Projekt: *NWTiS_2020_2021 v1.0*

Sustav se treba sastojati od sljedećih elemenata:

1. web aplikacija ({korisnicko_ime}_aplikacija_1) predstavlja središnji autentikacijski i autorizacijski servis za ostale aplikacije. Svoj rad temelji se na poslužitelju na mrežnoj utičnici (socket server) na određenom vratima (portu) (postavkom se određuje). Poslužitelj treba biti realiziran kao višedretveni sustav s ograničenim brojem dretvi (postavkom se određuje). Aplikacija ima dvije tablice u svojoj bazi podataka: korisnici i ovlasti. Tablica korisnika sastoji se od stupaca za korisničko ime, lozinku, prezime i ime. Treba sadržavati podatke za minimalno 10 korisnika. Tablica ovlasti/prava sastoji se od stupaca za korisničko ime (veza prema tablici korisnici), područje rada (oznaka određene aplikacije ili dijela unutar aplikacije) i status (ako je važeće/aktivno ili nije). Treba sadržavati podatke za minimalno 7 korisnika i za svakog od njih pridružene ovlasti za min 2 područja. Minimalno jedan korisnik treba imati pridružene ovlasti za sva područja. Zahtjev se temelji na komandama (isključivo u jednom retku). Svaki zahtjev treba biti upisan u tablicu dnevnika rada u bazi podataka koji se nalazi u {korisnicko_ime}_aplikacija_2, što se obavlja slanjem zahtjeva RESTful web servisu. U dnevnik se upisuju vrijeme primitka komande, sadržaj komande, status odgovora (OK ili ERROR nn). Komande mogu biti sljedećih vrsta:

- **ADD korisnik lozinka "Prezime" "Ime"**
  - Npr. ADD pero 123456 "Kos" "Pero"
  - Zadatak: dodavanje korisnika
  - Provjerava postoji li korisnik i ako korisnik postoji vraća određenu informaciju s kodom pogreške. Ako ne postoji dodaje ga u tablicu korisnika u bazu podataka.
  - Korisniku se vraća odgovor OK.
  - Npr. OK

- **AUTHEN korisnik lozinka**
  - Npr. AUTHEN pero 123456
  - Zadatak: autentikacija korisnika tj. provjera korisničkog imena i lozinke
  - Provjerava postoji li korisnik i njemu pridružena lozinka u tablici korisnika. Ako ne postoji, vraća određenu informaciju s kodom pogreške. Ako je u redu, provjerava postoji li važeća sjednica za korisnika. Ako postoji, produljuje joj vrijeme trajanja za vrijednost trenutnog vremena uvećano za postavku sjednica.trajanje. Ako ne postoji, kreira sjednicu s podacima o id sjednice, korisničkom imenu, trenutnom vremenu (ms), vremenu do kada traje sjednica (trenutnom vrijeme + postavka sjednica.trajanje) i maksimalnom broju zahtjeva koje korisnik može realizirati tijekom svoje sjednice (postavkom se određuje). Korisniku se vraća odgovor OK id_sjednice vrijemeDoKadaTrajeSjednica maksBrojZahtjeva (ili brojPreostalihZahtjeva). Vrijeme je u ms.
  - Npr. OK 7 1619827200 30

- **LOGOUT korisnik id_sjednice**
  - Npr. LOGOUT pero 7
  - Zadatak: odjavljivanje korisnika
  - Provjerava postoji li korisnik i ako korisnik ne postoji vraća određenu informaciju s kodom pogreške. Ako postoji korisnik provjerava da li postoji važeća sjednica s id_sjednice. Ako nema, vraća određenu informaciju s kodom pogreške. Ako postoji postavlja da je vrijeme trajanja isteklo u trenutno vrijeme i da je raspoloživi broj zahtjeva 0. Nakon toga briše sjednicu.
  - Korisniku se vraća odgovor OK.
  - Npr. OK

