zadanim mrežnim vratima i radi u višedretvenom načinu rada s asinkronim slanjem/primanjem poruka putem kanala. PosluziteljZaVozila za svakog klijenta (Simulator-Vozila) kreira novu virtualnu dretvu koja izvršava objekt klase RadnikZaVozila. U tom objektu se prima komanda i ispituje se njena ispravnost. Ispravna komanda odlazi na obradu u kojoj se ispituje da li se e-vozilo nalazi u području pojedinog radara. Ako je u području radara, šalje se komanda poslužitelju PosluziteljRadara. U ovom obliku asinkronog rada objekt RadnikZaVozila ne šalje odgovor svom klijentu.

Za poslužitelja PosluziteljZaVozila ulogu klijenta ima program SimulatorVozila. On se spaja na mrežnu utičnicu od PosluziteljZaVozila, otvara datoteku s podacima za vožnju e-vozila, čita redak po redak, priprema podatke za komandu (dodaje id vozila i redni broj retka) i asinkrono šalje komandu bez da čeka na odgovor.

Za poslužitelja PosluziteljZaKazne ulogu klijenta ima i program Klijent. On se spaja na mrežnu utičnicu od PosluziteljZaKazne, šalje komandu za dohvat kazne zadanog e-vozila i intervala vremena ili broj kazni u zadanom intervalu vremena.

Svaki program koristi za svoj rad vlastitu datoteku s postavkama tako da su podaci izdvojeni od programa i jedan program može se izvršavati s različitim datotekama postavki kako bi se dobili različiti uvjeti rada programa bez da postoji potreba za izmjenom programskog koda i njegovim kompiliranjem.

Svi poslužitelji (i njihovi radnici) kod provjere ispravnosti primljene komande trebaju koristiti dozvoljene izraze (eng. Regular Expression). Ne smije se koristiti cijepanje komande (funkcije split , tokenize i sl).

Za sljedeće klase treba pripremiti jedinične testove: CentralniSustav, RadnikZaVozila, RadnikZaRadare i PosluziteljKazni.

Varijante redoslijeda pokretanja programa:

Br	Varijanta 1	Varijanta 2	Varijanta 3	Varijanta 4
1	CentralniSustav	CentralniSustav	CentralniSustav	CentralniSustav
2	PosluziteljKazni	PosluziteljRadara R1	SimulatorVozila V1	PosluziteljKazni
3	PosluziteljRadara R1	PosluziteljKazni	Klijent V1	PosluziteljRadara R1
4	SimulatorVozila V1	SimulatorVozila V1	PosluziteljRadara R1	PosluziteljRadara R2
5	Klijent V1	Klijent V1	PosluziteljKazni	SimulatorVozila V1
6			SimulatorVozila V2	SimulatorVozila V2
7			Klijent V1	Klijent V1
8			Klijent V2	Klijent V2

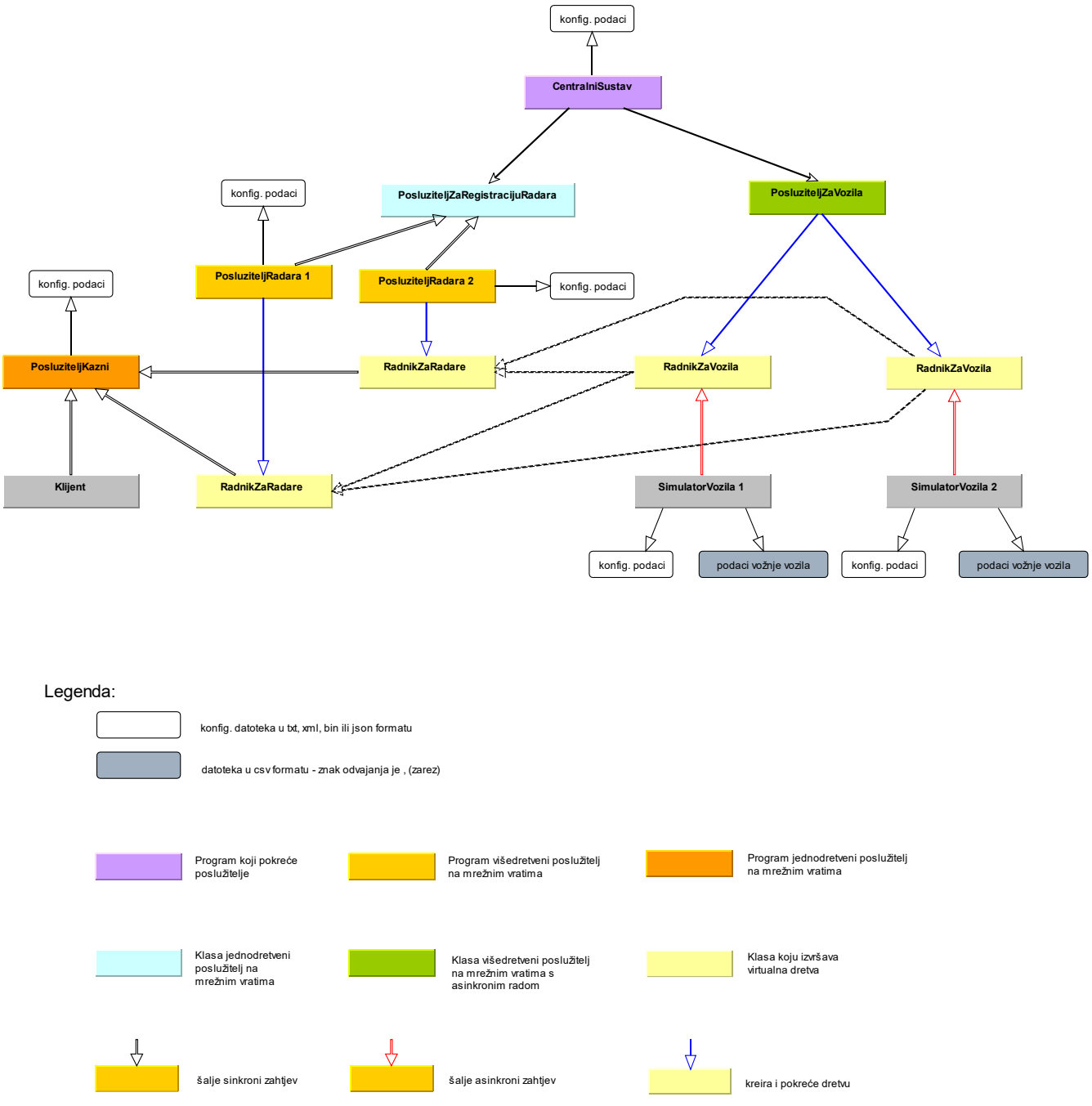
Varijante 1 i 2 su gotovo iste jer se PosluziteljKazni i PosluziteljRadara mogu izvršavati u bilo kojem redoslijedu sve dok se ne pokrene SimulatorVozila.

