Projekt: *NWTiS_2018_2019 v1.3*

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, ali samo u situaciji kada nema ni jednog aviona koje mu je pridružen. 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_2019 (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 i avioni. 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_2019 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_2019 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 avione koji su poletjeli s pojedinog aerodroma
- 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)
- dodati i obrisati avione pojedinom aerodromu 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_2019/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", "avion": "icao24", "oznaka": "callsign", "poruka": "tekst poruke", "vrijeme": "dd.MM.yyyy hh.mm.ss.SSSS"}
```

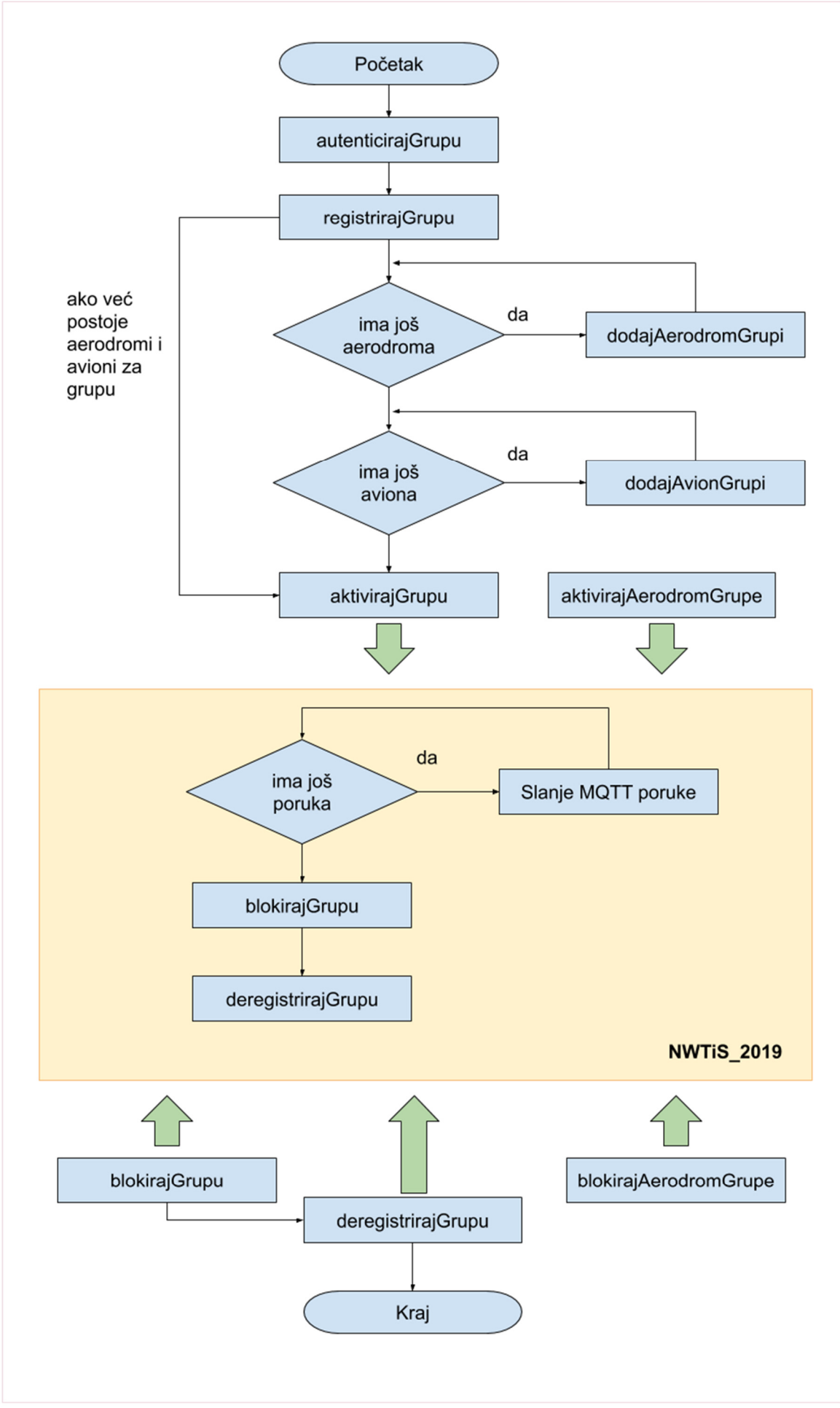
Adresa web konzole MQTT poslužitelja: http://nwtis.foi.hr:61680/

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_2019

Sustav se treba sastojati od sljedećih aplikacija:

1. web aplikacija (`{korisnicko_ime}_aplikacija_1`) bavi se komuniciranjem s NWTiS_2019 te ujedno pruža sučelje ostalim aplikacijama prema NWTiS_2019. U pozadinskom modu (putem slušača aplikacije), pokreće dretvu koja radi u pravilnim ciklusima (postavkom se određuje pravilni vremenski interval preuzimanja podataka pri čemu je jedinica minuta, npr. 2 min, 5 min, 10 min, 30 min, 60 min, ...). Dretva za izabran skup aerodroma preuzima avione koji su s njih poletjeli unutar kliznog vremenskog intervala i imaju određen aerodrom za slijetanje. Podaci se upisuju u bazu podataka u kronološkom redoslijedu prema atributu firstSeen. Inicijalni početak kliznog intervala i trajanje intervala određeni su postavkama. Ako postoji datoteka s podacima rada dretve (naziv datoteke određen je postavkama, a datoteke se treba nalaziti u WEB-INF direktoriju aplikacije) tada se vrijednost početka kliznog intervala postavlja na preuzetu vrijednost iz datoteke. Iz datoteke se preuzima i redni broj ciklusa, a ako