.

Univerzitet u Beogradu
Matematički fakultet

REST servisi u Javi i njihova primena u aplikaciji telekomunikacionog operatera

Master rad

Mentor: dr Vladimir Filipović

Student: Bojana Zečević
br. indeksa: 1108/2011

Septembar 2015.

SADRŽAJ

1 UVOD

Elektronski računi (elektronske fakture) sve više zamenjuju konvencionalne, papirne račune. Korisnici na željenu e-mail adresu dobijaju elektronski račun koji je istog sadržaja kao i papirni. Usluga odabira elektronskog računa se sve više usvaja, kako u Srbiji, tako i u drugim zemljama.

Kompanija SBB (Serbia BroadBand) je svojim korisnicima omogućila da jednostavnim prijavljivanjem na neki od veb portala (https://www.mojsbb.rs ili https://www.mojtotaltv.tv) ili prijavom u prodajno-korisničkim objektima (PKO) aktiviraju uslugu elektronskog računa. Odabirom ovakvog načina prijema računa, korisnicima više neće stizati papirni računi, a aplikacija će im elektronskim putem poslati račune u odgovarajućem trenutku u mesecu. Prednosti uvođenja ovog servisa su smanjenje troškova štampe računa, smanjena potrošnja papira i to što omogućava korisnicima da dobiju svoj račun gde god da se nalaze. Korisnici račune mogu sačuvati na svom računaru i tako imati celu istoriju svojih računa.

Usluga elektronskog slanja računa je u SBB-u omogućena od aprila 2013. godine. Što se obima aktivnosti tiče, trenutno ima preko 50 000 korisnika u 3 kompanije (SBB, TotalTV i KDS Novi Sad) koji su aktivirali ovu uslugu. Fakturisanje se vrši početkom svakog meseca za prethodni mesec, a zatim se računi šalju korisnicima.

Aplikacija e_bill za obezbeđenje usluge slanja elektronskog računa je interno razvijena na Java platformi korišćenjem REST veb servisa, a ja sam bila zadužena za projektovanje baze podataka, kreiranje REST veb servisa i pozadinskog sistema koji omogućuje funkcionalnosti prijavljivanja i odjavljivanja korisnika za uslugu elektronskog računa, promenu e-mail adrese, slanja elektronskog računa, masovnog slanja elektronskih računa, kao i za izradu veb interfejsa ka operateru.

U prvom delu ovog rada biće opisani osnovni principi REST veb servisa, oganičenja ove tehnologije i arhitekturni elementi u okviru ovog servisa, dok će u drugom delu akcenat biti na praktičnoj primeni u okviru razvoja aplikacije za slanje elektronskog računa.

Cilj ovog rada je da doprinese većoj ekspanziji novog načina slanja i primanja računa.

2 ELEKTRONSKI RAČUN

Elektronski račun ili elektronska faktura predstavlja elektronsku verziju papirnog računa. To je račun koji se izdaje, prima i obrađuje elektronskim putem i šalje uz pomoć elektronske infrastrukture, najčešće Interneta.

Jedna od glavnih prednosti slanja elektronskog, u odnosu na papirni račun, jeste smanjenje izuzetno velikih finansijskih troškova, koji nastaju štampanjem i obračunom PTT usluga. Takođe, smanjuje se vreme od izdavanja do prijema računa, a smanjenom potrošnjom papira se doprinosi očuvanju životne sredine i daje se doprinos razvoju ekološke svesti.

Postoje dva modela elektronskog računa [15]:

- Model direktnog plaćanja - odnosi se na pristup u kome potrošači vrše plaćanja direktno na izvršavaocu računa čije račune oni primaju na sajtu firme koja je izdala račun.

- Model konsolidacije - odnosi se na pristup kod koga plaćanje vrši agregator ili konsolidator, obično sa posrednikom u radu sa potrošačima i bankama. Ovaj model omogućava potrošaču isplate više različitih faktura izdavaocima koji su registrovani da primaju uplate.

Račun ili faktura predstavlja dokument koji izdaje kompanija (prodavac) za neku uslugu (proizvod) koju pruža korisniku (kupcu). On obavezuje kupca da, u skladu sa ugovorom koji je sklopio sa kompanijom, izvrši uplatu u iznosu naznačenom na računu.

Račun se sastoji od sledećih podataka [16]:

- Naziv, adresa i PIB/OIB obveznika – izdavaoca računa
- Mesto i datum izdavanja i redni broj računa
- Naziv, adresa i PIB/OIB obveznika – primaoca računa
- Vrsta i količina isporučenih dobara ili vrsta i obim usluga
- Datum prometa dobara i usluga i visinu avansnih plaćanja
- Iznos osnovice
- Poreska stopa koja se primenjuje

- Iznos PDV-a koji je obračunat na osnovicu
- Napomena o poreskom oslobođenju

3 REST VEB SERVISI

3.1 Veb servisi

Prema definiciji WWW Konzorcijuma (World Wide Web Consortium, W3C), veb servis je softverski sistem dizajniran da podrži interoperabilnu interakciju mašina preko mreže [19]. Interfejs veb servisa je opisan u formatu koji može mašina da obradi (posebno WSDL). Drugi sistemi mogu da ostvare komunikaciju sa veb servisom putem HTTP protokola, pri čemu mogu koristiti SOAP (engl. Simple Object Access Protocol) protokol.

3.2 REST Veb Servisi

Američki naučnik Roj Filding (Roy Fielding) je u svojoj doktorskoj disertaciji [3] pod nazivom „Arhitekturni stil i dizajn mrežno-orijentisane softverske ahitekture“, objavljenoj 2000. godine, prvi put pomenuo pojam REST.

Po njegovim rečima:

„Motivacija za razvoj REST-a je pravljenje modela arhitekture koji opisuju kako bi Veb trebalo da radi, takvog da može da posluži kao orijentir pri definisanju standarda protokola za Veb“.

REST je skraćenica od REpresentational State Transfer („prenos stanja reprezentacije“) i predstavlja skup ograničenja koja, kada se primene na dizajn sistema, kreiraju softverski arhitekturni stil. [11]

Ograničenja koja definišu REST sistem, kažu da on mora da bude:

- **klijent-server**

  Ovo ograničenje se odnosi na interakciju klijenta i servera, koji su odvojeni jednoznačnim interfejsom. Zaduženja klijenta i servera su skroz odvojena, pa se mogu nezavisno razvijati i menjati, dok se interfejs između njih ne menja.

- **bez stanja (stateless)**

  Svaki klijentski zahtev ka serveru mora da sadrži sve potrebne informacije za razumevanje zahteva i ne može da iskoristi sadržaje koji već postoje na serveru. Zbog toga klijent čuva stanje sesije. [3]

- **da podrži keširanje**

  Ovo ograničenje zahteva da se podaci u okviru odgovora na klijentski zahtev implicitno ili eksplicitno označe kao oni koji se mogu keširati ili oni koji se ne mogu kreširati. Ako je u odgovoru označeno da se može keširati, tada je klijent slobodan da te podatke koristi i za kasnije, ekvivalentne zahteve. [3]

- **uniformno dostupan**

  Glavna odlika kojom se REST arhitekturni stil razlikuje od ostalih mrežno orijentisanih stilova je uniformni interfejs među komponentama.Time se razdvaja implementacija servisa od usluga koje pruža. Obezbeđivanje uniformnog interfejsa se postiže uvođenjem sledeća četiri ograničenja: identifikacija resura, manipulacija resursima kroz njihove reprezentacije, poruke koje same sebe opisuju i HTTP kao način perzistencije stanja aplikacije.[3]

- **slojevit**

  Komunikacija klijenta i servera ne mora da bude direktna, već mogu da postoje hijerarhijski uređeni slojevi. Komponenta jednog sloja može da komunicira samo sa komponentama sloja sa kojim interaguje. Slojevitost doprinosi poboljšanju skalabilnosti omogućujući balansiranje i obezbeđivanje zajedničkog keša. [18]

- **da obezbedi kod na zahtev (opciono ograničenje)**

  Ovo ograničenje je jedino opciono. REST dozvoljava proširenje klijentske funkcionalnosti izvršavanjem koda u formi apleta ili skripti. Na ovaj način se pojednostavljuje implementacija klijentske strane i smanjuje vidljivost. [3]

Ova ograničenja ne diktiraju koja vrsta tehnologije se koristi, već samo definišu kako se podaci prenose između komponenata. U tom smislu, RESTful sistem se može implementirati u bilo kojoj mrežnoj arhitekturi. [11]

3.3 Resursi

Resurs je sve adresabilno na veb-u. Pod adresabilnošću se podrazumeva da se resursu može pristupiti ili da se može preneti između klijenta i servera. [11]

Neki od primera resursa su:

- zaposleni neke kompanije
- kursna lista današnjeg dana
- temperatura u Beogradu
- račun korisnika SBB-a za određeni mesec

Ukoliko se uzme u razmatranje račun korisnika SBB-a za neki mesec, slanjem zateva za određeni mesec dobijamo jedinstvenu reprezentaciju resursa. Međutim, može se desiti i da se prilikom različitih zahteva dobije ista reprezentacija. Tako, na primer, zahtevi:

1. račun korisnika sa brojem ugovora 123456789 za mesec maj 2015. godine
2. najveći račun korinika sa brojem ugovora 123456789 u 2015. godini

će vratiti istu reprezentaciju resursa, ukoliko je upravo maj bio mesec u kome je dati korisnik zabeležio najveću potrošnju u 2015. godini.