U varijanti 3 simulacija će započeti, a kako nema radara neće doći do generiranja kazni. Osim ako se ne pokrene PosluziteljRadara za radara R1 prije nego što završi rad SimulatorVozila za vozilo V1. Ako je zatim došlo do kazne, a još nije pokrenut PosluziteljKazni doći će do pogreške jer se PosluziteljRadara R1 ne može spojiti na PosluziteljKazni. Nakon pokretanja PosluziteljKazni može se pokrenuti SimulatorVozila za vozilo V2.

U varijanti 4 pokrenut je PosluziteljRadara za dva radara (R1 i R2) i SimulatorVozila za dva e-vozila (V1 i V2).

Program Klijent nema smisla pokrenuti dok nije pokrenut poslužitelj PosluziteljKazni.

Shema sustava prikazana je na slici.



Slika 2. Shema sustava

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 CentralniSustav, nastavnički PosluziteljRadara i studentski PosluziteljKazni. Zatim se pokrene nastavnički SimulatorVozila i studentski Klijent. Ili bilo koja druga kombinacija studentskih i nastavničkih poslužitelja i klijenata. 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".

Pokretanje programa **CentralniSustav** sadrži jedan argument:

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

Npr.

`NWTiS_DZ1_CS.txt`

CentralniSustav pokreće poslužitelje PosluziteljZaRegistracijuRadara i PosluziteljZaVozila kao dvije zasebne dretve.

Klijent se spaja na **PosluziteljZaRegistracijuRadara** putem mrežne utičnice te šalje komandu (završava s \n) poslužitelju na temelju postavki i traži izvršavanja određene akcije i vraća odgovor (završava s \n)

- **`RADAR id adresa mreznaVrata gpsSirina gpsDuzina maksUdaljenost`**
  - npr. `RADAR 1 localhost 8010 46.29950 16.33001 100`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li su podaci tog zahtjeva spremljeni u memoriji u kolekciji R. Ako nisu, upisuje ih u kolekciju R (tj. registracija radara) i vraća OK.
  - Npr. `OK`

- **`RADAR OBRIŠI id`**
  - npr. `RADAR OBRIŠI 1`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li su podaci tog zahtjeva spremljeni u memoriji u kolekciji R. Ako jesu, briše ih iz kolekcije R (tj. deregistracija radara) i vraća OK.
  - Npr. `OK`

- **`RADAR OBRIŠI SVE`**
  - npr. `RADAR OBRIŠI SVE`
  - Provjera da li ispravni podaci. Ako su ispravni briše sve radare iz kolekcije R (tj. deregistracija svih radara) i vraća OK.
  - Npr. `OK`

Kodovi pogrešaka za poslužitelja za registraciju radara su:

- Kada format komande nije ispravan, vraća odgovor ERROR 10 tekst (tekst objašnjava razlog pogreške)
- Kada već postoji podaci za radar id u kolekciji, vraća odgovor ERROR 11 tekst (tekst objašnjava razlog pogreške).
- Kada ne postoji podaci za radar id u kolekciji, vraća odgovor ERROR 12 tekst (tekst objašnjava razlog pogreške).
- Kada nešto drugo nije u redu vraća odgovor ERROR 19 tekst (tekst objašnjava razlog pogreške).

