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:

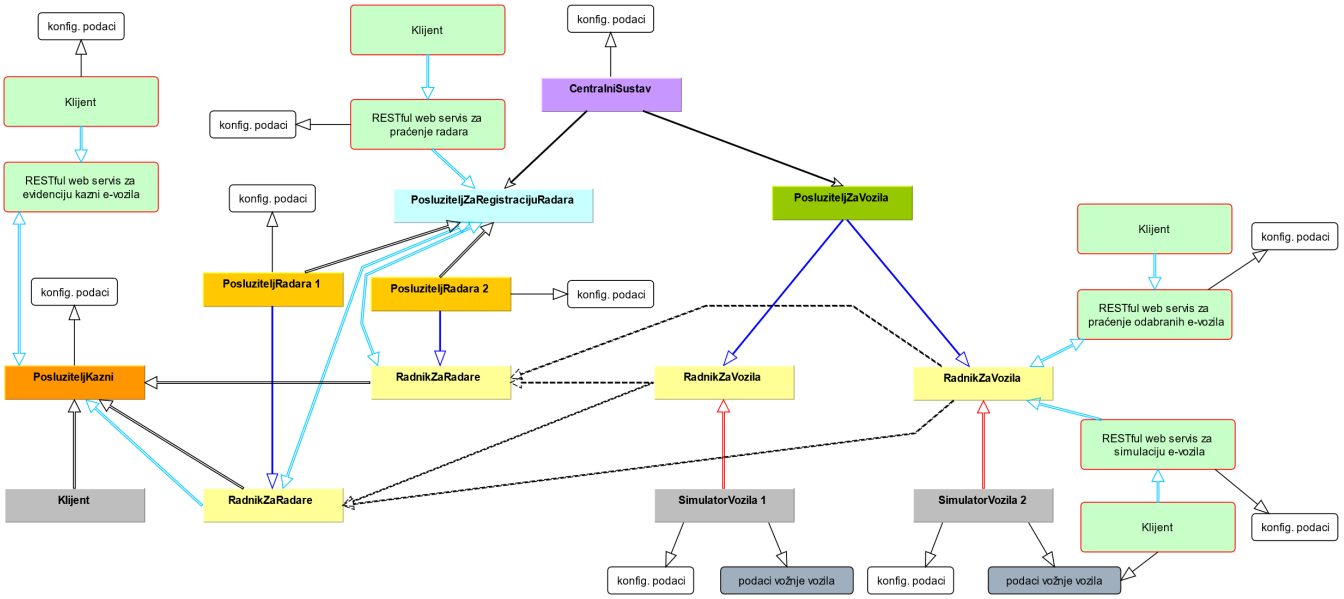
```
{LDAP_korisnik}_vjezba_07_dz_2
        {LDAP_korisnik}_vjezba_07_dz_2.pdf
        {LDAP_korisnik}_vjezba_07_dz_2_app
        {LDAP_korisnik}_vjezba_07_dz_2_lib_konfig
        {LDAP_korisnik}_vjezba_07_dz_2_lib
        {LDAP_korisnik}_vjezba_07_dz_2_lib_rest
        {LDAP_korisnik}_vjezba_07_dz_2_servisi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

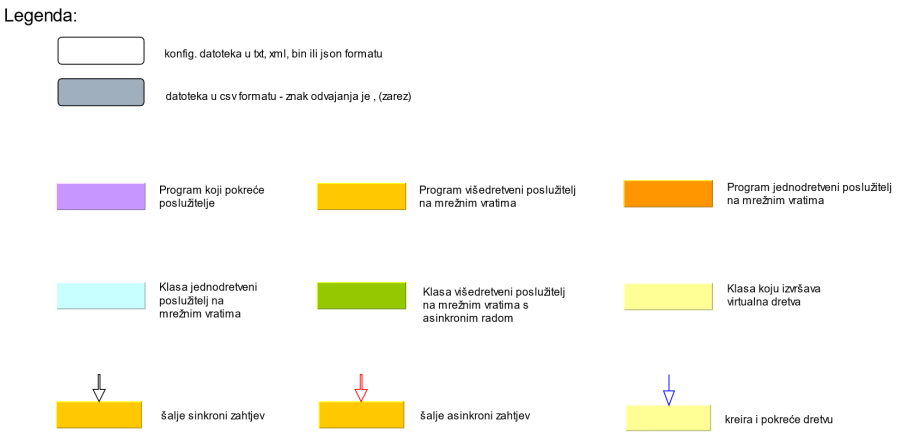
Naziv Maven modula	Uloga Maven modula i kako je nastao	Verzija
`{LDAP_korisnik}_vjezba_07_dz_2_app`	Aplikacija za rad na mrežnoj utičnici. Nastala iz **`{LDAP_korisnik}_vjezba_04_dz_1_app.`** Obrisani paketi podaci i pomocnici. Dodana ovisnost na **`{LDAP_korisnik}_vjezba_07_dz_2_lib`**	1.1.0
`{LDAP_korisnik}_vjezba_07_dz_2_lib_konfig`	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 PosluziteljKazni 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 PosluziteljZaRegistraciju s kojim jednosmjerno komunicira. Podatke sprema u memoriji.
