Predmet: Napredne Web tehnologije i servisi
Ak. god.: 2019./2020.

Vježba: 11. Upoznavanje s web servisima, razvoj Web aplikacije koja pruža SOAP i REST web servise, razvoj Web aplikacije uz korištenje JSF

Kreiranje 11. vježbe (direktorij **{LDAP_korisničko_ime}_zadaca_2**). U nastavku se direktorij za vježbu simbolički označava kao **{vježba}**.

Pokretanje programa NetBeans.

Instalirati certifikate za web mjesta u Glassfish (5.1.0). Njegov korijenski direktorij označen je kao {glassfish}:

1. preuzeti iz Moodle-a certifikate za web mjesta
2. otvoriti Command Prompt kao administrator i izvršiti:
    a. keytool -import -alias LocationIQ -keystore "{glassfish}\glassfish\domains\domain1\config\cacerts.jks" -file LocationIQ.cer
    b. keytool -import -alias OpenSky –keystore "{glassfish}\glassfish\domains\domain1\config\cacerts.jks" -file OpenSky.cer
    c. keytool -import -alias OpenWeatherMap –keystore "{glassfish}\glassfish\domains\domain1\config\cacerts.jks" -file OpenWeatherMap.cer

Lozinka za spremište je: changeit

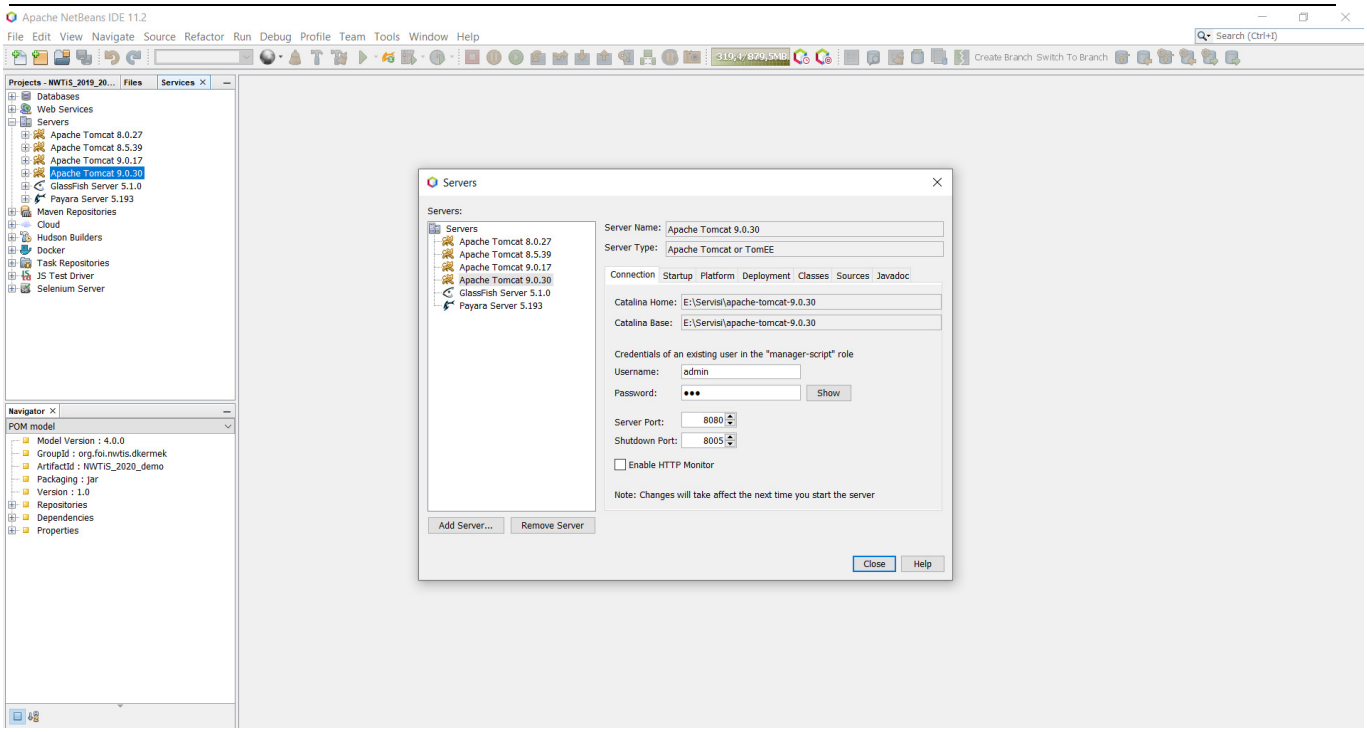
Certifikati se znaju mijenjati zbog zastare i sl. Ako certifikat nije isti kao kod Glassfisha tada web servis neće raditi pa treba provjeriti njihovu valjanost na web mjestima. I kod promjene preuzeti novu verziju i obaviti gornji postupak.

Pokretanje poslužitelja Glassfish (Services/Servers/Glassfish Server/Start). Predlaže se s Debug Mode.

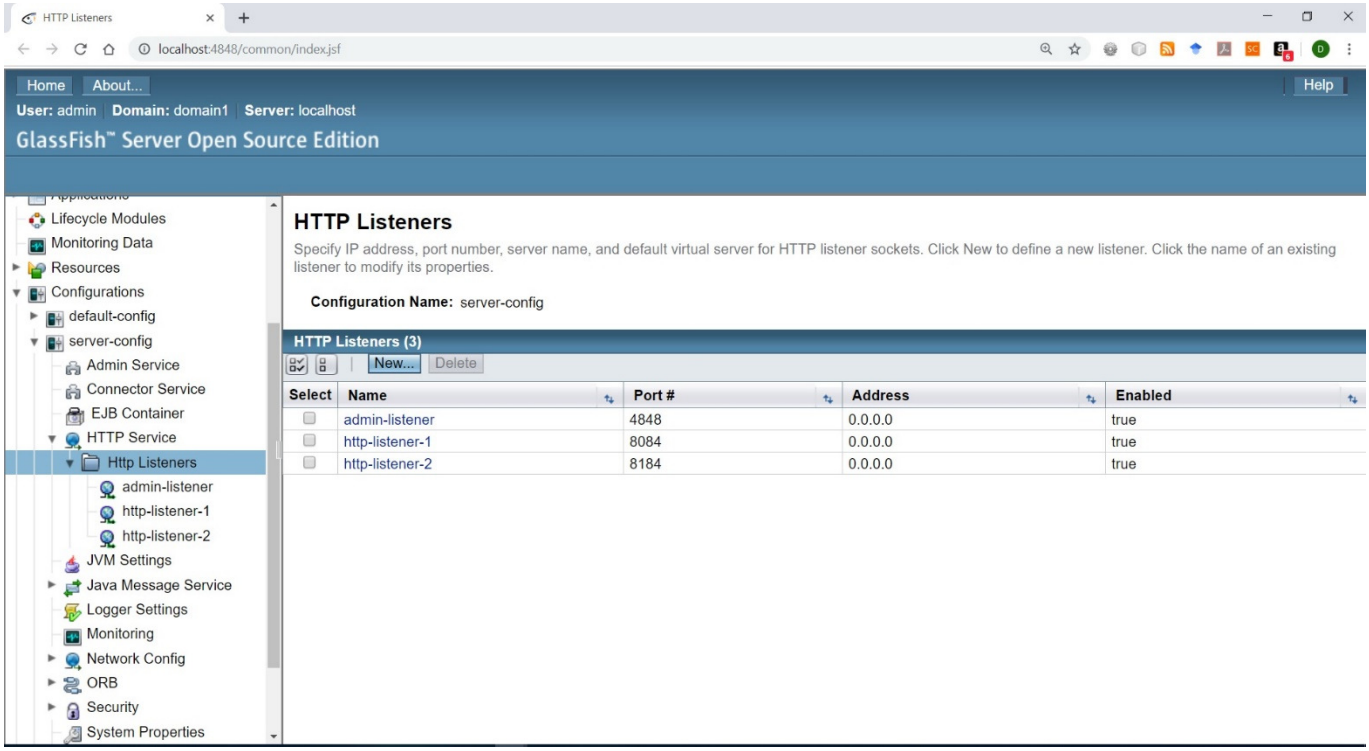
Pokretanje poslužitelja Tomcat (Services/Servers/Apache Tomcat/Start). Predlaže se s Debug Mode.

