Projekt: *NWTiS_2020_2020 v1.0*

Postoji sustav za upravljanje aerodromima i avionima. Na početku sustav nema pridruženo ni jedan aerodrom. Pojedini aerodrom može se dodati u sustav tako da se pošalje zahtjev. Odabran aerodrom može se obrisati iz sustava. Svaki aerodrom određen je svojim jednoznačnim identifikatorom (icao), a sadrži podatke o nazivu, državi (iso_contry), geo lokaciji na bazi naziva.

Upravljanje aerodromima obavlja se u aplikaciji NWTiS_2020 (nastavnička aplikacija) koja je fizički odvojena od ostalih aplikacija i temelji se na web servisu pod nazivom AerodromiWS. Osnovu čini korisnik koji predstavlja grupu, a njemu se mogu pridružiti (i obrisati) aerodromi kjoje on prati. Web servis prima informacije za određeni skup aerodroma koji su pridruženi pojedinom korisniku te prosljeđuje poruke 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 aerodromima u grupi. Integracija sustava NWTiS_2020 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 AerodromiWS iz aplikacije NWTiS_2020 može pozivati samo aplikacija {korisnicko_ime}_aplikacija_1. Radi se o sljedećim aktivnostima:

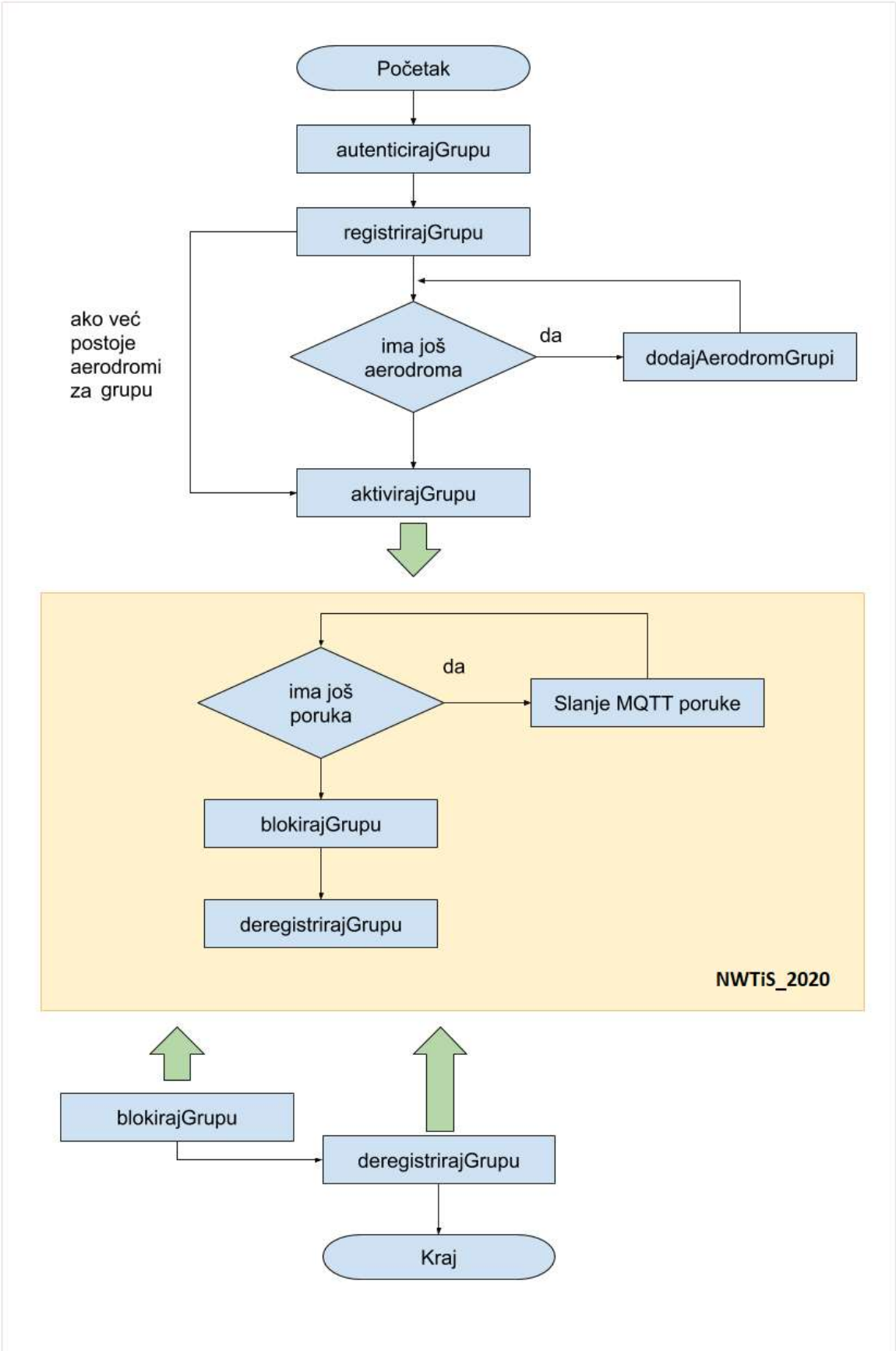
- registrirati svoju grupu što dovodi do kreiranja i pokretanje dretve za slanje MQTT poruka za avione aktivnih aerodroma iz njegove grupe. Grupa je inicijalno blokirana tako da dretva čeka aktiviranje. Korisnik može imati samo jednu aktivnu registraciju. Sljedeći zahtjev će biti zaustavljen na monitoru dok se ne završi prethodna registracija (nakon što se pošalju sve poruke ili korisnik zahtjeva deregistraciju)
- odjaviti (deregistrirati) svoju grupu što dovodi prekida i brisanja dretve za slanje MQTT poruka.
- aktivirati svoju grupu što dovodi do izvršavanje dretve koja šalje MQTT poruke za aerodrome
- blokirati svoju grupu što dovodi do postavljanja dretve u stanje čekanja
- dobiti status svoje grupe
- dodati, obrisati, aktivirati, blokirati aerodrom iz svoje grupe (jedan, više ili sve)
- dobiti aerodrome iz svoje grupa
- dobiti status odabranog aerodroma iz svoje grupe

