Vježba 11. / Zadaća 3.

Naziv: sustav aplikacija s korištenjem web servisa openweathermap.org i Google Maps API

Sustav je sastoji od dva projekta ({LDAP\_korisničko\_ime}\_zadaca\_3\_1 i {LDAP\_korisničko\_ime}\_zadaca\_3\_2). Prvi projekt ima 4 dijela:

1. u korisničkom dijelu potrebno je unositi pojedinačna parkirališta za koje će se preuzimati metorološki podaci. Prvo se unese naziv parkirališta i adresa, zatim se pokrene akcija koja preuzima njeni geolokacijski podaci putem Google Maps API. Slijedi pokretanje akcije za prikaz geolokacijski podataka. Nakon toga je akcija za spremanje podataka o parkiralištu u tablicu baze podataka (PARKIRALISTA). Zadnja akcija je preuzimanje važećih meteoroloških podataka za parkiralište na bazi njegovih geolokacijskih podataka te njihov prikaz na ekranu korisnika.
2. u pozadinskoj dretvi preuzimaju se u pravilnim intervalima meteorološki podaci putem REST web servisa openweathermap.org za izabrani skup parkirališta i pohranjuju se u tablicu u bazi podataka (METEO)
3. pruža SOAP web servis za meteorološke podatke spremljenih parkirališta. Operacije se temelje na podacima koje se nalaze u tablici METEO u bazi podataka. Potrebne su sljedeće operacije:
   * daje popis svih parkirališta i njihovih geo lokacija u application/json formatu
   * dodaj parkiralište (šalje se naziv i adresa koja služi za preuzimanje geo lokacije)
   * svi meteo podaci za parkiralište u intervalu (šalju se id parkirališta, od i do tipa long za timestamp, vraća se niz objekata klase Meteo)  (id je identifikator parkirališta u tablici PARKIRALISTAu bazi podataka)
   * zadnji meteo podaci za parkiralište (šalje se id parkirališta, vraća se objekat klase Meteo)  (id je identifikator parkirališta u tablici PARKIRALISTAu bazi podataka)
   * važeći meteo podaci za parkiralište (šalje se id parkirališta, vraća se objekat klase Meteo)  (id je identifikator parkirališta u tablici PARKIRALISTAu bazi podataka). Podaci se preuzimaju od Web servisa openweathermap.org
   * min i max važeća temperatura za parliralište u intervalu (šalju se id parkirališta, od i do tipa long za timestamp, vraća se niz od dva elementa tipa float)  (id je identifikator parkirališta u tablici PARKIRALISTAu bazi podataka)
4. pruža dva REST web servis za parkirališta, spremljene podatke o parkiralištima i meteo podacima:
   * GET metoda - osnovna adresa - vraća popis svih parkirališta, njihovih adesa i geo lokacija u application/json formatu. Struktura odgovora je u sljedeća: {"odgovor": [{...},{...}...], "status": "OK" | "ERR", <ako je "ERR" onda se  dodaje "poruka": poruka>}.
   * POST metoda - osnovna adresa - dodaj parkiralište (šalje se naziv i adresa koja služi za preuzimanje geo lokacije) Šalju se podaci u application/json formatu. Vraća odgovor u application/json formatu. Struktura odgovora je u sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se  dodaje "poruka": poruka>}.
   * GET metoda - na bazi putanje {id} za izabrano parkiralište vraća važeće meteorološke podatke (temp, vlaga, tlak) na bazi poziva REST web servisa openweathermap.org. id je identifikator parkirališta u tablici PARKIRALISTA u bazi podataka). Vraća podatke u application/json formatu. Sadržaj je u obliku Struktura odgovora je u sljedeća: {"odgovor": [{...}], "status": "OK" | "ERR", <ako je "ERR" onda se  dodaje "poruka": poruka>}.
   * PUT metoda - na bazi putanje {id} za izabrano parkiralište ažurira podatke (šalju se naziv i adresa, koja služi za preuzimanje geo lokacije). Šalju se podaci u application/json formatu. Vraća podatke u application/json formatu. Struktura odgovora je u sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se  dodaje "poruka": poruka>}.
   * DELETE metoda - na bazi putanje {id} briše izabrano parkiralište. Vraća podatke u application/json formatu. Struktura odgovora je u sljedeća: {"odgovor": [], "status": "OK" | "ERR", <ako je "ERR" onda se  dodaje "poruka": poruka>}.

