Projekt: *NWTiS_2017_2018 v1.2*

Postoji sustav za upravljanje parkiralištima. Na početku sustav nema pridruženo ni jedno parkirašte. Pojedino parkiralište može se dodati u sustav tako da se pošalje zahtjev. Odabrano parkirališta može se obrisati iz sustava, ali samo u situaciji kada nema ni jednog vozila koje ga trenutno koristi. Svako parkiralište određeno je svojim jednoznačnim identifikatorom, a sadrži podatke o nazivu, adresi, geo lokaciji na bazi adrese, broju parkirnih mjesta, broju ulazno-izlaznih mjesta.

Upravljanje parkiralištima obavlja se u aplikaciji NWTiS_2018 koje je fizički odvojeno od ostalih aplikacija (nastavnička aplikacija) i temelji se na web servisu pod nazivom Parkiranje. Njegova je uloga da prima informacije od određenog skupa parkirališta (ulaz i izlaz vozila) koja su pridružena pojedinom korisniku (to se naziva grupa), te da ih prosljeđuje korisniku u obliku MQTT poruka na njegovu temu (topic). Korisnik se treba pretplatiti na svoju temu kako bi mogao preuzeti/primati poruke. Korisnik može upravljati svojom grupom kao i pojedinim parkiralištima u grupi. Integracija sustava NWTiS_2018 s ostalim dijelovima sustava realizirana je pomoću operacija web servisa. Svaki poziv operacije web servisa mora sadržavati podatke o korisničkom imenu i korisničkoj lozinci aktivnog korisnika web servisa. Tu se radi o podacima koji se koriste prilikom autentikacije za NWTiS video materijale i NWTiS_svn Subversion. Operacije SOAP web servisa Parkiranje iz aplikacije NWTiS_2018 može pozivati samo aplikacija {korisnicko_ime}_aplikacija_1. Radi se o sljedećim aktivnostima:

- registrirati svoju grupu što dovodi do kreiranja i pokretanje dretve za slanje MQTT poruka o aktivnim parkiralištima iz njegove grupe. Grupa je inicijalno blokirana tako da dretva čeka aktiviranje.
- odjaviti (deregistrirati) svoju grupu što dovodi prekida i brisanja dretve za slanje MQTT poruka.
- aktivirati svoju grupu što dovodi do izvršavanje dretve koja simulira ulaze i izlaze iz parkirališta
- blokirati svoju grupu što dovodi do postavlja dretve u stanje čekanja.
- dobiti status svoje grupe
- dodati, obrisati, aktivirati, blokirati parkiralište iz svoje grupe (jedno, više ili sve)
- dobiti parkirališta iz svoje grupa
- dobiti status odabranog parkirališta iz svoje grupe

Adresa WSDL: http://nwtis.foi.hr:8080/NWTiS_2018/Parkiranje?wsdl

Adresa MQTT poslužitelja: http://nwtis.foi.hr/

Port MQTT poslužitelja: 61613

Za autentikaciju se koriste podaci za NWTiS video materijale i NWTiS_svn Subversion.

Naziv teme: `"/NWTiS/{korisnickoIme}"`

Sadržaj MQTT poruke je u formatu JSON: `{"parkiraliste": d{1-6}, "vozilo": registracija, "akcija": 0 (ulaz) | 1 (izlaz), "vrijeme": "dd.MM.yyyy hh.mm.ss.zzz"}`

Adresa web konzole MQTT poslužitelja: http://nwtis.foi.hr:61680/

Za autentikaciju se koriste podaci za NWTiS video materijale i NWTiS_svn Subversion.