je nema tada je 0. Kraj intervala dobije se dodavanjem trajanja intervala početku intervala. Nakon svakog ciklusa dretve potrebno je povećati redni broj ciklusa i spremiti u datoteku rada dretve podatke o sljedećem početku intervala (to je vrijednost trenutnog kraja intervala) i rednom broju ciklusa. Ako je vrijeme početka intervala jednako ili veće od trenutnog vremena, potrebno je početak intervala postaviti na inicijalnu vrijednost. Podaci se upisuju u application/json formatu: {″pocetakIntervala″: vrijeme, ″redniBrojCiklusa″: broj}. Prije upisa u tablicu AIRPLANES u bazi podataka treba provjeriti da li postoji takav podatak u tablici za taj avion na bazi njegovog icao24 i forstSeen. Ako postoji nije potrebno upisati. Podaci o avionima preuzimaju se putem opensky-network.org web servisa.

   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 aerodroma i za svaki od njih minimalno 100 preuzetih podataka o avionima 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) (postavkom 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. Treba biti jasno koji dio aplikacije je upisao pojedini zaspis u dnevnik. 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; {PAUZA; | KRENI; | PASIVNO;
   | AKTIVNO; | STANI; | STANJE; }
   ```

   Objašnjenje komandi:

   - `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 podataka za aerodrome 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 podataka za aerodrome 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 podataka za aerodrome 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 podatke za aerodrome, 12 - preuzima sve komanda i ne preuzima podatke za aerodrome, 13 – preuzima samo poslužiteljske komanda i preuzima podatke za aerodrome, 14 – preuzima samo poslužiteljske komanda i ne preuzima podatke za aerodrome.

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 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, "komanda": "sadržaj komande za poslužitelja",
"vrijeme": "dd.MM.yyyy hh.mm.ss.SSSS"}
```

gdje je id redni broj JMS poruke koja se šalje.

