UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

M A S T E R R A D

Implementacija SMS usluge mobilne telefonije „Kredit alarm“ korišćenjem veb servisa u Microsoft .NET tehnologiji

Višnja Tarbuk

Beograd, decembar 2015

UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET
Katedra za računarstvo i informatiku

M A S T E R R A D

Implementacija SMS usluge mobilne telefonije „Kredit alarm“ korišćenjem veb servisa u Microsoft .NET tehnologiji

Student: Višnja Tarbuk, 1159/2011

Mentor: Prof. dr Dušan Tošić
Komisija: Prof. dr Nenad Mitić**,** Prof. dr Miroslav Marić

Tema je odobrena od strane Katedre za računarstvo i informatiku na Nastavno-naučnom veću 25.09.2015. godine. Datum odbrane: ____.12.2015. godine.

Zahvaljujem se profesorima Matematičkog fakulteta u Beogradu – profesoru Dušanu Tošiću, mentoru, na izdvojenom vremenu i nesebičnoj podršci, profesoru Nenadu Mitiću na veoma korisnim savetima i ukazanim smernicama, profesoru Miroslavu Mariću na sugestijama i profesoru Saši Malkovu, Prodekanu za nastavu, na organizacionoj podršci tokom izrade ovog rada.

Takođe se zahvaljujem porodici i prijateljima na podršci i savetima tehničke i jezičke prirode.

Višnja Tarbuk, dipl. mat.

Apstrakt

U ovom radu je predstavljen način na koji je implementirana i kako se u praksi koristi SMS usluga mobilne telefonije „Kredit alarm“. Opisan je kompletan razvojni ciklus projekta – najpre je predstavljena arhitektura sistema, a zatim i kompletan proces implementacije usluge. Prikazana je struktura baze podataka, način korišćenja uskladištenih procedura u programskoj logici, način na koji se pomoću veb servisa vrši transfer SMS poruka, proces izrade izveštaja, a dat je i spisak testnih slučajeva koji su korišćeni u završnom testiranju. Na kraju rada je prikazano kako se usluga koristi na mobilnom telefonu i date su statistike vezane za broj korisnika usluge.

Abstract

This paper presents the implementation and practical use of SMS mobile telephony service "Credit alarm". It describes the entire project development life cycle – starting with the system architecture, and then the complete process of service implementation. It presents the database structure, the way stored procedures are used in programming logic, the way SMS messages are transferred using web services, the implementation of reporting process, as well as the list of test cases used in acceptance test. At the end of the paper, how the service is used in praxis on a mobile device is demonstrated, and statistics related to service usage are presented.

Sadržaj

1 Predgovor

SMS usluga je danas zastupljena na svim modelima mobilnih telefona i stoga operateri posebnu pažnju poklanjaju razvoju aplikacija vezanih za ovaj kanal komunikacije sa korisnikom. Za razvoj aplikacija pogodno je koristiti savremena okruženja poput Microsoft .NET i Java EE razvojnog okruženja. U ovom radu će biti predstavljeno kako je .NET tehnologija korišćena za implementaciju SMS usluge „Kredit alarm“.

Cilj rada je da predstavi način na koji se implementiraju i kako funkcionišu mobilni servisi na kratkim brojevima. Servis „Kredit alarm“ izabran je zbog svoje jednostavne logike. Ovim je omogućeno da fokus rada budu veze unutar sistema, komunikacija sa korisnikom i protok SMS poruka, a funkcionalnost same usluge stavljena je u drugi plan.

U radu će biti predstavljen kompletan razvojni ciklus projekta i u skladu sa tim, rad je podeljen na poglavlja čiji redosled oslikava proces nastajanja aplikacije.

Na početku rada, u poglavlju *Uvod*, dat je pregled osnovnih pojmova vezanih za mobilnu telefoniju i njene korisnike, a zatim je opisana funkcionalnost same usluge „Kredit alarm“.

U poglavlju *Osnovni pojmovi* pojašnjeni su pojmovi iz teorije i dat je opis tehnologija korišćenih u implementaciji.

U poglavlju *Implementacija* prikazani su detalji implementacije programskih delova aplikacije. Poglavlje počinje opisom načina kretanja korisničkog zahteva kroz sistem i veza između delova sistema. Zatim je dat je prikaz arhitekture sistema na kome je usluga „Kredit alarm“ implementirana, a nakon toga opis programskih delova aplikacije. Prikazana je struktura baze podataka, a sledi opis glavne aplikacije do koje korisnički zahtevi stižu putem 3 kanala – SMS kanalom, USSD menijem i putem mobilne aplikacije. U nastavku je detaljnije opisana funkcionalnost SMS kanala, a poglavlje završava prikazom procesa izveštavanja i testiranja.

U poglavlju *Primeri korišćenja usluge* prikazano je nekoliko praktičnih primera korišćenja usluge, a u poglavlju *Statistike* dati su dijagrami vezani za statistike korišćenja servisa.

U *Zaključku* su navedeni postignuti ciljevi rada i predstavljeni planovi za buduća unapređenja aplikacije.

Na kraju rada dat je spisak slika, zatim slede spisak skraćenica i korišćene literature.

Svi programski delovi aplikacije predstavljeni u ovom radu razvijeni su od strane autora – dizajn baze podatka, implementacija objekata baze podataka, kao i projektovanje i razvoj potrebnih veb servisa i formi. U trenutku pisanja rada (novembar 2015) aplikacija se aktivno koristi kod lokalnog operatera mobilne telefonije.

2 Uvod

2.1 Vrste korisnika mobilne telefonije

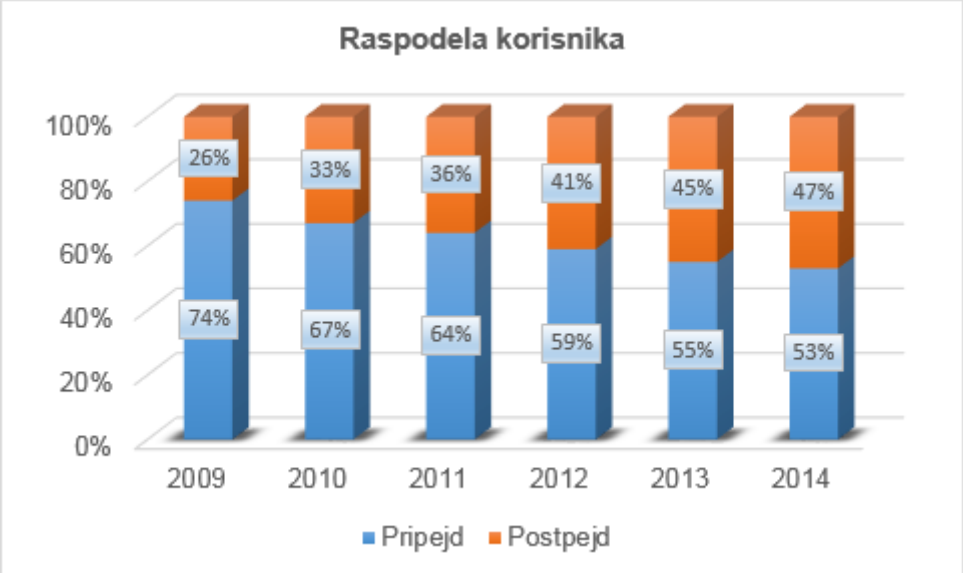
U zavisnosti od načina na koji plaćaju račune za svoje mobilne usluge, korisnici mobilne telefonije mogu se podeliti u dve kategorije – pripejd i postpejd.

