Projekt: *NWTiS_2022_2023 v1.0*

Sustav se treba sastojati od sljedećih elemenata:

1. aplikacija **{korisnicko_ime}_aplikacija_1** predstavlja servis upravljanje radom cijelog projekta i za izračunavanje udaljenosti između aerodroma koji koriste ostale aplikacije. Svoj rad temelji na poslužitelju na mrežnoj utičnici (socket server) na određenim mrežnim vratima (portu) (postavkom se određuje). Poslužitelj treba biti realiziran kao višedretveni sustav s ograničenim brojem dretvi (postavkom se određuje). Aplikacija nema bazu podataka. Aplikacija može primiti komandu KRAJ u bilo kojem načinu rada i treba prekinuti rad poslužitelja. Aplikacija može primiti komandu STATUS u bilo kojem načinu rada i treba vratiti trenutni status poslužitelja. Aplikacija je na početku u statusu pauze i čeka svoju inicijalizaciju koja se provodi komandom INIT. Poslužitelj dok ne primi komandu INIT odbija ostale standardne komande. Nakon INIT može primiti sve komande dok ponovno ne pređe u pauzu. Na početku poslužitelj NE ispisuje podatke o svom radu na standardni izlaz. Komandom INFO odobrava se (DA) ili zabranjuje (NE) ispis na standardni izlaz. Komande koje treba implementirati:

- **STATUS**
  - Npr. STATUS
  - Zadatak: vraća trenutni status poslužitelja
  - Korisniku se vraća odgovor OK n. Status n: 0 – pauza, 1 – aktivan
  - Npr. OK 2

- **KRAJ**
  - Npr. KRAJ
  - Zadatak: prekida prijema novih komandi, čeka na završetak rada aktivnih dretvi i na kraju prekida rad poslužitelja
  - Korisniku se vraća odgovor OK.
  - Npr. OK
- **INIT**
  - Npr. INIT
  - Zadatak: inicijalizacija poslužitelja i prijelaz u aktivni status, postavljanje brojača obrađenih zahtjeva za izračun udaljenosti na 0
  - Korisniku se vraća odgovor OK
  - Npr. OK
- **PAUZA**
  - Npr. PAUZA
  - Zadatak: prebacivanje poslužitelja u pauziranje
  - Korisniku se vraća odgovor OK i broj zahtjeva za izračun udaljenosti koje je obradio od zadnjeg prijelaza u aktivni status
  - Npr. OK 27

- **INFO [DA|NE]**
  - Npr. INFO DA
  - Zadatak: ako je DA time se dopušta ispis primljene komande na standardni izlaz. Ako je NE time se prekida ispis primljene komande na standardni izlaz.
  - Korisniku se vraća odgovor OK
  - Npr. OK

- **UDALJENOST gpsSirina1 gosDuzina1 gpsSirina2 gosDuzina2**
  - npr. UDALJENOST 46.30771 16.33808 46.02419 15.90968
  - Provjera da li ispravni podaci. Ako su ispravni izračunava udaljenost između dviju lokacija na temelju njihovih GPS pozicija. Algoritam treba uzeti u obzir zakrivljenost Zemlje. Vraća broj km u formatu ddddd.dd. Korisniku se vraća odgovor `OK broj.` Ako se radi o ispravnom zahtjevu povećava broj zahtjeva za izračun udaljenosti.
  - Npr. `OK 271.4`

Kodovi pogrešaka su:

- Kada poslužitelj pauzira a stigla je komanda (PAUZA, INFO, UDALJENOST) vraća odgovor ERROR 01 tekst (tekst objašnjava razlog pogreške)
- Kada je poslužitelj aktivan a stigla je komanda (INIT) vraća odgovor ERROR 02 tekst (tekst objašnjava razlog pogreške)
- Kada je stigla komanda INFO DA a poslužitelj ispisuje podatke o svom radu na standardni izlaz vraća odgovor ERROR 03 tekst (tekst objašnjava razlog pogreške)
- Kada je stigla komanda INFO NE a poslužitelj ne ispisuje podatke o svom radu na standardni izlaz vraća odgovor ERROR 04 tekst (tekst objašnjava razlog pogreške)
- U slučaju bilo koje ostale pogreške vraća odgovor ERROR 05 tekst (tekst objašnjava razlog pogreške)