Predstavaljanje reprezentacije resursa, odnosno trenutnog stanja određenog podatka, može se predstaviti različitim formama. Za sve zahteve biće isti URI (engl. Uniform Resource Identifier), a forma reprezentacije zavisiće od zahteva klijenta. Ukoliko klijent zahteva reprezentaciju koja je pregledna i čitljiva, obično će onda biti u HTML formi, a ukoliko to nije važno (na primer, kod zahteva drugih veb servisa), onda se obično predstavlja u XML (engl. Extensible Markup Language) ili JSON (engl. JavaScript Object Notation) formi. Deteljno razmatranje Jave, XML-a i JSON-a izlazi iz okvira ovog rada. Više informacija o Javi mogu da se nađu u knjizi [4], o XML-u u knjizi [2], a o JSON-u u knjizi [12].

3.4 REST metode

Komunikacija između klijenta i servera se vrši putem mrežnih protokola, najčeće HTTP-a.

Postoje četiri metode koje se koriste kod REST servisa:

1. **GET** – *Metoda koja služi za dohvatanje resursa.*

   Klijent kreira zahtev koji sadrži GET metodu, identifikator podatka za koji se traži reprezentacija, kao i tip formata u kome će dobiti odgovor. Zatim se serveru šalje zahtev, gde ga prihvata REST okvir, koji ga dalje prosleđuje Java kodu, koji ga obrađuje i od koga se dobija tražena reprezentacija. Ona se konvertuje u prosleđeni format, a onda se preko HTTP-a vraća klijentu.

   Bitne karakteristike GET metode su idempotentnost i bezbednost. Idempotentnost znači da, bez obrizira na to koliko puta se primenjuje operacija, rezultat će biti isti, dok bezbednost znači da se pozivima GET metode ne menja stanje na serveru. [1]

2. **POST** – *Metoda koja služi za kreiranje resursa*

   Klijent kreira zahtev sa URI-jem i POST metodom koja se prosleđuje serveru. REST okvir ga prihvata i prosleđuje Java kodu, koji vrši kreiranje resursa. Kada se proces kreiranja završi, server vraća odgovor klijentu.

   POST nije ni idempotentna ni bezbedna operacija, jer mu je pri svakom zahtevu dozoljeno da modifikuje servis na jedinstven način. [1]

3. **PUT** - *Metoda koja služi za ažuriranje resursa*

   Klijent kreira zahtev sa URI-jem za koji se vrši ažuriranje PUT metodom. REST okvir ga prihvata i prosleđuje Java kodu, koji pronalazi resurs i vrši ažuriranje, a zatim se klijentu vraća odgovor.

   PUT metoda je idempotentna, što znači da slanje istog zahteva neće imati uticaja na osnovni servis. [1] Zbog ove karakteristike, PUT metodu se, osim za kreiranje, može koristiti i za ažuriranje.

4. **DELETE –** *Medota koja služi za brisanje resursa*

   Zahtev klijenta se sastoji od URI-ja koji želimo obrisati i DELETE metode. Zahtev se šalje serveru, čiji ga REST okvir prihvata i prosleđuje Java kodu. Vrši se brisanje resursa i vraćanja odgovora klijentu. Delete je takođe idempotentna metoda.

3.5 Spojnice

REST koristi razne vrste spojnica (engl. connector) da enkapsulira aktivnosti pristupanja i prenosa reprezentacija resursa. Spojnice predstavljaju apstraktni interfejs za komunikaciju između komponenata. Oni povećavaju jednostavnost komunikacije, tako što potpuno odvajaju zaduženja i što sakrivaju implementaciju resursa i komunikacione mehanizme. Uopštenost

interfejsa takođe omogućava zamenu komponenata: ako korisnik pristupa sistemu samo preko apstraktnog interfejsa, implementacija može biti zamenjena bez uticaja na korisnike. Pošto spojnica upravlja mrežnom komunikacijom, informacija može biti poslata kroz više interakcija kako bi se povećala efikasnost i funkcionalnost. [3]

Sve REST interakcije su bez stanja. To znači da svaki zahtev sadrži sve potrebne informacije, da bi spojnica mogla da ga razume, nezavisno od bilo kojih zahteva koji mu prethode. Ovo ograničenje ostvaruje četiri funkcije: 1) otklanja potrebu da spojnice čuvaju stanje aplikacije između zahteva, čime se smanjuje potrošnja fizičkih resursa i poboljšava skalabilnost; 2) omogućava da se interakcije odvijaju paralelno, bez potrebe da procesni mehanizam razume semantiku interakcije; 3) omogućava posredniku da vidi i razume zahtev u izolaciji, što može biti neophodno kada se servisi dinamički preuređuju; 4) prisiljava sve informacije koje bi mogle da se ponovo koriste da budu prisutne u svakom zahtevu. [3]

Primarni tipovi spojnica su server i klijent. Bitna razlika između njih je što klijent inicira komunikaciju šaljući zahtev, dok server osluškuje i odgovara na zahteve u cilju da obezbedi pristup svojim uslugama. [3]

Treća vrsta spojnica, keš spojnica, može biti smešten na interfejs klijenta ili servera kako bi se sačuvali keširani odgovori date interakcije, kako bi mogli biti ponovo korišćeni u drugim interakcijama. Korišćenjem keša se smanjuje kašnjenje u komunikaciji i on je najčešće implementiran u okviru adresnog prostora spojnice koja ga koristi.[3]

Rezolver, tip spojnice, prevodi delimične ili kompletne identifikatore resursa u mrežne adrese, potrebne za uspostavljanje među komponentne konekcije. Da bi inicirao zahtev, Web pretraživač mora uzeti ime hosta iz URI-a i ,koristeći DNS resolver, dobije adresu Internet protokola. Poslednja vrsta spojnice je tunel, koji jednostavno prenosi komunikaciju preko konekcionih granica kao što su *firewall* i mrežni *gateway*. [3]

3.6 Komponente

U REST komponente spadaju: izvorni server, gateway, proxy i korisnički agent. *Korisnički agent* koristi klijentsku spojnicu da otpočne slanje zahteva i postane jedini koji prima zahtev. Najčešći primer za to je Veb pretraživač koji obezbeđuje pristup informacijama servisa i pruža odgovore na osnovu svojstava aplikacije. *Izvorni server* koristi serversku spojnicu da upravlja zahtevanim resursom. To je definitivno izvor za reprezentacije resursa, i mora biti krajnji primalac bilo kog zahteva koji nastoji da modifikuje vrednost svojih resursa.

Svaki izvorni server obezbeđuje generički interfejs za svoje resurse zasnovan na hijerarhiji resursa. *Proxy* komponenta je posrednik odabran od strane klijenta da obezbedi interfejs enkapsulaciju drugih servisa, prevoda podataka, poboljšanja performansi ili zaštite sigurnosti. *Gateway* komponenta je posrednik koga nameće mreža ili izvorni server da obezbedi interfejs enkapsulaciju drugih servisa, prevoda podataka, poboljšanja performansi ili zaštite sigurnosti. [3]

3.7 REST sa Javom (JAX-RS)

Java programski API, koji ispunjava ograničenja REST sistema i služi za kreiranje veb servisa, naziva se JAX-RS API (ili JAX-311). Jedan od osnovnih ciljeva njegovog nastanka je bio da učini RESTful veb servise jednostavnijim. JAX-RS obezbeđuje spojnice za veb servise kroz Java anotacije – anotacije automatski generišu kod neophodan za klase koji ih koriste da se povežu sa određenim radnim okvirom.[11]

3.8 Anotacije

Anotacije se u programskom jeziku Java pojavljuju u unapređenom JDK 5. Java anotacija je forma sintaksnog metapodatka koji može biti dodat na Java izvorni kod. Anotacije se mogu dodavati iznad klasa, metoda, promenljivih, parametara i paketa, tako što se neposredno iznad njihove definicije stavlja et-znak (@) posle koga slede naziv anotacije, a u zavisnosti od same anotacije, nakon toga može da sledi i zagrada u kojoj su smešteni elementi i njihove vrednosti. Anotacije se prevode u klasne datoteke (engl. class files) i dostupne su procesoru ili korisniku prilikom izvršavavanja. U zavisnosti od potrebe, moguće je kreirati i anotacije koje se koriste samo u kodu i one se ne prevode, a mogu se kreirati i anotacije koje se prevode, ali nisu dostupne za vreme izvršavanja. Neke upotrebe anotacija su: automatska izrada izveštaja, generisanje koda ili XML datoteka, primenjuju se u veb servisima, a takođe su i bitna informacija za kompilator.

@PATH anotacija

Anotacija @Path podešava putanju na osnovni URL+/naziv_putanje. Osnovni URL se bazira na imenu aplikacije, servletu i URL paternu koji se nalazi u web.xml konfiguracionoj datoteci.