Adresa WSDL: http://nwtis.foi.hr:8080/NWTiS_2020/AerodromiWS?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 application/json: {″korisnik″: ″korisnik″, ″aerodrom″: ″icao″, ″poruka″: ″tekst poruke″, ″vrijeme″: ″dd.MM.yyyy hh.mm.ss.SSSS″}

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=11332
Više o MQTT porukama na adresama:
https://activemq.apache.org/apollo/documentation/mqtt-manual.html



Slika 1.Shema procesa korištenja sustava NWTiS_2020

Sustav se treba sastojati od sljedećih elemenata:

1. web aplikacija ({korisnicko_ime}_aplikacija_1) bavi se komuniciranjem s NWTiS_2020 tako da ostalim aplikacijama pruža sučelje prema NWTiS_2020.

   Preuzimanje podataka u {korisnicko_ime}_aplikacija_2 temelji se na statusu poslužitelja na mrežnoj utičnici (socket servera) na određenom vratima (portu) (postavkom se određuje). Kada poslužitelj primi zahtjev od klijenta (osim kod dodavanja/registracije korisnika) prvo treba obaviti autentikaciju korisnika te ako je u redu može se nastaviti s radom. Aplikacija ima samo jednu tablicu s korisnicima u bazi podataka. Tablica korisnika sastoji se od stupaca za korisničko ime i lozinku, a treba sadržavati podatke za minimalno 10 korisnika. Provođenje zahtjeva treba biti treba slijedno. Zahtjev se temelji na komandama (isključivo u jednom retku), koje mogu biti sljedećih vrsta:

   Komande za poslužitelja:

   KORISNIK korisnik; LOZINKA lozinka; {DODAJ; | PAUZA; | RADI; | KRAJ; | STANJE; }

   Objašnjenje komandi:

   1. 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.
   2. DODAJ; – dodaje korisnika u tablicu korisnika u bazi podataka. Vraća OK 10; ako korisnik ne postoji i bilo je uspješno dodavanje, odnosno ERR 12; ako korisnik postoji.
   3. PAUZA; – privremeno prekida preuzimanje podataka za aerodrome. Vraća OK 10; ako je bio u aktivnom radu, odnosno ERR 13; ako je bio u pasivnom radu.
   4. RADI; – nastavlja s preuzimanjem podataka za aerodrome. Vraća OK 10; ako je bio u pasivnom radu, odnosno ERR 14; ako je bio u aktivnom radu.
   5. KRAJ; – potpuno prekida preuzimanje podataka za aerodrome i preuzimanje komandi. I završava rad. Vraća OK 10;
   6. STANJE; – vraća trenutno stanje poslužitelja. Vraća OK dd; gdje dd znači: 11 – preuzima podatke za aerodrome, 12 - ne preuzima podatke za aerodrome.