Podesiti postavke http portova:

1. Tomcat: 8080
2. Glassfish: 8084



Slika 1. Izgled postavki za Tomcat poslužitelj



Slika 2. Izgled postavki za Glassfish poslužitelj

Vježba_11 – Zadaća 2: Web aplikacije s SOAP i REST web servisima uz korištenje JSF za korisničko sučelje

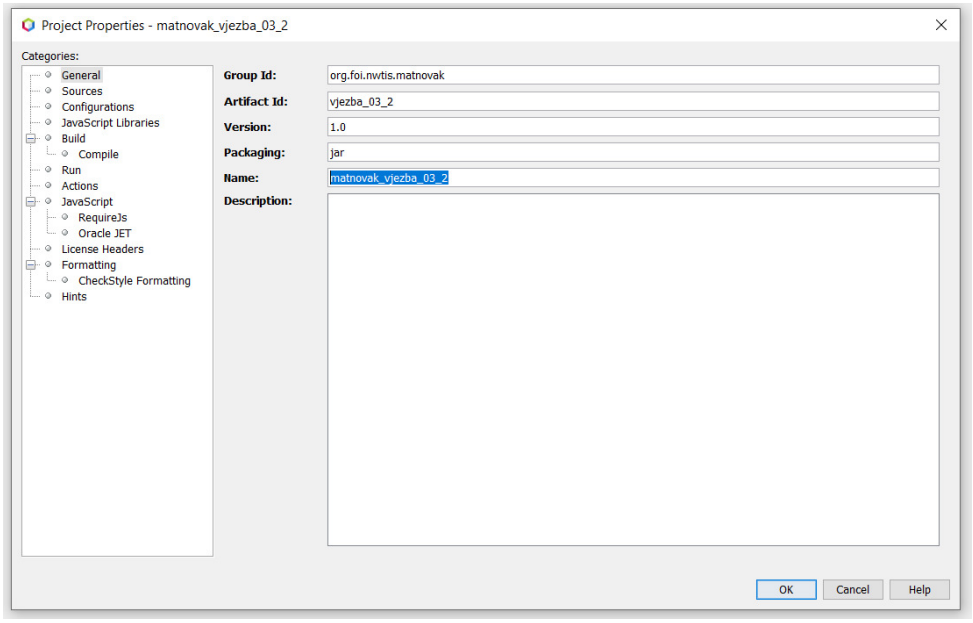
Naziv projekta: **{LDAP_korisničko_ime}_zadaca_2**

Korijenski direktorij treba biti **{LDAP_korisničko_ime}_zadaca_2**

Za rad s postavkama treba koristiti Java biblioteke iz vjezba_03_2 i vjezba_06_1. Rad se temelji na preuzimanju podataka s 3 web servisa: **LocationIQ, Open Weather Map, OpenSky Network**. Kao sučelje za te servise postoji biblioteka NWTiS_2020_REST_lib, a u njoj postoje klase koje se bave pojedinim web servisom. Vlastite klase i metode trebaju biti komentirane u Javadoc formatu. **Projekt isključivo treba predati u formatu NetBeans projekta.** Prije predavanja projekta potrebno je napraviti Clean na svim projektima. Zatim cijelu zadaća (uključujući tri projekta iz zadaće te projekte vjezba_03_2 i vjezba_06_1) saženi u **.zip** (NE .rar) format s nazivom **{LDAP_korisničko_ime}_zadaca_2.zip** i predati u Moodle. Uključiti izvorni kod, primjere datoteka konfiguracijskih podataka (.xml) i popunjeni obrazac za zadaću pod nazivom **{LDAP_korisničko_ime}_zadaca_2.[doc | pdf]** (u direktoriju **{LDAP_korisničko_ime}_zadaca_2**).

Sustav je sastoji od tri projekta ({LDAP_korisničko_ime}_zadaca_2_1, {LDAP_korisničko_ime}_zadaca_2_2 i {LDAP_korisničko_ime}_zadaca_2_3).

Kako su za izvršavanje projekata iz zadaće potrebni vaši projekti vjezba_03_2 i vjezba_06_1 potrebno je kod njih odraditi malu promjenu u nazivu (vidljivi dio u NetBeans). Otvori se Properties za projekt (npr. vjezba_03_2) te se u dijelu Name podesi da izgleda (slika) u obliku ({LDAP_korisničko_ime}_ vjezba_03_2. Kako je autor Matija onda je naziv matnovak_vjezba_03_2. Tako napraviti i za vjezba_06_1 pa će kod Matije biti matnovak_vjezba_06_1. Oba projekta kopirajte u direktorij {LDAP_korisničko_ime}_zadaca_2) tako da budu zajedno s 3 projekta zadaće.



Uvodne informacije o pojmovima:

- ICAO[1],[2] – predstavlja 4 znakovno/brojčanu oznaku aerodroma. U podacima OpenSky Network on se označava kao ident zbog čega je u tablici AIRPORTS atribut ident kao idenfitikator/primarki ključ a on se tumači kao ICAO. Klasa Airport se neposredno odnosi na tablicu AIRPORTS tako da ima iste nazive atributa kao što su nazivi stupaca u tablici. Informacije o pojedinom aerodromu na temelju ICAO mogu se naći na (npr. EDDF za Frankfurt na Maini)
  https://opensky-network.org/airport-profile?icao=EDDF
- ICAO24[3] predstavlja 24 bitu oznaka avion, šalje ju avion svojim transporderom u različite sustave koji praće avione.