- **GRANT korisnik id_sjednice podrucje korisnikTrazi**
  - Npr. GRANT pero 8 pregledAerodroma mato
  - Zadatak: dodavanje/aktiviranje područja za korisnika, služi kao dozvola
  - Provjerava postoji li korisnik i ako korisnik ne postoji vraća određenu informaciju s kodom pogreške. Ako postoji korisnik provjerava da li postoji važeća sjednica s id_sjednice. Ako nema, vraća određenu informaciju s kodom pogreške. Ako ima smanjuje broj preostalih zahtjeva u sjednici, ako je broj = 0 vraća određenu informaciju s kodom pogreške. Ako je broj > 0 provjerava ima li korisnik korisnikTrazi pridruženo područje. Ako ima i aktivno je vraća određenu informaciju s kodom pogreške. Ako ima i nije aktivno tada ga mijenja u aktivno. Ako nema dodaje ga u tablicu.
  - Korisniku se vraća odgovor OK.
  - Npr. OK

- **REVOKE korisnik id_sjednice podrucje korisnikTrazi**
  - Npr. REVOKE pero 9 pregledAerodroma mato
  - Zadatak: oduzimanje/deaktiviranje područja za korisnika korisnikTrazi
  - Provjerava postoji li korisnik i ako korisnik ne postoji vraća određenu informaciju s kodom pogreške. Ako postoji korisnik provjerava da li postoji važeća sjednica s id_sjednice. Ako nema, vraća određenu informaciju s kodom pogreške. Ako ima smanjuje broj preostalih zahtjeva u sjednici, ako je broj = 0 vraća određenu informaciju s kodom pogreške. Ako je broj > 0 provjerava ima li korisnik korisnikTrazi pridruženo područje. Ako nema, vraća određenu informaciju s kodom pogreške. Ako postoji i nije aktivno, vraća određenu informaciju s kodom pogreške. Ako ima i aktivno je, tada ga mijenja u neaktivno.
  - Korisniku se vraća odgovor OK.
  - Npr. OK

- **RIGHTS korisnik id_sjednice korisnikTrazi**
  - Npr. RIGHTS pero 6 mato
  - Zadatak: vraćanje liste područja koja su pridružena korisniku korisnikTrazi
  - Provjerava postoji li korisnik i ako korisnik ne postoji vraća određenu informaciju s kodom pogreške. Ako postoji korisnik provjerava da li postoji važeća sjednica s id_sjednice. Ako nema, vraća određenu informaciju s kodom pogreške. Ako ima smanjuje broj preostalih zahtjeva u sjednici, ako je broj = 0 vraća određenu informaciju s kodom pogreške. Ako je broj > 0 provjerava ima li korisnik korisnikTrazi aktivno područje/a. Ako nema, vraća određenu informaciju s kodom pogreške.
  - Korisniku se vraća odgovor OK podučje1 područje2 …
  - Npr. OK aerodromi korisnici

- **AUTHOR korisnik id_sjednice podrucje**
  - Npr. AUTHOR pero 7 pregledAerodroma
  - Zadatak: autorizacija korisnika za određeno područje tj. provjera da li korisnik ima pridruženo/aktivno područje
  - Provjerava postoji li korisnik i ako korisnik ne postoji vraća određenu informaciju s kodom pogreške. Ako postoji korisnik provjerava da li postoji važeća sjednica s id_sjednice. Ako nema, vraća određenu informaciju s kodom pogreške. Ako ima smanjuje broj preostalih zahtjeva u sjednici, ako je broj = 0 vraća određenu informaciju s kodom pogreške. Ako je broj > 0 provjerava ima li aktivno područje. Ako nema, vraća određenu informaciju s kodom pogreške.
  - Korisniku se vraća odgovor OK.
  - Npr. OK