U slučaju ispravne komande za grupu treba pozvati pripadajuću operaciju SOAP web servisa AerodromiWS iz aplikacije NWTiS_2019 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 podacima za aerodrome i avione koji su putem pozadinske dretve spremljeni u bazu podataka ili se poziva vanjski web servis za meteo

podatke. Za podatke o letovima aviona koristi se tablica AIRPLANES u bazi podataka u koju pozadinska dretva upisuje podatke. Za podatke o aerodromima koristi se tablica MYAIRPORTS u bazi podataka u koju REST web servis upisuje podatke. Na raspolaganju stoje dvije grupe operacije:

Osnovne:

- zadnji preuzeti podaci o avionima za izabrani aerodrom. Vraća AvionLeti.
- posljednjih n (n se unosi) podataka o avionima za izabrani aerodrom. Vraća List<AvionLeti>.
- podaci o letovima avionima koji su poletjeli s izabranog aerodroma u nekom vremenskom intervalu (od datuma, do datuma kao timestamp) u vremenskom slijedu kako su avioni prolazili (firstSeen). Vraća List<AvionLeti>.
- podaci o letovima izabranog aviona u nekom vremenskom intervalu (od datuma, do datuma kao timestamp) u vremenskom slijedu kako je avion prolazio (firstSeen). Vraća List<AvionLeti>.
- podaci o letovima izabranog aviona u nekom vremenskom intervalu (od datuma, do datuma kao timestamp). Vraća List<String> naziva aerodroma u vremenskom slijedu kako je avion prolazio (firstSeen).
- važeći meteorološki podaci za izabrani aerodrom (putem openweathermap.org web servisa). Vraća MeteoPodaci.
- dodaj korisnika. Prima Korisnik. Vraća true ako je uspješno ili false.
- ažuriraj korisnika. Prima Korisnik. Vraća true ako je uspješno ili false.
- podaci o korisnicima. Vraća List<Korisnik>.

Napredne:

- podaci o potrebnim letovima od izabranog polazišnog aerodroma do izabranog odredišnog aerodroma u nekom vremenskom intervalu (od datuma, do datuma kao timestamp). Vraća List<AvionLeti> za letove avione u vremenskom slijedu kako je avion prolazio između pojedinihi aerodroma. Ovo bi se mogao koristiti kao planer putovanja od jednog aerodroma do drugog aerodroma pri čemu nije unaprijed poznato kroz koje je ostale aerodrome potrebno prolaziti. Postoji mogućnost direktne veze/leta između ta dva aerodroma no ovdje se prvenstveno očekuje da treba obaviti barem jedno presjedanje. Potrebno je voditi brigu da vrijeme lastSeen kod leta aviona u jedan odredišni aerodrom i vrijeme firstSeen kod leta aviona s njega kao novog polazišnog aerodroma u sljedeći odredišni aerodrom bude veća od minimalno potrebno vremena za presjedanj (postavnom je određeno)
- udaljenost u km između dva izabrana aerodroma
- aerodromi koji su udaljeneni od izabranog aerodroma unutar određenih granica u km (npr. 200 i 2000).

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 aerodromima koja se spremaju u tablicu vlastitih aerodroma u bazi podataka. Svaki zahtjev treba biti autenticiran pomoću parametara (korisnik i lozinka) prema korisničkim podacima u bazi podataka i upisan u tablicu zajedničkog dnevnika rada u bazi podataka. Na raspolaganju stoje:

- GET metoda - osnovna adresa - vraća popis svih aerodroma, a za svaki aerodrom icao, naziv, državu i geo lokaciju u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...},{...}...], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
- POST metoda - osnovna adresa - dodaje aerodrom (šalje se icao kod). Šalju se podaci u application/json formatu {"icao": "icao kod"}. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.