U aplikaciji e_bill, neposredno ispred definicije klase ActivateEBillAction, postavlja se anotacija @Path

```
@Path(value = "/ebill/sbb")

public class ActivateEBillAction{ . . . }
```

Kako je naziv aplikacije e_bill, a u web.xml-u imamo

```
<context-param>
        <param-name>resteasy.servlet.mapping.prefix</param-name>
        <param-value>/resteasy</param-value>

</context-param>
```

Na ovaj način se postiže da putanja do svih resursa (ukoliko se aplikacija nalazi na lokalnoj mašini) sa kojima komunicira ova klasa bude:

http://localhost:8080/e_bill/resteasy/ebill/sbb/

U istoj klasi se i iznad metoda nalazi anotacija @Path, na primer kod metode:

```
@POST
@Produces(MediaType.APPLICATION_JSON)
@Path(value = "/addNewEBill")
public Response addNewEBill(
        @FormParam(value = "contractNo") String contractNo,
        @FormParam(value = "email") String email,
        @FormParam(value = "company") String company,
        @FormParam(value = "insertInputSystem") String insertInputSystem,
        @FormParam(value = "insertOperater") String insertOperater,
        @FormParam(value = "operaterCode") String operaterCode){…}
```

Ovim se postiže da se resursi za poziv ove metode nalaze na adresi:

http://localhost:8080/e_bill/resteasy/ebill/sbb/addNewEBill

pa će primer poziva ove metode izgledati ovako:

http://localhost:8080/e_bill/resteasy/ebill/sbb/addNewEBill?contractNo=123456789&email=bojana.zecevic@sbb.co.rs&company=SBB&insertInputSystem=PORTAL&insertOperater=bojana.zecevic

@POST anotacija

Java anotacija @POST ukazuje na to da će dati Java metod biti odgovor na HTTP POST zahtev.

Na primeru:

```
@POST
@Produces(MediaType.APPLICATION_JSON)
@Path(value = "/addNewEBill")
public Response addNewEBill(
        @FormParam(value = "contractNo") String contractNo,
        @FormParam(value = "email") String email,
        @FormParam(value = "company") String company,
        @FormParam(value = "insertInputSystem") String insertInputSystem,
        @FormParam(value = "insertOperater") String insertOperater,
        @FormParam(value = "operaterCode") String operaterCode){…}
```

to znači da ce se prilikom prijavljivanja novog korisnika na uslugu elektronskog računa, kreirati resurs (na lokaciji /addNewBill) za tog korisnika.

@GET anotacija

Ova anotacija ukazuje na to da će metod biti odgovor na HTTP GET zahtev.

Tako će se pri pozivu metoda:

```
@GET
@Produces(MediaType.APPLICATION_JSON + ";charset=" + "UTF-8")
@Path(value = "/verifyEBill")
public Response verifyEBill(@QueryParam(value = "mailId") String mailId){ . . . }
```

dohvatati resurs čiji je mailId prosleđen kao parametar.

@PUT anotacija

Prilikom poziva metoda koji je anotiran sa @PUT anotacijom, ažuriraće se trenutna vrednost resursa za parametar koji je prosleđen metodi.

Na primer, ukoliko imamo metodu sa sledećim kodom:

```
@PUT
@Path(value = "/updateEBillContract")
public Response updateEBillContract(@QueryParam(value = "contractNo") String contractNo){ . . . }
```

ažuriraće se podaci za broj ugovora koji joj se prosledi.

@DELETE anotacija

Metode koje su anotirane ovom anotacijom će vršiti brisanje resursa sa neke lokacije, u zavisnosti od parametra koji se prosledi metodi. Na primer, ukoliko imamo metodu:

```
@DELETE
@Path(value = "/deleteEBill")
public Response deleteEBill(@QueryParam(value = "contractNo") String contractNo){ . . . }
```

obrisaće se podaci o elektronskom računu za broj ugovora koji joj se prosledi.

@PRODUCES anotacija

Ova anotacija omogućuje radnom okviru da zna koji tip reprezentacije da vrati klijentu. @PRODUCES se koristi zajedno sa anotacijama @GET, @POST i @PUT.[11] Prilikom slanja HTTP zahteva, klijent šalje i @PRODUCES(tip_sadržaja), pa ukoliko se kao tip pošalje application/xml, onda će server vratiti odgovor tog tipa. Ukoliko je potrebno da odgovor bude u JSON formatu, onda je iznad poziva potrebno postaviti anotaciju @PRODUCES(“application/json”)

@CONSUMES anotacija

Anotacija @Consumes se koristi da odredi koji MIME tip medija reprezentacije resursa može da prihvati (konzumira) od strane klijenta. Ukoliko se primenjuje na nivou klase, svi odgovori metoda prihvataju određen MIME tip. Ukoliko se anotacija primenjuje na nivou metoda, ona prevazilazi anotaciju na nivou klase.[8]

@FormParam anotacija

Ova anotacija omogućava da se pročitaju vrednosti parova naziv/vrednost koje se koriste u okviru POST i PUT zahteva.

Na primeru:

```
@POST
@Produces(MediaType.APPLICATION_JSON)
@Path(value = "/addNewEBill")
public Response addNewEBill(
        @FormParam(value = "contractNo") String contractNo,
        @FormParam(value = "email") String email,
        @FormParam(value = "company") String company,
        @FormParam(value = "insertInputSystem") String insertInputSystem,
        @FormParam(value = "insertOperater") String insertOperater,
        @FormParam(value = "operaterCode") String operaterCode){…}
```

To znači da se prilikom slanja ovakvog zahteva, klijent se konektuje na naš URI kroz POST zahtev koji kao parametre sadrži skupove parova naziv/vrednost. Kada radni okvir obradi zahtev, imamo pristup svim vrednostima koje smo definisali u metodu addNewEBill.

4 IMPLEMENTACIJA APLIKACIJE ZA SLANJE ELEKTRONSKIH RAČUNA

4.1 Poslovni zahtevi

Potrebno je kreirati aplikaciju za slanje elektronskih računa korisnicima u PDF formatu. Takođe, potrebno je omogućiti korisnicima da se prijavljuju i odjavljuju sa servisa elektronskim putem ili prijavom u prodajno-korisničkim objektima (PKO). Svim prijavljenim korisnicima aplikacija treba da pošalje račune elektronskim putem u odgovarajućem trenutku u mesecu.

Aplikacija treba da smanji izuzetno velike finansijske troškove koji nastaju konvencionalnim slanjem računa korisnicima. Slanjem računa elektronskim putem, aplikacija treba da smanji vreme od izdavanja računa do prijema.

4.2 Funkcionalni zahtevi

1. ***Prijavljivanje na uslugu E-BILL***. Aplikacija dobija informacije o prijavama korisnika pomoću veb portala https://mojsbb.rs ili https://mojtotaltv.tv, ili na osnovu unosa u interni portal koji je napravio operater u PKO pomoću CRM aplikacije. Aplikacija treba da upamti sve prijavljene korisnike na jedinstvenom spisku. Treba osigurati konzistentnost sistema (nije moguće prijaviti se na uslugu dok traje fakturisanje i/ili slanje računa).

2. ***Odjavljivanje korisnika.*** Treba omogućiti korisnicama da se odjave sa usluge elektronskog slanja računa tako da to ne narušava konzistentnost sistema (nije moguće odjaviti se dok traje fakturisanje i/ili slanje računa), bilo elektronskim putem, bilo u PKO, kako bi aplikacija mogla da ukloni odjavljene korisnike sa spiska.

3. ***Obrada prijava.*** Spisak prijavljenih korisnika, aplikacija treba da prosledi SAP sistemu, koji će kreirati račune i vratiti ih aplikaciji u odgovarajućem formatu. Takođe, pri odjavi korisnika sa usluge, aplikacija treba da obavesti SAP sistem kako bi se prekinulo generisanje računa.

4. ***Slanje i arhiviranje računa.*** Po dobijanju generisanih računa, aplikacija treba da ih pošalje svim prijavljenim korisnicima u odgovarajućem trenutku u mesecu i da ih arhivira na odgovarajući način.

5. ***Interfejs ka operateru.*** Operateru treba omogućiti pregled, pretragu i prebacivanje računa, zatim slanje pojedinačnih i svih računa, kao i pregled statistike o broju uspešno i neuspešno poslatih računa, kao i vođenje statistike o samim operaterima i teritorijalnim direkcijama. Takođe, svaki operater treba da bude zaseban korisnik na sistemu.

4.2.1 Prijavljivanje korisnika za uslugu elektronskog računa putem portala

Jedan od načina za prijavljivanje na uslugu elektronskog računa je putem veb portala https://mojsbb.rs ili https://mojtotaltv.tv . Potrebno je da se korisnik na uloguje sa svojim korisničkim imenom i lozinkom i da pristupi kartici E-račun (Slika 1).

Slika 1. Prikaz korisnikovog portala i kartice koju treba da izabere

Zatim treba da klikne dugme kojim se započinje proces prijave na uslugu korišćenja (Slika 2).

Slika 2. Prikaz dugmeta koje korisnik treba da klikne prilikom prijave na uslugu

Nakon klika, korisniku će biti omogućeno da upiše e-mail adresu na koju želi da prima elektronske račune (Slika 3).