Pripejd korisnici sami uplaćuju novac na svoj mobilni račun i ograničeni su uplaćenom sumom. Kada korisnik potroši kredit sa svog pripejd računa, potrebno je da račun ponovo dopuni, kako bi bio u mogućnosti da nastavi sa korišćenjem mobilnih usluga.

Postpejd korisnici plaćaju račun za mobilne usluge jednom mesečno po unapred dogovorenoj tarifi, odnosno za unapred dogovoreni paket usluga. Nemaju potrebu da prate potrošnju sredstava na svom računu jer će im račun biti zadužen na kraju meseca, a korišćene usluge automatski obračunate.

Kada je brojnost korisnika u pitanju, prema podacima RATEL-a sa kraja 2014. godine (pogledati [6]), ukupan broj korisnika mobilne telefonije na području Srbije je 9,34 miliona, od čega 53% čine pripejd, a 47% postpejd korisnici. Ranijih godina procenat pripejd korisnika je bio znatno veći, ali se tokom poslednjih 5 godina beleži konstantan rast broja postpejd korisnika (*Slika 1*).

Postpejd korisnici se prema operateru obavezuju ugovorom, stoga je operaterima u interesu da povećaju broj ovakvih korisnika i time sebi obezbede siguran prihod za duži vremenski period. Imajući ovo u vidu, operateri nude povoljnije cene mobilnih usluga za postpejd, nego za pripejd tarife. Pokazuje se da je upravo cena jedan od faktora koji utiče na porast broja postpejd korisnika. Drugi faktor je taj što postpejd tarife pružaju konstantnu dostupnost usluga mobilnog saobraćaja. Podsetimo, kod pripejd tarifa, mogućnost korišćenja usluga prestaje sa gubitkom kredita na pripejd računu korisnika.



Slika 1. Raspodela pripejd / postpejd korisnika

2.2 Usluga mobilne telefonije „Kredit alarm“

Praćenje stanja kredita pripejd korisnicima često može biti opterećujuće, a u isto vreme je korisniku veoma važno da u svakom trenutku zna sa koliko kredita raspolaže.

Usluga „Kredit alarm“ (pogledati [7], [8]) namenjena je pripejd korisnicima mobilne telefonije i omogućava im da bezbrižno koriste sredstva sa svog računa ne razmišljajući o stanju preostalog kredita – o tome brine „Kredit alarm“.

„Kredit alarm“ omogućava da u trenutku kada se iznos kredita smanji ispod određenog limita, korisnik bude obavešten SMS porukom. Na taj način se korisnik pravovremeno upozorava da se iznos njegovog kredita smanjio, kako bi znao da je vreme da dopuni svoj račun, a sve to bez potrebe da ulazi u aplikacije za proveru stanja kredita i time troši dodatno vreme.

Iznos limita utvrđen je na osnovu tržišnih cena razgovora i SMS poruka, a definisana suma je 50,00 dinara.

3 Osnovni pojmovi

U ovom poglavlju biće predstavljene teorijske osnove i tehnologije korišćene za razvoj aplikacije.

Kod je napisan u programskom jeziku *C#* korišćenjem *Microsoft Visual Studio 2010* razvojnog okruženja sa verzijom *.NET 3.5*. U implementaciji su korišćeni veb servisi i Windows forme.

Glavni tok aplikacije, odnosno prijem i slanje SMS poruka urađen je pomoću veb servisa, dok su forme korišćene za automatizaciju procesa koji treba da se izvršavaju u određenim vremenskim razmacima, nezavisno od glavne aplikacije. Parsiranje dolaznih SMS poruka urađeno je pomoću regularnih izraza.

Regularni izrazi (eng. *regular expression*) koriste se za pretraživanje stringova i predstavljaju uzorke za pronalaženje kombinacija karaktera unutar datog stringa (pogledati [9]). Pri imenovanju uzorka, moguće je navesti konkretan karakter, klasu kojoj dati karakter pripada ili grupu od više uzastopnih karaktera.

Klase karaktera definisane su u okviru jezika regularnih izraza, a označavaju se specijalnim niskama karaktera, tj. specijalnim meta karakterima. Dva najčešća tipa klasa definišu se meta karakterima koji počinju sa „\" ili su okvireni uglastim zagradama „[" i „]". Klase koje počinju obrnutom kosom crtom predstavljaju skup karaktera određenog tipa. Primeri ovih klasa su: \w – osnovni karakteri (mala i velika slova engleskog alfabeta, cifre od 0 do 9 i donja crta), \d – cifre od 0 do 9, \s – blanko karakteri. Klase uokvirene uglastim zagradama predstavljaju opsege ili grupe karaktera. Primeri ovih klasa su: [a-z] – mala slova engleskog alfabeta, [A-Z] – velika slova engleskog alfabeta, [0-9] – jednocifreni brojevi, [niz_karaktera] – bilo koji karakter iz datog niza, [^niz_karaktera] – bilo koji karakter koji se ne nalazi u datom nizu. Regularni izrazi su zasnovani na karakterima, odnosno svaki od prethodno navedenih regularnih izraza opisuje tačno jedan karakter uzorka koji se traži.

U okviru jezika regularnih izraza definisan je još jedan skup meta karaktera, takozvani džoker karakteri. Najčešći džoker karakteri su „?", „*" i „+", a omogućavaju izbor nula ili više karaktera iz prethodno navedene klase ili jednostavno označavaju nula ili više pojava prethodno navedenog karaktera. Karakter „?" označava najviše jednu pojavu, karakter „*" označava nula ili više pojava, dok „+" označava najmanje jednu pojavu datog karaktera. Primeri korišćenja džoker karaktera su: \d+ – niz od jedne ili više cifara, [a-z]* – niz od nula ili više malih slova, c? – najviše jedna pojava slova *c*, colo?r – označava reči *color* i *colour*.

Džoker karakteri ne pružaju informacije o tačnom broju karaktera. Za ovu namenu u okviru regularnih izraza definisane su vitičaste zagrade „{" i „}". U okviru vitičastih zagrada potrebno je navesti željeni broj pojava karaktera, odnosno željenu dužinu uzorka koji se traži. Na primer: {n} – karakter je ponovljen tačno *n* puta, {n,} – karakter je ponovljen najmanje *n* puta, {m, n} – karakter se ponavlja najmanje *m*, a ne više od *n* puta.

Regularni izrazi nalaze primenu i u mobilnoj telefoniji, a izraz za proveru ispravnosti MSISDN broja glasi *06\d{7,8}*.

Veb servisi su aplikacije koje omogućavaju pozive udaljenih metoda preko HTTP protokola. Programsko povezivanje distribuiranih softverskih komponenti je na ovaj način moguće bez obzira na kojoj su platformi implementirane, na kom su programskom jeziku napisane ili na kojoj se platformi izvršavaju. Komunikacija sa veb servisom se obavlja pomoću SOAP protokola, standarda zasnovanog na XML-u. Definisan je kao jednostavni protokol za razmenu informacija između računara, bez obzira na njihov operativni sistem, programsko okruženje ili objektni model, a sadrži detalje o tome kako pozivi metoda mogu da se izvrše putem HTTP protokola na predvidiv način. Jezik za opis veb servisa naziva se WSDL. WSDL pomoću XML šema daje spisak metoda koje veb servis izlaže, parametre koje te metode zahtevaju, kao i odgovarajuće tipove. Definiše sve detalje koji su neophodni za komunikaciju sa servisom – format poruke, transportne protokole i lokaciju servisa.