Drugi projekt sastoji se od korisničkog dijela u kojem se mogu obaviti dvije aktivnosti:

* unositi pojedinačna parkirališta za koje će se preuzimati metorološki podaci. Unese se naziv parkirališta i adresa. Postoje dva gumba "Upiši SOAP" i "Upiši REST". Svaki od njih pokreće akciju koja poziva određenu operaciju web servisa (SOAP i REST) iz prvog projekta.
* preuzeti parkirališta za koje se prikupljaju meteorološki podaci u prvom projektu te se prikazuju u obliku padajućeg izbornika s mogućim odabirom više elemenata.
  + ako se odabere samo jedno parkiralište tada se mogu aktivirati gumbi "Preuzmi REST", "Ažuriraj REST"i "Briši REST". Prvim se pokreće akcija koja će preuzeti podatke o odabranom parkiralištu. Šalju se podaci u application/json formatu. Zatim će se prikazati njegov naziv i adresa unutar podataka za unos u obrascu. Drugim se pokreće akcija koja će ažurirati podatke o odabranom parkiralištu. Koriste se podaci za unos u obrascu. Šalju se podaci u application/json formatu. Trećim se pokreće akcija koja će obrisati odabrano parkiralište.
  + ako se odaberu minimalno dva parlirališta tada se može aktivirati gumb "Preuzmi SOAP",  kojim se pokreće akcija koja će preuzeti sve pohranjenje meteorološke podatke za odabrano parkiralište putem SOAP web servisa iz prvog projekta. Preuzeti meteorološki podaci prikazuju se u obliku tablice.

APPID (APIKEY) za openweathermap treba biti spremljen u konfiguracijskoj datoteci.

Klase i metode trebaju biti komentirane u javadoc formatu.**Projekt se isključivo treba predati u formatu NetBeans projekta.** Prije predavanja projekta potrebno je napraviti Clean na projektu. Zatim cijeli projekt sažeti u .zip (NE .rar) format s nazivom Prije predavanja projekta potrebno je napraviti Clean na projektu. Zatim cijeli projekt sažeti u .zip (NE .rar) format s nazivom **{LDAP\_korisničko\_ime}\_zadaca\_3.zip** i predati u Moodle. Uključiti izvorni kod, primjere datoteka konfiguracijskih podataka i popunjeni obrazac za zadaću pod nazivom{LDAP\_korisničko\_ime}\_zadaca\_3.doc (u korijenskom direktoriju projekta).

Pregledi trebaju biti vizualno sređeni pomoću css-a.

Metode u klasama NE smiju imati više od 25 linija programskog koda, u što se ne broji definiranje metode, njenih argumenata i lokalnih varijabli. U jednoj liniji može biti jedna instrukcija. Linija ne može ima više od 132 znaka.