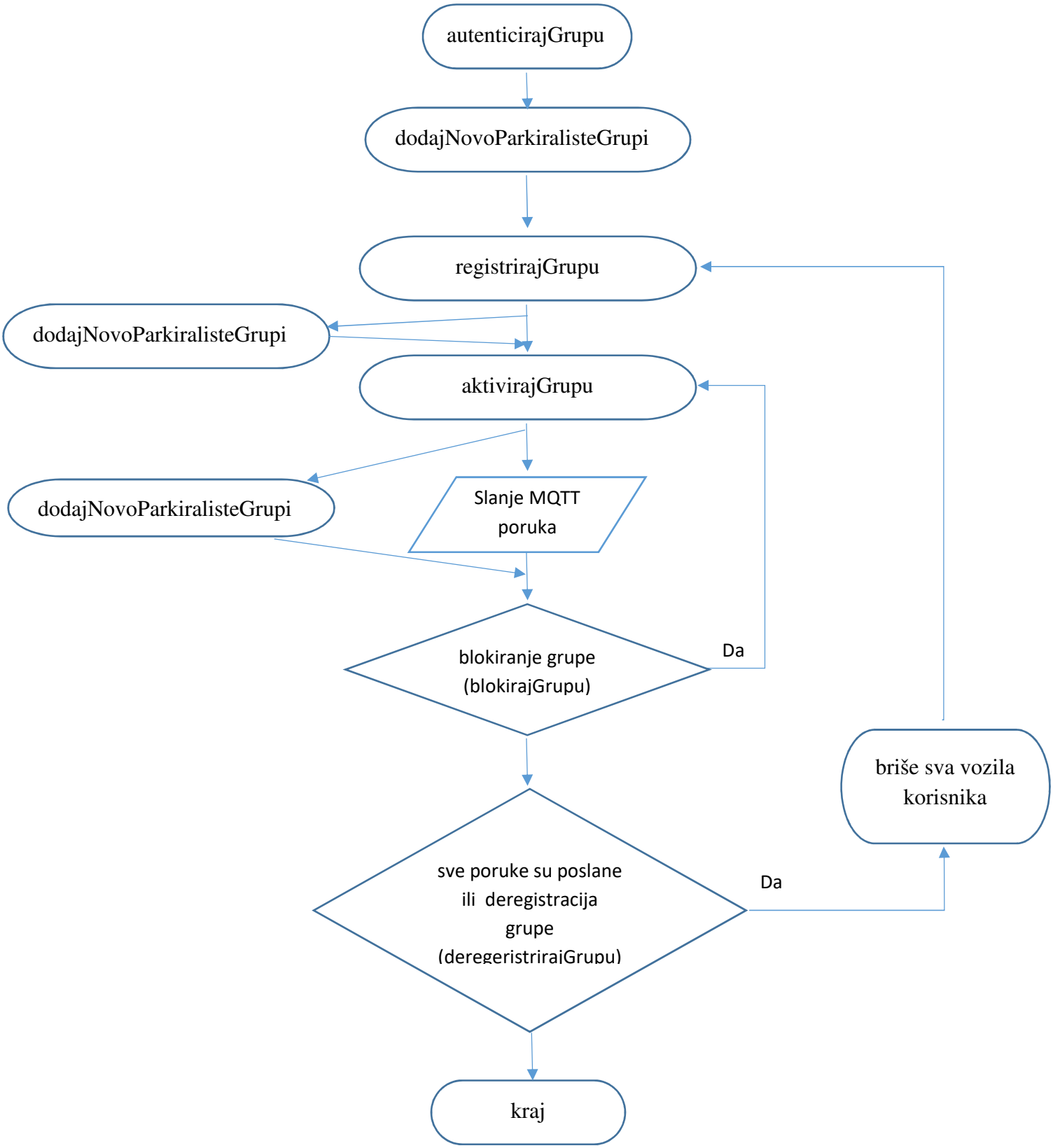
Poslužitelj MQTT poruka je Apollo (https://activemq.apache.org/apollo/).

Adresa NetBeans primjera za preuzimanje MQTT poruka (iz Apollo):
https://elf.foi.hr/mod/resource/view.php?id=51243

Više o MQTT porukama na adresama:

https://activemq.apache.org/apollo/documentation/mqtt-manual.html

http://mqtt.org/



Sustav se treba sastojati od sljedećih aplikacija:

1. web aplikacija (`{korisnicko_ime}_aplikacija_1`) bavi se komuniciranjem s NWTiS_2018 te ujedno pruža sučelje ostalim aplikacijama prema NWTiS_2018. U pozadinskom modu (putem slušača aplikacije), pokreće dretvu (konfiguracijom se određuje pravilni vremenski interval preuzimanja podataka pri čemu je jedinica sekunda, npr. 30 sec, 100 sec, 2 min, 10 min, 30 min, 60 min, ...) koja preuzima važeće meteorološke podatke od openweathermap.org web servisa (u prilogu se nalazi opis postupka) za izabrani skup parkirališta na bazi njihovih lokacijskih podataka. APPID (APIKEY) za openweathermap treba biti spremljen u konfiguracijskoj datoteci.

   Ova aplikacija jedina sadrži podatke o korisnicima (korisničko ime, lozinku, prezime i ime). Tablica korisnika treba sadržavati podatke za minimalno 10 korisnika.