   Komande za grupu:

   KORISNIK korisnik; LOZINKA lozinka; GRUPA { PRIJAVI; | ODJAVI; | AKTIVIRAJ; | BLOKIRAJ; | STANJE; AERODROMI icao_1, icao_2, icao_3,...,icao_n;}

   1. 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.
   2. GRUPA PRIJAVI; – registrira grupu. Vraća OK 20; ako nije bila registrirana (ne postoji), odnosno ERR 20; ako je bila registrirana.
   3. GRUPA ODJAVI; – odjavljuje (deregistrira) grupu. Vraća OK 20; ako je bila registrirana, odnosno ERR 21; ako nije bila registrirana.
   4. GRUPA AKTIVIRAJ; – aktivira grupu. Vraća OK 20; ako nije bila aktivna, ERR 22; ako je bila aktivna odnosno ERR 21; ako ne postoji.
   5. GRUPA BLOKIRAJ; – blokira grupu. Vraća OK 20; ako je bila aktivna, ERR 23; ako nije bila aktivna odnosno ERR 21; ako ne postoji

6. GRUPA6 STANJE; – vraća status grupe. Vraća OK dd; gdje dd znači: 21 – grupa je aktivna, 22 – grupa blokirana odnosno ERR 21; ako ne postoji.
7. GRUPA AERODROMI icao_1, icao_2, icao_3,...,icao_n; – postavlja nove aerodrome za grupu, grupa mora biti blokirana da bi se mogla izvršiti ova komanda. Prvo briše sve postojeće aerodrome grupe i zatim postavlja nove aerodrome za grupu. Vraća OK 20; ako je sve u redu, ERR 31; ako grupa nije blokirana, odnosno ERR 32; ako je bilo koji drugi problem.

U slučaju ispravne komande za grupu treba pozvati pripadajuću operaciju SOAP web servisa AerodromiWS iz aplikacije NWTiS_2020 te vratiti potrebne podatke u odgovoru.

U slučaju da je primljena ispravna komanda potrebno obaviti njen zadatak i na kraju poslati REST zahtjev POST metodom na putanji "/komande" u {korisnicko_ime}_aplikacija_3 uz parametre komanda i vrijeme.

2. EJB modul ({korisnicko_ime}_modul_1) koji sadrži Entity Bean (EB), Stateless Sesssion Bean (SlSB) - Fasade. Ovo je osnova za **JEDINI NAČIN** na koji se može pristupiti bazi podataka (putem ORM). Ovaj EJB modul moraju koristiti {korisnicko_ime}_aplikacija_2, {korisnicko_ime}_aplikacija_3 i {korisnicko_ime}_aplikacija_4.

3. enterprise aplikacija ({korisnicko_ime}_aplikacija_2) koja ima EJB i Web module (po potrebi). Osnovni EJB modul je ({korisnicko_ime}_modul_1) za rad s bazom podataka. Ostali EJB modul(i) može/gu se kreirati prema potrebi i strukturi rješenja koju odabare pojedini student/ica.

Prvi zadatak realizira se u EJB modulu ({korisnicko_ime}_modul_2) koji u pozadinskom modu u ciklusima s pravilnim intervalima radi određeni skup poslova. U svakoj iteraciji/ciklusu prvo provjerava smije li obaviti svoj posao. To radi pomoću komunikacije s poslužiteljem na mrežnoj utičnici koji je realiziran u {korisnicko_ime}_aplikacija_1. Ako ne smije, tada spava preostalo vrijeme do kraja ciklusa. Ako smije, tada preuzima podatke o polascima/letovima aviona (samo za one koji imaju određen aerodrom slijetanja) za izabrani skup aerodroma koje prate korisnici, zovemo iz vlastiti aerodromi (tablica MYAIRPORTS) putem REST servisa OpenSky (koristi se klasa OSKlijent iz biblioteke NWTiS_2020_REST_lib) i pohranjuju se u tablicu u bazi (AIRPLANES). Na početku se preuzima iz konfiguracije podatak o početnom datumu polazaka aviona. Jedan aerodrom može se javiti u kolekcijama aerodroma više korisnika. Podaci za pojedini aerodrom trebaju se samo jednom preuzeti u pojedinom ciklusu. Prije preuzimanja podataka za pojedini aerodrom potrebno je provjeriti u tablici MYAIRPORTSLOG ima li zapis tog aerodroma za datum koji se obrađuje u ciklusu. Ako postoji onda se preskače aerodrom. Podaci o letovima pojedinog aerodroma preuzimaju se za cijeli dan datuma koji se obrađuje u ciklusu. Nakon preuzimanja podataka za pojedini aerodrom potrebno je upisati u tablicu MYAIRPORTSLOG da je odrađeno preuzimanje za taj aerodrom, datum i broj letova aviona. Na kraju svakog ciklusa datum se povećava za jedan dan. Interval je određen postavkama, kao i vremenski razmak (pauza) između preuzimanja podataka za dva aerodroma u jednom ciklusu. Ako se stigne do važećeg datuma tada pauzira jedan dan.