- **LIST korisnik id_sjednice korisnikTrazi**
  - Npr. LIST pero 7 mato
  - Zadatak: vraćanje podataka za odabranog korisnika
  - Provjerava postoji li korisnik i ako korisnik ne postoji vraća određenu informaciju s kodom pogreške. Ako postoji korisnik provjerava da li postoji važeća sjednica s id_sjednice. Ako nema, vraća određenu informaciju s kodom pogreške. Ako ima smanjuje broj preostalih zahtjeva u sjednici, ako je broj = 0 vraća određenu informaciju s kodom pogreške. Ako je broj > 0 provjerava postoji li korisnik s korisničkim imenom korisnikTrazi. Ako nema, vraća određenu informaciju s kodom pogreške.
  - Korisniku se vraća odgovor OK "korisnik\tprezime\time". Odgovor predstavlja zapis korisnika unutar navodnika gdje su podaci odvojeni znakom tab tj. \t
  - Npr. OK "mato\tMedved\tMato“

- **LISTALL korisnik id_sjednice**
  - Npr. LISTALL pero 7
  - Zadatak: vraćanje podataka svih korisnika
  - Provjerava postoji li korisnik i ako korisnik ne postoji vraća određenu informaciju s kodom pogreške. Ako postoji korisnik provjerava da li postoji važeća sjednica s id_sjednice. Ako nema, vraća određenu informaciju s kodom pogreške. Ako ima smanjuje broj preostalih zahtjeva u sjednici, ako je broj = 0 vraća određenu informaciju s kodom pogreške.
  - Korisniku se vraća odgovor OK "korisnik1\tprezime1\time1" "korisnik2\tprezime2\time2" ... Odgovor predstavlja pojedinačne zapise korisnika unutar navodnika gdje su podaci odvojeni znakom tab tj. \t
  - Npr. OK "pero\tKos\tPero" "mato\tMedved\tMato"

Kodovi pogrešaka su:

- Kada nema slobodne dretve vraća odgovor ERROR 01 tekst (tekst objašnjava razlog pogreške)
- Kada format komande nije ispravan vraća odgovor ERROR 10 tekst (tekst objašnjava razlog pogreške)
- Kada korisnik ili lozinka ne odgovaraju vraća odgovor ERROR 11 tekst (tekst objašnjava razlog pogreške).
- ~~Kada je broj preostalih zahtjeva u sjednici = 0 vraća odgovor ERROR 12 tekst (tekst objašnjava razlog pogreške).~~
- Kada postoji aktivno područje vraća odgovor ERROR 13 tekst (tekst objašnjava razlog pogreške).
- Kada ne postoji aktivno područje vraća odgovor ERROR 14 tekst (tekst objašnjava razlog pogreške).
- Kada nema važeću sjednicu vraća se odgovor ERROR 15 tekst (tekst objašnjava razlog pogreške).
- Kada je broj preostalih zahtjeva u sjednici = 0 vraća odgovor ERROR 16 tekst (tekst objašnjava razlog pogreške).
- Kada nema traženog korisnika vraća se odgovor ERROR 17 tekst (tekst objašnjava razlog pogreške).
- Kada je bilo koja druga pogreška vraća se odgovor ERROR 18 tekst (tekst objašnjava razlog pogreške).

2. web aplikacija ({korisnicko_ime}_aplikacija_2) bavi se preuzimanjem podataka iz vanjskih servisa te pružanjem vlastitog RESTful web servisa. Ova aplikacije NEMA tablicu korisnika! Potrebno je učitati postavke koje se koriste u radu (poslužitelj, port, …).

   Prvi zadatak je da se u prvoj pozadinskoj dretvi u ciklusima s pravilnim intervalima preuzimaju podaci o polascima aviona (samo za one koji imaju određen aerodrom slijetanja) za izabrani skup aerodroma koje prate korisnici, zovemo ih vlastiti aerodromi (tablica MYAIRPORTS) putem RESTful servisa OpenSky (koristi se klasa OSKlijent iz biblioteke NWTiS_2021_REST_lib). Tablica MYAIRPORTSLOG služi za kontrolu preuzetih podataka pojedinog aerodroma za određeni dan kako se ne bi isti podaci više puta preuzimali. Podaci o letovima aviona s pojedinog aerodroma preuzimaju se za cijeli dan pojedinog datuma koji se obrađuje u ciklusu dretve. Nakon preuzimanja podataka o letovima aviona s pojedinog aerodroma potrebno ih je spremiti u tablicu u bazi podataka (AIRPLANES). Struktura tablica ista je kao i kod laboratorijskih vježbi.