Napomene:

1. UNIX timestamp[4] (broj sekundi od 00:00:00 UTC 01.01.1970.) razlikuje se od java.sql.Timestamp[5] (uzimamo referentnu metodu getTime() koja vraća broj milisekundi od 00:00:00 UTC 01.01.1970.) jer ima precizniji oblik (milisekunda umjesto sekunda). Određeni podaci u tablicama baze podataka temelje se na UNIX timestamp pa imaju kao tip podatka integer (npr. firstSeen u tablici airplanes). Kod drugih je važnije preciznije vrijeme pa se koristi java.sql.Timestamp (npr. stored u tablici airplanes). Kod operacija JAX-WS web servisa vremenski parametri UNIX timestamp. Tako da kod poziva operacija treba voditi brigu o ispravnom obliku podataka. U klasi OSKlijent postoje metode koje u parametru(ima) primaju UNIX timestamp (sekunde, tip long). Sada su dodane metode koje imaju u parametru(ima) tip java.sql.Timestamp. U njima se interno pozivaju metode s parametrima long.
2. U tablici AIRPORTS u stupcu COORDINATES podaci su upisani u obliku longitude, latitude. Tako da ih treba voditi brigu kod preuzimanja podataka iz tablice i pretvaranju u objekt klase Lokacija.
3. Za pretvaranje UNIX timestamp u standardni format i obratno preporučuje se https://www.epochconverter.com/
4. Biblioteka NWTiS_2020_REST_lib i dalje se razvija zbog čega je poželjno provjeriti ima li u NWTiS riznici/repository novih verzija u odnosu na onu koju koristite. API dokumentacija nalazi se na http://nwtis.foi.hr/NWTiS/video/NWTiS_2020_REST_lib/apidocs/
5. Dodatne informacije za Java EE, JAX-RS, Jersey, JSF
   a. https://javaee.github.io/tutorial/
   b. https://www.oracle.com/java/technologies/java-ee-glance.html
   c. https://eclipse-ee4j.github.io/jersey/
   d. https://eclipse-ee4j.github.io/jersey.github.io/documentation/latest/
   e. https://github.com/jax-rs
   f. https://javaserverfaces.github.io/whats-new-in-jsf23.html

[1] https://en.wikipedia.org/wiki/ICAO_airport_code
[2] http://airportsbase.org/ICAO.php
[3] http://www.virtualradarserver.co.uk/documentation/Glossary/Icao24.aspx
[4] https://tools.ietf.org/html/rfc3339
[5] https://docs.oracle.com/javase/8/docs/api/java/sql/Timestamp.html

Prvi projekt ({LDAP_korisničko_ime}_zadaca_2_1} izvršava se na poslužitelju, ima vlastitu bazu podataka i ima sljedeće dijelove:

1. u pozadinskoj dretvi preuzimaju se u ciklusima s pravilnim intervalima podaci o polascima 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 podataka (AIRPLANES). Dretva na početku 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 dretve. Prije preuzimanja podataka za pojedini aerodrom potrebno je provjeriti u tablici MYAIRPORTSLOG ima li zapis tog aerodroma za datum koji se obrađuje u ciklusu dretve. Ako postoji onda se preskače aerodrom. Podaci o letovima pojedinog aerodroma preuzimaju se za cijeli dan datuma koji se obrađuje u ciklusu dretve. Nakon preuzimanja podataka za pojedini aerodrom potrebno je upisati u tablicu MYAIRPORTSLOG da je odrađeno preuzimanje za taj aerodroma i datum. Na kraju svakog ciklusa dretve datum se povećava za jedan dan. Interval dretve određen je postavkama, kao i vremenski razmak (pauza) između preuzimanja podataka za dva aerodroma u jednom ciklusu. Ako dretva stigne do važećeg datuma tada pauzira jedan dan.
2. pruža JAX-WS (SOAP) web servis za podatke o korisnicima, aerodromina i avionima za spremljene aerodrome. Operacije se temelje na podacima koji se nalaze u tablicama KORISNICI, AIRPORTS, MYAIRPORTS, MYAIRPORTSLOG i AIRPLANES u bazi podataka te na pozivu metoda klasa iz NWTiS_2020_REST_lib. Ako nije navedeno tada se podaci preuzimaju iz baze podazaka. Potrebno se držati zadanih osobina i realizirati sljedeće operacije:
    1. sve operacije (osim kod dodavanja novog korisnika) moraju kod slanja zahtjeva pripremiti potrebne podatke za autentikaciju (šalje se korisničko ime, lozinka kao prva dva parametra), vraća Boolean. Tako sve operacija imaju barem ta dva parametra, Kod onih operacija koje imaju dodatne parametre to piše.
    2. provjerava korisničke podatke za autentikaciju, vraća Boolean
    3. dodaje novog korisnika (šalje Korisnik), vraća Boolean. Podaci za datumKreiranja i datumPromjene nemaju ulogu te se ignoriraju. Ostali podaci u objektu Korisnik ne smiju biti null ili prazni.
    4. ažurira korisničke podatke (šalje Korisnik), vraća Boolean. Ako je lozinka u objektu Korisnik null ili je prazna, tada se ona ne mijenja. Korisničko ime u objektu mora odgovarati parametru za korisničko ime. Podaci za datumKreiranja i datumPromjene nemaju ulogu te se ignoriraju. Ostali podaci u objektu Korisnik ne smiju biti null ili prazni. – neobavezni dio, za više bodova ako postoji registracija i ažuriranje korisnika u trećem projektu
    5. vraća popis svih korisnika, vraća java.util.List<Korisnik>. Kod lozinke se stavlja"******" zbog privatnosti i sigurnosti. Ako nema ni jednog korisnika vraća null. – neobavezni dio, za više bodova

6. vraća popis svih korisnika koji imaju vlastite aerodrome, vraća java.util.List<Korisnik>. Ako nema ni jednog korisnika vraća null. – neobavezni dio, za više bodova
7. vraća popis aerodroma koji imaju sličan naziv koji se traži (šalje naziv), vraća java.util.List<Aerodrom>. Ako nema ni jednog aerodroma vraća null.
8. vraća popis aerodroma koji su iz određene države (šalje kod države), vraća java.util.List<Aerodrom>. Ako nema ni jednog aerodroma vraća null.
9. vraća popis svih vlastitih aerodroma, vraća java.util.List<Aerodrom>. Ako nema ni jednog aerodroma vraća null.
10. provjerava da li je aerodrom u njegovoj kolekciji aerodroma koje prati, vraća Boolean (šalje ICAO kod)
11. vraća traženi aerodrom iz vlastitih aerodroma, vraća Aerodrom (šalje se ICAO kod). Ako nema aerodroma vraća null.
12. dodaje aerodrom u vlastite aerodrome (šalje se ICAO kod), vraća Boolean. Mora postojati aerodrom u tablici AIRPORTS s tim kodom (ICAO = ident).
13. vraća popis svih 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 aviona vraća null.
14. vraća najveću visinu leta odabranog aviona i njenu geo lokaciju (šalje se ICAO24 kod, vrijeme) kao UNIX timestamp, vraća LetPozicija. Ako nema ni jedne pozicije leta aviona vraća null. Podaci svih pozicija leta preuzimaju se od web servisa Open Sky Network putem klase OSKlijent i njegove metode getTracks(...). Potrebno je pronaći koja od njih ima najveću vrijednost te se ona vraća. – neobavezni dio, za više bodova ako se koristi u trećem projektu