Naziv projekta: **{LDAP\_korisničko\_ime}\_zadaca\_3**

1. Kreiranje 11. vježbe/zadaće 3 (direktorij {LDAP\_korisničko\_ime}\_zadaca\_3). U nastavku se direktorij za vježbu simbolički označava kao {vježba}).
2. Kreiranje projekta **{LDAP\_korisničko\_ime}\_zadaca\_3\_1** kao Java Web aplikaciju, server **Tomcat 8.\***, Java EE verzija: **Java EE 7**, kontekst **{LDAP\_korisničko\_ime}\_zadaca\_3\_1**, (Frameworks: Ne)
3. Kreirati pakete **org.foi.nwtis.{LDAP\_korisnik}.ws.klijenti, org.foi.nwtis.{LDAP\_korisnik}.ws.serveri, org.foi.nwtis.{LDAP\_korisnik}.rest.klijenti, org.foi.nwtis.{LDAP\_korisnik}.rest.serveri, org.foi.nwtis.{LDAP\_korisnik}.web.podaci, org.foi.nwtis.{LDAP\_korisnik}.web.slusaci**
4. Kreirati direktorij lib u projektu
5. Kopiranje biblioteke dist\vjezba\_03\_2.jar iz projekta **vjezba\_03\_2** u direktorij lib
6. Kopiranje biblioteke dist\vjezba\_06\_1.jar iz projekta **vjezba\_06\_1** u direktorij lib
7. Dodavanje biblioteke (jar/folder) lib\vjezba\_03\_2.jar u projekt
8. Dodavanje biblioteke (jar/folder) lib\vjezba\_06\_1.jar u projekt
9. Kopirati  glassfish-4.1.?\glassfish\modules\javax.json.jar u direktorij lib
10. Dodavanje biblioteke (jar/folder) lib\gson.jar u projekt
11. Dodavanje biblioteke (jar/folder) lib\javax.xson.jar u projekt
12. Kopiranje biblioteke gson.jar iz projekta {LDAP\_korisnicko\_ime}\_zadaca\_2 u direktorij lib
13. Dodavanje biblioteka Java DB Driver, Java EE7 API Library, JAX-RS, JAX-WS, Jersey
14. Preuzeti primjere datoteke konfiguracija za JavaDB (NWTiS.db.config\_2.xml) iz vjezba\_06\_2 te kopirati u direktorij WEB-INF
15. kreirati **web.xml** (New file/Web/Standard Deployment Descriptor) i upisati u web.xml inicijalni parametar konteksta **konfiguracija**
16. kreirati slušaca konteksta **SlusacAplikacije** (anotirani bez uključivanja u web.xml) i kopirati kod funkcije iz vjezba\_08\_1 ili  {LDAP\_korisničko\_ime}\_zadaca\_2. Vrijednost ServletContext spremiti u varijablu klase kako bi se jednostavno pristupilo podacima iz konteksta (aplikacije). Dodati getter za varijablu.
17. izgraditi, isporučiti, izvršiti i testirati aplikaciju
18. izvršiti poziv REST web servisa Google Maps API <http://maps.google.com/maps/api/geocode/json?address=xxx&sensor=false> (umjesto xxx upusati adresu po želji) npr. <http://maps.google.com/maps/api/geocode/json?address=Croatia,Vara%C5%BEdin,Pavlinska%202&sensor=false>
19. preuzeti geolokacijske podatke (latitude i longitude)
20. izvršiti poziv REST servisa openweathermap.org za važeće meteorološke podatke <http://api.openweathermap.org/data/2.5/weather?lat=aaa.aaa&lon=bbb.bbb&units=metric&lang=hr&APIKEY=nnn>  (umjesto aaa.aaa upsati latitude, umjesto bbb.bbb upisati longitude, umjesto nnn upisati APPID)
21. preuzeti priložene klase za servis (Klase: GMRESTHelper, OWMRESTHelper, MeteoPodaci, Lokacija, Parkiraliste)
22. kreirati novi SOAP web servis **GeoMeteoWS**(New/Other/Web Services/Web Service)  u paketu org.foi.nwtis.{LDAP\_korisnik}.ws.serveri
23. slijedi prikaz dijaloškog okvira (Slika 1.) u vezi uključivanja GEOMETEO web servisa u projekt. Odabrati Yes.
24. otvoriti klasu GeoMeteoWS i odabrati Design
25. kliknuti na operaciju hello i obrisati operaciju hello
26. promjeniti u Properites za projekt Source/Binary/Format u 1.8 (ako nije)
27. preuzeti sa Moodla iz zadaće 2. SQL za kreiranje tablice PARKIRALISTA i izvršiti u JavaDB bazi podataka nwtsi\_g{1,2,3,4}
28. preuzetu SQL datoteku meteo.sql izvršiti u JavaDB bazi podataka nwtsi\_g{1,2,3,4}
29. dodati operaciju dajSvaParkiralista(), a vraća java.util.List<Parkiraliste>
30. izgraditi i isporučiti aplikaciju
31. testirati web servis, desni klik na web servis i Test Web Service, otići na adresu [http://localhost:xxxx/{LDAP\_korisničko\_ime}\_zadaca\_3\_1/GeoMeteoWS?WSDL](http://localhostxxxx/) i upamtiti adresu