2. web aplikacija **{korisnicko_ime}_aplikacija_2** učitava konfiguracijske podatke putem slušača aplikacije kod pokretanje aplikacije i upisuje ih u atribut konteksta pod nazivom „konfig“. Naziv datoteke konfiguracijskih podataka zapisan je u web.xml kao inicijalni parametar konteksta „konfiguracija“ (npr. datoteka NWTiS.db.config_2.xml). Nakon učitanih konfiguracijskih podataka provjerava status poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1. Ako nije aktivan, program prekida rad. Aplikacija pruža RESTful (JAX-RS) servis pod krajnjom točkom „api“. Svi zahtjevi trebaju se upisivati u tablicu „DNEVNIK“ putem aplikacijskog filtera (vrsta AP2). Raspoloživi aerodromi nalaze se u tablici „AIRPORTS“. Udaljenosti između dva aerodroma uz udaljenosti unutar država nalaze se u tablici „AIRPORTS_DISTANCE_MATRIX“. Postavkama se određuje adresa i mrežna vrata poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1.

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. POST metode RESTful web servisa šalje podatke u application/json formatu sa strukturom koja je zadana kod pojedine operacije ili je ostavljena na vlastiti izbor.

RESTful servis koji pruža web aplikacija ima aerodrome kao resurs (putanja „aerodromi“). Treba implementirati sljedeće operacije:

a. get metoda – osnovna putanja, može imati parametre upita pod nazivima traziNaziv, traziDrzavu, odBroja i broj. Parametar traziNaziv služi za filtriranje aerodroma na temelju sličnosti naziva aerodroma (SQL upit LIKE %traziNaziv%). Parametar traziDrzavu služi za filtriranje aerodroma na temelju točne oznake države u kojoj je aerodrom. Oba parametra mogu biti prazna i tada nema filtriranja. Može jedan biti prazan i tada se filtriranje provodi po drugom parametru koji nije prazan. Mogu oba parametra imati vrijednosti i tada oba služe za filtriranje. Vraća podatke za aerodrome uz straničenje. Ako nisu upisani parametri odBroja i broj tada je odBroja jednak 1, a broj jednak 20.

b. get metoda – putanja {icao}. Varijabilni dio icao je parametar putanje. Vraća podatke za odabrani aerodrom.

c. get metoda – putanja {icaoOd}/{icaoDo}. Varijabilni dio icaoOd i icaoDo su parametri putanje. Vraća podatke za udaljenosti između odabranih aerodroma unutar država preko kojih se leti.

d. get metoda – putanja {icao}/udaljenosti. Varijabilni dio icao je parametar putanje. Može imati parametre upita pod nazivima odBroja i broj. Vraća podatke za udaljenosti između odabranog aerodroma i svih ostalih aerodroma uz straničenje. Ako nisu upisani parametri tada je odBroja jednak 1, a broj jednak 20.

e. get metoda – putanja {icaoOd}/izracunaj/{icaoDo}. Varijabilni dio icaoOd i icaoDo su parametri putanje. Šalje zahtjev na {korisnicko_ime}_aplikacija_1 s komandom
UDALJENOST gpsSirina1 gosDuzina1 gpsSirina2 gosDuzina2
Podatke za GPS lokacije oba aerodroma preuzima iz tablice „AIRPORTS“. Vraća podatke za udaljenost između odabranih aerodroma.

f. get metoda – putanja {icaoOd }/udaljenost1/{icaoDo}. Varijabilni dio icaoOd i icaoDo su parametri putanje. Prvo šalje zahtjev na {korisnicko_ime}_aplikacija_1 s komandom
UDALJENOST gpsSirina1 gosDuzina1 gpsSirina2 gosDuzina2
Podatke za GPS lokacije oba aerodroma preuzima iz tablice „AIRPORTS“. S dobivenim

podatkom za udaljenost između odabranih aerodroma traže se svi aerodromi koji su u državi u kojoj je aerodrom icaoDo i čija je udaljenosti manja od dobivene udaljenosti odredišnog aerodroma. Za udaljenost između dva aerodroma šalje zahtjev na {korisnicko_ime}_aplikacija_1 s komandom
UDALJENOST gpsSirina1 gosDuzina1 gpsSirina2 gosDuzina2
Podatke za GPS lokacije oba aerodroma preuzima iz tablice „AIRPORTS“. Vraća podatke za udaljenosti između odabranog aerodroma i svakog aerodroma iz države iz koje je icaoDo čija je udaljenosti manja od udaljenosti između icaoOd i icaoDo.