Windows Forms predstavlja biblioteku klasa za rad sa grafičkim okruženjem u Microsoft .NET tehnologiji. Aplikacije ovog tipa nasleđuju klasu *Form*, a prvenstveno služe za pravljenje formi, odnosno prozora u Windows radnom okruženju. Međutim, ovde su forme iskorišćene za pravljenje izvršnih fajlova koji će se automatski pokretati u određenim vremenskim trenucima. Ovo je postignuto tako što se pri samom pokretanju aplikacije uradi zatvaranje prozora forme, čime je dobijena logika bez grafičkog okruženja.

U projektu je korišćen sistem za upravljanje bazom podataka *Microsoft SQL Server 2008 R2*, a suština programske logike napisana je pomoću uskladištenih procedura i okidača.

Uskladištena procedura (eng. *stored procedure*) predstavlja niz SQL naredbi povezanih u logičku celinu, a čuva se kao zaseban objekat u bazi podataka (pogledati [1], [2]). Procedure, osim SQL naredbi, mogu sadržati parametre, promenljive, kao i konstrukcije za kontrolu toka izvršavanja.

U ovom projektu, za svaku uskladištenu proceduru implementirana je po jedna statička metoda koja je poziva, a sve metode ove vrste smeštene su u jednu klasu. Na ovaj način nema potrebe za kreiranjem objekata i dodatnim utroškom memorije, već se metode pozivaju direktno preko imena klase.

Okidač (eng. *trigger*) je proceduralni kod u vidu niza SQL naredbi koji se automatski izvršava kao rezultat specifičnih događaja nad specifičnom tabelom ili pogledom u bazi podataka (pogledati [1], [2]). Izvršavanje okidača u SQL Server bazi se može postaviti pre ili posle operacija unosa, izmene ili brisanja podataka za tabele, dok su za poglede na raspolaganju trigeri koji se okidaju umesto izvršavanja operacija unosa, izmene ili brisanja podataka.

Pri projektovanju baze podataka vođeno je računa o strukturi tabela i njihovih veza upotrebom odgovarajućih indeksa i primarnih i stranih ključeva. Postoje i log tabele koje se pune tokom rada aplikacije paralelno sa upisom u osnovne tabele, a namenjene su procesu izveštavanja i praćenju grešaka.

Primarni ključ (eng. *primary key*, PK) predstavlja kolonu ili kombinaciju kolona koje jedinstveno određuju red u tabeli (pogledati [3]). Dva različita reda u tabeli ne mogu imati istu vrednost primarnog ključa. Primarni ključ ne može sadržati NULL vrednosti, a tabela može imati imajviše jedan primarni ključ.

Strani ključ (eng. *foreign key*, FK) predstavlja kolonu ili kombinaciju kolona jedne tabele koje jedinstveno određuju red u drugoj tabeli (pogledati [3]). Služi za uspostavljanje veza između podataka dve tabele i obezbeđuje referencijalni integritet u bazi podataka. Veza između dve tabele postiže se na način da strani ključ jedne tabele pokazuje na primarni ključ druge tabele.

Indeksi su strukture podataka koje omogućavaju efikasnu pretragu sadržaja tabela u bazi podataka (pogledati [1], [2]).

Izveštaji su implementirani pomoću uskladištenih procedura postavljenih u **automatske poslove** (eng. *jobs*). Za upravljanje poslovima korišćen je deo Microsoft SQL Servera koji se naziva *SQL Server Agent* (pogledati [10]). Ovaj program nudi veliki broj opcija i mogućnosti. Potrebno je izabrati jednu ili više željenih uskladištenih procedura, a mogu se izvršavati u jednom ili više koraka sa najrazličitijim vremenskim opcijama.

Proces razvoja konstantno je bio praćen procesom testiranja aplikacije.

Testiranje je proces koji za zadatak ima evaluaciju napisanog programskog koda, odnosno da utvrdi da li se za različite ulazne parametre dobijaju očekivane izlazne vrednosti (pogledati [11]).

U procesu testiranja od strane programera, korišćeni su jedinični testovi (eng. *unit test*) kako bi se proverila ispravnost funkcionisanja pojedinih blokova koda. Završno testiranje rađeno je pomoću testa prihvatanja (eng. *acceptance test*). Ova druga vrsta testiranja najčešće se obavlja od strane klijenata ili korisnika aplikacije u cilju provera da li aplikacija zadovoljava prvobitno postavljene korisničke zahteve (pogledati [11]).

4 Implementacija

U ovom poglavlju će biti opisan način na koji je implementirana usluga „Kredit alarm“. Biće dat prikaz arhitekture sistema, strukture baze podataka i programskih delova aplikacije. Aplikacija se sastoji iz nekoliko modula koji svaki za sebe čine odvojene programske celine.

4.1 Tok izvršavanja

Akcije koje su korisniku na raspolaganju su prijava i odjava korišćenja usluge. Na raspolaganju su mu tri kanala komunikacije sa aplikacijom – SMS, USSD meni i mobilna Android aplikacija. Korisnik šalje zahtev ka sistemu na jedan od tri načina – šalje SMS poruku sa unapred definisanim sadržajem, tj. sintaksom na za to predviđeni broj, bira odgovarajuću opciju iz USSD menija ili vrši izbor odgovarajuće usluge putem mobilne aplikacije.

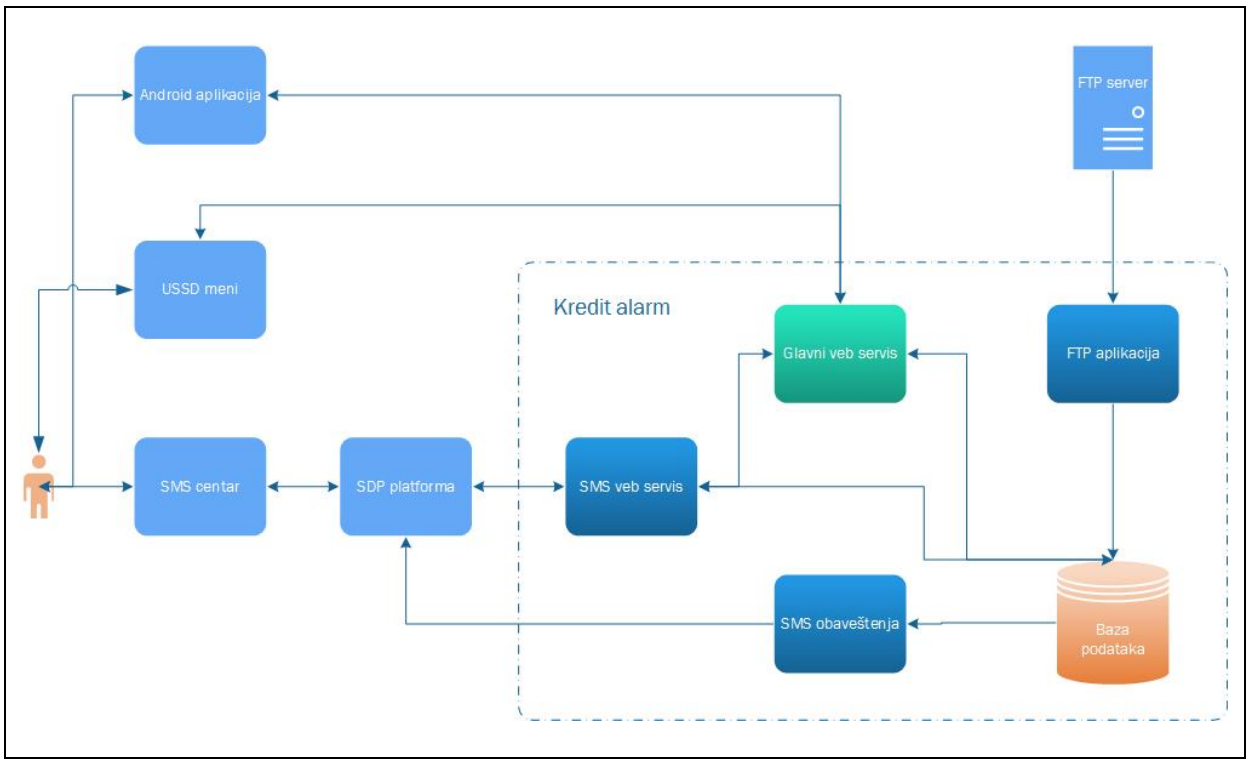
U slučaju prijave za korišćenje usluge, tj. aktivacije, sistem prihvata zahtev korisnika, obrađuje ga i ukoliko je zahtev uspešan, korisniku se aktivira usluga „Kredit alarm“. Korisnik od sistema, putem kanala kojim je inicirao komunikaciju, dobija povratnu poruku o uspešno ili neuspešno izvršenoj prijavi za korišćenje usluge. Dok god je korisniku usluga aktivna, dobijaće SMS obaveštenje kada mu se iznos kredita na pripejd računu smanji ispod dogovorenog limita od 50,00 dinara.