   Drugi zadatak je da se u drugoj pozadinskoj dretvi preuzimaju u ciklusima s pravilnim intervalima važeći meteorološki podaci (temperatura, vlaga, tlak, brzina vjetra, smjer vjetra) za izabrani skup aerodroma koje prate korisnici putem RESTful servisa OpenWeatherMap (koristi se klasa OWMKlijent iz biblioteke NWTiS_2021_REST_lib). Nakon preuzimanja meteoroloških podataka pojedinog aerodroma potrebno ih je spremiti u tablicu u bazi podataka (METEO).

   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
   4. METEO – 7 dana za svaki aerodrom iz MYAIRPORTS. Ciklus preuzimanja ne smije biti kraći od 10 min, a ni dulji od 60 min.

   Aplikacija se smatra nevažećom (0 bodova) ukoliko ne preuzima podatke o letovima i/ili meteo podatke ili nisu prikupljeni podaci prema spomenutim minimalnim količinama. Potrebno je pripremiti datoteku s SQL upitima kojima će se prikazati broj zapisa u tablicama, broj zapisa po danu i sl. kako bi se dokazalo ispunjavanje gornjih uvjeta.

   Treći zadatak aplikacije je da pruža RESTful (JAX-RS) web servis za resurse: korisnici (putanja „korisnici"), aerodromi (putanja „aerodromi"), aerodromi koje prati pojedini korisnik (putanja „mojiAerodromi"), dnevnik rada (putanja „dnevnik"). Potrebno se držati zadanih osobina i realizirati sljedeće operacije:
   1. sve operacije RESTful web servisa moraju kod slanja zahtjeva pripremiti podatke za autentikaciju u atribute zaglavlja pod nazivima „korisnik" i „lozinka". Svaki zahtjev prolazi autentikaciju. To se jedino ne koristi kod dodavanja u resurs „korisnici" i dodavanja u resurs „dnevnik". Za provjeru autentikacije koristi se slanje komande prema {korisnicko_ime}_aplikacija_1.

2. sve operacije RESTful 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 ili je ostavljena na vlastiti izbor.
3. sve operacije RESTful web servisa vraćaju odgovor u application/json formatu pomoću klase Response tako da vraćaju status izvršene operacije i objekt tražene klase kao entitet.
4. Resurs „korisnici“:
    1. GET metoda - osnovna adresa - vraća kolekciju korisnika, a za korisnika podaci trebaju odgovarati atributima klase Korisnik. Za ovo se koristi slanje komande prema {korisnicko_ime}_aplikacija_1.
    2. POST metoda - osnovna adresa - dodaje korisnika u bazu podataka. Ulazni podaci za korisnika trebaju odgovarati atributima klase Korisnik. Za ovo se koristi slanje komande prema {korisnicko_ime}_aplikacija_1. Šalju se samo oni podaci koji su potrebni.
    3. GET metoda - putanja "{korisnik}" - vraća podatke izabranog korisnika. Za korisnika podaci trebaju odgovarati atributima klase Korisnik. Za ovo se koristi slanje komande prema {korisnicko_ime}_aplikacija_1.
5. Resurs „aerodromi“:
    1. GET metoda - osnovna adresa - vraća kolekciju aerodroma, a za aerodrom podaci trebaju odgovarati atributima klase Aerodrom. Za filtriranje podataka mogu se koristiti parametri: naziv (može se koristiti LIKE upit na tablici s %), drzava (točno mora odgovarati),
    2. GET metoda - putanja "{icao}" - vraća podatke izabranog aerodroma. Podaci aerodroma trebaju odgovarati atributima klase Aerodrom.
    3. GET metoda - putanja "{icao}/letovi" - vraća broj prikupljenih letova aviona s izabranog aerodroma.
    4. GET metoda - putanja "{icao}/letovi/{dan}" - vraća prikupljene letova aviona s izabranog aerodroma za određeni dan. Dan je u formatu gggg-mm-dd.
    5. GET metoda - putanja "{icao}/meteoDan/{dan}" - vraća prikupljene meteo podatke za izabrani aerodrom za određeni dan. Dan je u formatu gggg-mm-dd.
    6. GET metoda - putanja "{icao}/meteoVrijeme/{vrijeme}" - vraća prikupljene meteo podatke za izabrani aerodrom na određeno vrijeme ili prve nakon njega. Vrijeme je u timestamp/long formatu.
6. Resurs „mojiAerodromi“:
    1. GET metoda - osnovna adresa - vraća kolekciju aerodroma (bez duplikata) za koje se prikupljaju podaci o letovima aviona. Za aerodrom podaci trebaju odgovarati atributima klase Aerodrom.
    2. GET metoda - putanja "{icao}/prate" - vraća kolekciju korisnika koji su pretplaćeni za podatke izabranog aerodroma. Za korisnika podaci trebaju odgovarati atributima klase Korisnik. Za podatke o korisnicima koristi slanje komande prema {korisnicko_ime}_aplikacija_1.
    3. GET metoda - putanja "{korisnik}/prati" - vraća kolekciju aerodroma koje prati izabrani korisnik. Za aerodrom podaci trebaju odgovarati atributima klase Aerodrom.