   Tablica u bazi podataka treba sadržavati podatke za minimalno 10 parkirališta i za svako od njih minimalno 100 preuzetih meteoroloških podataka u vremenskom intervalu duljim od zadnjih 72 sata od termina obrane projekta.

   Upravljanje pozadinskom dretvom i izvršavanje predviđenih operacija provodi se putem poslužitelja na mrežnoj utičnici (socket servera) na određenom vratima (portu) (konfiguracijom se određuje). Kada poslužitelj primi zahtjev od klijenta prvo treba obaviti autentikaciju korisnika prema bazi podataka te ako je u redu može se nastaviti s radom. Zatim zapisuje podatke u tablicu zajedničkog dnevnika rada u bazi podataka. Aplikacija ima samo jednu tablicu dnevnika rada u koju podatke upisuju svi dijelovi aplikacije koji to trebaju zapisivati u dnevnik rada. Provođenje spomenutih operacija treba biti putem dretvi kako bi se ostvalira paralelnost obrade zahtjeva jer se ne smije utjecati na sposobnost poslužitelja da primi i obrađuje nove zahtjeve tj. treba izbjeći slijedno obrađivanje. Određene operacije višedretvenog sustava moraju biti međusobno isključive. Zahtjev se temelji na komandama (isključivo u jednom retku), koje mogu biti sljedećih vrsta:

   Komande za poslužitelja:

   ```
   KORISNIK korisnik; LOZINKA lozinka; DODAJ "prezime" "ime";

   KORISNIK korisnik; LOZINKA lozinka; AZURIRAJ "prezime" "ime";

   KORISNIK korisnik; LOZINKA lozinka; {PAUZA; | KRENI; | PASIVNO;
   | AKTIVNO; | STANI; | STANJE; | LISTAJ; }
   ```

   Objašnjenje komandi:

   - `KORISNIK korisnik; LOZINKA lozinka; DODAJ "prezime" "ime";` – Vraća OK 10; ako ne postoji korisnik i uspješno je dodan odnosno ERR 10; ako već postoji korisnik.
   - `KORISNIK korisnik; LOZINKA lozinka; AZURIRAJ "prezime" "ime";` – Vraća OK 10; ako postoji korisnik i uspješno je ažuriran odnosno ERR 10; ako ne postoji korisnik.
   - `KORISNIK korisnik; LOZINKA lozinka;` – autentikacija korisnika. Vraća ERR 11; ako ne postoji korisnik ili ne odgovara lozinka. Ako su podaci u redu i nema nastavka komande vraća OK 10; Odnosno ako ima, prelazi na obradu ostalog dijela komande.
   - `PAUZA;` – privremeno prekida preuzimanje ostalih komande osim za poslužitelja. Vraća OK 10; ako nije bio u pauzi, odnosno ERR 12; ako je bio u pauzi.
   - `KRENI;` – nastavlja s preuzimanjem svih komandi. Vraća OK 10; ako je bio u pauzi, odnosno ERR 13; ako nije bio u pauzi.

- `PASIVNO;` – privremeno prekida preuzimanje meteoroloških podataka od sljedećeg ciklusa Vraća OK 10; ako je bio u aktivnom radu, odnosno ERR 14; ako je bio u pasivnom radu.
- `AKTIVNO;` – nastavlja s preuzimanjem meteoroloških podataka od sljedećeg ciklusa. Vraća OK 10; ako je bio u pasivnom radu, odnosno ERR 15; ako je bio u aktivnom radu.
- `STANI;` – potpuno prekida preuzimanje meteoroloških podataka i preuzimanje komandi. I završava rad. Vraća OK 10; ako nije bio u postupku prekida, odnosno ERR 16; ako je bio u postupku prekida.
- `STANJE;` – vraća trenutno stanje poslužitelja. Vraća OK dd; gdje dd znači: 11 – preuzima sve komanda i preuzima meteo podatke, 12 - preuzima sve komanda i ne preuzima meteo podatke, 13 – preuzima samo poslužiteljske komanda i preuzima meteo podatke, 14 – preuzima samo poslužiteljske komanda i ne preuzima meteo podatke.
- `LISTAJ;` – vraća podatke svih korisnika. Vraća OK 10; [{″ki″: d{1-6} ″prezime″: prezime, ″ime″: ime,},...]; odnosno ERR 17; ako ne postoji.