Slika 3. Prikaz polja za unos e-mail adrese

Klikom na dugme ispod polja (Slika 4),

Slika 4. Dugme za slanje zahteva za prijavu na uslugu elektronski račun

korisniku treba prikazati uslove korišćenja usluge, na koje on treba da pristane (Slika 5).

Slika 5. Prikaz uslova korišćenja

Po pristanku na uslove korišćenja korisniku se prikazuje poruka o uspešnom slanju zahteva za prijavu na uslugu, kao i informacija o verifkacionom e-mailu koji treba da bude poslat na prijavljenu adresu (Slika 6).

Slika 6. Prikaz informacije o uspešnoj prijavi

Po okončanju prijave, aplikaciji E-BILL ista treba da se pošalje uz pomoć veb servisa, pri čemu ta prijava treba da se zabeleži u bazi podataka, zahtev treba da se postavi na status WAITING(0) (Slika 7) i korisniku treba da se pošalje verifikacioni link sa rokom važenja na ostavljenu e-mail adresu (Slika 8).

id [PK] serial	contract_no character var	company_id integer	email character varying(320)	status_id integer	insert_ timest
2234	123456789	1	bojana.zecevic@sbb.co.rs	0	2014-0

Slika 7. Prikaz unosa u bazi prilikom registrovanja zahteva za prijavu na uslugu

Slika 8. Prikaz verifikacionog e-maila

Ako se verifikacija izvrši u predviđenom roku, korisniku treba poslati e-mail sa obaveštenjem da je uspešno aktivirao uslugu (Slika 9), zahtev treba da se prebaci u status ACTIVE (1) za korisnikov broj ugovora koji treba da se prosledi SAP sistemu, kako bi se ubuduće generisali elektronski računi za taj ugovor (Slika 10).

Slika 9. Prikaz e-maila koji potvrđuje uspešnu aktivaciju usluge

id [PK] serial	contract_no character var	company_id integer	email character varying(320)	status_id integer	insert_ti timestan
2234	123456789	1	bojana.zecevic@sbb.co.rs	1	2014-06

Slika 10. Prikaz unosa u bazi nakon uspešne aktivacije usluge

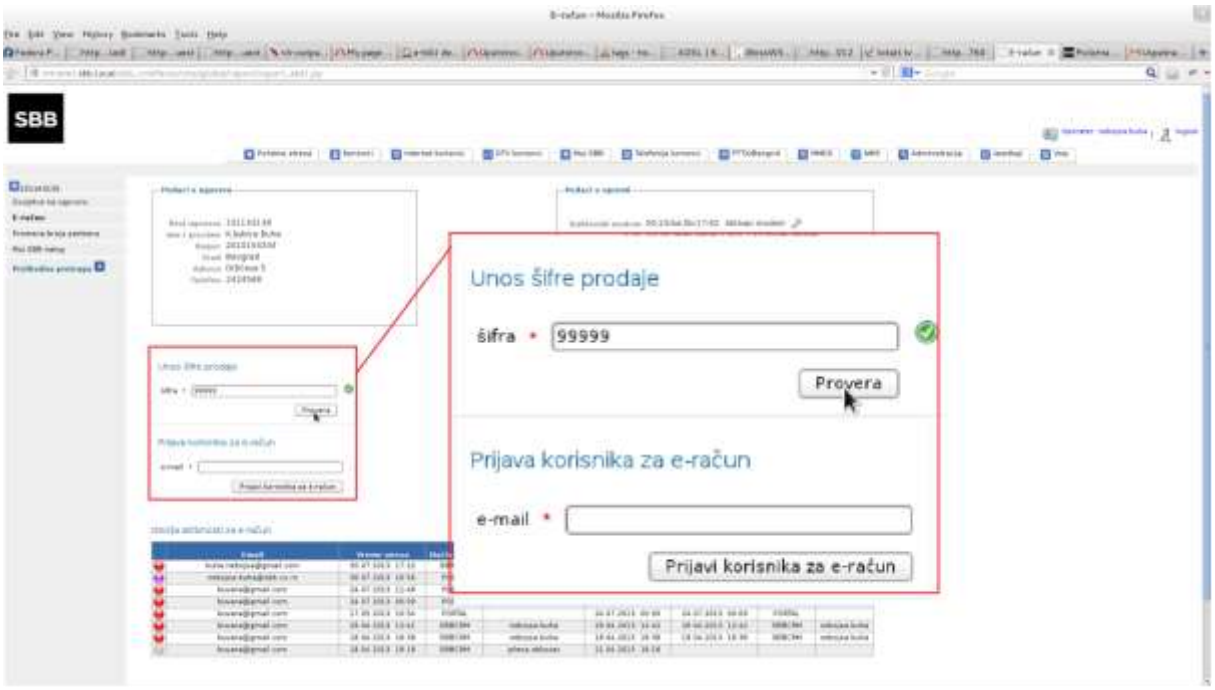
U suprotnom, u bazi podataka treba postaviti status zahteva na EXPIRED(3) (Slika 11).

id [PK] serial	contract_no character var	company_id integer	email character varying(320)	status_id integer	insert_time timestamp w
2234	123456789	1	bojana.zecevic@sbb.co.rs	3	2014-06-13

Slika 11. Prikaz unosa u bazi nakon isticanja vremena za aktivaciju usluge

4.2.2 Prijavljivanje korisnika preko CRM-a u PKO

Korisnik treba da popuni zahtev za prijavljivanje na uslugu elektronski račun u PKO koju operater unosi pomoću CRM aplikacije. Nakon uspešne autentifikacije, operater treba da unese prijavljenu e-mail adresu u za to predviđeno polje i klikom na dugme *Prijavi korisnika* prijavljivanje treba da se završi (Slika 12).



Slika 12. Prikaz prijave korisnika pomoću CRM aplikacije

Po okončanju unosa prijave, aplikaciji E-BILL ista treba da se pošalje uz pomoć veb servisa, pri čemu ta prijava treba da se zabeleži u bazi podataka, zahtev treba da se postavi na status WAITING (0) i korisniku treba da se pošalje verifikacioni link sa rokom važenja na ostavljenu e-mail adresu. Ako se verifikacija izvrši u predviđenom roku, korisniku treba poslati e-mail sa obaveštenjem da je uspešno aktivirao uslugu, zahtev treba da se prebaci u status ACTIVE (1) za korisnikov broj ugovora koji treba da se prosledi SAP sistemu kako bi se ubuduće generisali elektronski računi za taj ugovor. U suprotnom, u bazi podataka treba postaviti status zahteva na EXPIRED (3). (Slike 7– 11)

4.2.3 Odjavljivanje korisnika preko portala

Korisnik u okviru portala https://mojsbb.rs ili https://mojtotaltv.tv treba da izabere karticu e-račun i da klikom na odgovarajuće dugme pokrene proces za odjavljivanje sa usluge (Slika 13) .

Slika 13 . Prikaz odjavljivanja korisnika putem portala

Korisnik zatim treba da potvrdi otkazivanje usluge (Slika 14).

Slika 14. Prikaz forme za potvrdu otkazivanje usluge

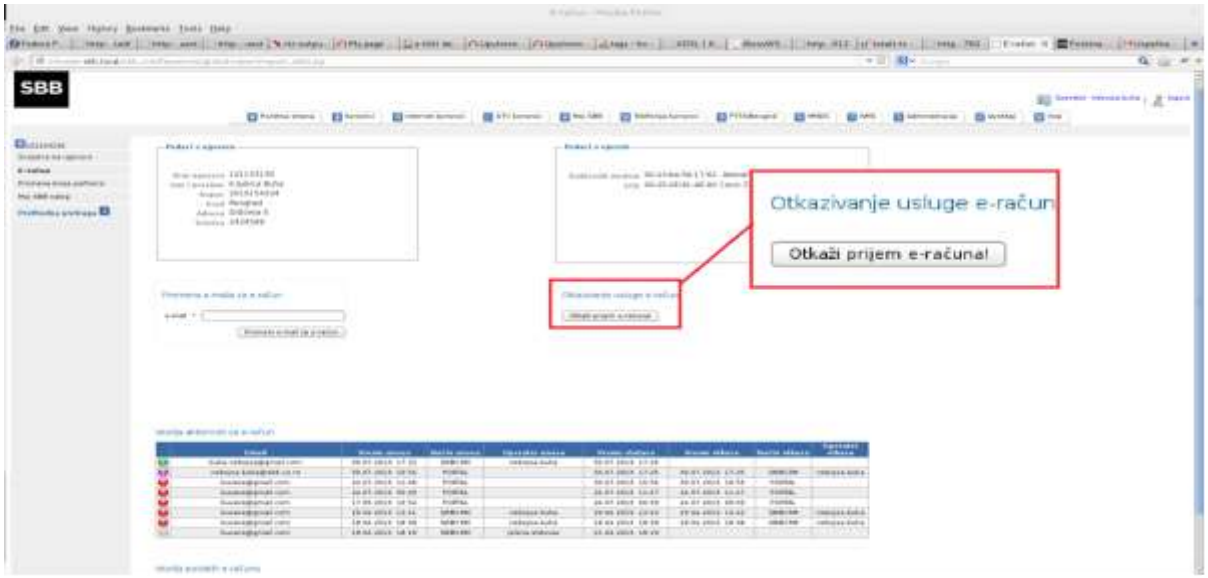
Odjavom sa usluge, aplikaciji E-BILL treba preko veb servisa da se pošalje informacija o odjavljivanju sa usluge, zahtev u bazi treba da se postavi na status CANCELLED (2) za korisnikov broj ugovora koji treba da se prosledi SAP sistemu kako bi se otkazalo generisanje računa za taj ugovor (Slika 15).

id [PK] serial	contract_no character var	company_id integer	email character varying(320)	status_id integer	insert_time timestamp w
2234	123456789	1	bojana.zecevic@sbb.co.rs	2	2014-06-13

Slika 15. Prikaz unosa u bazi nakon otkazivanja usluge

4.2.4 Odjavljivanje korisnika preko CRM-a u PKO

Korisnik treba da popuni odjavu u PKO koju operater unosi pomoću CRM aplikacije (Slika 16).