g. get metoda – putanja {icaoOd }/udaljenost2. Varijabilni dio icaoOd je parametar putanje. Mora imati parametre upita pod nazivima drzava i km. Vraća aerodrome koji su u određenoj državi (parametar drzava) i čija je udaljenosti manja od zadane (parametar km) u odnosu na zadani aerodrom icaoOd. Za udaljenost između dva aerodroma šalje zahtjev na {korisnicko_ime}_aplikacija_1 s komandom
UDALJENOST gpsSirina1 gosDuzina1 gpsSirina2 gosDuzina2
Podatke za GPS lokacije oba aerodroma preuzima iz tablice „AIRPORTS“. Vraća podatke za udaljenosti između odabranog aerodroma i svakog aerodroma iz određene države čija je udaljenosti manja od zadane.

RESTful servis koji pruža web aplikacija za upravljanje poslužiteljem na mrežnoj utičnici (putanja „nadzor“). Treba implementirati sljedeće operacije:

a. GET metoda - osnovna adresa – šalje komandu STATUS na poslužitelj {korisnicko_ime}_aplikacija_1. Vraća status 200 i pripadajući JSON sadržaj {status: broj, opis: tekst} ako je komanda uspješna. Vraća status 400 ako nije uspješna uz tekst pogreške koji je primljen od poslužitelja.

b. GET metoda - putanja "{komanda}" - šalje komandu [KRAJ | INIT | PAUZA] na poslužitelj {korisnicko_ime}_aplikacija_1. Vraća status 200 i pripadajući JSON sadržaj {status: broj, opis: tekst} ako je komanda uspješna. Vraća status 400 ako nije uspješna uz tekst pogreške koji je primljen od poslužitelja.

c. GET metoda - putanja "INFO/{vrsta}" - šalje komandu INFO uz vrstu [DA|NE] na poslužitelj {korisnicko_ime}_aplikacija_1. Vraća status 200 i pripadajući JSON sadržaj {status: broj, opis: tekst} ako je komanda uspješna. Vraća status 400 ako nije uspješna uz tekst pogreške koji je primljen od poslužitelja.

RESTful servis koji pruža web aplikacija za pristup do zapisa u tablici „DNEVNIK“ (putanja „dnevnik“). Treba implementirati sljedeće operacije:

a. get metoda – osnovna putanja, može imati parametre upita pod nazivima vrsta (JAX-RS, JAX-WS, UI), odBroja i broj. Parametar vrsta služi za filtriranje aerodroma na temelju porijekla upisa podataka. Vraća podatke za dnevnik uz straničenje. Ako nisu upisani parametri tada je odBroja jednak 1, a broj jednak 20.

b. post metoda – osnovna putanja. Upisuje zapis u tablicu.

Za testiranje RESTful servisa potrebno je pripremiti Postman ili Soap ili Insomnia datoteku ili sh datoteku s curl komandama koja sadrži pozive svih implementiranih operacija. **Aplikacija se smatra nevažećim (0 bodova) ukoliko ne postoji funkcionalna datoteka za testiranje**.