4. POST metoda - putanja "{korisnik}/prati" – odabranom korisniku dodaje aerodrom kojeg će pratiti. Za aerodrom podaci trebaju odgovarati atributima klase Aerodrom.
5. DELETE metoda - putanja "{korisnik}/prati/{icao}" – odabranom korisniku briše aerodrom tako da ga više ne prati.

7. Resurs „dnevnik“:
    1. POST metoda - osnovna adresa - dodaje zapis u dnevnik rada u bazu podataka.
    2. GET metoda - putanja "{korisnik}" - vraća kolekciju zapisa u dnevnik rada za izabranog korisnika. Mogu se koristiti parametri od i do za određivanje vremenskog intervala za koji se uzimaju zapisi dnevnika rada. Mogu se koristiti parametri pomak i stranica za određivanje broja zapisa koji se preskače i broj zapisa koji se preuzima.
    3. GET metoda - putanja "{korisnik}/broj" - vraća broj zapisa u dnevniku rada za izabranog korisnika. Mogu se koristiti parametri od i do za određivanje vremenskog intervala za koji se uzimaju zapisi dnevnika rada.

Za testiranje web servisa potrebno je pripremiti Postman datoteku ili SoapUI datoteku ili sh datoteku s curl komandama koja sadrži pozive svih implementiranih operacija. **Treći dio aplikacije smatra se nevažećim (0 bodova) ukoliko ne postoji funkcionalna datoteka za testiranje**.

3. web aplikacija ({korisnicko_ime}_aplikacija_3) ima dva zadatka. Potrebno je učitati postavke koje se koriste u radu (poslužitelj, port, …).

   Prvi zadatak bavi se administrativnim poslovima kroz korisničko sučelje realizirano s Jakarta MVC. Ova aplikacija nema vlastitu bazu podataka. Svoj rada temelji na slanju komandi prema {korisnicko_ime}_aplikacija_1 i slanju zahtjeva na RESTful web servis iz {korisnicko_ime}_aplikacija_2. Potrebno se držati zadanih osobina i realizirati sljedeće dijelove:

   - korisnički dio (osim registracije i prijavljivanja korisnika) treba biti otvoren samo za prijavljene korisnike. Koristi se aplikacijski filtar za ograničavanje pristupa neprijavljenih korisnika.
   - pogled 3.1:
     - registraciju korisnika. Poziva se operacija RESTful web servisa iz {korisnicko_ime}_aplikacija_2.
   - pogled 3.2:
     - prijavljivanje korisnika. Za ovaj dio potrebno je provesti autentikaciju korisnika. Šalje se komanda na {korisnicko_ime}_aplikacija_1. Ako je bilo uspješno prijavljivanje korisnika spaja se na WebSocket krajnju točku „/login" i šalje se poruka s podacima o korisniku i aplikaciji s koje je obavio prijavljivanje. Otvara se izbornik koji daje poveznice na poglede (3.4, 3.5 i 3.6) te na odjavljivanje korisnika. Kod odjavljivanja šalje se poruka na WebSocket krajnju točku „/login" s podacima o korisniku i aplikaciji te se veza zatvara.
   - pogled 3.3:
     - dodavanje/aktiviranje područja za korisnika. Za ovaj dio potrebno je provesti autorizaciju korisnika. Prijavljeni korisnik mora imati aktivno područje „administracija" da bi mogao pokrenuti ovaj pogled. Šalje se komanda na {korisnicko_ime}_aplikacija_1. Prikazani su svi korisnici i raspoloživa područja. Korisnici se dobiju pozivom operacije RESTful web servisa iz {korisnicko_ime}_aplikacija_2. Korisnik može odabrati jednog korisnika te mu dodati/aktivirati ili oduzeti/deaktivirati odabrano područje. Šalje se komanda na {korisnicko_ime}_aplikacija_1.
   - pogled 3.4:
     - pregled korisnika koji prate određeni aerodrom. Za ovaj dio potrebno je provesti autorizaciju korisnika. Prijavljeni korisnik mora imati aktivno područje „administracijaAerodroma" da bi mogao pokrenuti ovaj pogled. Šalje se komanda na {korisnicko_ime}_aplikacija_1. Prikazani su aerodromi koje korisnici prati. Korisnik može odabrati jedan aerodrom te prikazati korisnike koji prate odabrani aerodrom. Korisnik može odabrati jedan aerodrom i može prestati pratiti odabrani aerodrom. Korisnik može upisati ICAO aerodroma (ili putem odabira iz izbornika svih aerodroma) i dodati ga u skup aerodroma koje on prati. Ispisuje se naziv aerodroma koji je odabran. Sve aktivnosti provode se pozivom operacije RESTful web servisa iz {korisnicko_ime}_aplikacija_2. Kod dodavanja praćenja aerodroma ili prestanak praćenja aerodroma šalje se JMS poruka u red NWTiS_{korisnicko_ime}_1 s podacima o korisniku, aerodromu i akciji.

- pogled 3.5:
    - slanje slobodno upisane komande na {korisnicko_ime}_aplikacija_1.

Drugi zadatak je pružanje krajnje točke za WebSocket „/vrijeme“, koja u pravilnim vremenskim intervalima (postavkom se određuje, između 10 i 60 sek) šalje poruku svim prijavljenim klijentima o trenutnom vremenu na poslužitelju (dd.mm.gggg hh:mm:ss).

4. enterprise aplikacija ({korisnicko_ime}_aplikacija_4) koja ima EJB i Web modul. Potrebno je učitati postavke koje se koriste u radu (poslužitelj, port, …).

   EJB modul ({korisnicko_ime}_modul_4_1) zadužen je za poslovnu logiku i sadrži potrebne Singleton Session Bean (SB), Stateful Sesssion Bean (SB) i Message-Driven Bean.

   Prvi zadatak je da se preuzimaju JMS poruke poslane kod dodavanja aerodroma korisniku za praćenje ili prestanak praćenja aerodroma. 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. Postoji mogućnost pražnjenja svih spremljenih JMS poruka.