Napomena: za testiranje, izvršavanje i bodovanje MORAJU biti pripremljeni i vlastitom metodom prikupljeni sljedeći minimalni brojevi zapisa po tablicama:

1. MYAIRPORTS – 20 jedinstvenih aerodroma
2. MYAIRPORTSLOG – 60 dana za svaki aerodrom iz MYAIRPORTS
3. AIRPLANES – 200.000 letova aviona

Aplikacija se smatra nevažećom (0 bodova) ukoliko ne preuzima podatke o letovima ili nisu prikupljeni podaci prema spomenutim minimalnim količinama.

Drugi zadatak web aplikacije je pružanje vlastitog JAX-WS (SOAP) web servisa. Sve operacije moraju kod slanja zahtjeva imati kao prva dva parametra korisničko ime i lozinku (potrebni podaci za autentikaciju) Ostali parametri pišu kod pojedine operacije koje imaju dodatne parametre. Potrebno se držati zadanih osobina i realizirati sljedeće operacije:

1. vraća popis aerodroma koji imaju sličan naziv koji se traži (šalje naziv), vraća java.util.List<Aerodrom>. Ako nema ni jednog aerodroma vraća null.
2. vraća popis aerodroma koji su iz određene države (šalje kod države), vraća java.util.List<Aerodrom>. Ako nema ni jednog aerodroma vraća null.
3. vraća popis svih vlastitih aerodroma (koje prati korisnik), vraća java.util.List<Aerodrom>. Ako nema ni jednog aerodroma vraća null.
4. vraća popis svih letova aviona koji su polijetali sa zadanog aerodroma u određenom razdoblju (šalje se ICAO kod, od, do). Od i do su UNIX timestamp, vraća java.util.List<AvionLeti>. Ako nema ni jednog leta aviona vraća null.
5. vraća popis svih letova izabranog aviona u određenom razdoblju (šalje se ICAO24 kod, od, do). Od i do su UNIX timestamp, vraća java.util.List<AvionLeti>. Ako nema ni jednog leta aviona vraća null.
6. vraća udaljenost u km između dva izabrana aerodroma (šalje se ICAO_1 kod i ICAO_2 kod)

7. vraća popis vlastitih aerodroma koji su udaljeneni od izabranog aerodroma unutar određenih granica u km (npr. 200 i 2000), (šalje se ICAO kod), vraća java.util.List<Aerodrom>. Ako nema ni jednog aerodroma vraća null.

Svaki zahtjev prema ovom web servisu treba biti upisan u tablicu dnevnika rada u bazi podataka.

Potrebno je pripremiti Postman ili SoapUI datoteku za testiranje web servisa koja sadrži pozive svih implementiranih operacija. **Drugi zadatak aplikacije smatra se nevažećim (0 bodova) ukoliko ne postoji funkcionalna datoteka za testiranje**.

Treći zadatak je pružanje krajnje točke za WebSocket, koja sadrži sljedeće operacije:

1. u pravilnim intervalima (konfiguracija) šalje se poruka o ukupnom broju aerodroma za koje se preuzimaju podaci o letovima
2. prima poruku o dodavanju aerodroma određenom korisniku. Obavlja se:
    1. dodaje aerodrom koji prati korisnik
    2. upis u tablicu dnevnika rada u bazi podataka
    3. šalje JMS poruku u red poruka: NWTiS_{korisnicko_ime}_1. Poruka treba biti u obliku TextMessage. Sadržaj poruke je u "application/json" formatu sa sljedećom sintaksom:

        {"id": broj, "korisnik": "id korisnika", "aerodrom": "ICAO aerodroma", "vrijeme": "dd.MM.yyyy hh.mm.ss.SSSS"}

        gdje je id redni broj JMS poruke koja se šalje u taj red.