Slika 16. Prikaz otkazivanja usluge pomoću CRM aplikacije

Po okončanju unosa odjave, aplikaciji E-BILL treba preko veb servisa da se pošalje informacija o odjavljivanju sa usluge, zahtev u bazi treba da se postavi na status CANCELLED (2) za korisnikov broj ugovora koji treba da se prosledi SAP sistemu kako bi se otkazalo generisanje računa za taj ugovor (Slika 15).

4.2.5 Promena e-mail adrese korisnika preko portala

Korisnik u okviru portala https://mojsbb.rs ili https://mojtotaltv.tv treba da izabere karticu e-račun i da klikom na označeno dugme pokrene proces promene e-mail adrese na koju želi da prima elektronske račune (Slika 17).

Slika 17. Prikaz promene e-mail adrese putem portala

Korisnik zatim treba da unese novu e-mail adresu na koju želi da prima elektronske račune i klikne na dugme "Promeni e-mail" (Slika 18).

Slika 18. Prikaz polja za unos nove e-mail adrese

Po okončanju prijave, aplikaciji E-BILL ista treba da se pošalje uz pomoć veb servisa, pri čemu zahtev za promenom treba da se zabeleži u bazi podataka, usluga treba da se postavi na status CHANGE_MAIL (4) i korisniku treba da se pošalje verifikacioni link sa rokom važenja na ostavljenu e-mail adresu (Slika 19).

id [PK]	contract_no	company_id integer	email character varying(320)	status_id integer	insert_ti timestan
2234	123456789	1	bojana.zecevic@sbb.co.rs	4	2014-06-
2238	123456789	1	bojana_zecevic@yahoo.com	1	2014-06-

Undo change of data.

Slika 19. Prikaz unosa u bazi pri zahtevu za promenu e-mail adrese

Ako se verifikacija izvrši u predviđenom roku, korisnika treba obavestiti da je uspešno aktivirao uslugu, zahtev treba da se prebaci na status ACTIVE (1) za korisnikov broj ugovora i novopotvrđenu adresu, dok se stara prijava sa starom e-mail adresom prebacuje u status CANCELLED_BY_CHANGE_EMAIL (5) (Slika 20).

id [PK] serial	contract_no character var	company_id integer	email character varying(320)	status_id integer	insert_time timestamp w
2234	123456789	1	bojana.zecevic@sbb.co.rs	5	2014-06-13
2238	123456789	1	bojana_zecevic@yahoo.com	1	2014-06-14

Slika 20. Prikaz unosa u bazi nakon uspešne promene e-mail adrese

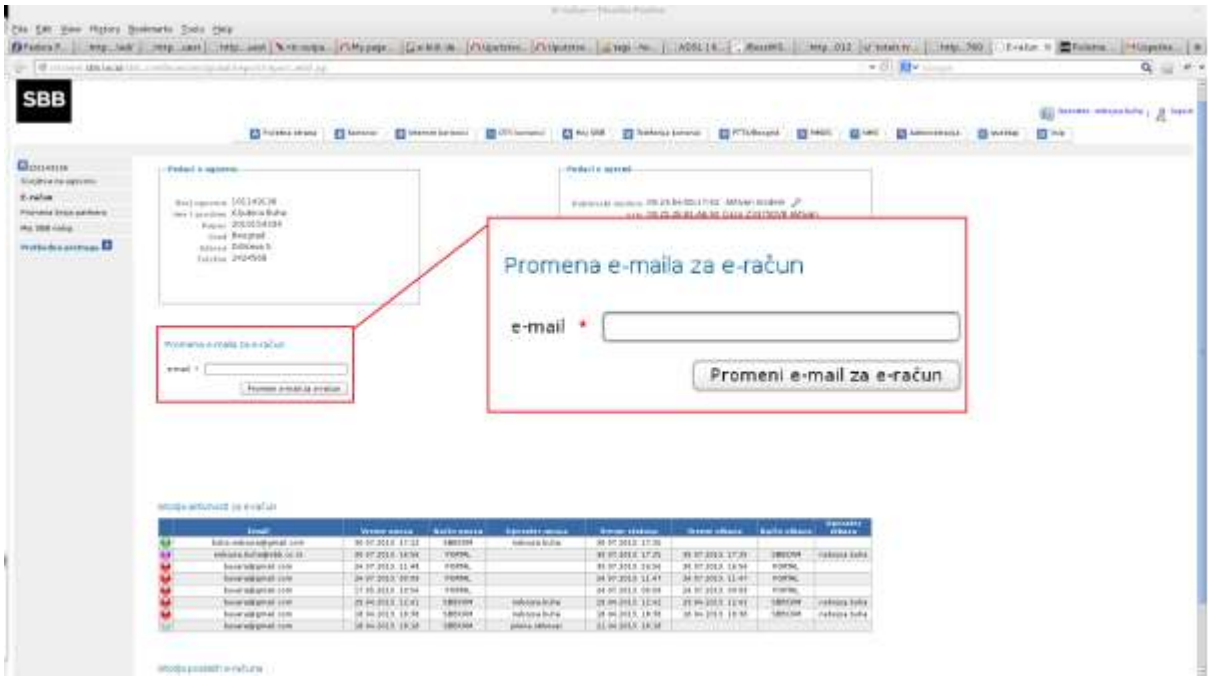
U suprotnom, u bazi podataka treba postaviti status usluge na EXPIRED_CHANGE_MAIL (6) na novoprijavljenoj adresi, dok usluga treba da ostane u statusu ACTIVE na staroj adresi (Slika 21).

id [PK] serial	contract_no character var	company_id integer	email character varying(320)	status_id integer	insert_time timestamp
2234	123456789	1	bojana.zecevic@sbb.co.rs	1	2014-06-13
2238	123456789	1	bojana_zecevic@yahoo.com	6	2014-06-14

Slika 21. Prikaz unosa u bazi prilikom isteka verifikacionog perioda za promene e-mail adrese

4.2.6 Promena e-mail adrese preko CRM-a u PKO

Korisnik podnosi zahtev za promenom e-mail adrese u PKO koju operater unosi pomoću CRM aplikacije (Slika 22).



Slika 22. Prikaz promene e-mail adrese putem CRM-a

Po okončanju prijave, aplikaciji E-BILL ista treba da se pošalje uz pomoć veb servisa, pri čemu zahtev za promenom treba da se zabeleži u bazi podataka, usluga treba da se postavi na status CHANGE_MAIL i korisniku treba da se pošalje verifikacioni link sa rokom važenja na ostavljenu e-mail adresu. Ako se verifikacija izvrši u predviđenom roku, korisniku treba poslati e-mail sa obaveštenjem da je uspešno aktivirao uslugu, usluga treba da se prebaci na status ACTIVE za korisnikov broj ugovora i novopotvrđenu adresu, dok se stara prijava sa starom e-mail adresom prebacuje u status CANCELLED_BY_CHANGE_EMAIL. U suprotnom, u bazi podataka treba postaviti status usluge na EXPIRED_CHANGE_MAIL na novoprijavljenoj adresi, dok usluga treba da ostane u statusu ACTIVE na staroj adresi. (Slike 19-21)

4.2.7 Slanje elektronskog računa za konkretan broj ugovora

Nakon generisanja računa u elektronskom obliku koje vrši Sektor za poslovno-informacioni sistem i njihovog skladištenja na unapred dogovorenoj lokaciji putem ftp protokola, aplikacija E-BILL te račune treba da prebaci na lokaciju sa koje se šalju korisnicima.

Nakon unosa broja ugovora od strane operatera i klika na dugme pošalji račun, aplikacija E-BILL treba da pronađe račun koji odgovara tom broju ugovora i da ga pošalje korisniku na e-mail adresu koju je ostavio prilikom prijave na uslugu.

Ukoliko je račun generisan u odgovarajućem formatu i u bazi podataka postoji zapis o prijavi korisnika sa tim brojem ugovora, račun treba da se pošalje na prijavljenu e-mail adresu.