U slučaju odjave korišćenja usluge, tj. deaktivacije, sistem prihvata zahtev korisnika, obrađuje ga i ukoliko je zahtev uspešan, korisniku se deaktivira usluga „Kredit alarm“. Korisnik od sistema, putem kanala kojim je inicirao komunikaciju, dobija povratnu poruku o uspešno ili neuspešno izvršenoj odjavi usluge.

Do neuspešne prijave ili odjave dolazi u slučaju kada korisnik pošalje SMS poruku sa neispravnom sintaksom, ukoliko broj telefona korisnika nije pripejd, ukoliko je došlo do prekida u komunikaciji između različitih delova sistema ili u slučaju pojave bilo koje druge vrste sistemske greške. Takođe se vodi računa da korisnik, koji već ima aktivnu uslugu, ne može vršiti ponovnu aktivaciju ili da korisnik, kome je usluga deaktivirana, ne može vršiti ponovnu deaktivaciju.

4.2 Arhitektura sistema

Na *Slici 2* dat je prikaz arhitekture celokupnog sistema. Prikazane su komponente aplikacije „Kredit alarm“, kao i spoljašnje komponente koje komuniciraju sa aplikacijom.



Slika 2. Arhitektura sistema „Kredit alarm“

Komponente sistema su:

- **Android aplikacija** – jedan od kanala za iniciranje usluge
- **USSD meni** – jedan od kanala za iniciranje usluge
- **SMS centar** – platforma koja je ispostavljena prema korisniku, prva tačka komunikacije SMS kanala sa sistemom za obradu SMS poruka
- **SDP platforma** – platforma na kojoj se definišu kratki brojevi za SMS poruke i veb servisi koji vrše obradu pristiglih SMS poruka
- **SMS veb servis** – veb servis koji služi za obradu zahteva pristiglih putem SMS kanala
- **Glavni veb servis** – veb servis koji služi za obradu zahteva pristiglih sa sva tri kanala (SMS, USSD, android aplikacija)
- **FTP server** – server na koji se postavljaju fajlovi sa spiskom MSISDN-ova korisnika čiji iznos kredita se smanjio ispod 50,00 dinara
- **FTP aplikacija** – aplikacija koja vrši obradu fajlova postavljenih na FTP server
- **SMS obaveštenja** – aplikacija koja korisnicima šalje SMS obaveštenja o padu kredita
- **Baza podataka** – baza u kojoj se evidentiraju zahtevi pristigli na SMS i glavni veb servis, kao i podaci dobijeni sa FTP servera

Glavni veb servis, SMS veb servis, FTP aplikacija, SMS obaveštenja i baza podataka čine sistem „Kredit alarm“.

4.3 Baza podataka

Za sve delove sistema „Kredit alarm“ korišćena je jedinstvena baza podataka. Tabele koje se tiču samo SMS kanala imaju ekstenziju *Sms*.

Svaka tabela sadrži kolonu *DatumUnosa* tipa *datetime*, a kolona služi za evidenciju unosa redova u tabelu. Podrazumevana vrednost ove kolone je trenutni datum i vreme.

Osnovne tabele u bazi su *Zahtev* i *Korisnik*.

U tabelu *Zahtev* beleže se kompletni podaci o zahtevima koji stižu ka glavnom veb servisu – kanal sa kog stiže zahtev, vrsta akcije koju inicira korisnik, broj telefona sa koga stiže zahtev, podatak o tipu korisnika, indikator uspešnosti zahteva, kao i vreme pristizanja zahteva sa preciznošću do milisekundi. Struktura tabele data je na *Slici 3*.

Column Name	Data Type
ZahtevId	bigint
KanalId	int
TipKorisnikaId	int
TipZahtevaId	int
Msisdn	varchar(12)
Ispravan	bit
StatusId	int
DatumUnosa	datetime

Slika 3. Struktura tabele Zahtev

U tabeli *Korisnik* vrši se evidencija svih pripejd korisnika koji su uspešno koristili uslugu. Tabela pored osnovnih podataka o korisniku sadrži i indikator aktivnosti tipa *bool*, odnosno *bit*, kako je imenovan na SQL Serveru. Na ovaj način u svakom trenutku je moguć uvid u spisak korisnika koji trenutno koriste uslugu. Struktura tabele data je na *Slici 4*.

Column Name	Data Type
KorisnikId	bigint
Msisdn	varchar(12)
DatumPrijave	datetime
DatumOdjave	datetime
Aktivan	bit
DatumUnosa	datetime
DatumIzmene	datetime

Column Name	Data Type
KorisnikJRId	bigint
KorisnikId	bigint
Msisdn	varchar(12)
DatumPrijave	datetime
DatumOdjave	datetime
Aktivan	bit
DatumUnosa	datetime
DatumIzmene	datetime
Akcija	varchar(10)
DatumAkcije	datetime

Slika 4. Struktura tabela Korisnik i KorisnikJR

Tabela *Korisnik* ima i svoju tzv. žurnal ili arhivsku tabelu (*Slika 4*), tj. tabelu u kojoj se beleži istorija svih njenih promena. Naziv ove tabele je *KorisnikJR*, a popunjava se pomoću okidača nakon svake operacije ažuriranja nad tabelom *Korisnik*. Struktura žurnal tabele predstavlja strukturu izvorne tabele na koju su dodate kolone za skladištenje vrste akcije (u konkretnom slučaju „*update*“) i tačnog vremena kada se akcija izmene dogodila. Moguće je dodati i podatke o korisniku koji je izvršio izmenu, kao i sve druge podatke za koje arhitekte sistema procene da su od značaja. Sistem na ovaj način ima informaciju o svim promenama statusa pojedinačnog korisnika.