3. web aplikacija **{korisnicko_ime}_aplikacija_3** učitava konfiguracijske podatke putem slušača aplikacije kod pokretanje aplikacije i upisuje ih u atribut konteksta pod nazivom „konfig“. Naziv datoteke konfiguracijskih podataka zapisan je u web.xml kao inicijalni parametar konteksta „konfiguracija“ (npr. datoteka NWTiS.db.config_3.xml). Nakon učitanih konfiguracijskih podataka provjerava status poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1. Ako nije aktivan, program prekida rad. Ako je sve u redu potrebno je pokrenuti dretvu. Kod zaustavljanja aplikacije potrebno je zaustaviti dretvu. Dretva radi u ciklusima identičnog trajanja (dretva ima svoj radni dio u ciklusu, a spava preostalo vrijeme trajanja ciklusa). Zadano je trajanje za ciklus (postavka „ciklus.trajanje“, u sekundama). Dretva ima zadatak da u svakom ciklusu preuzme polaske aviona na određeni dan za skup aerodroma koji je određen tablicom AERODROMI_LETOVI. Na kraju svakog ciklusa šalje se JMS poruka s TextMessage sadržajem „Na dan: dd.mm.gggg preuzeto ukupno x letova aviona. JMS poruka šalje se u red poruka „jms/NWTiS_{korisnicko_ime}“. Početni dan je određen postavkom preuzimanje.od, a završni dan je određen postavkom preuzimanje.od. U svakom ciklusu dan se povećava za 1. Dretva na početku provjerava zadnji zapis u tablici kako bi se utvrdio za koji dan je bilo zadnje preuzimanje. Ako je zadnji dan veći od dana iz postavke preuzimanje.od tada je zadnji dan uvećan za jedan dan s kojim se nastavlja rad. Ako je zadnji dan manji od dana iz postavke preuzimanje.od tada je dan iz postavke preuzimanje.od s kojim se nastavlja rad. Podaci o polascima letova upisuju se u tablicu LETOVI_POLASCI. Za preuzimanje podataka o polascima letova s aerodroma koristi se Java biblioteka nwtis_rest_lib v 3.0.0 koja se nalazi u repozitoriju http://nwtis.foi.hr:8088/repository/nwtis_2023/[1]. Biblioteka se temelji na RESTful servisu koji pruža OpenSky Network te predstavlja njegov omotač. JavaDoc biblioteke nalazi se na adresi https://nwtis.foi.hr/NWTiS/apidocs/nwtis_rest_lib/3.0.0/[2]. Za korištenje OpenSky Network i nwtis_rest_lib potrebno je obaviti registraciju kako bi se dobili korisnički podaci i spremili u konfiguracijsku datoteku (postavke „OpenSkyNetwork.korisnik“ i „OpenSkyNetwork.lozinka“).

[1] Dok poslužitelj neće biti ponovno osposobljen vrijede upute koje su dane kod 3. zadaće i nalaze se na adresi https://elf.foi.hr/mod/page/view.php?id=111289

[2] Dok poslužitelj neće biti ponovno osposobljen može se preuzeti zip datoteka JavaDoc koja je dana kod 3. zadaće i nalazi se na adresi https://elf.foi.hr/mod/resource/view.php?id=111306

Aplikacija se smatra nevažećim (0 bodova) ukoliko ne postoji:

- minimalno 20 aerodroma za preuzimanje (EBBR, EDDF, EDDM, EGGP, EGLL, EIDW, EPWA, GCLP, HEGN, LDZA, LEBL, LEPA, LFPG, EDDS, LIPZ, LOWW, LTBJ, LZLH, LJLJ, OMDB). Može više aerodroma no prethodno prikazani moraju biti.
- minimalno 50.000 preuzetih polazaka s aerodroma (bez ponavljanja istih)
- minimalno 10 dana u slijedu u cijelosti za koje su preuzeti podaci polazaka za sve aerodrome
- datoteka s pripremljen SQL komandama kojima se dokazuju:
  - prethodni uvjeti
  - broj preuzetih podataka po danima za sve aerodrome sortirano po danu (npr. 01.01.2023. – 123, 02.01.2023. – 307,…
  - broj preuzetih podataka po danima za sve aerodrome pojedinačno i sortirano po aerodromu i danu (npr. LDZA – 01.01.2023. – 10, LDZA – 02.01.2023. – 24,…, LOWW – 01.01.2023. – 98, LOWW – 02.01.2023. – 217,…,
  - broj preuzetih podataka po danima za odabrani aerodrom sortirano po danu npr. LDZA (npr. LDZA – 01.01.2023. – 10, LDZA – 02.01.2023. – 24,…)
  - broj preuzetih podataka za sve aerodrome pojedinačno sortirano po aerodromu na odabrani dan npr. 01.01.2023 (npr. LDZA – 01.01.2023. – 10, LOWW – 01.01.2023. – 98,…,