Ukoliko u bazi podataka postoji aktivna prijava na uslugu slanja elektronskog računa, a ne postoji generisan elektronski račun za taj broj ugovora, u log tabeli treba da se zabeleži neuspešno slanje elektronskog računa pri čemu je opis greške "Nije pronađen pdf račun".

Ukoliko u bazi ne postoji aktivna prijava za uneti broj ugovora, ništa se neće desiti.

4.2.8 Masovno slanje elektronskih računa

Klikom operatera na odgovarajuće dugme, E-BILL aplikacija treba da pošalje elektronske račune svim prijavljenim korisnicima kojima je usluga elektronski račun u bazi podataka aktivna. Proces se smatra uspešnim ako dođe do kraja bez prekida. Po završetku procesa treba proveriti da li postoje računi koji nisu poslati, kao i da li postoje korisnici kojima računi nisu kreirani.

Ukoliko postoje neposlati računi, njihov spisak treba e-mailom poslati Sektoru za poslovno-informacioni sistem, na osnovu čijeg odgovora treba ponovo poslati račune.

Ukoliko postoje prijavljeni korisnici u bazi kojima je usluga elektronski račun aktivna, a nije im generisan račun, u log tabeli treba zabeležiti njihove brojeve ugovora sa porukom o grešci koja glasi "Nije pronađen pdf račun".

4.3 Arhitektura sistema

Na osnovu zahteva odlučeno je da aplikacija bude razvijena u Java programskom jeziku uz upotrebu Maven Project plugin-a, pri čemu će baza podataka biti Postgres, dok će interfejs ka operateru biti realizovan putem veb-a upotrebom JavaServer Faces tehnologije (jsf1.2) i

radnog okvira Trinidad. Pri radu sa bazom podataka biće korišćena Hibernate tehnologija. Na osnovu prijavljenih ugovora SAP sistem će generisati pdf račune i tako generisane ih vraćati nazad aplikaciji, koja će ih dalje slati korisnicima.

4.3.1 Maven

Maven predstavlja Java alat koji služi za izgradnju i upravljanje projektom. Jezgro za izgradnju projekta čini konfiguraciona datoteka POM (engl. Project Object Model), koja osim osnovnih informacija o samom projektu (naziv projekta, identifikator artefakta, verzija projekta,..) sadrži i zavisnosti (engl. dependency). Ukoliko se radi o projektu koji sadrži module, svaki od njih će sadržati svoj POM, a u glavnom POM-u će biti definisan svaki od tih modula. Više informacija o Maven-u se mogu naći u knjizi [5].

4.3.2 PostgreSQL

PostgreSQL (Postgres) je objektno-relacioni sistem za upravljanje bazama podataka, koji može biti slobodno korišćen, modifikovan i distribuiran od strane svakog korisnika za bilo kakavu upotrebu, bila ona privatna, komercijalna ili akademska. [17]

PostgreSQL je prvobitno bilo moguće pokrenuti na UNIX-olikim platformama, ali je kasnije omogućena portabilnost, pa se sada može pokrenuti i na operativnim sistemima Mac-u, Solaris-u i Windows-u. [10]

PostgreSQL sadrži moćan objektno-relacioni model podataka, bogat izbor vrsta podataka, laku nadogradivost, kao i nadograđeni set naredbi SQL jezika. [18] Više informacija o PostgreSQL-u se može naći u knjizi [6].

4.3.3 JavaServer Faces

JavaServer Faces (JSF) tehnologija predstavlja Java aplikativno okruženje i danas je deo Java EE (engl. Enterprise Edition) standarda. Arhitektura JSF-a integriše MVC (Model – View – Controller) dizajnerski obrazac, pa se i razvoj Veb aplikacija razdvaja na ova tri sloja. To može biti prilično značajno ukoliko tim programera radi na razvoju aplikacije.

4.3.4 MyFaces Trinidad

Apač MyFaces Trinidad je JSF radni okvir koji uključuje veliku biblioteku poslovnih komponenata. Trenutno podržava verzije JSF 1.1, JSF 1.2 i JSF 2.0. Više o MyFaces Trinidadu, kao i biblioteci etiketa koje on sadrži se može naći na veb adresi [14].

4.3.5 Moduli aplikacije

Aplikacija treba da se sastoji od tri glavna dela:

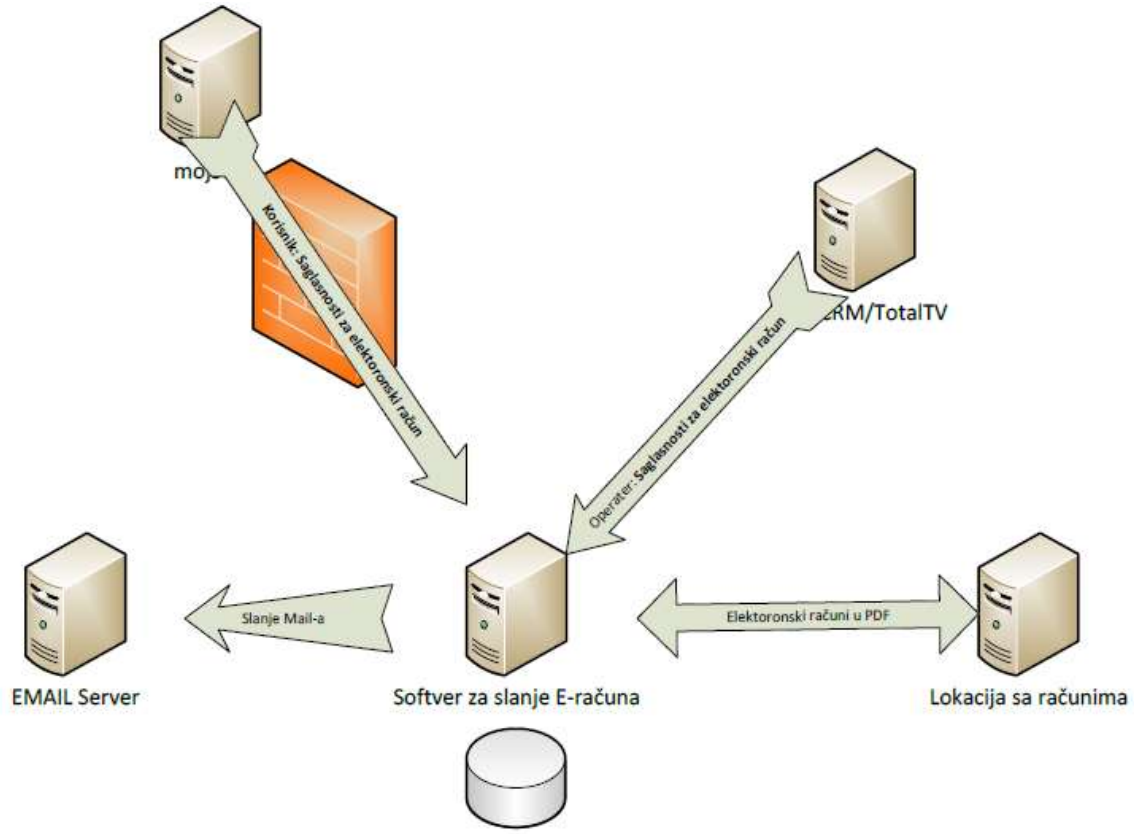
1. DataAccessObject (DAO) modul
2. WEB modul
3. Enterprise Archive (EAR) modul

DAO modul treba da bude pristupna tačka aplikacije koja će ostalim modulima pomoću svojih servisa omogućiti pristup i manipulisanje bazom prijavljenih korisnika. Takođe, DAO modul treba da vrši i komunikaciju sa SAP sistemom, kao i samo slanje elektronskih računa korisnicima na e-mail adresu koju su naveli u prijavi. Slanje e-email poruka korisnicima treba da bude definisano kao poseban interfejs, kao i komunikacija sa SAP-om.

WEB modul treba da realizuje interfejs ka operateru, kao i da omogući prijavljivanje i odjavljivanje korisnika sa usluge slanja elektronskih računa elektronskim putem pomoću portala https://mojsbb.rs ili https://mojtotaltv.tv ili preko CRM aplikacije. Takođe, WEB modul treba da vodi računa o verifikaciji e-mail adresa prijavljenih korisnika koja treba da bude moguća samo u unapred definisanom vremenskom okviru. Sva komunikacija WEB modula sa bazom podataka treba da bude realizovana preko servisa DAO modula.

EAR modul je neophodan deo kako bi ceo projekat mogao da se upakuje u jednu jedinu datoteku, kako bi čitava aplikacija mogla da se postavi na server, na njemu podigne i pokrene. EAR modul treba da ima informacije o svim drugim modulima i da ih integriše u jednu celinu. Takođe, ovaj modul treba da ima informacije o svim data source-ovima, kao i konekcijama na bazu (neophodno zbog hibernate tehnologije), zatim modul treba da bude

svestan o tome koji su procesi CronService-a iz DAO modula aktivni, a koji ne. Čitav EAR modul treba da se automatski generiše iz razvojnog okruženja po standardu kompanije.