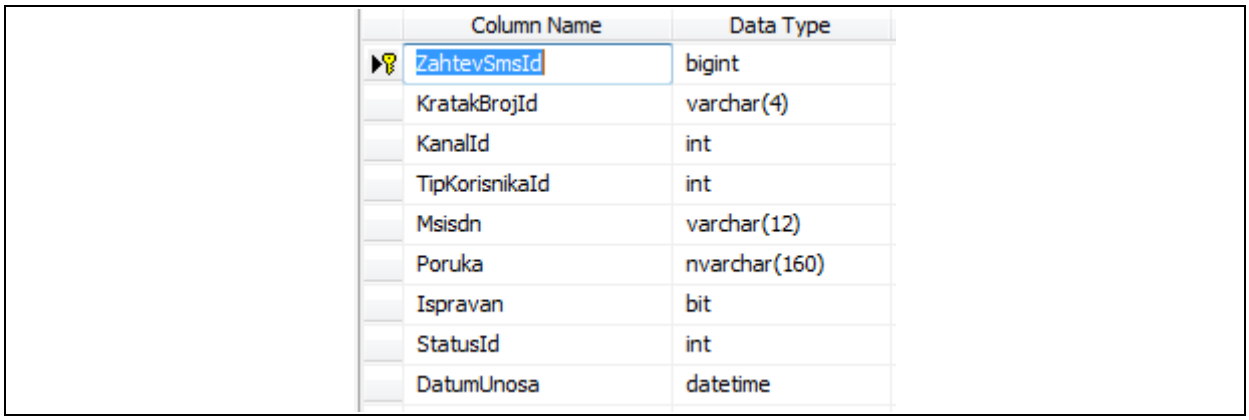
Naredni segment koda (*Slika 5*) predstavlja pomenuti triger nad tabelom *Korisnik*.

```
CREATE TRIGGER [dbo].[tru_Korisnik] ON [dbo].[Korisnik]
AFTER UPDATE

AS
BEGIN
      declare @datumSad datetime = getdate()

      update Korisnik
      set DatumIzmene = @datumSad
      where KorisnikId in (select KorisnikId from inserted)

      INSERT INTO [KreditAlarm].[dbo].[KorisnikJR]
            ([KorisnikId]
            ,[Msisdn]
            ,[DatumPrijave]
            ,[DatumOdjave]
            ,[Aktivan]
            ,[DatumUnosa]
            ,[DatumIzmene]
            ,[Akcija]
            ,[DatumAkcije])

     SELECT [KorisnikId]
            ,[Msisdn]
            ,[DatumPrijave]
            ,[DatumOdjave]
            ,[Aktivan]
            ,[DatumUnosa]
            ,[DatumIzmene]
            ,'UPDATE'
            ,@datumSad

     FROM deleted
END
```

Slika 5. Triger nad tabelom Korisnik

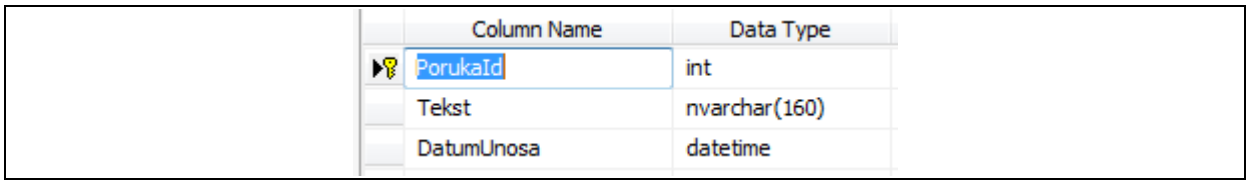
Deleted i *inserted* su ključne reči SQL Servera, a označavaju respektivno redove iz tabele *Korisnik* pre i posle izvršene akcije izmene.

U tabelu *ZahtevSms* beleže se zahtevi pristigli putem SMS kanala. Na SMS kanalu filtriraju se sintakse pristiglih SMS poruka, odnosno vrši se provera da li je poslata poruka u skladu sa predefinisanom formom. U pomenutu tabelu, pored ostalih podataka o zahtevu, beleži se indikator ispravnosti sintakse. Struktura tabele data je na *Slici 6*.



Slika 6. Struktura tabele ZahtevSms

SMS poruke, koje sistem šalje ka korisniku, evidentirane su u tabeli *Poruka* (*Slika 7*).



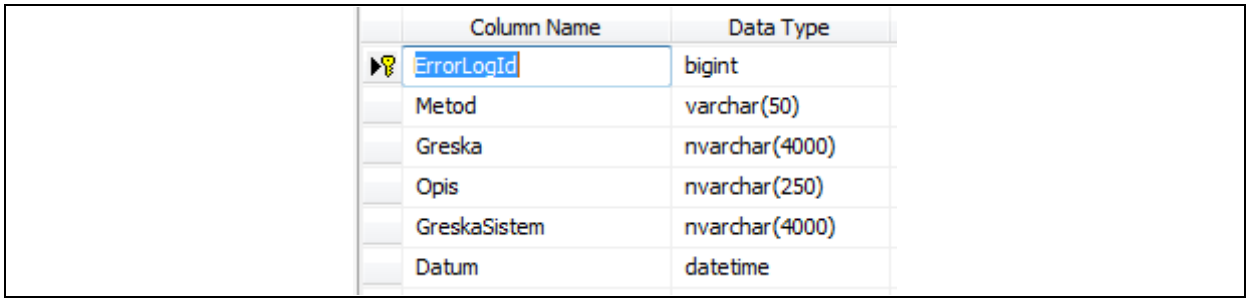
Slika 7. Struktura tabele Poruka

Svakoj tekstuelnoj poruci dodeljen je numerički identifikator (primarni ključ iz pomenute tabele), a na osnovu ovog identifikatora vrši se odabir poruke koju je u određenoj situaciji potrebno poslati korisniku. Naredni segment koda (*Slika 8*) prikazuje kako se, po uspešnoj aktivaciji usluge, korisniku šalje SMS poruka sa identifikatorom 4.

```
smsId = 4; // uspesna aktivacija
sendSmsText = dtPoruka.Select("PorukaId=" + smsId)[0]["Tekst"].ToString();
```

Slika 8. Izbor teksta poruke na osnovu identifikatora

Svaka neregularna situacija na sistemu beleži se u tabelu *ErrorLog*. Tabela sadrži podatak o vremenu nastanka greške, naziv metode u kojoj se neregularnost pojavila, originalnu poruku o grešci koju sistem prijavljuje, kao i prilagođen opis greške ukoliko je grešku bilo moguće unapred predvideti. Struktura tabele data je na *Slici 9*.



Slika 9. Struktura tabele ErrorLog

Spisak vrsta kanala, akcija, tipova korisnika dat je u pojedinačnim šifarnicima. Svaki šifarnik je strukture *Id*, *Naziv*, *Opis*, *DatumUnosa*.

Za svaku od tabela postoji uskladištena procedura koja vrši upis u tabelu, a nazivi ovih procedura su oblika *sp_<ime_tabele>Insert*.

Prefiksi u nazivima procedura označavaju vrstu akcije koja je u proceduri podržana, tako da procedure koje služe za DML operacije počinju sa *sp_*, *ac_* je prefiks za procedure u kojima se odigrava određena akcija ili logika (aktivacija, deaktivacija korisnika, kao i razne pretrage), dok su prefiksom *mail_* označene uskladištene procedure koje služe za izveštavanje putem mail-a.