Drugi projekt ({LDAP_korisničko_ime}_zadaca_2_2} izvršava se na poslužitelju, ne koristi bazu podataka, komunicira s JAX-WS web servisom iz prvog projekta. Potrebno se držati zadanih osobina i realizirati sljedeće operacije:

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"}.`
4. GET metoda - osnovna adresa - vraća popis vlastitih aerodroma, a za aerodrom podaci trebaju odgovarati atributima klase Aerodrom. Poziva operaciju JAX-WS web servisa iz prvog projekta.
5. POST metoda - osnovna adresa - dodaje aerodrom korisniku koji je pozvao operaciju (šalje se ICAO kod). Šalje se {"icao": "icao kod"}. Poziva operaciju JAX-WS web servisa iz prvog projekta.
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 iz prvog projekta.
7. GET metoda - putanja `"{icao}"` - vraća podatke izabranog aerodroma ako je pridružen korisniku koji je pozvao operaciju. Podaci aerodroma trebaju odgovarati atributima klase Aerodrom. Poziva operaciju JAX-WS web servisa iz prvog projekta.
8. PUT, POST i DELETE metode - osnovna adresa, na bazi putanja - nije dozvoljeno pozivati ako nisu prethodno spomenute. Vraća pogrešku.

Treći projekt ({LDAP_korisničko_ime}_zadaca_2_3} izvršava se na poslužitelju. Potrebno se držati zadanih osobina i realizirati sljedeće dijelove:

1. korisnički dio treba biti podržan za više jezika (hr, en, de ili koji treći) uz odabir važećeg jezika u posebnom pogledu (p1) – neobavezni dio, za više bodova
2. korisnički dio treba biti podržan zajedničkim predloškom (izbornik, sadržaj) – neobavezni dio, za više bodova
3. korisnički dio (osim prijavljivanja i registracije korisnika) treba biti otvoren samo za prijavljene korisnika. Koristi se filter za ograničavanje pristupa. Filter – neobavezni dio, za više bodova
4. u korisničkom dijelu potrebno je imati pogled (p2) za registraciju korisnika. Poziva se operacija JAX-WS web servisa iz prvog projekta. – neobavezni dio, za više bodova
5. u korisničkom dijelu potrebno je imati pogled (p3) za prijavljivanje korisnika. Poziva se operacija JAX-WS web servisa iz prvog projekta.
6. u korisničkom dijelu potrebno je imati pogled (p4) za dodavanje aerodroma u vlastitu kolekciju. Moguća su dva načina selekcije aerodroma:
    1. unese se naziv aerodroma, (min 3 znaka) zatim se putem gumba "Preuzmi aerodrome prema nazivu" pokreće akcija koja preuzima podatke o aerodromima koji imaju sličan naziv. Poziva se operacija JAX-RS web servisa iz drugog projekta.
    2. unese se kod države (min 2 znaka) zatim se putem gumba " Preuzmi aerodrome iz države" pokreće akcija koja preuzima podatke o aerodromima iz upisane države.. Poziva se operacija JAX-RS web servisa iz drugog projekta.

    Zatim se pozove operacija JAX-RS web servisa iz drugog projekta za dohvat vlastitih aerodroma. Iz prve kolekcije aerodroma treba obrisati aerodrome koji se nalaze u kolekciji vlastitih aerodroma tako da ostanu aerodromi koji se mogu dodati. Dobiveni podaci prikazuju se o obliku padajućeg izbornika sa 7 vidljivih elemenata. Slijedi pokretanje akcije putem gumba "Dodaj aerodrom" kojom se odabrani aerodrom dodaje u vlastitu kolekciju aerodroma. Poziva se operacija JAX-RS web servisa iz drugog projekta. Može se umjesto padajućeg izbornika koristiti tablica aerodroma u kojoj svaki redak ima gumb za dodavanje.

7. u korisničkom dijelu potrebno je imati za pogled (p5) koji ima tri dijela. Gornji dio čini blok element koji zauzima 15% visine ekrana i u kojem se nalazi unos intervala vremena (od i do) u standardnom hrvatskom 24 satnom obliku dd.mm.gggg hh:mm. Potrebno ih je kasnije pretvoriti u UNIX timestamp. Srednji dio je blok element koji zauzima 60% visine ekrana i prikazuje aerodrome iz vlastite kolekcije u obliku tablice. Ukoliko je tablica veća od predviđenog prostora tada se kd blok elementa pojavljuje okomiti klizač kako bi se mogla vidjeti cijela tablica. Tablica prikazuje ICAO, naziv, državu te dva gumba. Prvi gumb služi za pokretanje akcije za prikaz aviona koji su poletjeli s izabranog aerodroma u traženom intervalu. Podaci se prikazuju u pogledu koji se otvara u novoj kartici u pregledniku. Drugi gumb služi za pokretanje akcije za preuzimanje geolokacijskih i meteo podataka izabranog aerodroma. U donjem dijelu je blok element koji zauzima preostalu visinu ekrana. U njemu

se nalazi prostor za usporedni prikaz geolokacijskih (iz baze podataka i od web servisa, klasa LIQKlijent, na bazi naziva aerodroma) i meteoroloških podataka izabranog aerodroma (web servis, klasa OWMKlijent).

8. u korisničkom dijelu potrebno je imati za pogled (p6) koji prikazuje tablicu s avionima koji su poletjeli s odabranog aerodroma unutar upisanog intervala. Prikazuju se podaci icao24, callsign, prvo i zadnje vrijeme (standardni hrvatski 24 satni oblik), polazni i odredišni aerodrom, gumb za pokretanje akcije za prikaz najveće visina leta i njene geo lokacije. Poziva se operacija JAX-WS web servisa iz prvog projekta. Za vrijeme uzima se srednje vrijeme između prvog i zadnjeg vremena. Podaci se prikazuju ispod tablice. Dio s gumbom za prikaz najveće visine leta – neobavezni dio, za više bodova.

Preporučeni koraci za prvi projekt:

1. Kreiranje projekta **{LDAP_korisničko_ime}_zadaca_2_1** u Group Id: **org.foi.nwtis.{LDAP_korisnik}** kao Java Web aplikaciju, server Glassfish, Java EE verzija: Java EE 8, kontekst {LDAP_korisničko_ime}_zadaca_2_1
2. U Properties/Sources provjeriti Source/Binary Format za 1.8
3. Kreirati pakete org.foi.nwtis.{LDAP_korisnik}.web.slusaci, org.foi.nwtis.{LDAP_korisnik}.web.dretve, org.foi.nwtis.{LDAP_korisnik}.web.podaci, org.foi.nwtis.{LDAP_korisnik}.ws.serveri
4. Kopirati datoteke konfiguracije iz projekta vjezba_09_1 ili vjezba_10_1 s direktorija src/main/webapps/WEB-INF u direktorij src/main/webapps/WEB-INF. Potrebno je dopuniti s novim postavkama prema Tablica 1. Može se preuzeti pripremljena proširenja datoteka s Moodlea.
5. Otvoriti projekt vjezba_06_1 te u BP_Konfiguracija dodati get metodu za konfig. Napraviti Clean/Build.
6. Dodati ovisnost/dependecy na vjezba_06_1-1.0.jar. Provjeriti verziju Jave kod vjezba_06_1 i podesiti da bude usklađena s verzijom Jave koja se koristi na poslužiteljima Tomcat i Glassfish.
7. Ako ne postoji kreirati opisnik isporuke **web.xml** (New/Other/Web/Standard Deployment Descriptor)
8. Upisati u **web.xml** inicijalni parametar konteksta **konfiguracija** (NWTiS.db.config_2.xml)
9. Obisati klase JavaEE8Resource i JAXRSConfiguration
10. U Files obrisati direktorij src/main/resources
11. Kopirati slušaca konteksta **SlusacAplikacije** iz projekta vjezba_09_1 ili vjezba_10_1. U ovom slučaju želi se koristiti bez opisnika isporuke pa se dodaje anotacija @WebListener (ako ne postoji)
12. Izgraditi, isporučiti, izvršiti i testirati aplikaciju
13. U **pom.xml** dodati NWTiS riznicu (repository) prije <dependencies>

```
<repositories>
    <repository>
        <id>NWTiS</id>
        <name>NWTiS</name>
        <url>http://nwtis.foi.hr:8088/repository/NWTiS/</url>
    </repository>
</repositories>
```

14. Dodati ovisnost/dependency na NWTiS_2020_REST_lib
15. Preuzeti datoteke myairports.sql, myairports_podaci.sql, myairportslog.sql i izvršiti u NetBeans na vezi za JavaDB i bazi podataka za grupu kojoj pripadate. Pretpostavlja se da postoji tablica AIRPORTS iz vježbe 6. . Ako je nema onda treba preuzeti datoteke airports_baza.sql, airports_podaci.sql i izvršiti u NetBeans na vezi za JavaDB i bazi podataka za grupu kojoj pripadate
16. Kreirati klasu **PreuzimanjeLetovaAvionaAerodroma** koja nasljeđuje klasu Thread
17. Dodati standardne metode (Insert Code/Override Methods..) i označiti interrupt(), run(), start()
18. Dodati atribut BP_Konfiguracija konf
19. Dodati konstruktor koji prima parametar konf (Insert Code/Constructor) i označiti atribut konf
20. Dodati atribute za objekt klase OSKlijent, korisničko ime i lozinku za OpenSky Network
21. Dodati atribute za početni datum preuzimanja, trajanje ciklusa dretve i trajanje pauze između dva aerodroma
22. U metodi start() preuzeti potrebne parametre iz konfiguracije (prethodni atributi)
23. Kreira se objekt klase OSKlijent s korisničkim podacima.
24. Dodati privatni atribut za krajPreuzimanja - Boolean
25. U metodi run() otvoriti petlju koja se izvršava dok nije kraj preuzimanja
26. Podese se potrebni podaci za početni datum preuzimanja koji predstavlja od vremena, te se izračunava do vremena dodajući trajanje jednog dana od vremenu. Može se koristiti klasa SimpleDateFormat[6]

[6] https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html

27. U petlji se dohvate svi aerodromi za koje treba preuzeti podatke o letovima aviona. U petlji se provjerava je li već obrađen aerodrom za datum i ako nije preuzimaju se podaci pomoću metode getDepartures iz klase OSKlijent i zapisuju se u potrebnu tablicu. Zapisuje se u potrebnu tablicu da je obrađen aerodrom za datum. Odrađuje se pauza između dva aerodroma. Na kraju petlje izračunavaju se novi od i do vremena tj. pomiču se za 1 dan. Dretva pauzira potrebno vrijeme da se osigura pravilan vremenski ciklus. Za rad s bazom podataka može se iskoristiti klasa AirportDAO iz projekta vjezba_09_1 ili vjezba_10_1.
28. U **SlusacAplikacije** kreirati objekt dretve PreuzimanjeLetovaAvionaAerodroma i pokrenuti ju. Kod zaustavljanja aplikacije potrebno je zaustaviti dretvu da ne ostane raditi kao tzv. „deamon".
29. Izgraditi, isporučiti, izvršiti i testirati aplikaciju
30. Pretpostavlja se da postoji tablica KORISNICI iz vježbe 8. Ako je nema onda treba preuzeti datoteku korisnici.sql i izvršiti u NetBeans na vezi za JavaDB i bazi podataka za grupu kojoj pripadate.
31. Kreirati klasu za JAX-WS web servis (New/Web Services/Web Service) pod nazivom **Zadaca2**.
32. Promijeniti metodu hello u dodajKorisnika, vraća Boolean, promijeniti parametar u noviKorisnik tipa Korisnik. Za rad s bazom podataka može se iskoristiti klasa KorisnikDAO iz projekta vjezba_08_1. Potrebno je podesiti za novu klasu Korisnik.
33. Treba dohvatiti podatke iz konfiguracija. Razmisliti na koje načine se može obaviti?
34. Najednostavniji način je injektirati kontekst kao varijablu u klasu i preko atributa konkteksta "BP_Konfig" dohvatiti konfiguraciju

```
@Inject
ServletContext context;
```

35. Kreira se objekt KorisnikDAO i pozove metoda dodajKorisnika(...)
36. Izgraditi, isporučiti, izvršiti i testirati aplikaciju. Dio operacije web servisa može se testirati pomoću ugrađenog modula za testiranje JAX-WS web servisa u NetBeans: Web Service/Zadaca2/Test Web Service. Problem je kod metoda koje primaju objekt(e) klasa. Tu može pomoći Postman. Pripremljena je datoteka koja se može preuzeti iz Moodlea. Ona sadrži kolekciju zahtjeva koji su pripremljeni na temelju nastavničkih podataka. Potrebno je otvoriti datoteku u uređivaču po izboru i promijeniti podatke u vlastite. Nakon toga se učita u Postman i mogu se izvršavati pojedini zahtjevi. Osim Postman može se koristiti i SoapUI[7].
37. Kreirati novu operaciju JAX-WS webservisa (Insert Code.../Add Web Service Operation) pod nazivom provjeraKorisnika, vraća Boolean, ima parametre korisnik i lozinka tipa String. Poziva se metoda dohvatiKorisnika(...) iz klase KorisnikDAO.
38. Na sličan način realiziraju se ostale metode (azurirajKorisnika, sviKorisnici, korisniciAerodroma, dohvatiAerodromeNaziv, dohvatiAerodromeDrzava, mojiAerodromi, imamAerodrom, mojAerodrom, dodajMojAerodrom, poletjeliAvioniAerodrom) koje rade samo s korisnicima i aerodromima. Može se koristiti i Copy/Paste.
39. Kreirati novu operaciju JAX-WS webservisa (Add Web Service Operation) pod nazivom najvecaVisinaLetaAviona, vraća LetPozicija, ima parametre icao24 tipa String i UNIX timestamp zaVrijeme. Kopirati parametre iz metode provjeraKorisnika(...)
40. Poziva metodu getTracks klase OSKlijent. U vraćenom objektu postoji atribut path koji sadrži kolekciju putanje iz leta aviona. Potrebno je pronaći koji element ima najveću visinu te ga vratiti.
41. Izgraditi, isporučiti, izvršiti i testirati aplikaciju