Slika 23. Dijagram organizacije aplikacije

4.3.6 Model baze podataka

Model baze podataka koja se koristi u aplikaciji sadrži sledeće tabele:

- Contract
- Company
- Input_system
- Contract_bills
- Code_book

- Tabela ***Contract*** sadrži sve potrebne informacije o korisničkim ugovorima i ima strane ključeve ka tabelama Company, Input_system i Code_book. Atributi tabele su:

Naziv	Opis	Napomena
Id	primarni ključ	generiše sistem, not null
contract_no	broj ugovora	not null
company_id	id kompanije kojoj korisnik pripada	not null, strani ključ ka tabeli Company
email	e-mail adresa na koju korisnik želi da prima elektronske račune	not null
status_id	status usluge za dati broj ugovora	integer
insert_time	vreme unosa zahteva prvi put od strane korisnika ili operatera	not null
insert_operator	operater koji je uneo zahtev (username)	
insert_input_system_id	portal (1) ili CRM (2)	not null, strani ključ ka tabeli Input_system
end_time	vreme otkazivanja usluge	
end_operator	operater koji je otkazao uslugu (username)	
end_input_system_id	portal(1) ili CRM(2)	strani ključ ka tabeli Input_system
code_book	za koga je operater uneo (osoba koja je stvarno prihvatila zahtev korisnika)	strani ključ ka tabeli Code_book
sap_time	trenutak u kome je izvršen poziv ka sap-u (otkazivanje ili aktivacija)	

status_time	vreme promene statusa	

Tabela 1. Prikaz atributa tabele Contract

Status može biti:

Kod	Značenje	Opis
0	waiting	čeka se potvrda
1	active	usluga je aktivna na datom ugovoru za datu e-mail adresu
2	cancelled	usluga je otkazana
3	expired	korisnik nije potvrdio aktivaciju usluge u predviđenom roku
4	change_email	čeka se potvrda promene e-mail adrese
5	cancelled by change email	usluga je otkazana za dati ugovor na toj e-mail adresi usled promene adrese
6	expired change email	korisnik nije potvrdio promenu mejl adrese u predviđenom roku

Tabela 2. Prikaz mogućih statusa ugovora

Tabela **Contract_bills** sadrži informacije o računima koje aplikacija dobija od SAP sistema, i nema stranih ključeva ka drugim tabelama. Predstavlja log tabelu za slanje računa korisnicima, uspešnu ili neuspešnu. Atributi tabele su:

Naziv	Opis	Napomena
Id	primarni ključ	generiše sistem, not null
contract_no	broj ugovora	not null
Year	Godina	not null
Month	Mesec	not null
file_path	putanja na disku do datoteke	not null
insert_time	vreme kada se ugovor upiše u tabelu iz koje se kasnije šalju ugovori	not null
send_time	vreme kada je izvršeno slanje računa za taj ugovor	
send_result	uspešan (ok), neuspešan (opis greske)	
Status	govori o statusu računa	not null

Tabela 3. Prikaz atributa tabele Contract_bills

Status može biti:

Kod	Značenje
0	samo unet, pripremljen za slanje
1	uspešno poslato
2	neuspešno poslato

Tabela 4. Prikaz mogućih statusa računa

Tabela **Code_book** sadrži informacije o operateru koji unosi prijave u sistem, i nema stranih ključeva ka drugim tabelama. Atributi tabele su:

Naziv	Opis	Napomena
id	primarni ključ	generiše sistem, not null
naziv_td	naziv teritorijalne direkcije kojoj pripada operater	
naziv_pj	naziv prodajne jedinice	not null
kanal_prodaje	kanal prodaje	
ime_prezime	ime i prezime operatera	
sifra	šifra operatera	not null
status_id	aktivan (1), neaktivan (2)	not null
partner	ako je partner nema ime, prezime itd, već ime kompanije koja uzima zahteve za e-račun	

Tabela 5. Prikaz atributa tabele Code_book

Tabela **Input_system** sadrži informacije o načinu unosa zahteva u sistem. Atributi tabele su:

Naziv	Opis	Napomena
id	primarni ključ	generiše sistem, not null
name	portal ili CRM	

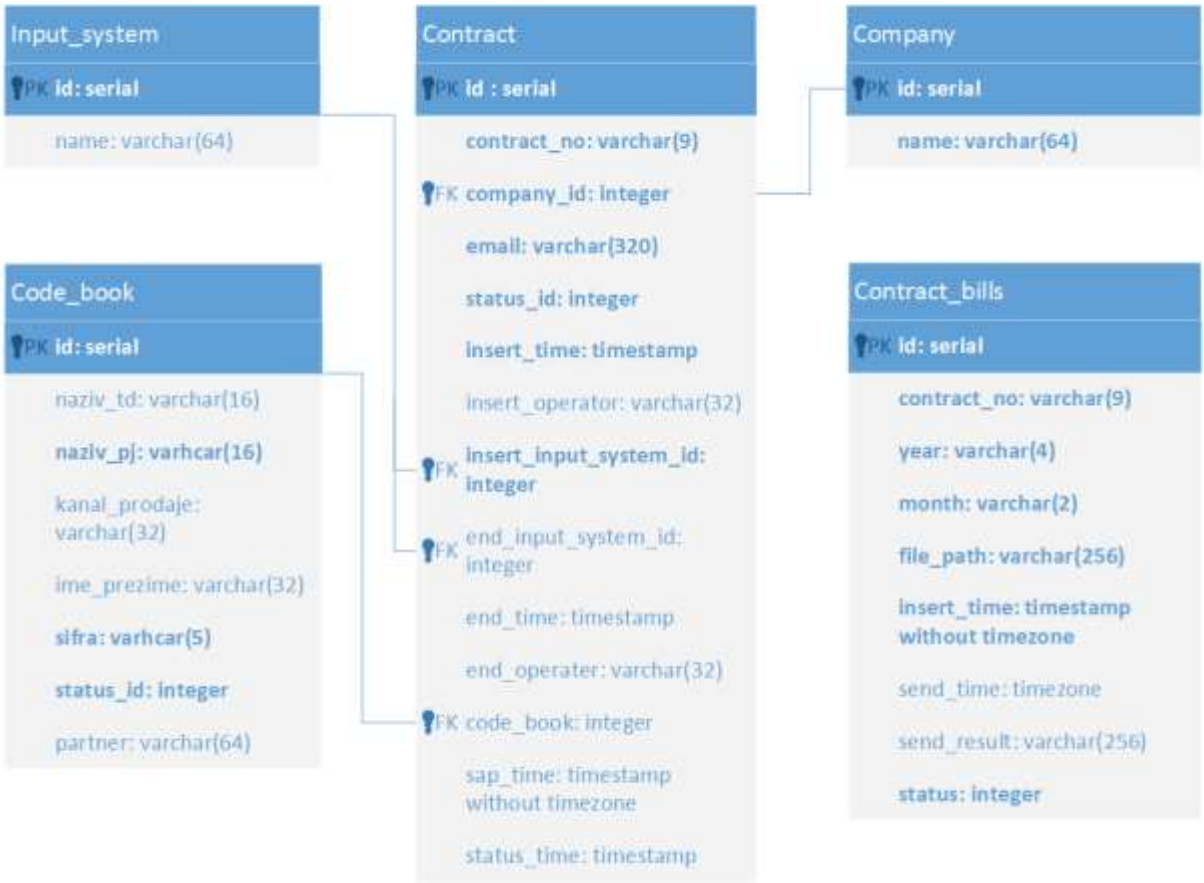
Tabela 6. Prikaz atributa tabele Input_system

Tabela **Company** sadrži informacije o kompaniji (SBB ili TotalTv). Atributi tabele su:

Naziv	Opis	Napomena
id	primarni ključ	integer, generiše sistem
name	sbb ili totalTV	not null

Tabela 7. Prikaz atributa tabele Company

Baza podataka se dijagramom može prikazati na sledeći način:



Slika 24. Model baze podataka

Ovakvom organizacijom tabela su ispunjeni prethodno zacrtani ciljevi izrade projekta. Nije bilo potrebe za korišćenjem uskladištenih procedura i pogleda.

5 ZAKLJUČAK

Sa razvojem informacionih tehnologija i uređaja, svakodnevno se razvijaju servisi, koji korisnicima nude mnogobrojne pogodnosti. Ukoliko je reč o poslovnim aplikacijama, zbog jednostavnog korišćenja i mogućnosti pristupa sa različitih uređaja (računara, tableta, mobilnih telefona,…) dobija se velika ušteda u vremenu i novcu, kako za samog korisnika, tako i za kompaniju koja tu uslugu pruža.

Jedan od primera servisa, za koji se poslednih godina sve više korisnika opredeljuje, je dobijanje elektronskog računa na željenu e-mail adresu. Osim što korisnici na ovaj način nisu vezani za svoje mesto boravka i dobijaju račun bilo gde da se nalaze, on im omogućava da na svome računaru sačuvaju i istoriju svojih računa. Sa druge strane, uvođenjem slanja elektronskog računa, uvećava se prihod kompanije, jer se smanjuju troškovi štampanja računa. Ono što je za obe strane bitno jeste da će ranije biti dostavljen račun, jer se postiže velika ušteda vremena ukoliko se računi ne štampaju u papirnoj formi. Zbog toga sve više kompanija radi na razvoju i pružanju ovakve vrste usluge svojim korisnicima.