Klijent se spaja se na **PosluziteljZaVozila** putem mrežne utičnice i šalje komandu (završava s \n) poslužitelju na temelju postavki i traži izvršavanja određene akcije i ne čeka na odgovor:

- **VOZILO id broj vrijeme brzina snaga struja visina gpsBrzina tempVozila postotakBaterija naponBaterija kapacitetBaterija tempBaterija preostaloKm ukupnoKm gpsSirina gpsDuzina**
  - npr. `VOZILO 1 101 1708073749078 0.02 0.8086 0.02 214.2 1.337297 19 93 40.43 7314 20 27.9 816.458 46.286644 16.35285`
  - Provjera da li ispravni podaci. Ako su ispravni provjerava da li se nalazi u dosegu radara. Ako se nalazi tada se poslužitelju PosluziteljRadara šalju podaci o vožnji ~~i vraća OK~~.
  - ~~Npr. OK~~

  Kodovi pogrešaka su:

  - Kada format komande nije ispravan, vraća odgovor ERROR 20 tekst (tekst objašnjava razlog pogreške)
  - Kada nešto drugo nije u redu vraća odgovor ERROR 29 tekst (tekst objašnjava razlog pogreške).

Pokretanje programa **PosluziteljRadara** sadrži jedan argument ili tri argumenta:

```
datoteka.(txt | xml | bin | json)
datoteka.(txt | xml | bin | json) OBRIŠI id
datoteka.(txt | xml | bin | json) OBRIŠI SVE
```

Npr.

```
NWTiS_DZ1_R1.txt
NWTiS_DZ1_R1.txt OBRIŠI 1
NWTiS_DZ1_R1.txt OBRIŠI SVE
```

Ako je jedan argument tada se radar registrira PosluziteljZaRegistracijuRadara i otvara se mrežna utičnica na zadanim mrežnim vratima te se čeka na klijente. Ako su tri argumenta pri čemu je 3. argument cijeli broj, tada se briše radar s tim id. Inače se brišu svi radari.

Klijent se spaja na PosluziteljRadara putem mrežne utičnice i šalje komandu (završava s \n) poslužitelju na temelju postavki i traži izvršavanja određene akcije i vraća odgovor (završava s \n):

- **VOZILO id vrijeme brzina gpsSirina gpsDuzina**
  - npr. `VOZILO 1 1711348009 21.767 46.286602 16.353136`
  - Provjera da li ispravni podaci. Ako su ispravni i ako je brzina veća od maksimalno dozvoljene brzine (maksBrzina) za radar tada rada aktivira praćenje e-vozila. U praćenju ako je brzina e-vozila veća od maksimalno dozvoljene brzine te dulje od maksimalnog dozvoljenog vremena (maksTrajanje), tada radar šalje poslužitelju PosluživateljKazni poruku s podacima o prekršaju brzine vožnje od e-vozila. Ako u praćenju brzina e-vozila padne ispod maksimalno dozvoljene brzine prekida se praćenje e-vozila i vraća OK.
  - Npr. `OK`

  Kodovi pogrešaka su:

  - Kada format komande nije ispravan, vraća odgovor ERROR 30 tekst (tekst objašnjava razlog pogreške)
  - Kada PosluživateljKazni nije aktivan, vraća odgovor ERROR 31 tekst (tekst objašnjava razlog pogreške).
  - Kada nešto drugo nije u redu vraća odgovor ERROR 39 tekst (tekst objašnjava razlog pogreške).

Pokretanje programa **PosluziteljKazni** sadrži jedan argument:

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

Npr.

`NWTiS_DZ1_PK.txt`

Otvara se mrežna utičnica na zadanim mrežnim vratima te se čeka na klijente.

Klijent se spaja na PosluziteljKazni putem mrežne utičnice i šalje komandu (završava s \n) poslužitelju na temelju postavki i traži izvršavanja određene akcije i vraća odgovor (završava s \n):

- **`VOZILO id vrijemePocetak vrijemeKraj brzina gpsSirina gpsDuzina gpsSirinaRadara gpsDuzinaRadara`**
  - npr. `VOZILO 1 1711348009 1711368009 21.767 46.286608 16.353131 46.286602 16.353136`
  - Provjera da li ispravni podaci. Ako su ispravni u evidenciju kazni upisuje podatke za e-vozilo, ispisuje na ekran podatke o kazni i vraća OK.
  - Npr. `OK`

- **`VOZILO id vrijemeOd vrijemeDo`**
  - npr. `VOZILO 1 1711348009 1711355209`
  - Provjera da li ispravni podaci. Ako su ispravni u evidenciji kazni traži podatke o e-vozilu unutar zadanog vremena od-do. Ako postoje kazne vraća zadnju (najsvježiju) kaznu OK vrijeme brzina gpsSirinaRadar gpsDuzinaRadar.
  - Npr. `OK 1711348009 21.767 46.286602 16.353136`