32. testirati operaciju web servisa. Kartica Services/Web Services pa desna tipka na mišu Create Group/NWTiS, desna tipka na mišu pa Add Web Service... i kopirati adresu WSDL-a. Otvoriti web servis i odabrati operaciju, desna tipka na mišu i Test Method i Submit
33. dodati **index.jsp**  (izgled kao na slici 5)
34. u web.xml promijeniti da je index.jsp početna stranica (Pages/Welcome Files)
35. obrisati index.html
36. dodati u**index.jsp** obrazac za unos naziva parkirališta i njegove adrese i gumba za akcije
37. kreirati servlet DodajParkiraliste(paket org.foi.nwtis.{LDAP\_korisnik}.web) koji će primiti podatke i obaviti potrebne akcije
38. dohvat geo podataka nakon akcije na gumbu
39. spremanje podataka o parkiralištu u tablicu baze podataka PARKIRALISTA
40. dohvat važećih meteopodataka upisane adrese parkirališta
41. kreirati dretvu PreuzmiMeteoPodatke koja će za spremljena parkirališta odnosno njihove geo lokacije preuzimati meterološke podatke i spremati u odgovarajuću tablicu pod nazivom METEO. Dretvu treba startati u slušaču aplikacije kod kreiranje konteksta. Interval dretve određen je konfiguracijskim podatkom.
42. dodati i dovršiti dodatne operacije web servisa GeoMeteoWS:
    * dajSvaParkiralista()  - vraća List<Parkiraliste> sa svim parkiralištima koja se nalaze u tablici PARKIRALISTA
    * dodajParkiraliste(Parkiraliste)  - dodaje novo parkiralište u tablicu PARKIRALISTA
    * dajSveMeteoPodatke(int) - vraća sve spremljene podatke iz baze podataka za uneseno parkirališta, ukoliko nema podataka vraća null
    * dajSveMeteoPodatke(int, long, long) - vraća sve spremljene podatke iz baze podataka za uneseno parkirališta i interval (timestamp), ukoliko nema podataka vraća null
    * dajZadnjeMeteoPodatke(int) - vraća posljednje spremljene meteo podatake iz baze podatka za uneseno parkiralište ukoliko nema podataka vraća null
    * dajVazeceMeteoPodatke(int) - vraća važeće meteo podatake na bazi web servisa  za uneseno parkiralište ukoliko nema podataka vraća null
    * dajMinMaxTemp(int, long, long) - vraća min i max važeće temperature iz baze podatka za uneseno parkiralište i interval, ukoliko nema podataka vraća null
43. testirati kreirane operacije
44. kreirati RESTful web servis u paketu org.foi.nwtis.{LDAP\_korisnik}.rest.serveri (Other/Web Services/RESTful Web Services from Patterns/Simple Root Resource/ (Slika 2) s nazivom putanje **meteo**  klasa **MeteoREST**, tip application/json, Representation Clakk String.(Slika 3.)
45. u klasi MeteoREST u metodi getJson() staviti da vraća popis svih parkirališta koja se nalaze u tablici PARKIRALISTA iz baze podataka u application/json formatu. Potrebno je napraviti upit prema bazi podataka. Pripremiti odgovor u application/json formatu i zadanom strukturom.
46. u metodi postJson(String) staviti da dodaje parkiralište. Prima podatke u application/json formatu. Pripremiti odgovor u application/json formatu i zadanom strukturom.
47. u metodi putJson(String) i delete Json staviti da vraća pogrešku. Pripremiti odgovor u application/json formatu i zadanom strukturom.
48. dodati metodu:
    * @GET  
      @Produces(MediaType.APPLICATION\_JSON)  
      @Path("{id}")  
      public String getJson(@PathParam("id") String id) {  
          return ...  
      }
49. na isti način pripremiti ostale metode koje će imati u putanje {id} za post, put i delete metode
50. u metodi getJson() staviti da vraća na bazi putanje {id} važeće meteorološke podatke izabranog IoT uređaja. Pripremiti odgovor u application/json formatu i zadanom strukturom.
51. u metodi putJson(@PathParam("id") String id) staviti da ažurira parkiralište. Prima podatke u application/json formatu. Pripremiti odgovor u application/json formatu i zadanom strukturom.
52. u metodi deleteJson(@PathParam("id") String id) staviti da briše parkiralište. Pripremiti odgovor u application/json formatu i zadanom strukturom.
53. izgraditi i isporučiti aplikaciju
54. testirati RESTful web servis (RESTFul Test Services/Test RESTFul Test Services)