U okviru ovog master rada data je analiza i arhitektura aplikacije, razvijene od autora ovog rada, za elektronsko slanje računa. Opisani su poslovni i funkcionalni zahtevi i arhitektura sistema, a prikazana je i struktura baze podataka. Detaljno je obrađena centralna tema rada, REST servisi, a osim toga, opisani su i softverski alati koji su korišćeni u implementaciji.

REST je od strane programera ocenjen kao veoma dobar model arhitekture. Rad nad HTTP protokolom, jednostavna implementacija različitih rešenja, platformska i programska nezavisnost, odličan rad nad velikim brojem klijenata, kao i jednostavna primena metoda GET, POST, PUT i DELETE, samo su neki od razloga zbog čega mnoge velike kompanije koriste ovaj model. Bitna karakteristike su nepostojanje stanja, keširanje i uniformna dostupnost, a za klijenta je značajno i to što odgovor može dobiti u čitljivoj formi (JSON, XML, HMTL....).

Dalje unapređenje aplikacije se može vršiti u više pravaca, među kojima su:

- Izrada stranice za logovanje u aplikaciju, čime bi se omogućilo da se određenim operaterima dodaju veća prava u odnosu na ostale operatere. Osim toga, mogle bi se beležiti u bazu podataka i akcije operatera.

- Parcijalno slanje e-mailova, čime bi se ubrzao proces slanja. Ne bi bilo potrebe da se čeka na generisanje svih računa.

- Automatsko slanje na e-mail izveštaja o broju prijavljeniih korisnika za uslugu primanja elektronskog računa.

6 LITERATURA

[1] Burke Bill, *RESTful Java with JAX-RS 2.0*, Second Edition, O'Reilly, 2013.

[2] Evjen Bill, Sharkey Kent, Thangarthinam Thiru, Kay Michael, Vernet Alessandro, Ferguson Sam, *Professional XML*, wrox, 2007.

[3] Fielding Roy, *Architectural Styles and the Design of Network-based Software Architectures*, PhD Thesis, University of California, Irvin, CA,USA, 2000.

[4] Horton Ivor, *Java 2 JDK 5 Od početka*, CET, 2006.

[5] Massol Vincent, Van Zyl Jason, Porter Brett, Casey John, Sanchez Carlos, *Better builds with Maven*, Library ress, April 2007.

[6] Obe Regina, Hsu Leo, *PostgreSQL: Up and Running*, O'Reilly, 2012.

[7] Oracle Corporation, Jersey – RESTFul Web Services in Java, https://jersey.java.net/ , 14.avgust 2015.

[8] Oracle Corporation and/or its affiliates, RESTful Web Services Developer's Guide, http://docs.oracle.com/cd/E19776-01/820-4867/ggqqr/

[9] Oracle Corporation and/or its affiliates, The Java EE 6 Tutorial, http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

[10] PostgreSQL Tutorial Website, PostgreSQLTutorial, http://www.postgresqltutorial.com/what-is-postgresql/

[11] Sandoval Jose, *RESTful Java WebServices*, Packt Pubishing, 2009.

[12] Sriparasa Sai Srinivas, *JavaScript and JSON Essentials*, Packt Publishing, oktobar 2013.

[13] Vogella GmbH, REST with Java (JAX-RS) using Jersey –Tutorial, http://www.vogella.com/tutorials/REST/article.html#rest_httpmethods, 20. avgust 2014.

[14] The Apache Software Foundation, Apache MyFaces Trinidad, https://myfaces.apache.org/trinidad/, 27. februar 2012.

[15] Wikipedia, Elektronska faktura, http://sr.wikipedia.org/wiki/Elektronska_faktura, 07. jun 2015.

[16] Wikipedia, Faktura (račun), http://sh.wikipedia.org/wiki/Faktura_(ra%C4%8Dun), 28.avgust 2014.

[17] Wikipedia, PostgreSQL, http://sh.wikipedia.org/wiki/PostgreSQL, 09. mart 2013.

[18] Wikipedia, Representational state transfer, http://en.wikipedia.org/wiki/Representational_state_transfer, 05. septembar 2015.

[19] W3, Web Services Architecture, http://www.w3.org/TR/ws-arch/#whatis, 11. februar 2004.

DODATAK A

Prilikom aktivacije elektronskog računa, prvi korak je validacija unetih parametara. Proverava se da li su svi potrebni parametri prosleđeni prilikom pozivanja REST servisa. Ukoliko nisu, vraća se odgovarajući objekat, koji u sebi sadrži id, kod i poruku. Ukoliko su prosleđeni svi parametri, vrši se unos novog reda u tabelu contract. Tom prilikom se proverava da li već postoji zahtev u bazi i ukoliko postoji, vraća se odgovarajući odgovor. Zatim se kreira i šalje mail za započinjanje procesa aktivacije.

```
@POST
@Produces(MediaType.APPLICATION_JSON)
@Path(value = "/addNewEBill")
public Response addNewEBill(@FormParam(value = "contractNo") String contractNo,
@FormParam(value = "email") String email, @FormParam(value = "company") String
company,@FormParam(value = "insertInputSystem") String insertInputSystem,
@FormParam(value = "insertOperater") String insertOperater, @FormParam(value =
"insertOperater") String insertOperater)
{

ActionResponse response = null;
Response webResponse = null;

boolean validateParam = contractNo != null && email != null && company != null &&
insertInputSystem != null;

if (!validateParam)
{
	return new Response(-1, "ERROR_NO_PARAM", "Nisu prosledjeni svi potrebni
	parametri!");
}

try
{
	response = adminService.addNewEBill(contractNo, company, email,
	insertInputSystem, insertOperater);

	if (response.getCode().equals("OK"))
	{
		String contractId =
		response.getReturnValues().get(EActionResponseReturnValues.CONTRACT_I
		D.getId());

		String encodedId = aes.encrypt(contractId);
		encodedId = URLEncoder.encode(encodedId, "ISO-8859-1");

		String host = "";

		if (company.equals("TOTALTV"))
		host = "http://www.mojtotaltv.tv/ttv_home";
```

```
            else if (company.equals("SBB"))
            host = "https://www.mojsbb.rs/sbb_home";

            else if (company.equals("KDS"))
            host = "http://www.mojkds.rs/kds_home";

            VerifyConfirmationEmailDTO verifyConfirmationEmailDTO = new
            VerifyConfirmationEmailDTO(host + "/resteasy/ebill/" +
            company.toLowerCase() + "/verifyEBill?mailId=" + encodedId,
            EFirm.valueOf(company),

            EInputSystem.valueOf(insertInputSystem));

            String senderAdr;

            if (company.equals("SBB"))
                  senderAdr = EEmail.SBB.getEmail();

            else if (company.equals("TOTALTV"))
                  senderAdr = EEmail.TOTALTV.getEmail();

            else senderAdr = "office@kablovska.co.rs";

      mailManager.sendActivateEmail(email, senderAdr, "Potvrda registracije",
      verifyConfirmationEmailDTO, EFirm.valueOf(company));

      }

                  webResponse = new Response();

                  webResponse.setId(response.getId());
                  webResponse.setCode(response.getCode());
                  webResponse.setMessage(response.getMessage());
                  webResponse.setReturnValues(response.getReturnValues());

                  return webResponse;
            } catch (IOException ioe)
            {
                  adminService.removeContractRequest(contractNo);

                  StringWriter sw = new StringWriter();
                  PrintWriter pw = new PrintWriter(sw);
                  ioe.printStackTrace(pw);

                  String mailBody = "Exception <br/>" + sw.toString();
                  mailManager.sendEmailExceptionReport("webdev@sbb.co.rs",
"webdev@sbb.co.rs", "e_bill : FileNotFoundException", mailBody);

                  ioe.printStackTrace();
            } catch (Exception e)
            {
                  e.printStackTrace();
            }
            return webResponse;
      }
```

DODATAK B

Skripta za kreiranje jedne od tabela koja se koristi u aplikaciji za slanje elektronskih računa:

```
CREATE TABLE contract
(
  id serial NOT NULL,
  contract_no character varying(9) NOT NULL,
  company_id integer NOT NULL,
  email character varying(320) NOT NULL,
  status_id integer NOT NULL,
  insert_time timestamp without time zone NOT NULL,
  insert_operater character varying(32),
  insert_input_system_id integer NOT NULL,
  end_time timestamp without time zone,
  end_operater character varying(32),
  end_input_system_id integer,
  code_book_id integer,
  sap_time timestamp without time zone,
  status_time timestamp without time zone,
  CONSTRAINT pk_contract PRIMARY KEY (id),
  CONSTRAINT fk_contract_code_book FOREIGN KEY (code_book_id)
     REFERENCES code_book (id) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_contract_company FOREIGN KEY (company_id)
     REFERENCES company (id) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_contract_end_input_system FOREIGN KEY (end_input_system_id)
     REFERENCES input_system (id) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_contract_inser_input_system FOREIGN KEY (insert_input_system_id)
     REFERENCES input_system (id) MATCH SIMPLE
     ON UPDATE NO ACTION ON DELETE NO ACTION
```

```
)
WITH (
  OIDS=FALSE
);
```