   Web modul ({korisnicko_ime}_modul_4_2) 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 tzv. labele i dobiti jezične prijevode. Jezik se odabire na početnoj stranici aplikacije (pogled 4.1) i svi pogledi moraju imati poveznicu na taj pogled. Navigacija između pogleda mora biti indirektna na bazi faces-config.xml.. Korisniku na početku stoji na raspolaganju pogled 4.1. 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 4.1:
     - prijavljivanje korisnika. Šalje se komanda na {korisnicko_ime}_aplikacija_1. Ako je bilo uspješno prijavljivanje korisnika spaja se na WebSocket krajnju točku „/login“ i šalje se poruka s podacima o korisniku i aplikaciji s koje je obavio prijavljivanje. Sakriva se obrazac za prijavljivanje. Otvara se izbornik koji daje poveznice na ostale poglede te na odjavljivanje korisnika. Kod odjavljivanja šalje se poruka na WebSocket krajnju točku „/login“ s podacima o korisniku i aplikaciji te se veza zatvara. Sakrivaju se izbornici i otvara se obrazac za prijavljivanje. Izbornik za odabir jezika treba stalno biti aktivan.
   - pogled 4.2:
     - popis svih korisnika (straničenje, filtriranje). Poziva operaciju RESTful web servisa iz {korisnicko_ime}_aplikacija_2. Za ovaj dio potrebno je provesti autorizaciju korisnika. Prijavljeni korisnik mora imati aktivno područje „pregledKorisnik“ da bi mogao pokrenuti ovaj pogled. Šalje se komanda na {korisnicko_ime}_aplikacija_1.
   - pogled 4.3:
     - pregled uz straničenje i brisanje (svih) spremljenih JMS poruka iz reda čekanja NWTiS_{korisnicko_ime}_1. Za ovaj dio potrebno je provesti autorizaciju korisnika. Prijavljeni korisnik mora imati aktivno područje „pregledJMS“ da bi mogao pokrenuti ovaj pogled. Šalje se komanda na {korisnicko_ime}_aplikacija_1.

- pogled 4.4:
    - pregled dnevnika rada uz straničenje (konfiguracijom se određuje broj linija po stranici). Zbog broja zapisa ne preuzimaju se svi podaci odjednom kako bi se u JavaScript realiziralo straničenje. Potrebno je preuzimati samo podatke jedne stranice. Potrebno je minimalno pripremiti poveznice za skok na prvu stranicu, na prethodnu stranicu, na sljedeću stranicu i na zadnju stranicu. Može se odrediti vremenski interval (od-do) za koji se traže podaci iz dnevnika. Šalje se zahtjev na RESTful web servis iz {korisnicko_ime}_aplikacija_2. Za ovaj dio potrebno je provesti autorizaciju korisnika. Prijavljeni korisnik mora imati aktivno područje „pregledDnevnik" da bi mogao pokrenuti ovaj pogled. Šalje se komanda na {korisnicko_ime}_aplikacija_1.
- pogled 4.5:
    - na vrhu stranice prikazuje se podatak o ukupnom broju aktivnih korisnika i trenutnom vremenu na poslužitelju na kojem se izvršava {korisnicko_ime}_aplikacija_3. U nastavku slijedi tablični prikaz podataka aktivnih korisnika, aplikacija u kojoj su prijavljeni i vrijeme prijavljivanja. Ti podaci se osvježavaju temeljem WebSocket poruka iz {korisnicko_ime}_aplikacija_3 na krajnjim točkama „/login" i „/vrijeme". Za ovaj dio potrebno je provesti autorizaciju korisnika. Prijavljeni korisnik mora imati aktivno područje „pregledAktivnihKorisnika" da bi mogao pokrenuti ovaj pogled. Šalje se komanda na {korisnicko_ime}_aplikacija_1.
- pogled 4.6:
    - pregled vlastitih aerodroma (koje prati) u padajućem izborniku (ili tabličnom prikazu) na temelju slanja zahtjeva RESTful web servisu iz {korisnicko_ime}_aplikacija_2. Za odabrani aerodrom i upisan datum prikazuju se prikupljeni letovi aviona. Podaci se preuzimaju na temelju slanja zahtjeva RESTful web servisu iz {korisnicko_ime}_aplikacija_2. Za odabrani aerodrom i upisan datum (dd.mm.gggg) prikazuju se prikupljeni meteo podaci. Podaci se preuzimaju na temelju slanja zahtjeva RESTful web servisu iz {korisnicko_ime}_aplikacija_2. Za odabrani aerodrom i upisano vrijeme (dd.mm.gggg hh:mm:ss) prikazuju se prikupljeni meteo podaci. Podaci se preuzimaju na temelju slanja zahtjeva RESTful web servisu iz {korisnicko_ime}_aplikacija_2. Za ovaj dio potrebno je provesti autorizaciju korisnika. Prijavljeni korisnik mora imati aktivno područje „pregledAerodroma" da bi mogao pokrenuti ovaj pogled. Šalje se komanda na {korisnicko_ime}_aplikacija_1.