- GET metoda - na bazi putanje "{id}" - za izabrani aerodrom vraća podatke. Vraća podatke u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...}], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
- PUT metoda - na bazi putanje "{id}" - za izabrani aerodrom ažurira podatke (šalju se naziv i adresa, koja služi za preuzimanje geo lokacije). Šalju se podaci u application/json formatu. Vraća podatke u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
- DELETE metoda - na bazi putanje "{id}" - briše izabrani aerodrom. Vraća podatke u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
- GET metoda - na bazi putanje "{id}/avion" - za izabrani aerodrom vraća podatke njegovih aviona. Vraća podatke u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...},{...}...], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
- POST metoda - na bazi putanje "{id}/avion" - dodaje avion(e) aerodromu koji ima icao = id. Šalju se podaci u application/json formatu [{"icao24": "icao24 kod", ...}, ...]. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
- DELETE metoda - na bazi putanje ""{id}/avion/" - briše sve avione za aerodrom koji ima icao = id. Vraća podatke u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
- DELETE metoda - na bazi putanje ""{id}/avion/{aid}" – briše letove izabranog aviona aid = icao24 za aerodrom koji ima icao = id. Vraća podatke u application/json formatu. Struktura odgovora u sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.

Svi aerodromi preuzimaju se putem osnovne adrese. Za izabrani aerodrom šalje se njegov id tako da je to putanja `{id}` u URL strukturi poziva REST web servisa. Rad ovog REST web servisa za preuzimanje, dodavanje, ažuriranje i brisanje aerodroma temelji se na pozivima SOAP web servisa AerodromiWS iz NWTiS_2019 kako bi se podaci sinkronizirali s onima koji su u bazi podataka.

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 Entity Bean (EB), Stateless Sesssion Bean (SB) - Fasade, Singleton Session Bean i Stateful Sesssion Bean. Singleton SB služi za evidenciju aktivnih korisnika.

   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 aerodrome 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 aerodromu za koji korisnik prikuplja podatke. 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, sadržaju primljene MQTT poruke. Poruka treba biti u obliku TextMessage. Sadržaj poruke je u "application/json" formatu sa sljedećom sintaksom:

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

   gdje je id redni broj JMS poruke koja se šalje. 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, ime, 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, prijavljivanje korisnika, ažuriranje vlastitih podataka i vidjeti popis svih korisnika. Sve aktivnosti provode se pomoću REST servisa `{korisnicko_ime}_aplikacija_3`.
   - pogled 2:
     - pregled vlastitih aerodroma, dodavanje pomoću REST servisa `{korisnicko_ime}_aplikacija_1` i upravljati odabranim aerodromom (dodati, obrisati, aktivirati, blokirati, pregled statusa).
     - za odabrani aerodrom na temelju podataka SOAP web servisa iz `{korisnicko_ime}_aplikacija_1` pregled važećih meteoroloških podataka
   - pogled 3:
     - pregled vlastitih aerodroma pomoću REST servisa `{korisnicko_ime}_aplikacija_1`, unos početka i završetka intervala (u formatu dd-MM-yyyy HH:mm:ss), za odabrani aerodrom i u zadanom intervalu na temelju podataka SOAP web servisa iz `{korisnicko_ime}_aplikacija_1`

pregled letova aviona (podaci o polijetanju, slijetanju, aerodromu slijetanja) koji su poletjeli s toga aerodroma
    - za odabrani avion iz pregleda i u zadanom intervalu na temelju podataka SOAP web servisa iz `{korisnicko_ime}_aplikacija_1` pregled letova aviona (podaci o polijetanju, slijetanju, aerodromu slijetanja)
- pogled 4:
    - pregled vlastitih aerodroma pomoću REST servisa `{korisnicko_ime}_aplikacija_1`, za odabrani polazišni i odredišni aerodrom na temelju podataka SOAP web servisa iz `{korisnicko_ime}_aplikacija_1` pregled svih letova, njihovih polazišnih i odredišnih aerodroma, vremena polijetanja (firstSeen) i slijetanja (lastSeen) te vremena između dva leta. Vremena se prikazuju u formatu dd.MM.yyyy HH.mm.ss.