4. web aplikacija ({korisnicko_ime}_aplikacija_3) koja ima EJB i Web module. Osnovni EJB modul je ({korisnicko_ime}_modul_1) za rad s bazom podataka. Ostali EJB modul(i) može/gu se kreirati prema potrebi i strukturi rješenja koju odabare pojedini student/ica.

   Prvi zadatak je realizacija JAX-RS (RESTful) web servis za rad s aerodromima. Dio operacija obavlja se na temelju baze podataka, dio poziva operacije JAX-WS web servisa iz {korisnicko_ime}_aplikacija_2 ili šalje poruku na krajnju točku WebSocket-a iz {korisnicko_ime}_aplikacija_2. Svaki zahtjev treba biti upisan u tablicu dnevnika rada u bazi podataka. Potrebno se držati zadanih osobina:

   1. sve operacije JAX-RS (RESTful) web servisa moraju kod slanja zahtjeva pripremiti podatke za autentikaciju u atribute zaglavlja pod nazivima „korisnik“ i „lozinka“.
   2. sve operacije JAX-RS web servisa koje trebaju dodatne ulazne podatke (osim putem zaglavlja i parametara u adresi) šalju ih u application/json formatu sa strukturom koja je zadana kod pojedine operacije.
   3. sve operacije JAX-RS web servisa vraćaju odgovor u application/json formatu sa sljedećom strukturom odgovora:
      - ako je operacija u redu, atribut odgovor sadrži niz elemeneta.. Taj niz može biti prazan, može imati jedan ili više elemenata. Struktura elementa ovisi o pojedinoj operaciji. Tako neke vraćaju niz objekata jedne klase, neke objekt treće klase i sl.
        {"odgovor": [{...},{...}...], "status": "10", "poruka": "OK"}
      - ako operacija nije u redu, atribut odgovor sadrži prazan niz elemeneta.
        {"odgovor": [], "status": "40", "poruka": "tekst poruke"}.

   Potrebno je realizirati sljedeće operacije:

   1. GET metoda - osnovna adresa - vraća popis vlastitih aerodroma, a za aerodrom podaci trebaju odgovarati atributima klase Aerodrom. Poziva operaciju JAX-WS web servisa.
   2. POST metoda - osnovna adresa - dodaje aerodrom korisniku koji je pozvao operaciju (šalje se ICAO kod). Šalje se {"icao": "icao kod"}. Ne provodi stvarno dodavanje nego šalje poruku na krajnju točku WebSocket u {korisnicko_ime}_aplikacija_2.
   3. GET metoda - putanja "{icao}" - vraća podatke izabranog aerodroma ako je pridružen korisniku koji je pozvao operaciju. Podaci aerodroma trebaju odgovarati atributima klase Aerodrom.
   4. DELETE metoda - putanja "{icao}" - briše praćenje izabranog aerodroma za korisnika ako je pridružen korisniku koji je pozvao operaciju. Ako aerodrom ima pridružene letove aviona tada je operacija neuspješna.
   5. DELETE metoda - putanja "{icao}/avioni" - briše letove aviona izabranog aerodroma ako je pridružen korisniku koji je pozvao operaciju.
   6. GET metoda - putanja "/svi" - uz parametre naziv i drzava, vraća popis svih aerodroma koji imaju naziv koji sliči nazivu iz parametra ili su iz države. Ako je upisan naziv tada se po njemu pretražuje. Ako je upisana samo država tada se po njoj pretražuje. Ako nije upisan ni jedan parametar tada se uzimaju svi aerodromi na bazi naziva "%". Za aerodrom podaci trebaju odgovarati atributima klase Aerodrom. Poziva potrebnu operaciju JAX-WS web servisa.
   7. POST metoda - putanja "/komande" - uz parametre komanda i vrijeme. U slučaju da je primljen ispravni zahtjev potrebno je poslati JMS poruku u red poruka: NWTiS_{korisnicko_ime}_2. Poruka treba biti u obliku TextMessage. Sadržaj poruke je u "application/json" formatu sa sljedećom sintaksom:

      {"id": broj, "korisnik": "id korisnika", "komanda": "komanda iz parametra", "vrijemePrijema": "dd.MM.yyyy hh.mm.ss.SSSS", "vrijemeSlanja": "dd.MM.yyyy hh.mm.ss.SSSS"}

      gdje je id redni broj JMS poruke koja se šalje u taj red, vrijemePrijema je vrijeme iz parametra, a vrijemeSlanja je trenutno vrijeme.

Potrebno je pripremiti Postman ili SoapUI datoteku za testiranje web servisa koja sadrži pozive svih implementiranih operacija. **Prvi zadatak aplikacije smatra se nevažećim (0 bodova) ukoliko ne postoji funkcionalna datoteka za testiranje**.