[7] https://www.soapui.org/downloads/soapui/

Preporučeni koraci za drugi projekt:

1. Kreiranje projekta **{LDAP_korisničko_ime}_zadaca_2_2** u Group Id: **org.foi.nwtis.{LDAP_korisnik}** kao
2. Java Web aplikaciju, server Tomcat, Java EE verzija: Java EE 7
3. U Properties/Sources promijeniti Source/Binary Format na 1.8
4. U Run promijeniti Java EE Version: Java EE 8 Web
5. Otvoriti **pom.xml** i promijeniti kod javaee-web-api iz 7.0 u 8.0
6. Obrisati blokove gdje se spominje endorsed
7. Kreirati pakete org.foi.nwtis.{LDAP_korisnik}.ws.klijenti, org.foi.nwtis.{LDAP_korisnik}.rest.serveri, org.foi.nwtis.{LDAP_korisnik}.web.podaci
8. U pom.xml dodati NWTiS riznicu (repository) kao što je opisano u prvom projektu
9. Dodati ovisnosti/dependency za:

artifactId	groupId	version	D	K
NWTiS_2020_REST_lib	org.foi.nwtis	1.1	1	
javax.ws.rs-api	javax.ws.rs	2.1		
jersey-container-servlet	org.glassfish.jersey.containers	2.30		
jersey-server	org.glassfish.jersey.core	2.30		
jersey-hk2	org.glassfish.jersey.inject	2.30		
jersey-media-jaxb	org.glassfish.jersey.media	2.30		
jersey-media-json-jackson	org.glassfish.jersey.media	2.30		
jaxb-api	javax.xml.bind	2.3.1		
jaxb-impl	com.sun.xml.bind	2.3.1		
lombok	org.projectlombok	1.18		1
gson	com.google.code.gson	2.8.5		1
jaxws-api	javax.xml.ws	2.3.1		
jaxws-rt	com.sun.xml.ws	2.3.1		
javaee-web-api	javax	8.0		1

10. Kreirati JAX-RS web servis (New/Other/Web Services/RESTfull Web Services from Patterns), dalje, označiti Simple Root Resource, dalje, pod Path: **aerodromi**, Class Name: **Zadaca2RestAerodromi**, MIME Type: **application/json**, Representation Class: **Response**
11. Izgraditi, isporučiti, izvršiti.
12. Testirati se može pomoću ugrađenog modula za testiranje JAX-RS web servisa u NetBeans: RESTful Web Services/Zadaca2Rest/Test Resource Uri. I za ovaj dio zadaće pripremljena je datoteka za Postman za testiranje operacija. Ponoviti postupak pripreme kao što je opisan u prvom projektu.
13. Obrisati postojeću GET metodu i dodati metodu dajAerodomeKorisnika(...) koja prima podatke o korisniku u zaglavlju zahtjeva i na početaku vraća kolekciju ukodiranih aerodroma te izgleda (preuzeti s Moodlea):

```
@GET
@Consumes({MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_JSON})
      public Response dajAerodomeKorisnika(
              @HeaderParam("korisnik") String korisnik,
              @HeaderParam("lozinka") String lozinka) {
    java.util.List<Aerodrom> aerodromi = new ArrayList<>();
    aerodromi.add(new Aerodrom(
```

```
                "LDAZ", "Zagreb", "HR", new Lokacija("0.0", "0.0")));
        aerodromi.add(new Aerodrom(
                "LOWW", "Vienna", "AT", new Lokacija("0.0", "0.0")));
        aerodromi.add(new Aerodrom(
                "EDDF", "Frankfurt/M", "AT", new Lokacija("0.0", "0.0")));
        Odgovor odgovor = new Odgovor();
        odgovor.setStatus("10");
        odgovor.setPoruka("OK");
        odgovor.setOdgovor(aerodromi.toArray());
        return Response
                .ok(odgovor)
                .status(200)
                .build();
    }
```

14. Umjesto klase Odgovor s generičkim tipom za niz, može se koristiti klasa OdgovorAerodrom koja sadrži niz objekata klase Aerodrom. Možete probati
15. Izgraditi, isporučiti, izvršiti.
16. Testirati pomoću Test Resource Uri i/ili Postman
17. Kreirati JAX-WS klijenta (New/Other/Web Services/Web Service Client) kod označenog Project odaberemo Browse... te odaberemo {LDAP}_zadaca_2_1 i Zadaca2
18. Kreirati Java klasu **Zadaca2_1WS**.
19. Kreirati metodu List<Aerodrom> dajAerodomeKorisnika(...)
20. U metodi dodati poziv JAX-WS operacije (Insert Code/Call Web Service Operation) i odabrati dajAerodomeKorisnika
21. Generirani kod treba podesiti da odgovara ulaznim i izlaznim potrebama.
22. U metodi web servisa obrišu se ukodirani aerodromi, kreira se objekt klase Zadaca2_1WS te se poziva metoda dajAerodomeKorisnika(...) čiji rezultat se pridružuje varijabli aerodromi
23. Izgraditi, isporučiti, izvršiti.
24. Testirati pomoću Test Resource Uri ili Postman ili SoapUI.
25. Ako ste promijenili operacije u JAX-WS u prvom projektu potrebno je osvježiti JAX-WS klijenta a to će se obaviti osvježavanjem WSDL iz kojeg se generiraju pomoćne klase. (Web Service References/Zadaca2/Refresh.../označiti Also replace local wsdl.... i pokrenuti s Yes)
26. Dodati ostale operaciju na tom principu.

Preporučeni koraci za treći projekt:

1. Kreiranje projekta **{LDAP_korisničko_ime}_zadaca_2_3** u Group Id: **org.foi.nwtis.{LDAP_korisnik}** kao Java Web aplikaciju, server Glassfish, Java EE verzija: Java EE 8, kontekst {LDAP_korisničko_ime}_zadaca_2_3
2. U Properties/Sources provjeriti Source/Binary Format za 1.8
3. Kreirati pakete org.foi.nwtis.{LDAP_korisnik}., org.foi.nwtis.{LDAP_korisnik}.web.zrna, org.foi.nwtis.{LDAP_korisnik}.rest.klijenti, org.foi.nwtis.{LDAP_korisnik}.ws.klijenti
4. Obisati klase JavaEE8Resource i JAXRSConfiguration
5. U Files obrisati direktorij src/main/resources
6. Otvoriti Properties/Frameworks/Add. i odabrati JaveServer Faces, Libraries/Server Library: JSF 2.2, Configuration/JSF Servlet URL pattern: *.xhtml, Prefered Page Language: Facelets, Components: PrimeFaces
7. Ostali dio postavljanja JSF na 2.3 prema opisu u vježbi 9
8. U pom.xml za primefaces može se promijeniti verzija (testirano s 5.3 i 8.0)
9. U pom.xml dodati NWTiS riznicu (repository) kao što je opisano u prvom projektu
10. Dodati ovisnost/dependency na NWTiS_2020_REST_lib
11. Dodati ovisnost/dependency na lombok
12. Dodati ovisnost/dependecy na vjezba_06_1-1.0.jar. Provjeriti verziju Jave kod vjezba_06_1 i podesiti da bude usklađena s verzijom Jave koja se koristi na poslužiteljima Tomcat i Glassfish.
13. Kopirati datoteke konfiguracije iz projekta vjezba_09_1 ili vjezba_10_1 s direktorija src/main/webapps/WEB-INF u direktorij src/main/webapps/WEB-INF
14. Upisati u **web.xml** inicijalni parametar konteksta **konfiguracija** (NWTiS.db.config_2.xml)
15. Kopirati slušaca konteksta **SlusacAplikacije** prvog projekta bez aktiviranja dretve za preuzimanje podataka.
16. Kreirati JAX-WS klijenta (New/Other/Web Services/Web Service Client) kod označenog Project odaberemo Browse... te odaberemo {LDAP}_zadaca_2_1 i Zadaca2
17. Kopirati Java klasu **Zadaca2_1WS** iz drugog projekta.
18. Dodati potrebne operacija na isti način kao u drugom projektu. Na kraju možete nepotrebne metode obrisati. Ili kreirati novi Java projekt kao biblioteku te sve zajedničke klase premjestiti u novi projekt/biblioteku a nakon ga toga dodati u drugi i treći projekt dodati kao ovisnost. A stare klase obrisati.
19. Kreirati klasu za klijenta JAX-RS iz drugog projekta (New/Others/Web Services/RESTful Java Client) pod nazivom **Zadaca2_2RS**, Select the REST resource, From Project, odabereno drugi projekt (resurs aerodromi), bez autentikacije. Možemo obrisati nepotrebne metode. Potrebno je dodati varijable za korisnika i lozinku koji se u većini metoda šalju u zaglavlju. Dodati konstruktor s ta dva parametra. U svim metodama prije poziva HTTP metode .post(...) ili .get(...) dodati postavljanje zaglavlja

```
.header("korisnik", korisnik)
.header("lozinka", lozinka)
```

20. Kreirati JSF CDI Bean **PrijavaKorisnika** (New/JSF(JSF CDI Bean) (bez uključivanja u web.xml), scope: session. Ovo zrno nosi informaciju o prijavljenom korisniku.
21. Kreirati JSF **prijavaKorisnika.xhtml** i postaviti potrebne elemente.
22. Kreirati JSF CDI Bean **RegistracijaKorisnika** (New/JSF(JSF CDI Bean) (bez uključivanja u web.xml), scope: session (više odgovara request no ovo je elegantnije da se ne moraju podaci ponovo unositi kod više unosa)
23. Kreirati JSF **registracijaKorisnika.xhtml** i postaviti potrebne elemente
24. U **index.xhtml** postaviti potrebne poveznice na ostale poglede
25. Izgraditi, isporučiti, izvršiti i testirati.
26. Kreirati JSF CDI Bean **DodavanjeAerodroma** (New/JSF(JSF CDI Bean) (bez uključivanja u web.xml), scope: request

27. Za pristup podacima prijavljenog korisnika (korisničko ime i lozinka potrebni su kod većina operacija web servisa) može se koristiti injektiranje zrna PrijavaKorisnika putem kojeg se može doći do potrebnih podataka.

```
@Inject
PrijavaKorisnika prijavaKorisnika;
```

28. Kreirati varijable za korisnika, lozinku (String)
29. Kreirati varijable za naziv, drzavu (String), anotirati s @Getter @Setter
30. Kreirati varijablu aerodrome (<List<Aerodrom>), anotirati s @Getter
31. Dodati metodu preuzmiPodatkeKorisnika() u kojoj se preuzimaju podaci za korisnika i lozinku od injektiranog zrna PrijavaKorisnika
32. Dodati metode public String dajAerodromaNaziv() i dajAerodromaDrzava(). Obje pozivaju prethodnu metodu, kreiraju objekt klase Zadaca2_2RS uz prijenos korisnika i lozinke, pozivaju metodu za preuzimanje aerodroma s time da se pridružuje stvarne vrijednost nazivu ili državi prema odabiru akcije. Kod poziva metode potrebno je pridružiti klasu u koju će se učitati podaci koju su dohvaće od JAX-RS weg servisa. Klasa Odgovor sadrži generički tip za odgovor pa nije najbolja za rad s JSON. Zbog toga postoji klasa OdgovorAerodrom u kojoj je odgovor tip Aerodrom. Npr.

```
OdgovorAerodrom odgovor =
      zadaca2_2RS.dajAerodrome(OdgovorAerodrom.class,...);
```

33. Treba preuzeti aerodrome koje prati korisnik te ih maknuti iz dohvaćenih aerodroma na bazi naziva ili države tako da se prikažu samo oni aerodromi koje korisnik ne prati u tom trenutku
34. Dodati metodu public String dodajAerodromKorisniku(String icao) i njoj se poziva metoda za dodavanje odabranog aerodroma aktivnom korisniku.
35. Kreirati JSF **dodavanjeAerodroma.xhtml** i postaviti poveznice na ostale poglede. Tako isto napraviti u svim novim pogledima. Izgled može biti kao na Slika 3.
36. U <html dodati vezu na PrimeFaces

```
xmlns:p="http://primefaces.org/ui"
```

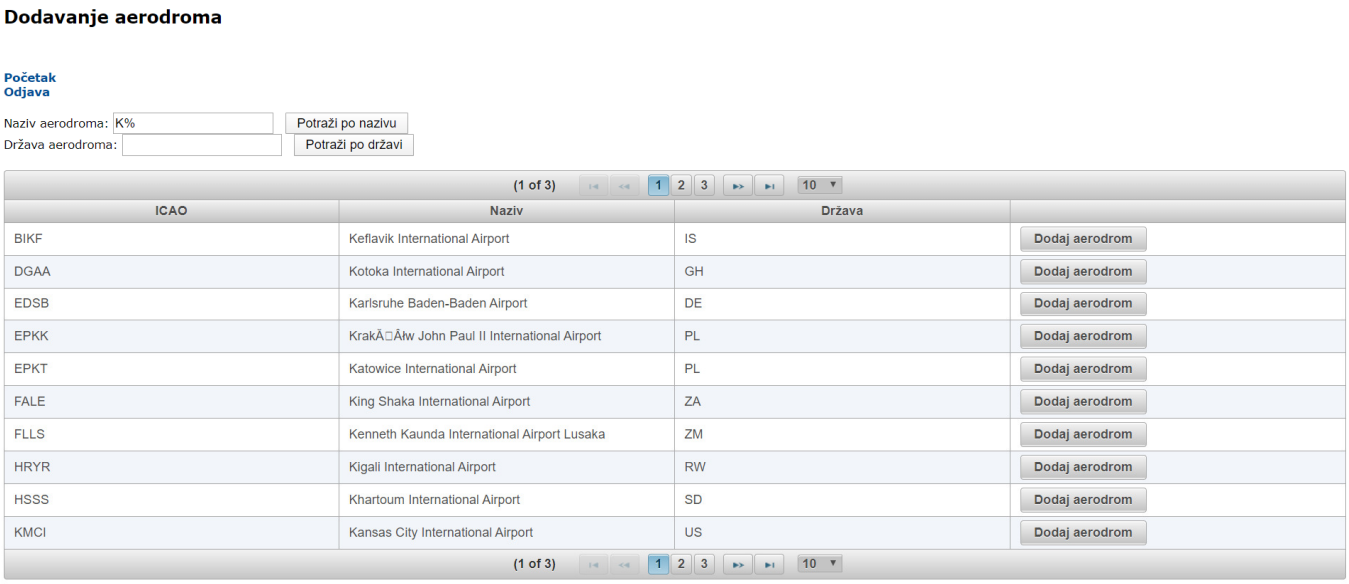
37. Postaviti potrebne elemente za unos podataka, za prikaz aerodroma koriste se <p:dataTable ...>, <p:column...>. Postoji primjer na predavanjima o JSF. Varijabla a se koristi u tablici za pojedini aerodrom u iteraciji za prikaz.
38. Za gumb se koristi <p:commandButton...> koji kod akcije poziva metodu kojom se prenosi odabrani aerodrom tj. njegov icao

```
#{dodavanjeAerodroma.dodajAerodromKorisniku(a.icao)}
```