    - za odabrani polazišni i odredišni aerodrom na temelju podataka SOAP web servisa iz `{korisnicko_ime}_aplikacija_1` pregled udaljenosti između njih.
- 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, pogled 3 i pogled 4 treba biti realizirana pomoću Ajax-a. Dodatni bodovi mogu se dobiti ako se u pogledu 2 za prikaz odabranih aerodroma i njihovih podataka (npr. koordinate, trenutna temp, udaljenost i sl.) koristi Google Maps JavaScript API. Dodatni bodovi mogu se dobiti ako se u pogledu 4 za prikaz linija između svih aerodroma koje treba proći između odabranog polazišnog i odredišnog aerodroma 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: primljene komande za poslužitelja 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 application/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. Autentikacija korisnika ostvaruje se komandom na mrežnoj utičnici. Ostale akcije koriste SOAP web servis `{korisnicko_ime}_aplikacija_1`. Na raspolaganju stoje:

   - GET metoda - osnovna adresa - preuzimanje svih korisnika. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...},{...}...], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}. Ne vraća se lozinka!
   - POST metoda - osnovna adresa - dodavanje jednog korisnika - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
   - GET metoda - na bazi putanje "{id}" - preuzimanje jednog korisnika - vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [{...}], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}. Ne vraća se lozinka!
   - GET metoda - na bazi putanje "{id}" i uz parametar "auth" - autentikacija jednog korisnika - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.
   - PUT metoda - na bazi putanje {id} - ažuriranje jednog korisnika - šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je sljedeća: {"odgovor": [], "status": "OK"} | {"status": "ERR", "poruka": "tekst poruke"}.

   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 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_2019 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.

Instalacijska, programska i komunikacijska arhitektura sustava:

{korisnicko_ime}_aplikacija_1:

- Razvojni alat (IDE) kod obrane projekta: NetBeans
- Web poslužitelj: Glassfish
- EE osobine: EE7 Web
- korisničko sučelje: JSP, JSF (facelets) ili PrimeFaces
- baza podataka: MySQL - naziv `nwtis_{korisnickoime}_bp_1`, treba sadržavati tablice: `korisnici`, `airports`, `myairports`, `airplanes`, `dnevnik` i ostale koje su potrebne za rad
- rad s bazom podataka: JDBC, SQL
- daje poslužitelja na mrežnoj utičnici (socket server)
- šalje JMS poruku u red poruka: `NWTiS_{korisnicko_ime}_1` nakon primanja komande za poslužitelj
- daje SOAP web servis za podatke o avionima za izabrani aerodrom i meteorološke podatke za izabrani aerodrom
- daje REST web servis za rad s aerodromima i avionima
- koristi SOAP web servis AerodromiWS iz aplikacije NWTiS_2019 za upravljanje aerodromima
- koristi OpenWeather Map REST web servis za preuzimanje meteoroloških podataka
- koristi LocationIQ REST web servis za preuzimanje geolokacijskih podataka za adresu
- koristi OpenSky Netwotk REST web servis za preuzimanje podataka o avionima za izabrane aerodrome

{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
- prima i obrađuje MQTT poruke za avione izabranih aerodroma
- šalje JMS poruku u red poruka: `NWTiS_{korisnicko_ime}_2` za primljenu MQTT poruku
- koristi REST web servis `{korisnicko_ime}_aplikacija_1` za upravljanje aerodromima
- koristi SOAP web servis `{korisnicko_ime}_aplikacija_1` za meteorološke podatke za odabrani aerodrom
- 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 upravljanje poslužiteljem i grupom
- koristi poslužitelja na mrežnoj utičnici, socket server `{korisnicko_ime}_aplikacija_1` za autenticiranje korisnika
- koristi SOAP web servis `{korisnicko_ime}_aplikacija_1` za dodavanje, ažuriranje i pregled korisnika

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**