Komande za grupu:

```
KORISNIK korisnik; LOZINKA lozinka; GRUPA {DODAJ; | PREKID; |
KRENI; | PAUZA; | STANJE;}
```

- `KORISNIK korisnik; LOZINKA lozinka;` – autentikacija korisnika. Vraća ERR 11; ako ne postoji korisnik ili ne odgovara lozinka. Ako su podaci u redu i nema nastavka komande vraća OK 10; Odnosno ako ima, prelazi na obradu ostalog dijela komande.
- `GRUPA DODAJ;` – registrira grupu. Vraća OK 20; ako nije bila registrirana (ne postoji), odnosno ERR 20; ako je bila registrirana.
- `GRUPA PREKID;` – odjavljuje (deregistrira) grupu. Vraća OK 20; ako je bila registrirana, odnosno ERR 21; ako nije bila registrirana.
- `GRUPA KRENI;` – aktivira grupu. Vraća OK 20; ako nije bila aktivna, ERR 22; ako je bila aktivna odnosno ERR 21; ako ne postoji.
- `GRUPA PAUZA;` – blokira grupu. Vraća OK 20; ako je bila aktivna, ERR 23; ako nije bila aktivna odnosno ERR 21; ako ne postoji
- `GRUPA STANJE;` – vraća status grupe. Vraća OK dd; gdje dd znači: 21 – grupa je aktivna, 22 – grupa blokirana odnosno ERR 21; ako ne postoji.

U slučaju da je primljena ispravna komanda za poslužitelja potrebno obaviti njen zadatak i na kraju poslati email poruku (adresa primatelja, adresa pošiljatelja i predmet poruke određuju se postavkama) u MIME tipu "text/json" ili "application/json" s informacijama o zahtjevu sa sljedećom sintaksom:

{"id": d{1-6}, "komanda": komanda, "vrijeme": "yyyy.MM.dd hh:mm:ss.zzz"}

gdje je id redni broj email poruke koja se šalje. U komandi koja se šalje email porukom ne prikazuje se podatak za lozinku.

U slučaju ispravne komande za grupu treba pozvati pripadajuću operaciju SOAP web servisa Parkiranje iz aplikacije NWTiS_2018 te vratiti potrebne podatke u odgovoru.

Drugi zadatak web aplikacije je pružanje vlastitog SOAP web servisa. Svaki zahtjev treba biti autenticiran prema korisničkim podacima u bazi podataka i upisan u tablicu zajedničkog dnevnika

rada u bazi podataka. SOAP web servis svoj rad temelji na meteorološkim podacima koji su putem pozadinske dretve spremljeni u bazu podataka ili se poziva vanjski web servis. Na raspolaganju stoje operacije:

- zadnje preuzeti meteorološki podaci za izabrano parkiralište
- posljednjih n (n se unosi) meteoroloških podataka za izabrano parkiralište
- meteorološki podaci za izabrano parkiralište u nekom vremenskom intervalu (od datuma, do datuma kao timestamp)
- važeći meteorološki podaci za izabrano parkiralište (putem openweathermap.org web servisa).

Potrebno je pripremiti u NetBeans-ima za testiranje vlastitog web servisa (u Web services kreirati novu grupu NWTiS i dodati vlastiti servis).

Treći zadatak je REST web servis za rad s parkiralištima koja se spremaju u tablicu parkiralištima u bazi podataka. Svaki zahtjev treba biti autenticiran prema korisničkim podacima u bazi podataka i upisan u tablicu zajedničkog dnevnika rada u bazi podataka. Na raspolaganju stoje:

- preuzimanje svih parkirališta - vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...},{...}...], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}.
- preuzimanje jednog parkirališta - vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...}], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}.
- dodavanje jednog parkirališta - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}.
- ažuriranje jednog parkirališta - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}.
- brisanje jednog parkirališta - vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}.
- preuzimanje svih vozila odabranog parkirališta - vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...},{...}...], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}.

Sva parkirališta preuzimaju se putem osnovne adrese. Za izabrano parkiralište šalje se njegov id tako da je to putanja `{id}` u URL strukturi poziva REST web servisa. Sva vozila odabranog parkirališta preuzimaju se putem osnovne adrese uz dodanu putanju `{id}/vozila/`. Rad ovog REST web servisa za preuzimanje, dodavanje, ažuriranje brisanje parkirališta i preuzimanje vozila temelji se na pozivima SOAP web servisa Parkiranje iz NWTiS_2018.