Drugi zadatak je povezan s MQTT porukama. Nakon uspješnog autenticiranja korisnik može odabrati rad iz ponuđenih izbornika svojih pogleda. Ako se registrira za prijem MQTT poruka za aerodrome iz korisnikove grupe tada se šalje komanda 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 aerodromu za koji korisnik prikuplja podatke. Ako je sve u redu povećava se broj primljenih MQTT poruka, nakon primljene skupine od n poruka (određeno konfiguracijom) slijedi slanje JMS poruke (naziv reda čekanja NWTiS_{korisnicko_ime}_3) s podacima o rednom broju JMS poruke koja se šalje, vremenu prijema i sadržaju zadnje primljene MQTT poruke u skupini. Poruka treba biti u obliku TextMessage. Sadržaj poruke je u "application/json" formatu sa sljedećom sintaksom:

{"id": broj, "korisnik": "id korisnika", "poruka": sadrzajMQTTPoruke, "vrijeme": "dd.MM.yyyy hh:mm:ss.SSS"}

gdje je id redni broj JMS poruke koja se šalje u taj red. Nakon odjavljivanja korisnika ili isteka sjednice (sesije) potrebno je deregistrirati korisnika za prijem MQTT poruka.

Treći zadatak je web modul koji treba realizirati putem JSF (facelets) ili PrimeFaces. Korisniku na početku stoji na raspolaganju pogled 1. Nakon uspješnog prijavljivanja korisnik može obavljati aktivnosti kroz određene poglede pri čemu svaki od njih sadrži poveznicu na odjavljivanje, nakon kojeg se vraća na prijavljivanje korisnika. Potrebni su sljedeći pogledi:

- pogled 1:
  - registracija korisnika uz validaciju podataka, prijavljivanje korisnika, popis svih korisnika (straničenje, filtriranje). Kod registracije korisnika potrebno je poslati komandu DODAJ; poslužitelju na mrežnoj utičnici u {korisnicko_ime}_aplikacija_1.
- 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_2020 pomoću poslužitelja na mrežnoj utičnici (socket servera) iz {korisnicko_ime}_aplikacija_1 i njime upravljati (registrirati, odjaviti, aktivirati, blokirati grupu, postavlja nove aerodrome za grupu). Obavlja se slanjem pripadajuće komande.
- pogled 3:
  - pregled spremljenih MQTT poruka prijavljenog korisnika uz straničenje (konfiguracijom se određuje broj linija po stranici). Korisnik može izvršiti brisanje svih svojih poruka.
- pogled 4:
  - pregled dnevnika prijavljenog korisnika rada uz straničenje (konfiguracijom se određuje broj linija po stranici).

5. enterprise aplikacija ({korisnicko_ime}_aplikacija_4) koja ima EJB i Web module. Osnovni EJB modul je ({korisnicko_ime}_modul_1) za rad s bazom podataka. Ostali EJB modul(i) može/gu se kreirati prema potrebi i strukturi rješenja koju odabare pojedini student/ica.

   Prvi zadatak realizira se u EJB modulu ({korisnicko_ime}_modul_3) koji sadrži Singleton Session Bean (SB), Stateful Sesssion Bean (SB) i Message-Driven Bean. Preuzimaju se tri vrste JMS poruka: poslane kod dodavanja aerodroma korisniku putem WebSocket poruke, poslane kod prijema komande za poslužitelju mrežne utičnice i poslane kod popunjavanja skupine MQTT poruka. 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.

   Drugi zadatak 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 (pogled 1) i svi pogledi moraju imati poveznicu na taj pogled. Navigacija između pogleda mora biti indirektna na bazi faces-config.xml.

   Potrebno je voditi evidenciju korisničkih zahtjeva pomoću aplikacijskog filtera s upisom u tablicu dnevnika rada 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 stoji na raspolaganju pogled 1. Nakon uspješnog prijavljivanja korisnik može obavljati aktivnosti kroz određene poglede pri čemu svaki od njih sadrži poveznicu na odjavljivanje, nakon kojeg se vraća na prijavljivanje korisnika. Potrebni su sljedeći pogledi:

   - pogled 1:
     - prijavljivanje korisnika
   - pogled 2:
     - pregled uz straničenje i brisanje (svih) korisnikovih spremljenih JMS poruka iz redova čekanja:
       - NWTiS_{korisnicko_ime}_1
       - NWTiS_{korisnicko_ime}_2
       - NWTiS_{korisnicko_ime}_3
   - pogled 3:
     - na vrhu stranice prikazuje se podatak o ukupnom broju aerodroma za koje se preuzimaju podaci o letovima i vrijeme zadnjeg osvježavanja podatka o broju aerodroma. Ti podaci se osvježavaju temeljem WebSocket poruka iz {korisnicko_ime}_aplikacija_2. Pregled vlastitih aerodroma na temelju podataka iz baze podataka. U svakom retku uz podatke o aerodromu treba prikazati broj korisnika koji ga prate (MYAIRPORTS), broj dana za koje su preuzeti podaci o letovima s tog aerodroma (MYAIRPORTSLOG), broj preuzetih letova s tog aerodroma (AIRPLANES). Pregled geolokacijskih na temelju naziva aerodroma i meteo podataka za odabrani aerodrom. Unos dijela naziva za pretraživanje/filtriranje svih aerodrom, prikaz filtriranih aerodroma u padajućem izborniku. Dodavanje odabranog aerodroma za korisnika. Šalje se poruka na krajnju točku WebSocket u {korisnicko_ime}_aplikacija_2.
   - pogled 4:
     - unos intervala vremena (od i do) u standardnom hrvatskom 24 satnom obliku dd.mm.gggg hh:mm. Pregled vlastitih aerodroma na temelju poziva operacije JAX-WS web servisa iz {korisnicko_ime}_aplikacija_2. Pregled letova aviona za odabrani aerodrom u zadanom intervalu na temelju poziva operacije JAX-WS web servisa iz {korisnicko_ime}_aplikacija_2. Prikazuju se podaci o avionu,

vremenu polijetanja, slijetanja, aerodromu slijetanja. Pregled letova za odabrani avion u zadanom intervalu na temelju poziva operacije JAX-WS web servisa iz {korisnicko_ime}_aplikacija_2. Prikazuju se podaci o avionu, vremenu polijetanja, slijetanja, aerodromu slijetanja

- pogled 5:
    - pregled vlastitih aerodroma u padajućem izborniku na temelju slanja zahtjeva JAX-RS web servisu iz {korisnicko_ime}_aplikacija_3. Brisanje praćenja odabranog aerodroma od aktivnog korisnika, na temelju slanja zahtjeva JAX-RS web servisu iz {korisnicko_ime}_aplikacija_3. Brisanje letova aviona odabranog aerodroma, na temelju slanja zahtjeva JAX-RS web servisu iz {korisnicko_ime}_aplikacija_3. Ispis aerodoma čija je udaljenost unutar granica min i maks u odnosu na odabrani aerodrom na temelju poziva operacije JAX-WS web servisa iz {korisnicko_ime}_aplikacija_2. Ispis udaljenosti između dva odabrana aerodoma na temelju poziva operacije JAX-WS web servisa iz {korisnicko_ime}_aplikacija_2. Unos min i maks udaljenosti u km.

Funkcionalnost korisničkog dijela za pogled 2, pogled 3, pogled 4 i pogled 5 treba biti realizirana pomoću Ajax-a. Dodatni bodovi mogu se dobiti ako se u pogledu 3 za prikaz odabranog aerodroma i njegovih podataka (koordinate, trenutna temp, vlaga i sl.) koristi Google Maps JavaScript API ili drugi prikaz u obliku zemljopisne mape.

Instalacijska, programska i komunikacijska arhitektura sustava:

{korisnicko_ime}_aplikacija_1:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Tomcat
- EE osobine: EE7 Web/EE 8 Web
- korisničko sučelje: JSP, JSF (facelets) ili PrimeFaces
- baza podataka: JavaDB - naziv nwtis_{korisnickoime}_bp_1
- rad s bazom podataka: JDBC, SQL
- daje poslužitelja na mrežnoj utičnici (socket server)
- šalje REST zahtjev na NWTiS_{korisnicko_ime}_3 nakon primanja komande za poslužitelj
- koristi SOAP web servis AerodromiWS iz aplikacije NWTiS_2020 za upravljanje aerodromima