39. Izgraditi, isporučiti, izvršiti i testirati.
40. Klasu DodavanjeAerodroma sa Refactor/Copy kopirati u **PregledAerodroma**
41. Potrebe su mala prilagođavanja metoda koje dohvaćaju podatke o aerodromima.
42. Kopirati JSF dodavanjeAerodroma.xhtml sa Copy/Paste i Refaktor/Rename u **pregledAerodroma.xthml.** Izgled može biti kao na Slika 4.
43. Obaviti potrebna prilagođavanja
44. Klasu PregledAerodroma sa Refactor/Copy kopirati u **PregledAviona**
45. Potrebe su mala prilagođavanja metoda koje dohvaćaju podatke o avionima.
46. Kopirati JSF pregledAerodroma.xthml sa Copy/Paste i Refaktor/Rename u **pregledAviona.xthml.** Izgled može biti kao na Slika 5.
47. Obaviti potrebna prilagođavanja
48. Izgraditi, isporučiti, izvršiti i testirati.

Tablica 1. Dodatne postavke u datoteci konfiguracije

Ključ	Vrijednost
LocationIQ.token	token za locationIQ
OpenWeatherMap.apikey	API key za Open Weather Map
OpenSkyNetwork.korisnik	korisničko ime za OpenSky Network
OpenSkyNetwork.lozinka	lozinka za OpenSky Network
preuzimanje.status	treba li preuzimati podatke o aerodromima (true-da, false-ne)
preuzimanje.ciklus	broj sekundi koliko traje svaki pravilan ciklus dretve za preuzimanje podataka aerodroma
preuzimanje.pauza	broj milisekundi za pauzu između preuzimanja podataka za dva aerodroma
preuzimanje.pocetak	dan od kojeg se počine preuzimanje, u formatu dd.mm.gggg
preuzimanje.kraj	dan do kojeg se radi preuzimanje, u formatu dd.mm.gggg



Slika 3. Izgled korisničkog sučelja za dodavanje aerodroma (pretraživanje po nazivu K%) (varijanta s tablicom)

Pregled aerodroma

Početak
Odjava

Od datuma: 25.04.2020 00:00 Do datuma: 26.04.2020 00:00

(1 of 4) 1 2 3 4 10

ICAO	Naziv	Država	
CYUL	Montreal / Pierre Elliott Trudeau International Airport	CA	Preuzmi avione / Preuzmi GPS i meteo
EBBR	Brussels Airport	BE	Preuzmi avione / Preuzmi GPS i meteo
EDDF	Frankfurt am Main Airport	DE	Preuzmi avione / Preuzmi GPS i meteo
EDDH	Hamburg Airport	DE	Preuzmi avione / Preuzmi GPS i meteo
EDDM	Munich Airport	DE	Preuzmi avione / Preuzmi GPS i meteo
EDDS	Stuttgart Airport	DE	Preuzmi avione / Preuzmi GPS i meteo
EDDT	Berlin-Tegel Airport	DE	Preuzmi avione / Preuzmi GPS i meteo
EGCC	Manchester Airport	GB	Preuzmi avione / Preuzmi GPS i meteo
EGGP	Liverpool John Lennon Airport	GB	Preuzmi avione / Preuzmi GPS i meteo
EGKK	London Gatwick Airport	GB	Preuzmi avione / Preuzmi GPS i meteo

(1 of 4) 1 2 3 4 10

GPS BP š: 53.353698 GPS BP d: -2.2749500274658203
GPS NA š: 53.35034205 GPS NA d: -2.2803692526643
Temp: 12.19
Vlaga: 71.0

Slika 4. Izgled korisničkog sučelja za pregled aerodoma koje prati korisnik pero. GPS lokacija i meteo podaci za aerodrom EGCC

Pregled aerodroma

Vrati se na aerodrome

(1 of 2) 1 2 10

ICAO24	Pozivni znak	1. vrijeme	2. vrijeme	PA	OA	
3c70aa	BCS974	25.04.2020 21:22:27	26.04.2020 06:07:59	EBBR	KCVG	Najveća visina
04015c	ETH3614	25.04.2020 21:20:51	26.04.2020 09:11:48	EBBR	VHHH	Najveća visina
06a1ed	QTR8248	25.04.2020 19:40:14	26.04.2020 01:34:51	EBBR	OTHH	Najveća visina
4bd8ac	THY6338	25.04.2020 19:23:33	25.04.2020 21:55:00	EBBR	LTBA	Najveća visina
3c5468	DLH8W	25.04.2020 15:35:02	25.04.2020 16:13:16	EBBR	EDDF	Najveća visina
89610a	UAE184	25.04.2020 15:19:31	25.04.2020 21:29:25	EBBR	OMDB	Najveća visina
06a2b1	QTR8195	25.04.2020 15:10:32	25.04.2020 21:12:56	EBBR	OTHH	Najveća visina
040168	ETH3723	25.04.2020 14:51:31	25.04.2020 21:25:39	EBBR	OMDW	Najveća visina
7812a3	CHH7974	25.04.2020 14:46:59	26.04.2020 01:30:21	EBBR	ZUCK	Najveća visina
780ef8	CHH8974	25.04.2020 14:18:18	26.04.2020 00:45:39	EBBR	ZUCK	Najveća visina

(1 of 2) 1 2 10

Vrijeme: 25.04.2020 08:40:21
Visina: 11582.0
GPS š: 55.1982
GPS d: 51.6021
Adresa: Мельничная улица, Тетвель, Rajon Nischnekamsk, Tatarstan, Volga Federal District, 423587, Russia

Slika 5. Izgled korisničkog sučelja za pregled aviona. Vrijeme, visina leta i GPS lokacija najveće visine leta aviona 780ef8