Četvrti zadatak je vidljivi dio web aplikacije koji se može realizirati putem JSP, JSF (facelets) ili PrimeFaces i treba biti zaštićen putem kontejnera na bazi vlastitog obrasca/forme za prijavljivanje uz pomoć JDBC pristupa do baze podataka te osiguranjem sigurnog kanala (SSL uz vlastiti certifikat s imenom i prezimenom studenta). Korisnik može obaviti:

- pogled 1:
  - pregled korisnika uz straničenje (konfiguracijom se određuje broj linija po stranici)
- pogled 2:
  - pregled zajedničkog dnevnika rada uz straničenje i filtriranje (vrsta zapisa, od-do) (konfiguracijom se određuje broj linija po stranici)

2. enterprise aplikacija (`{korisnicko_ime}_aplikacija_2`) koja ima EJB i Web module. Prvi dio je EJB modul koji sadrži Singleton Session Bean (SB) i Stateful Sesssion Bean. Kreiranje Singleton SB pokreće dretvu (konfiguracijom se određuje pravilni vremenski interval rada (jedinica je sekunda), npr. 5 sec, 20 sec, 100 sec, ...) koja provjerava pristigle poruke u poštanskom sandučiću (adresa poslužitelja, korisničko ime i lozinka definiraju se u konfiguracijskoj datoteci). Brisanje Singleton SB prekida dretvu i zaustavlja ju.

   Od pristiglih nepročitanih email poruka one koje imaju predmet poruke prema postavkama iz konfiguracijske datoteke i sadržaj poruke MIME tip "text/json" ili "application/json" nazivamo NWTiS porukama. Obrađene NWTiS poruke treba označiti da su pročitane i prebaciti u mapu/direktorij prema postavkama za NWTiS poruke. Ostale ne-NWTiS poruke treba ostaviti da su nepročitane. Na kraju svake iteracije obrade email poruka u kojoj je obrađena barem jedna NWTiS poruka treba poslati JMS poruku (naziv reda čekanja `NWTiS_{korisnicko_ime}_1`) s podacima o rednom broju JMS poruke koja se šalje, vremenu slanja prethodne JMS poruke, vremenu slanja trenutne JMS poruke, vremenu rada iteracije dretve, broju NWTiS poruka. Poruka treba biti u obliku ObjectMessage, pri čemu je naziv vlastite klase proizvoljan, a njena struktura treba sadržavati potrebne podatke koji su prethodno spomenuti. Red čekanja treba ima vlastiti brojač JMS poruka.

   Autenticiranje korisnika obavlja se u Stateful SB uz korištenje REST web servis `{korisnicko_ime}_aplikacija_3`. Nakon uspješnog autenticiranja korisnik može odabrati rad iz ponuđenih izbornika svojih pogleda. Ako se registrira za prijem MQTT poruka za vozila (ulaz/izlaz) na parkiralištima iz korisnikove grupe slanjem komande poslužitelju na mrežnoj utičnici iz `{korisnicko_ime}_aplikacija_1`. Primljeni podaci iz MQTT poruke upisuju se u tablicu MQTT poruka u bazi podataka. Nakon prijema MQTT poruke potrebno je provjeriti da li se radi o parkiralištu za koje je zadužen korisnik. Ako je ulaz na određeno parkiralište provjerava se ima li slobodnog mjesta na tom parkiralištu. Ako je izlaz iz određenog parkirališta provjerava se ima li vozila na tom parkiralištu. Ako je sve u redu, slijedi slanje JMS poruke (naziv reda čekanja `NWTiS_{korisnicko_ime}_2`) s podacima o rednom broju JMS poruke koja se šalje, korisniku, vremenu prijema poruke, sadržaj poruke iz atributa `"tekst"` primljene MQTT poruke, status akcije (odobreno, zabranjeno), opis statusa. Poruka treba biti u obliku ObjectMessage, pri čemu je naziv vlastite klase proizvoljan, a njena struktura treba sadržavati potrebne podatke koji su prethodno spomenuti. Red čekanja treba ima vlastiti brojač JMS poruka. Nakon odjavljivanja korisnika ili isteka sjednice (sesije) potrebno je deregistrirati korisnika za prijem MQTT poruka.

   Drugi dio je web modul koji treba realizirati putem JSF (facelets) ili PrimeFaces uz minimalno dvojezičnu varijantu (hrvatski i engleski jezik). To znači da svi statički tekstovi u pogledima trebaju biti označeni kao „labele“ i dobiti jezične prijevode. Jezik se odabire na početnoj stranici aplikacije i svi pogledi moraju imati direktnu poveznicu na tu adresu.