- **`STATISTIKA vrijemeOd vrijemeDo`**
  - npr. `STATISTIKA 1711348009 1711355209`
  - Provjera da li ispravni podaci. Ako su ispravni u evidenciji prekršaja traži podatke o broju kazni unutar zadanog vremena od-do. Ako postoje kazne vraća OK idVozilo brojKazni; idVozilo brojKazni; idVozilo brojKazni;...
  - Npr. `OK 1 0; 2 7; 3 2; 4 5;`

Kodovi pogrešaka su:

- Kada format komande nije ispravan, vraća odgovor ERROR 40 tekst (tekst objašnjava razlog pogreške)
- Kada e-vozilo id nema kazne u zadanom vremenu, vraća odgovor ERROR 41 tekst (tekst objašnjava razlog pogreške).
- Kada nešto drugo nije u redu vraća odgovor ERROR 49 tekst (tekst objašnjava razlog pogreške).

Pokretanje programa **SimulacijaVozila** sadrži tri argumenta:

```
datoteka1.(txt | xml | bin | json) datoteka2.csv id
```

Npr.

```
NWTiS_DZ1_SV.txt NWTiS_DZ1_V1.csv 1
```

Putem mrežne utičnice spaja se na PosluziteljZaVozila i asinkrono šalje komanda poslužitelju putem kanala na temelju postavki i traži izvršavanja određene akcije. Ne čeka odgovor od poslužitelja kod slanja podataka.

Pokretanje programa **Klijent** sadrži tri ili dva argumenta:

```
datoteka.(txt | xml | bin | json) id vrijemeOd vrijemeDo

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

Npr.