4. web aplikacija **{korisnicko_ime}_aplikacija_4** učitava konfiguracijske podatke putem slušača aplikacije kod pokretanje aplikacije i upisuje ih u atribut konteksta pod nazivom „konfig“. Naziv datoteke konfiguracijskih podataka zapisan je u web.xml kao inicijalni parametar konteksta „konfiguracija“ (npr. datoteka NWTiS.db.config_4.xml). Nakon učitanih konfiguracijskih podataka provjerava status poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1. Ako nije aktivan, program prekida rad. Ako je sve u redu nastavlja s radom. Ova aplikacija pruža JAX-WS (SOAP) servise na krajnjim točkama „korisnici“, „aerodromi“, „letovi“ i „meteo“ te pruža WebSocket krajnju točku „info“. Ova aplikacija koristi bazu podataka. Podaci o korisnicima nalaze se u tablici „KORISNICI“. Svi zahtjevi trebaju se upisivati u tablicu „DNEVNIK“ putem aplikacijskog filtera (vrsta AP4). Svi zahtjevi JAX-WS servisa (osim dodavanja korisnika) moraju proći autentikaciju na bazi korisničkog imena i lozinke. Ako nije uspješna okida vlastitu iznimku pod nazivom PogresnaAutentikacija(tekst). Potrebno se držati zadanih osobina i realizirati sljedeće metode:
    a. JAX-WS krajnja točka „korisnici“:
        i. List<Korisnik> dajKorisnike(String korisnik, String lozinka, String traziImeKorisnika, String traziPrezimeKorisnika) – vraća kolekciju korisnika koji zadovoljavaju filtere ime i prezime korisnika po principu sličnosti (SQL upit LIKE %traziImeKorisnika% odnosno LIKE %traziPrezimeKorisnika%). Oba parametra mogu biti prazna i tada nema filtriranja. Može jedan biti prazan i tada se filtriranje provodi po drugom parametru koji nije prazan. Mogu oba parametra imati vrijednosti i tada oba služe za filtriranje.
        ii. Korisnik dajKorisnika(String korisnik, String lozinka, String traziKorisnika) – vraća korisnika koji zadovoljava korisničko ime.
        iii. boolean dodajKorisnika(Korisnik korisnik) – dodaje korisnika. Nakon dodavanja šalje obavijest putem WebSocket-a o ukupnom broju upisanih korisnika.

    b. JAX-WS krajnja točka „aerodromi“:
        i. List<Aerodrom> dajAerodromeZaLetove(String korisnik, String lozinka) – vraća kolekciju aerodroma za koje su preuzimaju podaci o polascima (tablica AERODROMI_LETOVI).
        ii. boolean dodajAerodromZaLetove (String korisnik, String lozinka, String icao) – dodaje aerodrom s icao kako bi se za njega preuzimali podaci o polascima (tablica AERODROMI_LETOVI). Nakon dodavanja šalje obavijest putem WebSocket-a o ukupnom broju upisanih aerodroma.
        iii. boolean pauzirajAerodromZaLetove (String korisnik, String lozinka, String icao) – postavlja status aerodroma s icao da je preuzimanje u pauzi i da se ne preuzimaju podaci o polascima (tablica AERODROMI_LETOVI).
        iv. boolean aktivirajAerodromZaLetove (String korisnik, String lozinka, String icao) – postavlja status aerodroma s icao da se za njega ponovno preuzimaju podaci o polascima (tablica AERODROMI_LETOVI).

c. JAX-WS krajnja točka „letovi“:

   i. List<LetAviona> dajPolaskeInterval(String korisnik, String lozinka, String icao, String danOd, String danDo, int odBroja, int broj) – vraća kolekciju letova koji su poletjeli s aerodroma koji ima traženi icao u zadanom intervalu. vrijemeOd/Do je u formatu dd.mm.gggg. Parametri odBroja i broj služe za straničenje. Za letove podaci trebaju odgovarati klasi LetAviona. Podaci se preuzimaju iz tablice LETOVI_POLASCI koje je aplikacija {korisnicko_ime}_aplikacija_3 preuzela s OpenSkyNetwork.

   ii. List<LetAviona> dajPolaskeNaDan(String korisnik, String lozinka, String icao, String dan, int odBroja, int broj) – vraća kolekciju letova koji su poletjeli s aerodroma koji ima traženi icao na određeni dan, koji je u formatu dd.mm.gggg. Parametri odBroja i broj služe za straničenje. Za letove podaci trebaju odgovarati klasi LetAviona. Podaci se preuzimaju iz tablice LETOVI_POLASCI koje je aplikacija {korisnicko_ime}_aplikacija_3 preuzela s OpenSkyNetwork.