{korisnicko_ime}_aplikacija_2:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Glassfish
- EE osobine: EE8
- korisničko sučelje: JSF (facelets) ili PrimeFaces
- baza podataka: JavaDB – naziv nwtis_{korisnickoime}_bp_2
- rad s bazom podataka: ORM (EclipseLink), Criteria API - {korisnicko_ime}_modul_1
- preuzima podatke o avionima za izabrane aerodrome - {korisnicko_ime}_modul_2
- daje JAX-WS web servis za podatke o aerodromima i avionima
- daje krajnju točku za WebSocket za osvježavanje broja aerodroma za koje se preuzimaju podaci o letovima aviona i za prijem poruke kod dodavanja aerodroma korisniku
- šalje JMS poruku u red poruka: NWTiS_{korisnicko_ime}_1 kod prijema poruke na krajnjoj točki WebSocket-a za dodavanja aerodroma korisniku
- koristi poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1 za ispitivanje statusa preuzimanja podataka za aerodrome
- koristi OpenSky Netwotk REST web servis za preuzimanje podataka o avionima za izabrane aerodrome

{korisnicko_ime}_aplikacija_3:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Glassfish
- EE osobine: EE8
- korisničko sučelje: JSF (facelets) ili PrimeFaces
- baza podataka: JavaDB – naziv nwtis_{korisnickoime}_bp_2
- rad s bazom podataka: ORM (EclipseLink), Criteria API - {korisnicko_ime}_modul_1
- daje JAX-RS web servis za rad s aerodromima i avionima itd
- prima i obrađuje MQTT poruke za izabrane aerodrome
- šalje JMS poruku u red poruka: NWTiS_{korisnicko_ime}_2 nakon prijema zahtjeva u JAX-RS na putanji „/komande“
- šalje JMS poruku u red poruka: NWTiS_{korisnicko_ime}_3 nakon punjenja skupine MQTT poruka
- koristi poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1 za upravljanje poslužiteljem i grupom

{korisnicko_ime}_aplikacija_4:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Glassfish
- EE osobine: EE8
- korisničko sučelje: JSF (facelets) ili PrimeFaces
- baza podataka: JavaDB – naziv nwtis_{korisnickoime}_bp_2
- rad s bazom podataka: ORM (EclipseLink), Criteria API - {korisnicko_ime}_modul_1
- prima i sprema JMS poruke u redovima poruka NWTiS_{korisnicko_ime}_1, NWTiS_{korisnicko_ime}_2 i NWTiS_{korisnicko_ime}_3 - {korisnicko_ime}_modul_3
- koristi WebSocket iz NWTiS_{korisnicko_ime}_2 za osvježavanje broj aerodroma za koje se prikupljaju podaci

- koristi JAX-WS web servis iz NWTiS_{korisnicko_ime}_2 za podatke o aerodromima i avionima
- koristi JAX-RS web servis iz NWTiS_{korisnicko_ime}_3 za podatke o aerodromima i avionima0
- koristi OpenWeather Map REST web servis za preuzimanje meteoroloških podataka za aerodrom
- koristi LocationIQ REST web servis za preuzimanje geolokacijskih podataka za naziv aerodroma

Postupak za aktiviranje i korištenje web servisa locationiq.com

1. Dokumentacija: **https://locationiq.com/docs**
2. Upoznati se s ponuđenim modelima: **https://locationiq.com/pricing** **(FREE)**
3. Registrirati se (**https://locationiq.com/register** - Create your account)
4. Popuniti podatke
5. Zapisati podatke za žeton (token)

LocationIQ API – za dobivanje geolokacijskih podataka za adresu

JSON

https://eu1.locationiq.com/v1/search.php?key=YOUR_PRIVATE_TOKEN&q=SEARCH_STRING&format=json

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

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

JSON

https://us1.locationiq.com/v1/reverse.php?key=YOUR_PRIVATE_TOKEN&lat=LATITUDE&lon=LONGITUDE&format=json

Postupak za aktiviranje i korištenje web servisa opensky-network.org

1. Dokumentacija: **https://opensky-network.org/apidoc/**
2. Registrirati se (**https://opensky-network.org/login?view=registration**)
3. Popuniti podatke
4. Zapisati podatke za korisničko ime i lozinku

OpenSky Network API – za dobivanje podataka za letove aviona s aerodroma

JSON

https://USERNAME:PASSWORD@opensky-network.org/api/flights/departure
?airport=EDDF&begin=1517227200&end=1517230800

OpenSky Network API – za dobivanje podataka za letove aviona na aerodrom

JSON

https://USERNAME:PASSWORD@opensky-network.org/api/flights/arrival
?airport=EDDF&begin=1517227200&end=1517230800

OpenSky Network API – za dobivanje podataka za letove aviona

JSON

https://USERNAME:PASSWORD@opensky-network.org/api/flights/aircraft
?icao24=3c675a&begin=1517184000&end=1517270400

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**