   Potrebno je voditi evidenciju zahtjeva pomoću aplikacijskog filtera s upisom u tablicu zajedničkog dnevnika rada ove aplikacije u bazi podataka. Osim osnovnih podataka o zahtjevu (url, IP adresa, korisničko ime, vrijeme prijema) potrebno je spremiti trajanje obrade pojedinog zahtjeva.

   Korisniku na početku stoje na raspolaganju registracija ili prijavljivanje. Ova aplikacija nema tablicu korisnika u bazi podataka. Korisnik kod registracije upisuje korisničko ime (mora biti jedninstveno), prezime, lozinku i ponovljenu lozinku za provjeru, email adresu. Nakon uspješne provjere (validacije) podataka za registraciju korisnika, podaci se šalju REST servisu `{korisnicko_ime}_aplikacija_3`. Ako je dodavanje u redu javlja se poruka o tome. Odnosno poruka ako nije u redu. Nakon uspješnog prijavljivanja (šalje se zahtjev REST servisu `{korisnicko_ime}_aplikacija_3` za podatke o korisniku te se provjeravaju) korisnik može obavljati aktivnosti kroz određene poglede:

- pogled 1:
  - registracija korisnika uz validaciju podataka, ažuriranje vlastitih podataka i vidjeti popis svih korisnika (konfiguracijom se određuje broj linija po stranici). Sve aktivnosti provode su pomoću REST servisa `{korisnicko_ime}_aplikacija_3`.
- pogled 2:
  - pregled trenutnog statusa poslužitelja na mrežnoj utičnici (socket servera) iz `{korisnicko_ime}_aplikacija_1` i njime upravljati. Obavlja se slanjem pripadajuće komande.
  - pregled trenutnog statusa grupe u NWTiS_2018 pomoću poslužitelja na mrežnoj utičnici (socket servera) iz `{korisnicko_ime}_aplikacija_1` i njime upravljati (registrirati, odjaviti, aktivirati, blokirati grupu). Obavlja se slanjem pripadajuće komande.
- pogled 3:
  - pregled vlastitih parkirališta, dodavanje pomoću REST servisa `{korisnicko_ime}_aplikacija_1` i upravljati odabranim parkiralištem (dodati, obrisati, aktivirati, blokirati, pregled statusa, pregled vozila).
  - za odabrano parkiralište na temelju podataka SOAP web servisa iz {korisnicko_ime}_aplikacija_1:
    - pregled zadnjih meteoroloških podataka za izabrano parkiralište
    - pregled važećih meteoroloških podataka za izabrano parkiralište
- pogled 4:
  - pregled poruka u poštanskom sandučiću (adresa poslužitelja, korisničko ime i lozinka definiraju se u postavkama) uz straničenje (konfiguracijom se određuje broj linija po stranici). Pri tome može izabrati mapu/direktorij u kojem pregledava poruke. Nazivi mapa preuzimaju se od mape poštanskog sandučića korisnika email poslužitelja. Korisnik može izvršiti brisanje svih poruka aktivne (izabrane) mape.
- pogled 5:
  - pregled spremljenih MQTT poruka za korisnika uz straničenje (konfiguracijom se određuje broj linija po stranici). Korisnik može izvršiti brisanje svih svojih poruka.
- pogled 6:
  - pregled dnevnika rada uz straničenje (konfiguracijom se određuje broj linija po stranici).

Funkcionalnost korisničkog dijela za pogled 2 i pogled 3 treba biti realizirana pomoću Ajax-a. Dodatni bodovi mogu se dobiti ako se u pogledu 2 za prikaz odabranih parkirališta i njihovih podataka (npr. koordinate, trenutna temp, udaljenost i sl.) koristi Google Maps JavaScript API.

Pristup do podataka u bazi podataka treba biti realiziran putem ORM-a tj. putem session, entity bean-ova i criteria API.