```
NWTiS_DZ1_K.txt 1 1708073749078 1708074766471

NWTiS_DZ1_K.txt 1708073749078 1708074766471
```

Putem mrežne utičnice spaja se na PosluziteljKazni i sinkrono šalje komandu poslužitelju na temelju postavki i traži izvršavanja određene akcije.

Vježba: 7 – Zadaća 2. Telemetrija e-vozila: Sustav za praćenje i analizu vožnje e-vozilom s više poslužitelja na mrežnoj utičnici i RESTful web servisima

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.

`{LDAP_korisnik}_vjezba_04_dz_1_app`	`{LDAP_korisnik}_vjezba_07_dz_2_app`	1.0.0	1.1.0
`{LDAP_korisnik}_vjezba_04_dz_1_lib`	`{LDAP_korisnik}_vjezba_07_dz_2_lib_konfig`	1.1.0	1.4.0

Obavezno na svakom Maven modulu obrisati direktorij .settings 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:

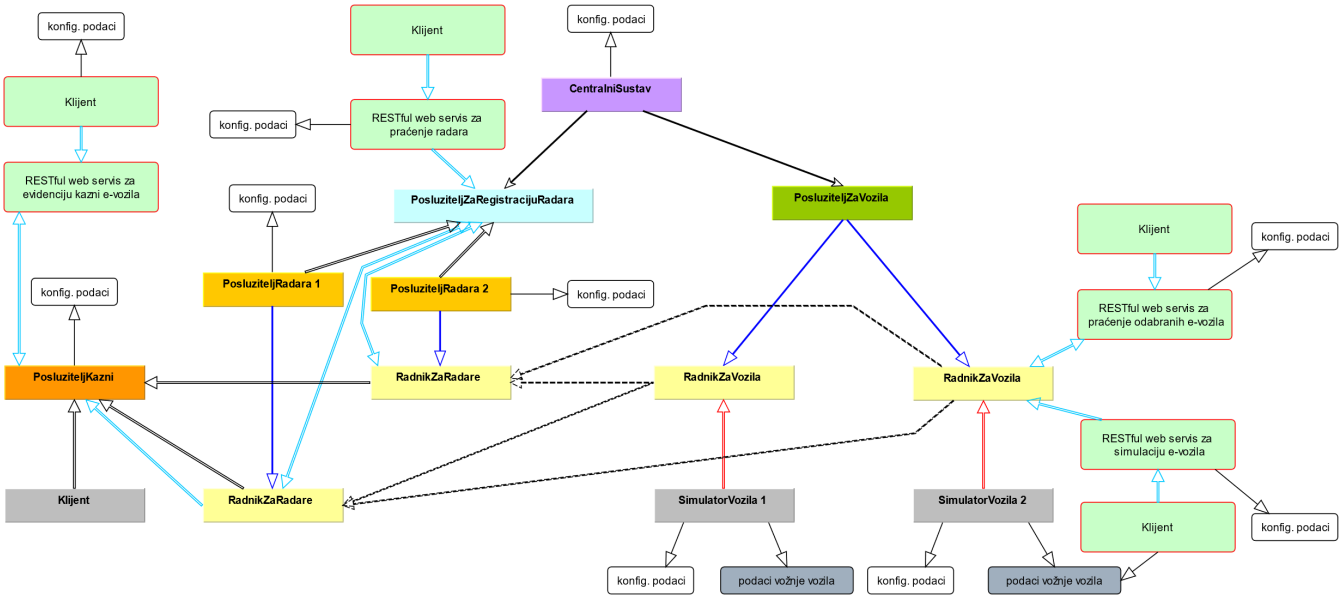
```
{LDAP_korisnik}_vjezba_07_dz_2
        {LDAP_korisnik}_vjezba_07_dz_2.pdf
        {LDAP_korisnik}_vjezba_07_dz_2_app
        {LDAP_korisnik}_vjezba_07_dz_2_lib_konfig
        {LDAP_korisnik}_vjezba_07_dz_2_lib
        {LDAP_korisnik}_vjezba_07_dz_2_lib_rest
        {LDAP_korisnik}_vjezba_07_dz_2_servisi
        {LDAP_korisnik}_vjezba_07_dz_2_klijenti
```

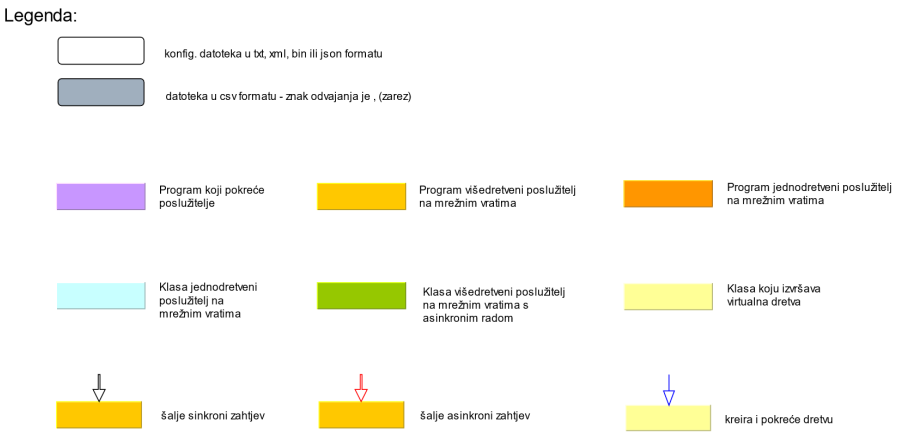
Naziv Maven modula	Uloga Maven modula i kako je nastao	Verzija
`{LDAP_korisnik}_vjezba_07_dz_2_app`	Aplikacija za rad na mrežnoj utičnici. Nastala iz **`{LDAP_korisnik}_vjezba_04_dz_1_app.`** Obrisani paketi podaci i pomocnici. Dodana ovisnost na **`{LDAP_korisnik}_vjezba_07_dz_2_lib`**	1.1.0
`{LDAP_korisnik}_vjezba_07_dz_2_lib_konfig`	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`	Knjižnica klasa za rad podacima. Nastala izdvajanjem paketa podaci i pomocnici iz **`{LDAP_korisnik}_vjezba_07_dz_2_app`**	1.0.0
`{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
`{LDAP_korisnik}_vjezba_07_dz_2_klijenti`	Web aplikacija podržana korištenjem Jakarta MVC s klijentima RESTful web servisa 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. Određeni poslužitelji dobit će dodatne funkcionalnosti putem kojih će se integrirati u rješenje.

Shema sustava prikazana je na slici.





Slika 1. Shema sustava

PosluziteljZaRegistracijuRadara ima dodatne komande:

- **`RADAR id`**
  - npr. `RADAR 1`
  - Provjera da li su ispravni podaci. Ako su ispravni, provjerava da li su postoji radar sa zadanim id u kolekciji R. Ako postoji vraća OK.
  - Npr. `OK`

- **`RADAR RESET`**
  - npr. `RADAR RESET`
  - Provjera da li ispravni podaci. Ako su ispravni, provjerava za svaki radar u kolekciji R da li je aktivan slanjem komande RADAR id tom radaru, gdje je id identifikator radara. Ako Radar nije aktivan, briše ga iz kolekcije R (deregistracija radara). Vraća OK n m. Broj n označava ukupan broj radara u trenutku primanja zahtjeva, a broj m označava broj radara koji nisu bili aktivni i obrisani su iz kolekcija R.
  - Npr. `OK 0 0`
  - Npr. `OK 3 1`

- **`RADAR SVI`**
  - npr. `RADAR SVI`
  - Provjera da li ispravni podaci. Ako su ispravni, prolazi po kolekciji R i za svaki radar priprema podatke. Vraća OK {`[id1 adresa1 mreznaVrata1 gpsSirina1 gpsDuzina1 maksUdaljenost1], [id2 adresa2 mreznaVrata2 gpsSirina2 gpsDuzina2 maksUdaljenost2]...}`