   iii. List<LetAviona> dajPolaskeNaDanOS(String korisnik, String lozinka, String icao, String dan) – vraća kolekciju letova koji su poletjeli s aerodroma koji ima traženi icao na određeni dan, koji je u formatu dd.mm.gggg. Za letove podaci trebaju odgovarati klasi LetAviona. Podaci se preuzimaju iz OpenSkyNetwork. Za preuzimanje podataka o polascima letova s aerodroma koristi se Java biblioteka nwtis_rest_lib v 3.0.0.

d. JAX-WS krajnja točka „meteo“:

   i. MateoPodaci dajMeteo(String icao) - dohvaća meteorološke podatke za zadani aerodrom iz primljenog argumenta. Kako bi se dobila GPS lokacija aerodroma šalje se GET zahtjev RESTful web servisu iz {korisnicko_ime}_aplikacija_2 na resursu „aerodromi“. S lokacijom aerodroma poziva se metoda getRealTimeWeather(lokacija) na objektu OWMKlijent.

e. WebSocket krajnja točka „info“:

   i. dajMeteo(info) - obavještavanje svojih aktivnih korisnika o trenutnom vremenu na poslužitelju aplikacije, broju korisnika i broju aerodroma za koje se prikupljaju podaci o polascima.

Za testiranje JAX-WS servisa potrebno je pripremit SoapUI datoteku koja sadrži pozive svih implementiranih operacija. **Aplikacija se smatra nevažećim (0 bodova) ukoliko ne postoji funkcionalna datoteka za testiranje**.

5. web aplikacija **{korisnicko_ime}_aplikacija_5** učitava konfiguracijske podatke putem slušača aplikacije kod pokretanje aplikacije i upisuje ih u atribut konteksta pod nazivom „konfig". Naziv datoteke konfiguracijskih podataka zapisan je u web.xml kao inicijalni parametar konteksta „konfiguracija" (npr. datoteka NWTiS.db.config_5.xml). Nakon učitanih konfiguracijskih podataka provjerava status poslužitelja na mrežnoj utičnici iz {korisnicko_ime}_aplikacija_1. Ako nije aktivan, program prekida rad. Ako je sve u redu nastavlja s radom. Aplikacija se bavi administrativnim poslovima kroz korisničko sučelje realizirano s Jakarta MVC. Ova aplikacija nema vlastitu bazu podataka. Svi zahtjevi trebaju se upisivati u tablicu „DNEVNIK" putem aplikacijskog filtera (vrsta AP5) tako da se šalje REST zahtjev POST metodom na {korisnicko_ime}_aplikacija_2. Svaki obrazac u bilo kojem pogledu treba POST metodom slati podatke. Svoj rad temelji na slanju zahtjeva na JAX-WS web servis iz {korisnicko_ime}_aplikacija_4 i slanju zahtjeva na RESTful/JAX-RS web servis iz {korisnicko_ime}_aplikacija_2. Aplikacija sadrži potrebne Singleton Session Bean (SiSB), Stateful Sesssion Bean (SfSB), Stateless Sesssion Bean (SlSB), Message-Driven Bean. Aplikacija preuzima JMS poruke iz reda poruka „jms/NWTiS_{korisnicko_ime}" koje su poslane kod završetka pojedinog ciklusa dretve u {korisnicko_ime}_aplikacija_3. Primljene JMS poruke spremaju se u kolekciju u SiSB. Postoji mogućnost pražnjenja svih primljenih/spremljenih JMS poruka. Potrebno se držati zadanih osobina i realizirati sljedeće dijelove:

   - pogled 5.1:
     - početni izbornik koji sadrži poveznice na ostale izbornike s predloškom 5.? (5.2, 5.3, 5.4, 5.5, 5.6 i 5.7)
   - pogled 5.2:
     - početni izbornik koji sadrži poveznice na ostale aktivnosti vezane uz korisnike
     - pogled 5.2.1:
       - registraciju korisnika. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4.
     - pogled 5.2.2:
       - prijavljivanje korisnika. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4.
     - pogled 5.2.3:
       - pregled korisnika uz filtiranje po imenu i prezimenu. Korisnici se dobiju slanjem zahtjeva JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4. Putem WebSocket poruke i JavaScript koda ažurira se ukupan broj upisanih korisnika.
   - pogled 5.3:
     - upravljanje poslužiteljem iz {korisnicko_ime}_aplikacija_1. Za svaku komandu (STATUS, KRAJ, INIT, PAUZA, INFO DA, INFO NE) postoji gumb koji pokreće akciju. Šalje se zahtjev JAX-RS web servisu iz {korisnicko_ime}_aplikacija_2. Interpretiraju se i zatim ispisuju primljeni podaci.

- pogled 5.4:
    - pregled svih primljenih JMS poruka (straničenje) iz reda čekanja „jms/NWTiS_{korisnicko_ime}“ koji su spremljeni u kolekciji SiSB. Postoji gumb „Obriši“ kojim se pokreće brisanje svih primljenih/spremljenih JMS poruka i osvježava se pregled JMS poruka.
- pogled 5.5:
    - početni izbornik koji sadrži poveznice na ostale aktivnosti vezane uz aerodrome
    - pogled 5.5.1:
        - pregled svih aerodroma uz filtriranje po nazivu aerodroma i državi, uz straničenje. Šalje se zahtjev JAX-RS web servisu iz {korisnicko_ime}_aplikacija_2. Osim podataka uz pojedini aerodrom stavlja se poveznica putem koje se aerodrom dodaje za preuzimanje podataka o polascima. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4. Putem WebSocket poruke i JavaScript koda ažurira se ukupan broj upisanih aerodroma za preuzimanje.
    - pogled 5.5.2:
        - pregled podataka izabranog aerodroma na temelju poveznice iz pogleda 5.5.1. Šalje se zahtjev JAX-RS web servisu iz {korisnicko_ime}_aplikacija_2. Osim podataka o aerodromu prikazuju se trenutni meteorološki podaci za njegovu lokaciju. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4.
    - pogled 5.5.3:
        - pregled aerodroma za koje se za preuzimaju podaci o polascima. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4. Uz pojedini aerodrom prikazuje se status preuzimanja podataka (Da/Pauza) te se stavlja poveznica putem koje se mijenja status preuzimanja za aerodrom. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4. Putem WebSocket poruke i JavaScript koda ažurira se ukupan broj upisanih aerodroma za preuzimanje.
    - pogled 5.5.4:
        - pregled udaljenosti između dva aerodroma unutar država preko kojih se leti te ukupna udaljenost. Unose se dva icao podatka. Šalje se zahtjev JAX-RS web servisu iz {korisnicko_ime}_aplikacija_2. Sljedeći podatak je izračunata udaljenost između dva aerodroma dobivena zbrojem udaljenosti unutar država.
    - pogled 5.5.5:
        - pregled udaljenosti između dva aerodroma. Unose se dva icao podatka. Šalje se zahtjev JAX-RS web servisu iz {korisnicko_ime}_aplikacija_2 koji za izračun udaljenosti koristi komandu UDALJENOST u {korisnicko_ime}_aplikacija_1.
    - pogled 5.5.6:
        - pregled aerodroma i udaljenosti do polaznog aerodroma unutar države odredišnog aerodroma koji su manje udaljeni od udaljenosti između polaznog i odredišnog aerodroma. Unose se dva icao podatka. Šalje se zahtjev JAX-RS web servisu iz {korisnicko_ime}_aplikacija_2 za izračun udaljenosti koji koristi komandu UDALJENOST u {korisnicko_ime}_aplikacija_1.