55. izvršiti RESTful http://localhost:xxxx/{LDAP\_korisnicko\_ime}\_zadaca\_3\_1/{web}resources/meteoREST/
56. izvršiti RESTful http://localhost:xxxx/{LDAP\_korisnicko\_ime}\_zadaca\_3\_1/{web}resources/meteoREST/1 i druge id parkirališta
57. instalirati dodatak (plugin) postman u preglednik. Potražiti u google npr. za preglednik Chrome "postman chrome" i zatim ga instalirati.
58. testirati gornje servise s različitim metodama
59. Kreiranje projekta **{LDAP\_korisnicko\_ime}\_zadaca\_3\_2** kao Java Web aplikaciju, server Glassfish, Java EE verzija: Java EE 7, kontekst {LDAP\_korisnicko\_ime}\_zadaca\_3\_2 , (Frameworks: JSF), Libraries/Server Library: JSF 2.\*, Configuration/JSF Servlet URL pattern: /faces/\* , Prefered Page Language: Facelets
60. kreirati pakete **org.foi.nwtis.{LDAP\_korisnik}.ws.klijenti,  org.foi.nwtis.{LDAP\_korisnik}.web.zrna**
61. kreirati novog klijenta web servisa (New/Other/Web Services/Web Service Client) za Web servis MeteoWS koji ima adresu WSDL na bazi projekta {LDAP\_korisnicko\_ime}\_zadaca\_3\_1
62. kreirati klasu **MeteoWSKlijent** u paketu org.foi.nwtis.{LDAP\_korisnik}.ws.klijenti
63. napraviti poziv operacije**dajSvaParkiralista**na web servisu GeoMeteoWS (Insert Code/Call Web Service Operation...)
64. promijeniti za metodu iz private u public
65. kreirati JSF MenagesBean **OperacijaParkiraliste** (razmisliti o primjeni session ili request)
66. dodati varijable za naziv i adresu tipa String, dodati gettere i settere
67. dodati varijablu za preuzimanje uređaja List<Parkiraliste>, dodati getter i setter
68. dodati varijablu za odabrana parkirališta, dodati getter i setter
69. dodati varijablu za preuzimanje meteo podataka List<Meteo>, dodati getter
70. dodati funkciju String upisiSOAP() za dodavanje parkirališta pomoću operacije SOAP web servisa. Ako se javi pogreška treba ispisati opis pogreške u elementu za poruke.
71. dodati funkciju String upisiREST() za dodavanje parkirališta pomoću operacije REST web servisa. Ako se javi pogreška treba ispisati opis pogreške u elementu za poruke.
72. dodati funkciju String preuzmiREST() za preuzimanje podataka odabranog parkirališta pomoću operacije REST web servisa. Podaci o nazivu i adresi prenose se u elemente u obrascu. Ako se javi pogreška treba ispisati opis pogreške u elementu za poruke.
73. dodati funkciju String azururajREST() za ažuriranje podataka odabranog parkirališta pomoću operacije REST web servisa. Podaci o nazivu i adresi preuzimaju se iz elemenata u obrascu. Ako se javi pogreška treba ispisati opis pogreške u elementu za poruke.
74. dodati funkciju String brisiREST() za brisanje odabranog parkirališta pomoću operacije REST web servisa. Ako se javi pogreška treba ispisati opis pogreške u elementu za poruke.
75. dodati funkciju String preuzmiSOAP() za preuzimanje podataka odabranog parkirališta pomoću operacije SOAP web servisa. Podaci o nazivu i adresi prenose se u elemente u obrascu. Ako se javi pogreška treba ispisati opis pogreške u elementu za poruke.
76. dodati funkciju String preuzmiMeteo() za preuzimanje meteo podataka odabranog parkirališta pomoću operacije SOAP web servisa. Podaci o meteo podacima prikazuju se u tablici u obrascu. Ako se javi pogreška treba ispisati opis pogreške u elementu za poruke.
77. kreirati JSF **operacijaParkiraliste.xhtml** (izgled kao na slici 6)
78. dodati elemente za unos podataka za naziv, adresu i poruke u obrascu i povezati ih na varijable u zrnu OperacijaParkiraliste
79. dodati element za izbornik u obrascu i povezati ga na varijable za podatke u izborniku i odabir elemenata u zrnu OperacijaParkiraliste
80. dodati potrebne gumbe i povezati njihove akcije na funkcije u zrnu OperacijaParkiraliste
81. dodati element <h:message... > za prikaz poruka (crvenom bojom naglašen element na slici)
82. dodati element <h:dataTable... > za prikaz preuzetih podataka
83. izgraditi i isporučiti aplikaciju

Slika 1.

Slika 2.

Slika 3. 

Slika 4.

Slika 5.