3. enterprise aplikacija (`{korisnicko_ime}_aplikacija_3`) koja ima EJB i Web module. Prvi dio je EJB modul koji sadrži Singleton Session Bean (SB), Stateful Sesssion Bean (SB) i Message-Driven Bean. Message-Driven Bean preuzima dvije vrste JMS poruka: za stanje obrade email poruka i primljene MQTT poruke. Primljene JMS poruke spremaju se u Singleton Session Bean (SB). Ako aplikacija prestaje s radom (brisanje Singleton SB), potrebno je poruke serijalizirati na vanjski spremnik (naziv datoteke u postavkama, smještena u WEB-INF direktoriju). Kada se aplikacija podiže (kreiranje Singleton Session Beana) potrebno je učitati serijalizirane poruke (ako postoji datoteka) u Singleton Session Bean. Autenticiranje korisnika obavlja se u Stateful SB uz korištenje REST web servis `{korisnicko_ime}_aplikacija_3. Na temelju JMS poruka za MQTT poruke potrebno je voditi evidenciju`

   Drugi dio je web modul koji pruža REST web servis u JSON formatu. REST web servis odnosi se na rad s korisnicima i on služi kao veza za ostale aplikacije prema `{korisnicko_ime}_aplikacija_1` koja sadrži tablicu korisnika u bazi podataka. Komunikacija se ostvaruje komandama s poslužiteljem na mrežnoj utičnici. Na raspolaganju stoje:

   - preuzimanje svih korisnika - vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...},{...}...], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}. Ne vraća se lozinka!
   - preuzimanje jednog korisnika - vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...}], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}. Ne vraća se lozinka!
   - dodavanje jednog korisnika - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}. Ne vraća se lozinka!
   - ažuriranje jednog korisnika - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}. Ne vraća se lozinka!
   - autentikacija jednog korisnika - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...}], "status": "OK" | "ERR", <ako je "ERR" onda se dodaje "poruka": poruka>}. Ne vraća se lozinka!

   Svi korisnici preuzimaju se putem osnovne adrese. Za izabranog korisnika šalje se njegovo korisničko ime tako da je to putanje `{korisnickoIme}` u URL strukturi poziva REST web servisa. Treba pravilno implementirati metode (GET, POST, PUT, DELETE) na osnovnoj adresi i putanji `{korisnickoIme}` tako da zabranjene ili nekorištene metode vraćaju pogrešku.

   Vidljivi dio aplikacije koji treba realizirati putem JSF (facelets) ili PrimeFaces. Korisniku na početku obavlja prijavljivanje. Nakon uspješnog prijavljivanja korisnik može obavljati aktivnosti kroz određene poglede:

   - pogled 1:
     - pregled i brisanje spremljenih JMS poruka iz reda čekanja `NWTiS_{korisnicko_ime}_1`). Pomoću websocketa treba obavijestiti pregled poruka da je stigla nova JMS poruka njegove vrste te treba osvježiti pregled.
   - pogled 2:
     - pregled i brisanje korisnikovih spremljenih JMS poruka iz reda čekanja `NWTiS_{korisnicko_ime}_2`). Pomoću websocketa treba obavijestiti pregled poruka da je stigla nova JMS poruka njegove vrste te treba osvježiti pregled.
   - pogled 3:
     - pregled trenutnog stanja svih parkirališta s ukupnim brojem mjesta, brojem zauzetih i brojem slobodnih mjesta. Pomoću websocketa treba obavijestiti pregled da je stigla nova JMS poruka njegove vrste te treba osvježiti pregled.

Instalacijska, programska i komunikacijska arhitektura sustava:

`{korisnicko_ime}_aplikacija_1:`

- Razvojni alat (IDE) kod obrane projekta: NetBeans
- Web poslužitelj: Tomcat
- EE osobine: EE7 Web
- korisničko sučelje: JSP, JSF (facelets) ili PrimeFaces
- baza podataka: MySQL - naziv `nwtis_{korisnickoime}_bp_1`, treba sadržavati tablice: `korisnici`, `parkiralista`, `dnevnik` i ostale koje su potrebne za rad
- rad s bazom podataka: JDBC, SQL
- šalje email poruku
- daje poslužitelja na mrežnoj utičnici (socket server)
- daje SOAP web servis za meteorološke podatke izabrano parkiralište
- daje REST web servis za rad s parkiralištima
- koristi SOAP web servis Parkiranje iz aplikacije NWTiS_2018 za upravljanje parkiralištima
- koristi openweathermap.org REST web servis za preuzimanje meteoroloških podataka
- koristi Google Maps API REST web servis za preuzimanje geolokacijskih podataka za adresu