  - Npr. `OK {}`
  - Npr. `OK {[1 localhost 8010 46.29950 16.33001 100]}`
  - Npr. `OK {[1 localhost 8010 46.29950 16.33001 100], [2 localhost 8011 46.28750 16.34201 123], [3 localhost 8012 46.29250 16.322201 300]}`

PosluziteljZaVozila ima dodatne komande:

- **`VOZILO START id`**
  - npr. `VOZILO START 1`
  - Provjera da li ispravni podaci. Ako su ispravni, provjerava postoji li e-vozilo s id u kolekciji e-vozila čiji se podaci o vožnji šalju na RESTful web servis za praćenje odabranih e-vozila. Ako ne postoji, dodaje e-vozilo s id u kolekciju e-vozila čiji se podaci o vožnji šalju na RESTful web servis za praćenje odabranih e-vozila. Vraća OK.
  - Npr. OK

- **`VOZILO STOP id`**
  - npr. `VOZILO STOP 1`
  - Provjera da li ispravni podaci. Ako su ispravni, provjerava postoji li e-vozilo s id u kolekciji e-vozila čiji se podaci o vožnji šalju na RESTful web servis za praćenje odabranih e-vozila. Ako postoji, briše e-vozilo s id iz kolekcije e-vozila čiji se podaci o vožnji šalju na RESTful web servis za praćenje odabranih e-vozila. Vraća OK.
  - Npr. OK

PosluziteljZaVozila ima dopunu komande:

- **`VOZILO id broj vrijeme brzina snaga struja visina gpsBrzina tempVozila postotakBaterija naponBaterija kapacitetBaterija tempBaterija preostaloKm ukupnoKm gpsSirina gpsDuzina`**
  - npr. `VOZILO 1 101 1708073749078 0.02 0.8086 0.02 214.2 1.337297 19 93 40.43 7314 20 27.9 816.458 46.286644 16.35285`
  - Provjera da li ispravni podaci. Ako su ispravni, provjerava postoji li e-vozilo s id u kolekciji e-vozila čiji se podaci o vožnji šalju na RESTful web servis za praćenje odabranih e-vozila. Ako postoji, šalje podatke POST metodom na RESTful web servis za praćenje odabranih e-vozila Ako je slanje bilo u redu ili e-vozilo nije u kolekciji e-vozila čiji se podaci o vožnji šalju na RESTful web servis za praćenje odabranih e-vozila, provjerava da li se e-vozilo nalazi u dosegu radara. Ako se nalazi, tada se poslužitelju PosluziteljRadara šalju podaci o vožnji.

Dodatni kodovi pogrešaka su:

- Kada POST metoda na RESTful web servis za dodavanje vožnje praćenih e-vozila nije uspješno obavljena, vraća odgovor ERROR 21 tekst (tekst objašnjava razlog pogreške)

PosluziteljRadara ima dodatne komande:

- **`RADAR RESET`**
  - npr. `RADAR RESET`
  - Provjera da li su ispravni podaci. Ako su ispravni, slanjem komande RADAR id na PosluziteljZaRegistracijuRadara provjerava ispravnost podataka, gdje je id njegov identifikator. Ako kao odgovor dobije OK znači da je sve u redu i vraća OK. Ako kao odgovor dobije ERROR 12 tada ponavlja registraciju i ako je odgovor u redu vraća OK.
  - Npr. `OK`

- **`RADAR id`**
  - npr. `RADAR 1`
  - Provjera da li su ispravni podaci. Ako su ispravni, provjerava odgovara li id njegovom identifikatoru. Ako odgovara, slanjem komande TEST na PosluziteljKazni provjerava njegovu aktivnost. Ako kao odgovor dobije OK znači da je sve u redu i vraća OK.
  - Npr. `OK`

Dodatni kodovi pogrešaka su:

- Kada PosluziteljZaRegistracijuRadara nije aktivan, vraća odgovor ERROR 32 tekst (tekst objašnjava razlog pogreške).