    - GET – vraća sve radare
    - GET/reset – pokreće postupak resetiranja svih poslužitelja slanjem komande poslužitelju PosluziteljZaRegistraciju
    - 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 PosluziteljZaRegistraciju
    - DELETE/{id} – briše podatke za radar radar sa zadanim id slanjem komande poslužitelju PosluziteljZaRegistraciju
- api/vozila - pokriva područje rada poslužitelja PosluziteljZaVozila/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 PosluziteljZaVozila
    - GET/vozilo/{id}/stop – prekida praćenje vožnje za e-vozilo sa zadanim id slanjem komande poslužitelju PosluziteljZaVozila
    - 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.

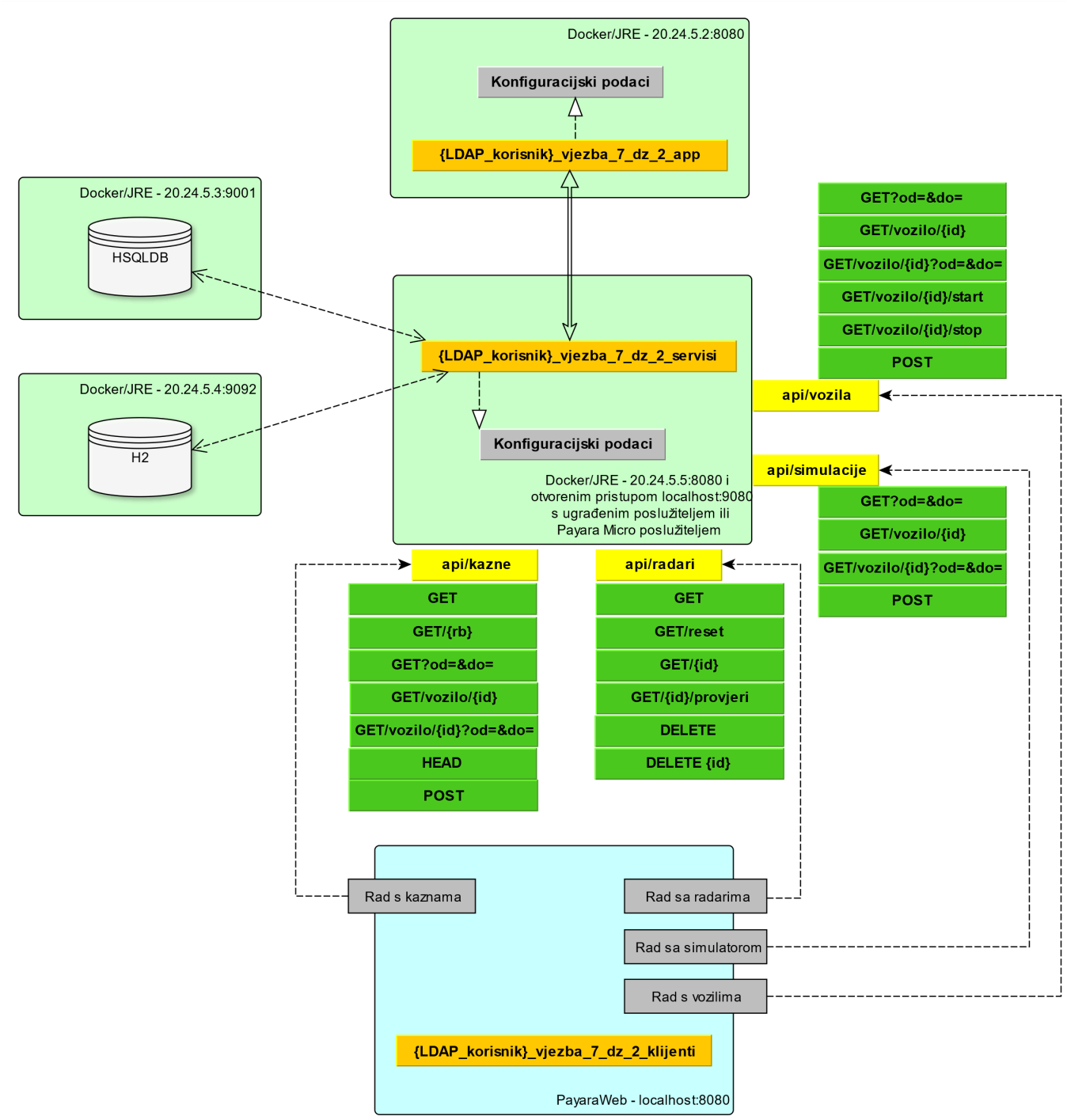
Sve metode RESTful web servisa moraju:

- 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