`{korisnicko_ime}_aplikacija_2:`

- Razvojni alat (IDE) kod obrane projekta: NetBeans
- Web poslužitelj: Glassfish
- EE osobine: EE7
- korisničko sučelje: JSF (facelets) ili PrimeFaces
- baza podataka: JavaDB – naziv `nwtis_{korisnickoime}_bp_2`, treba sadržavati tablice: `dnevnik, mqtt_poruke` i ostale koje su potrebne za rad (ne tablicu korisnika).
- rad s bazom podataka: ORM (EclipseLink), Criteria API
- James email poslužitelj
- pregledava i briše email poruke
- šalje JMS poruku u red poruka: `NWTiS_{korisnicko_ime}_1` nakom ciklusa obrade email poruka
- prima i obrađuje MQTT poruke za ulaz i izlaz vozila parkirališta
- šalje JMS poruku u red poruka: `NWTiS_{korisnicko_ime}_2` za MQTT poruku
- koristi poslužitelja na mrežnoj utičnici, socket server `{korisnicko_ime}_aplikacija_1` za upravljanje poslužiteljem i grupom
- koristi REST web servis `{korisnicko_ime}_aplikacija_1` za upravljanje parkiralištima
- koristi SOAP web servis `{korisnicko_ime}_aplikacija_1` za meteorološke podatke za odabrano parkiralište
- koristi REST web servis `{korisnicko_ime}_aplikacija_3` za pregled, upravljanje i autenticiranje korisnika

`{korisnicko_ime}_aplikacija_3:`

- Razvojni alat (IDE) kod obrane projekta: NetBeans
- Web poslužitelj: Glassfish
- EE osobine: EE7
- korisničko sučelje: JSF (facelets) ili PrimeFaces
- baza podataka: ne koristi bazu podataka.
- koristi JMS redove poruka `NWTiS_{korisnicko_ime}_1` i `NWTiS_{korisnicko_ime}_2` za preuzimanje, spremanje i pregled JMS poruka
- koristi websocket za osvježavanje pregleda poruka nakon prijema nove JMS poruke
- daje REST web servis za pregled, upravljanje i autenticiranje korisnika
- koristi poslužitelja na mrežnoj utičnici, socket server `{korisnicko_ime}_aplikacija_1` za pregled, upravljanje i autenticiranje korisnika

Postupak za aktiviranje i korištenje web servisa openweathermap.org:

1. Dokumentacija: **http://openweathermap.org/api**
2. Upoznati se s ponuđenim modelima: **http://openweathermap.org/price_detailes** (FREE)
3. Registrirati se (**http://openweathermap.org/register** - Register on the Sign up page)
4. Popuniti podatke
5. Zapisati podatke za APPID (API key)
6. Servisi:
   a. **Važeći vremenski podaci** - **http://openweathermap.org/current**

      Parametri:
      APIKEY=nnnnnn
      lokacijski:
      lat=nnn&lon=mmm
      units=metric
      mode=xml | (json je po osnovi)
      lang=jezik (kratica)

http://api.openweathermap.org/data/2.5/weather?lat=46.307768,&lon=16.338123&units=metric&lang=hr&APIKEY=nnnnnn

Parametri API odgovora važećeg vremena i prognoza: **http://openweathermap.org/weather-data#current**

Ikone za vremenske uvjete: **http://openweathermap.org/weather-conditions**

Google Maps API – za dobivanje geolokacijskih podataka za adresu

1. Dokumentacija: **https://developers.google.com/maps/documentation/geocoding/intro**
2. Upoznati se s ponuđenim modelima: **https://developers.google.com/maps/documentation/geocoding/usage-limits** (FREE)
3. Registrirati se (**https://developers.google.com/maps/documentation/geocoding/get-api-key** - GET STARTED)
4. Popuniti podatke
5. Zapisati podatke za APPID (API key)

JSON

https://maps.google.com/maps/api/geocode/json?address=varazdin&sensor=false&key=mmmmmmm

XML

https://maps.google.com/maps/api/geocode/xml?address=varazdin&sensor=false&key=mmmmmmm

Prije slanja adrese važno je obaviti njeno pretvaranje u HTTP URL format pomoću funkcije URLEncoder.encode(adresa, "UTF-8")

Google Maps API – za dobivanje adrese za geolokacijske podatake (reverse geocoding, address lookup)

JSON

https://maps.google.com/maps/api/geocode/json?latlng=xx.xxxxxx,yy.yyyyy&key=mmmmmmm

XML

https://maps.google.com/maps/api/geocode/xml?latlng=xx.xxxxxx,yy.yyyyy&key=mmmmmmm