Uskladištena procedura za upis novog reda u tabelu *Zahtev*, predstavljena je sledećim nizom naredbi (*Slika 10*):

```
CREATE PROCEDURE [dbo].[sp_ZahtevInsert]

      @kanalId int,
      @tipKorisnikaId int,
      @tipZahtevaId int,
      @msisdn varchar(12),
      @ispravan bit,
      @statusId int

AS
BEGIN
      declare @zahtevId bigint
      set @zahtevId = -1

      insert into [KreditAlarm].[dbo].[Zahtev]
            ([KanalId]
            ,[TipKorisnikaId]
            ,[TipZahtevaId]
            ,[Msisdn]
            ,[Ispravan]
            ,[StatusId])

      values
            (@kanalId,
             @tipKorisnikaId,
             @tipZahtevaId,
             @msisdn,
             @ispravan,
             @statusId)

      select @zahtevId = @@IDENTITY
      return @zahtevId
END
```

Slika 10. Procedura sp_ZahtevInsert

Kao što je ranije pomenuto, svakoj uskladištenoj proceduri u kodu odgovara po jedna statička metoda. Na primer, poziv procedure *sp_ZahtevInsert* dat je sledećim nizom naredbi (*Slika 11*):

```
// imenovanje konekcije ka bazi podataka
using (SqlConnection con = new SqlConnection(ConStr))
{
   try
   {
      // imenovanje procedure koja se poziva
      SqlCommand cmd = new SqlCommand("sp_ZahtevInsert", con);
      cmd.CommandType = CommandType.StoredProcedure;

      // parametri procedure
      if (!String.IsNullOrEmpty(msisdn))
         cmd.Parameters.Add(new SqlParameter("@msisdn", msisdn));
      else
         cmd.Parameters.Add(new SqlParameter("@msisdn", DBNull.Value));

       // ...

      cmd.Parameters.Add(new SqlParameter("@statusId", statusId));

      // povratna vrednost procedure
      SqlParameter pZahtevId = cmd.Parameters.Add("@zahtevId", SqlDbType.BigInt);
      pZahtevId.Direction = ParameterDirection.ReturnValue;

      // izvrsavanje procedure
      cmd.Connection.Open();
      cmd.ExecuteNonQuery();
      cmd.Connection.Close();

      // postavljanje povratne vrednosti metode u slucaju uspesnog izvrsenja
      zahtevId = Convert.ToInt64(pZahtevId.Value);
   }

   catch (Exception ex)
   {
      // postavljanje povratne vrednosti metode u slucaju greske
      zahtevId = -1;

      // logovanje zapisa o gresci
      ErrorLogInsertDAL("OperationDAL.ZahtevInsertDAL", "", ex.Message);
   }
}
```

Slika 11. Poziv procedure sp_ZahtevInsert

Gornja metoda za upis zahteva u kodu se poziva na sledeći način (*Slika 12*):

```
OperationBLL.ZahtevInsert(RequestActivate, msisdn, (rez == 0) ? true : false, rez);
```

Slika 12. Poziv metode ZahtevInsert

gde je *OperationBLL* klasa koja sadrži sve metode koje pozivaju uskladištene procedure.

4.4 Glavna aplikacija

Okosnicu celog sistema čini veb servis *KreditAlarmWS*, koji se u ovom radu naziva i glavna aplikacija ili glavni veb servis. Ovaj veb servis prihvata zahteve sa svih kanala i vrši njihovu dalju obradu, a u njemu je implementirana suština programske logike.

Metode koje se izlažu ka spoljnim sistemima nalaze se u datoteci sa ekstenzijom *.asmx* unutar klase koja nasleđuje sistemsku klasu `System.Web.Services.WebService`.

Metode veb servisa označene su direktivom `[WebMethod]` iznad potpisa i uglavnom imaju povratne vrednosti – celobrojne ili *DataTable* objekte (*Slika 13*).

```
[WebMethod]
public int ActivateCustomer(string msisdn, int channelId)
```

Slika 13. Potpis metode veb servisa

KreditAlarmWS sadrži dve metode – metodu za aktivaciju i metodu za deaktivaciju usluge. Ulazni parametri su identifikatori pristiglog zahteva – broj telefona i kanal sa koga stiže zahtev.

Na početku svake od metoda vrši se provera ulaznih parametara. Proverava se da li je šifra kanala odgovarajuća, da li je MSISDN odgovarajuće strukture i ukoliko jeste, da li je dati MSISDN pripejd broj. Ukoliko bilo koja od provera nije zadovoljena, zahtev se odbija. U suprotnom se nastavlja sa daljim tokom aplikacije.

Metoda za aktivaciju dalje proverava da li je za datog korisnika usluga već aktivna. Ukoliko jeste, zahtev se odbija, a u suprotnom se vrši poziv uskladištene procedure za aktivaciju usluge.

Slično, metoda za deaktivaciju vrši proveru da li za korisnika koji želi odjavu stvarno postoji aktivna usluga. Ukoliko ne postoji, zahtev za deaktivacijom se odbija, a u suprotnom poziva se uskladištena procedura za deaktivaciju usluge.

Uskladištene procedure za aktivaciju i deaktivaciju vrše ažuriranja nad tabelom *Korisnik*, tj. beleže izmene na indikatoru aktivnosti, kao i vrednostima datuma prijave, odnosno odjave usluge za korisnika sa datim MSISDN-om. Nad tabelom *Korisnik*, pored primarnog ključa, postoji i indeks nad kolonom MSISDN koji ubrzava ove operacije.

Nakon svake od operacija veb servisa beleži se zapis u tabelu *Zahtev* kako bi se održavala evidencija zahteva koji stižu na sistem.

4.5 SMS kanal

Kao što je ranije pomenuto, postoje tri kanala putem kojih sistem prihvata zahteve od strane korisnika. U ovom poglavlju biće opisana struktura i funkcionalnost SMS kanala, dok USSD meni i Android aplikacija u ovom radu neće biti razmatrani.

Na *SMS kanal* zahtevi korisnika stižu putem SMS poruka poslatih na kratke brojeve, a čine ga: *SMS centar*, *SDP platforma* i *SMS veb servis* (*Slika 2*). *SMS centar* i *SDP platforma* su spoljni sistemi, a u ovom poglavlju će biti detaljno predstavljen *SMS veb servis* koji će biti referenciran i kao *SMS aplikacija*.

SMS centar je sistem koji predstavlja prvu tačku komunikacije korisnika sa aplikacijama koje obrađuju SMS poruke. On vrši prijem poruka i usmerava ih dalje kroz sistem. Istim putem vrši se i transfer poruka od sistema ka korisniku.

Da bi proizvoljan SMS servis funkcionisao, neophodno ga je definisati na *SDP platformi*. Parametri koji se definišu su kratak broj i URL veb servisa. Korisnik inicira poziv sistema slanjem poruka predefinisane sadržine na unapred definisani kratak broj. Na *SDP platformi* nalazi se kolekcija svih zauzetih i raspoloživih kratkih brojeva. Svakom novom servisu *SDP platforma* dodeljuje novi kratak broj. Svakom kratkom broju platforma dodeljuje URL veb servisa koji će obrađivati SMS poruke koje pristižu na taj kratak broj. U konkretnom slučaju, definisanom kratkom broju aplikacije „Kredit alarm“, dodeljen je URL *SMS veb servisa*. Kada poruka preko *SMS centra* stigne do *SDP platforme*, poziva se veb servis odgovarajućeg kratkog broja i vrši dalju obradu pristigle poruke.