- pogled 5.5.7:
    - pregled aerodroma i udaljenosti do polaznog aerodroma unutar zadane države koje su manje od zadane udaljenosti. Unose se icao, oznaka države i broj km. Šalje se zahtjev JAX-RS web servisu iz {korisnicko_ime}_aplikacija_2 za izračun udaljenosti koji koristi komandu UDALJENOST u {korisnicko_ime}_aplikacija_1.
- pogled 5.6:
    - početni izbornik koji sadrži poveznice na ostale aktivnosti vezane uz letove
    - pogled 5.6.1:
        - pregled spremljenih letova s određenog aerodroma u zadanom intervalu, uz straničenje. Unose se icao, datum od i datum do. Datumi su u formatu dd.mm.gggg. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4.
    - pogled 5.6.2:
        - pregled spremljenih letova s određenog aerodroma na zadani datum, uz straničenje. Unose se icao, datum. Datum je u formatu dd.mm.gggg. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4.
    - pogled 5.6.3:
        - pregled letova s određenog aerodroma na zadani datum. Unose se icao, datum. Datum je u formatu dd.mm.gggg. Šalje se zahtjev JAX-WS web servisu iz {korisnicko_ime}_aplikacija_4 koji preuzima podatke iz OpenSkyNetwork
- pogled 5.7:
    - pregled zapisa dnevnika (straničenje). Unosi/odabere se vrsta (AP2, AP4, AP5).

Struktura projekta treba biti sljedeća:

```
{LDAP_korisnik}-projekt
      pom.xml
      {LDAP_korisnik}-obrazac_1.pdf - popunjeni obrazac za projekt
      {LDAP_korisnik}-obrazac_2.pdf - popunjeni obrazac za samoprocjenu projekta
      {LDAP_korisnik}_lib_1
            pom.xml
            src
                  main
                        ...
      {LDAP_korisnik}_lib_2
            pom.xml
            src
                  main
                        ...
      {LDAP_korisnik}_aplikacija_1
            pom.xml
            src
                  main
                        ...
      {LDAP_korisnik}_aplikacija_2
            pom.xml
            src
                  main
                        ...
      {LDAP_korisnik}_aplikacija_3
            pom.xml
            src
                  main
                        ...
      {LDAP_korisnik}_aplikacija_4
            pom.xml
            src
                  main
                        ...
      {LDAP_korisnik}_aplikacija_5
            pom.xml
            src
                  main
                        ...
```

Korijenski projekt treba sadržati ostale projekte kao module tako da se može pokrenuti priprema svih projekata na najvišoj razini.

Instalacijska i programska arhitektura sustava

Aplikacija {korisnicko_ime} _aplikacija_	**Razvojni alat** (IDE) kod obrane projekta	**Java**	**Poslužitelj**	**EE osobine**	**Korisničko sučelje**	**Baza podataka**	**Rad s bazom podataka**	**Namjena**
1	Eclipse s Maven	17	Vlastiti					Poslužitelj na mrežnoj utičnici (socket server)
2	Eclipse s Maven	17	Docker/Payara Micro	Jakarta EE10 Web		Docker/JRE HSQLDB nwtis_bp	JDBC, SQL *JPA*[3] *Criteria API*	RESTful/JAX-RS web servis
3	Eclipse s Maven	17	Payara Full	Jakarta EE10 Web		Docker/JRE HSQLDB nwtis_bp	JDBC, SQL *JPA*[4] *Criteria API*	Preuzima podatke o polascima aviona s izabranih aerodroma
4	Eclipse s Maven	17	Payara Full	Jakarta EE10		Docker/JRE HSQLDB nwtis_bp	JDBC, SQL *JPA*[5] *Criteria API*	JAX-WS web servis WebSocket krajnja točka
5	Eclipse s Maven	17	Payara Full	Jakarta EE10	Jakarta MVC *JSF*[6]			Pogledi za rad s: • korisnicima • primljenim JMS porukama • aerodromima • letovima

[3] Umjesto JDBC s SQL može se koristiti JPA (s Criteria API ili JPQL). U obrascu za procjenu nalazi se informacija o načinu bodovanja u slučaju korištenja JPA.
[4] Umjesto JDBC s SQL može se koristiti JPA (s Criteria API ili JPQL). U obrascu za procjenu nalazi se informacija o načinu bodovanja u slučaju korištenja JPA.
[5] Umjesto JDBC s SQL može se koristiti JPA (s Criteria API ili JPQL). U obrascu za procjenu nalazi se informacija o načinu bodovanja u slučaju korištenja JPA.
[6] Umjesto Jakarta MVC s JSP može se koristiti Jakarta MVC s JSF ili samostalni JSF bez Jakarta MVC. U obrascu za procjenu nalazi se informacija o načinu bodovanja u slučaju korištenja JSF.