- Kada broj id ne odgovara identifikatoru radara, vraća odgovor ERROR 33 tekst (tekst objašnjava razlog pogreške).
- Kada PosluziteljKazni nije aktivan, vraća odgovor ERROR 34 tekst (tekst objašnjava razlog pogreške).

PosluziteljKazni ima dodatne komande:

- **`TEST`**
  - npr. `TEST`
  - Provjera da li ispravni podaci. Ako su ispravni, vraća OK.
  - Npr. `OK`

PosluziteljKazni ima dopunu komande:

- **`VOZILO id vrijemePocetak vrijemeKraj brzina gpsSirina gpsDuzina gpsSirinaRadara gpsDuzinaRadara`**
  - npr. `VOZILO 1 1711348009 1711368009 21.767 46.286608 16.353131 46.286602 16.353136`
  - Provjera da li ispravni podaci. Ako su ispravni, u evidenciju kazni upisuje podatke za e-vozilo, ispisuje na ekran podatke o kazni, podaci o kazni šalju se POST metodom na RESTful web servis za evidenciju kazni e-vozila i vraća OK.
  - Npr. `OK`

  Dodatni kodovi pogrešaka su:

  - Kada POST metoda na RESTful web servis za evidenciju kazni e-vozila nije uspješno obavljena, vraća odgovor ERROR 42 tekst (tekst objašnjava razlog pogreške).

RESTful web servisi smješteni su u Maven modul **{LDAP_korisnik}_vjezba_07_dz_2_servisi**. Definirane su 4 krajnje točke/putanje koje slijede osnovnu putanju „nwtis/v1/“:

- api/kazne – pokriva područje rada poslužitelja PoslužiteljKazni s kojim dvosmjerno komunicira. Podatke o kazni zapisuje u tablicu Kazne u bazi podataka. Dio podataka koji se vraćaju potrebno je dohvatiti/filtrirati iz tablice Kazne u bazi podataka.
    - GET – vraća sve kazne
    - GET/{rb} – vraća kaznu s traženim rb
    - GET?od=&do= – vraća kazne koje su nastale unutar zadanog intervala
    - GET/vozilo/{id} – vraća kazne koje su nastale za zadano e-vozilo
    - GET/vozilo/{id}?od=&do= – vraća kazne koje su nastale za zadano e-vozilo i unutar zadanog intervala
    - HEAD – provjera da li radi poslužitelj
    - POST – dodaje novu kaznu
- api/radari - pokriva područje rada poslužitelja PoslužiteljZaRegistraciju s kojim jednosmjerno komunicira. Podatke sprema u memoriji.
    - GET – vraća sve radare
    - GET/reset – pokreće postupak resetiranja svih poslužitelja slanjem komande poslužitelju PoslužiteljZaRegistraciju
    - GET/{id} – vraća podatke za radar sa zadanim id
    - GET/{id}/provjeri – pokreće provjeru radara sa zadanim id
    - DELETE – briše podatke za sve radare slanjem komande poslužitelju PoslužiteljZaRegistraciju
    - DELETE/{id} – briše podatke za radar radar sa zadanim id slanjem komande poslužitelju PoslužiteljZaRegistraciju
- api/vozila - pokriva područje rada poslužitelja PoslužiteljZaVozila/RadnikZaVozila s kojim dvosmjerno komunicira. Podatke o vožnjama praćenih e-vozila zapisuje u tablicu PraceneVoznje u bazi podataka. Dio podataka koji se vraćaju potrebno je dohvatiti/filtrirati iz tablice PraceneVoznje u bazi podataka.
    - GET?od=&do= – vraća praćene vožnje koje su nastale unutar zadanog intervala
    - GET/vozilo/{id} – vraća praćene vožnje koje su nastale za zadano e-vozilo
    - GET/vozilo/{id}?od=&do= – vraća praćene vožnje nastale za zadano e-vozilo i unutar zadanog intervala