SMS aplikacija vrši proveru ispravnosti sintakse pristigle SMS poruke, a kao validni zahtevi tretiraju se samo oni sa ispravnom sintaksom. Ovakvi zahtevi se dalje obrađuju, dok se ostali zahtevi odbijaju, a korisnik biva obavešten o neispravnosti sintakse SMS porukom.

Prijem i slanje SMS poruka iz aplikacije vrši se pomoću metoda koje izlaže *SDP platforma*. Metode je, u zavisnosti od željene akcije, potrebno implementirati ili pozivati.

Za prijem poruka potrebno je implementirati SDP interfejs (pogledati [5]). Zbog toga glavna klasa SMS veb servisa ima sledeći potpis (*Slika 14*):

```
public class Service1 : System.Web.Services.WebService, IsmsNotificationBinding
```

Slika 14. Potpis glavne klase SMS veb serivsa

Interfejs `IsmsNotificationBinding` sadrži nekoliko metoda, a za potrebe prihvatanja pristiglih SMS poruka, potrebno je implementirati metodu `notifySmsReception` (*Slika 15*).

```
[WebMethod]
public notifySmsReceptionResponse notifySmsReception(notifySmsReception request)
```

Slika 15. Potrpis metode notifySmsReception

Nakon prihvatanja poruke, *SMS aplikacija* vrši njenu analizu. Iz poruke se izdvaja sadržaj, broj telefona pošiljaoca i kratak broj. Na osnovu broja telefona određuje se kom tipu korisnika pripada pošiljalac i ukoliko broj nije pripejd, zahtev se odbija, a korisnik o odbijanju biva obavešten SMS porukom.

Ukoliko broj jeste pripejd, vrši se provera sadržaja poruke, tj. da li poruka poštuje definisana pravila sintakse. U konkretnom slučaju, sintaksu predstavljaju ključne reči za aktivaciju i deaktivaciju usluge – ALARM i ALARM STOP. Provera sintakse vrši se ispitivanjem uzoraka pomoću regularnih izraza (pogledati [9]). Ukoliko poruka nema odgovarajuću sintaksu, korisnik o neregularnosti biva obavešten SMS porukom. Ukoliko je sintaksa poruke ispravna, u zavisnosti od vrste zahteva, poziva se odgovarajuća metoda glavnog veb servisa – za aktivaciju ili deaktivaciju usluge. Po uspešno ili neuspešno izvršenoj operaciji, korisnik se takođe obaveštava SMS porukom.

Za slanje SMS poruka od sistema ka korisniku, potrebno je pozvati SDP metodu *SendSms* iz klase sa zaglavljem (*Slika 16*):

```
public partial class SendSmsService :Microsoft.Web.Services3.WebServicesClientProtocol
```

Slika 16. Potpis klase SendSmsService

SendSms metoda kao parametre prihvata MSISDN primaoca, kratak broj i tekst poruke koja se šalje (pogledati [5]). Jednostavnim pozivom ove metode, poruka biva usmerena ka korisniku, a metoda kao povratnu vrednost daje indikator uspešnosti isporučivanja poruke. Sve poruke od sistema ka korisniku šalju se na ovaj način.

Istoriju svih pristiglih zahteva *SMS aplikacija* beleži u tabelu *ZahtevSms*, dok se sve odlazne poruke beleže u tabelu *OdgovorSms*. Na ovaj način aplikacija ima potpunu kontrolu nad protokom zahteva i omogućava podršku eventualnom rešavanju korisničkih reklamacija.

Operacije nad bazom podataka i u ovom delu sistema vrše se pomoću uskladištenih procedura – upisi u log tabele, kao i izlistavanja tabela sa informacijama o kratkom broju, definisanim sintaksama i tekstovima odlaznih poruka.

U projektu su korišćene dve vrste poziva procedura i stoga metode koje vrše ove pozive za povratnu vrednost imaju *DataTable* objekat ili numeričku vrednost tipa *int* ili *long*. Primeri ovih metoda dati su u naredna dva segmenta koda (*Slika 17*, *Slika 18*).

```
// objekat u koji se smesta povratna vrednost procedure
DataSet ds = new DataSet();

// imenovanje konekcije ka bazi podataka
using (SqlConnection con = new SqlConnection(ConStr))
{
	try
	{
		// imenovanje procedure koja se poziva; procedura izvrsava SELECT upit
		SqlCommand cmd = new SqlCommand("sp_KratakBrojList", con);
		cmd.CommandType = CommandType.StoredProcedure;

		 // izvrsavanje procedure
		 con.Open();
		SqlDataAdapter adapter = new SqlDataAdapter(cmd);

		 // smestanje rezultata procedure, tj. SELECT upita u DataSet objekat
		adapter.Fill(ds, "KratakBroj");
		con.Close();
	}

	catch (Exception ex)
	{
		 // postavljanje vrednosti DataSet objekta u slucaju greske
		ds = null;

		// logovanje zapisa o gresci
		ErrorLogInsertDAL("OperationDAL.KratakBrojListDAL", "", ex.Message);
	}
}

// postavljanje povratne vrednosti metode;
// povratna vrednost je prva tabela iz DataSet objekta ili null u slucaju greske
return (ds != null) ? ds.Tables[0] : null;
```

Slika 17. Metoda za poziv procedure koja vraća DataTable objekat

```
long rez = -1; // povratna vrednost metode

// imenovanje konekcije ka bazi podataka
using (SqlConnection con = new SqlConnection(ConStr))
{
	try
	{
		// imenovanje procedure koja se poziva
		SqlCommand cmd = new SqlCommand("sp_ZahtevSmsInsert", con);
		cmd.CommandType = CommandType.StoredProcedure;

		// parametri procedure
		if (!String.IsNullOrEmpty(msisdn))
			cmd.Parameters.Add(new SqlParameter("@msisdn", msisdn));
		else
			cmd.Parameters.Add(new SqlParameter("@msisdn", DBNull.Value));

		// ...

		if (!String.IsNullOrEmpty(incomingMessage))
			cmd.Parameters.Add(new SqlParameter("@poruka", incomingMessage));
		else
			cmd.Parameters.Add(new SqlParameter("@poruka", DBNull.Value));

		// povratna vrednost procedure
		SqlParameter pRez = cmd.Parameters.Add("@zahtevSmsId", SqlDbType.Int);
		pRez.Direction = ParameterDirection.ReturnValue;

		// izvrsavanje procedure
		cmd.Connection.Open();
		cmd.ExecuteNonQuery();
		cmd.Connection.Close();

		// postavljanje povratne vrednosti metode u slucaju uspesnog izvrsenja
		rez = Convert.ToInt64(pRez.Value);
	}

	catch (Exception ex)
	{
		// postavljanje povratne vrednosti metode u slucaju greske
		rez = -1;

		// logovanje zapisa o gresci
		ErrorLogInsertDAL("OperationDAL.ZahtevSmsInsertDAL", "", ex.Message);
	}
}

return rez;
```

Slika 18. Metoda za poziv procedure koja vraća primitivni tip podataka

4.6 FTP aplikacija

Podatke o korisnicima čiji je iznos kredita pao ispod dogovorenog limita od 50,00 dinara sistem „Kredit alarm“ dobija od eksternog partnerskog sistema. Podaci se nalaze u tekstuelnim datotekama, a eksterni sistem ih postavlja u malim vremenskim razmacima na FTP server sistema „Kredit alarm“.