Funkcionalnost korisničkog dijela za poglede 4.1, 4.2, 4.3, 4.4, 4.5 i 4.6 treba biti realizirana pomoću Ajax-a.

Drugi zadatak Web modula je pružanje krajnje točke za WebSocket „/login", koja prima poruke kod prijavljivanja i odjavljivanja korisnika te vodi evidenciju aktivnih korisnika u Singleton Session Bean (SB).

Instalacijska, programska i komunikacijska arhitektura sustava:

{korisnicko_ime}_aplikacija_1:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Tomcat
- EE osobine: Jakarta EE9 Web
- korisničko sučelje: nema
- baza podataka: JavaDB - naziv nwtis_{korisnickoime}_bp_1
- rad s bazom podataka: JDBC, SQL
- pruža poslužitelja na mrežnoj utičnici (socket server)
- šalje REST zahtjev na NWTiS_{korisnicko_ime}_2 nakon primanja komande

{korisnicko_ime}_aplikacija_2:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Tomcat
- EE osobine: Jakarta EE9 Web
- korisničko sučelje: nema
- baza podataka: MySQL – naziv nwtis_{korisnickoime}_bp_2
- rad s bazom podataka: JDBC, SQL
- preuzima podatke o letovima aviona s izabranih aerodroma
- preuzima podatke o meteo podacima za izabrane aerodrome
- pruža RESTful web servis za rad s korisnicima, aerodromima i dnevnikom rada
- koristi poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1 za autentikaciju i autorizaciju korisnika, prikupljanje podataka o korisnicima
- koristi OpenSky Netwotk REST web servis za preuzimanje podataka o letovima aviona za izabrane aerodrome
- koristi OpenWeatherMap REST web servis za preuzimanje meteo podataka za izabrane aerodrome

{korisnicko_ime}_aplikacija_3:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Glassfish EE Server
- EE osobine: Jakarta EE9
- korisničko sučelje: Jakarta MVC
- baza podataka: nema
- pruža WebSocket krajnju točku „/vrijeme“ za periodično slanje informacije o trenutnom vremenu na poslužitelju
- koristi WebSocket krajnju točku „/login“ iz NWTiS_{korisnicko_ime}_3 za osvježavanje aktivnih korisnika (kod prijave i odjave korisnika)
- šalje JMS poruku u red poruka: NWTiS_{korisnicko_ime}_1 nakon dodavanja aerodroma za praćenje ili prestanak praćenja aerodroma
- koristi poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1 za autentikaciju i autorizaciju korisnika, dodavanje i oduzimanje područja kao ovlasti
- koristi RESTful web servis iz {korisnicko_ime}_aplikacija_2 za popis korisnika

{korisnicko_ime}_aplikacija_4:

- Razvojni alat (IDE) kod obrane projekta: NetBeans s Maven
- Web poslužitelj: Glassfish EE Server
- EE osobine: Jakarta EE9
- korisničko sučelje: JSF (facelets) ili PrimeFaces
- baza podataka: nema
- prima i sprema JMS poruke u red poruka NWTiS_{korisnicko_ime}_1

- pruža WebSocket krajnju točku „/login“ za evidenciju aktivnih korisnika (kod prijave i odjave korisnika)
- koristi WebSocket krajnju točku „/vrijeme“ iz NWTiS_{korisnicko_ime}_3 za primanje informacije o trenutnom vremenu na poslužitelju
- koristi RESTful web servis iz {korisnicko_ime}_aplikacija_2 za popis korisnika
- koristi RESTful web servis iz {korisnicko_ime}_aplikacija_2 za pregled dnevnika
- koristi RESTful web servis iz {korisnicko_ime}_aplikacija_2 za pregled aerodroma, letova aviona i meteo podataka

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**