    - GET/vozilo/{id}/start – pokreće praćenje vožnje za e-vozilo sa zadanim id slanjem komande poslužitelju PoslužiteljZaVozila
    - GET/vozilo/{id}/stop – prekida praćenje vožnje za e-vozilo sa zadanim id slanjem komande poslužitelju PoslužiteljZaVozila
    - POST – dodaje novo praćenje vožnje za e-vozilo.

- api/simulacije - pokriva područje rada poslužitelja PoslužiteljZaVozila/RadnikZaVozila s kojim jednosmjerno komunicira. Podatke o vožnjama e-vozila zapisuje u tablicu Voznje u bazi podataka. Dio podataka koji se vraćaju potrebno je dohvatiti/filtrirati iz tablice Voznje u bazi podataka.
    - GET?od=&do= – vraća vožnje koje su nastale unutar zadanog intervala
    - GET/vozilo/{id} – vraća vožnje koje su nastale za zadano e-vozilo
    - GET/vozilo/{id}?od=&do= – vraća vožnje nastale za zadano e-vozilo i unutar zadanog intervala
    - POST – dodaje novu vožnju za e-vozilo.

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

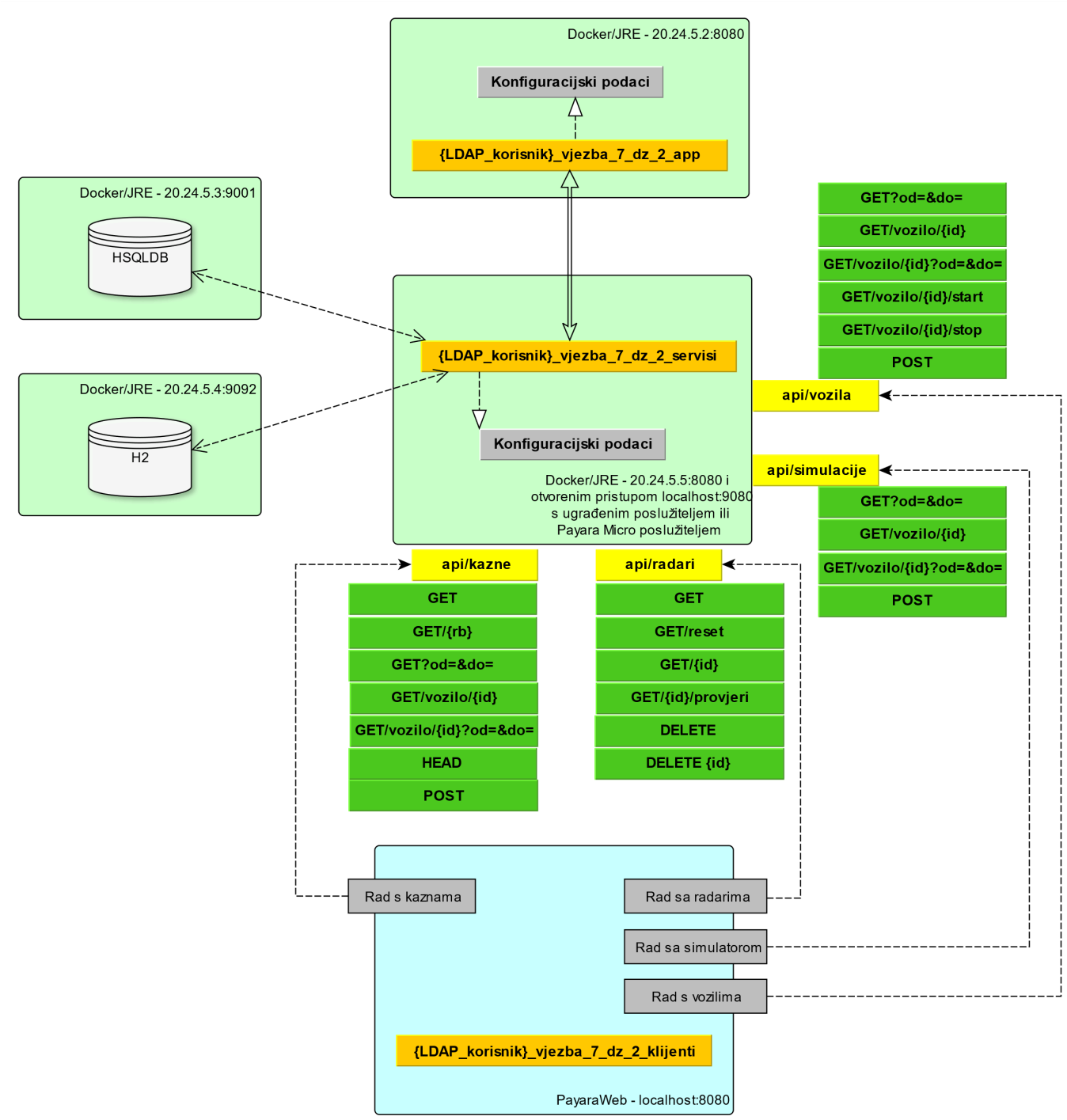
Sve metode RESTful web servisa moraju:

- kod vraćanja koristiti klasu Response
- primati podatke (kod POST metode) i vraćati podatke u obliku application/json.

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 nwtis i integrirani u Docker compose. Spajanje na poslužitelje u Docker kontejnerima mora se obavljati putem zadane statičke adrese pojedinog Docker kontejnera na prikazanoj slici. Jedino se može RESTful web servisima pristupiti putem lokalne adrese http://localhost:8080/.

Korisničko sučelje temelji se na primjeni Jakarta MVC. Za svaku od 4 krajnje točke potrebno je izraditi poseban kontroler i njemu pripadajuće poglede. POST metoda kod api/kazne nema doticaj s korisničkim sučeljem. Podaci za simulaciju u api/simulacija trebaju biti kopirani na direktorij WEB-INF. Postupak dolaska do datoteke na tom direktoriju nalazi se u primjeru s predavanja za Slušače i filtere. Za tablične prikaze podataka može se koristiti jQuery DataTables. Poželjno je da se povezuju podaci između pogleda putem poveznica. Npr. u prikazu podataka svih kazni kod pojedine kazne može se staviti poveznica da se prikazuju kazne za vozilo koje u tom retku. U tabličnom prikazu entiteta iz pojedine krajnje točke nike potrebno prikazati sve podatke, posebno ne kod vožnje. Zato se postavit poseban pogled koji prikazuje sve podatke samo jednog entiteta, a na taj pogled postoji poveznica koja se nalazi u tabličnom prikazu svih entiteta ili odabrane skupine entiteta.

U Maven modulu **{LDAP_korisnik}_vjezba_07_dz_2_app** potrebno je slati pozive na RESTful web service iz **{LDAP_korisnik}_vjezba_07_dz_2_servisi.** Zbog toga treba napraviti klijente za RESTful web service. Klijenti se radi na isti način kako je prikazano za **{LDAP_korisnik}_vjezba_07_dz_2_klijenti.** Potrebne ovisnosti za klijente RESTful web service u stolnoj aplikaciji prikazane su u primjeru s predavanja za Uvod u Jakarta EE i Jakarta EE Core Profile u datoteci pom.xml u Maven modulu nwtis.rest.aplikacija. Naravno, bez nwtis.rest.lib.



Slika 2. Instalacijska arhitektura sustava i metode RESTful web servisa