U datotekama se nalaze MSISDN-ovi korisnika i dodatni parametri vezani za spoljašnji sistem. Na osnovu ovih dodatnih parametara vrši se izdvajanje redova koji su od značaja za sistem „Kredit alarm“.

FTP aplikacija i aplikacija *SMS obaveštenja* implementirane su kao delovi automatskog posla *KreditAlarmObavestenje* na Windows okruženju koji se okida svakodnevno u malim vremenskim razmacima.

Klasa u kojoj je smeštena logika nasleđuje sistemsku klasu `System.Windows.Forms.Form`, a na učitavanju se vrši zatvaranje prozora forme, kao što je ranije u radu pomenuto. Ovo se postiže na sledeći način (*Slika 19*):

```
private void KreditAlarmObavestenje_Load(object sender, EventArgs e)
{
	// ...
	this.Close();
}
```

Slika 19. Zatvaranje prozora Windows forme

FTP aplikacija preuzima datoteke sa spiskom korisnika sa FTP servera, zatim sadržaj pojedinačne datoteke smešta u *DataTable* objekat i na kraju pomoću *bulk insert* operacije, u transakciji (pogledati [4]), zapisuje informacije u tabelu baze podataka. Ova tabela nazvana je *KorisnikSDP* i predstavlja početni skup podataka iz koga će se vršiti odabir korisnika kojima je potrebno slati SMS obaveštenja o padu kredita.

Bulk insert u tabelu *KorisnikSDP* dat je sledećim nizom naredbi (*Slika 20*):

```
using (SqlBulkCopy sqlBulkCopy =
                            new SqlBulkCopy(conn, SqlBulkCopyOptions.TableLock, tran))
{
       // imenovanje tabele na bazi u koju se vrsi upis podataka
       sqlBulkCopy.DestinationTableName = "KorisnikSDP";

       // velicina bloka za kopiranje, u najgorem slucaju moze biti dt.Rows.Count
       sqlBulkCopy.BatchSize = 20000;

       sqlBulkCopy.BulkCopyTimeout = 0; // neograniceno, inace je broj u sec

       // mapiranje izmedju kolona DataTable objekta i kolona tabele u bazi
       sqlBulkCopy.ColumnMappings.Add(0, "Msisdn");
       // ...
       sqlBulkCopy.ColumnMappings.Add(6, "Datum");
       sqlBulkCopy.ColumnMappings.Add(7, "Vreme");

       // upis bloka podataka DataTable objekta u bazu
       sqlBulkCopy.WriteToServer(dt);
       sqlBulkCopy.Close();
}

// nakon uspesnog kopiranja svih blokova, vrsi se potvrda transakcije
tran.Commit();
```

Slika 20. Operacija bulk insert

4.7 SMS obaveštenja

Aplikacija *SMS obaveštenja* izvršava se odmah nakon *FTP aplikacije* u okviru automatskog posla *KreditAlarmObavestenje*. Njena funkcionalnost obuhvata slanje SMS obaveštenja korisnicima u slučaju da se iznos njihovog kredita smanjio ispod dogovorenog limita.

Procedura na bazi podataka, iz tabele *KorisnikSDP*, vrši izbor MSISDN-ova gore pomenutih korisnika i kao rezultat vraća tabelu, podskup tabele *KorisnikSDP*. Aplikacija *SMS obaveštenja* prihvata rezultat ove procedure i obrađuje red po red rezultujuće tabele. Za svaki MSISDN iz rezultujućeg skupa poziva se metoda sistema SDP za slanje SMS poruka *SendSms* da korisnicima pošalje poruke obaveštenja, a zatim se vrši logovanje poslatih poruka u odgovarajuću tabelu.

Na ovaj način korisnici su informisani o smanjenju iznosa njihovog kredita ispod 50,00 dinara, tj. ispod granice dogovorenog limita, čime je i kompletan ciklus sistema „Kredit alarm“ zatvoren.

4.8 Izveštavanje

Proces izveštavanja u sklopu jednog projekta je veoma važan jer su upravo izveštaji jedini način na koji rukovodioci i klijenti dobijaju informacije o rezultatima rada programa i na osnovu njih su u mogućnosti da prave najrazličitije statistike i predviđanja.

U ovom projektu su od strane klijenta naručeni mesečni izveštaji, tj. izveštaji koji se isporučuju jednom mesečno i to svakog prvog u mesecu za prethodni mesec. U pitanju su sumarni izveštaji o broju korisnika i broju aktivacija i deaktivacija u toku jednog kalendarskog meseca. Pošto su u pitanju jednostavne sume, nije bilo potrebe za izradom korisničkog GUI-ja, već se traženi podaci korisnicima dostavljaju putem elektronske pošte.

Podaci o broju aktivnih korisnika dobijaju se iz tabele *Korisnik*, a kao osnova za dobijanje podataka o broju aktivacija i deaktivacija, korišćena je tabela *KorisnikZahtev*. Prilikom stizanja zahteva na sistem, beleženje svakog uspešnog zahteva za aktivacijom ili deaktivacijom vrši se, pored tabele *Zahtev*, i u tabelu *KorisnikZahtev*. Na ovaj način tabela *KorisnikZahtev* predstavlja podskup uspešnih zahteva tabele *Zahtev*, odnosno skup podataka relevantnih za izveštaje. Tabela *KorisnikZahtev* sadrži indeks po polju koje sadrži datum realizacije zahteva i na taj način je optimizovana za pretrage podataka u odnosu na datum, odnosno mesec.

Automatsko slanje mail-ova sa izveštajima implementirano je pomoću uskladištenih procedura postavljenih u automatske poslove korišćenjem programa *SQL Server Agent* (pogledati [10]). Procedure su podešene da se pokreću jednom mesečno i na taj način korisnici u traženo vreme dobijaju na svoje email adrese izveštaje zakačene u vidu priloga.

Za implementaciju je korišćena ugrađena SQL Server procedura *sp_send_dbmail*. Primer korišćenja dat je sledećim nizom naredbi (*Slika 21*):

```
EXEC msdb.dbo.sp_send_dbmail
@profile_name = 'SQL Server',
@recipients = 'Izvestaji@mail.com',
@subject = 'Kredit alarm - broj korisnika',
@query = @queryStr,
@attach_query_result_as_file = 1,
@query_attachment_filename = 'KA_BrojKorisnika.txt',
@body = N'Podaci su u prilogu.<br/><br/>--<br/>Pozdrav,<br/>Kredit alarm',
@body_format='HTML';
```

Slika 21. Upotreba procedure sp_send_dbmail

Kao parametri procedure prosleđuju se: lista primalaca odvojenih tačka zarezom, tekst SQL upita koji daje tražene podatke, parametri koji se tiču tela mail-a – formatiranje teksta i propratni sadržaj, podatak o načinu smeštanja izveštajnih podataka u mail i naziv datoteke sa izveštajnim podacima. Izraz *attachQueryResult = 1* definiše da će podaci biti poslati kao prilog mail-a smešteni u tekstuelnu datoteku. U tom slučaju potrebno je definisati i naziv datoteke u koju će podaci biti smešteni, što je prosleđeno kroz parametar *query_attachment_fileName*. Po definisanju parametara, procedura se pokreće pomoću ključne reči *exec*.

4.9 Testiranje

Za završno testiranje SMS aplikacije korišćen je test prihvatanja (eng. *acceptance test*), a spisak testiranih slučajeva dat je u narednoj tabeli (*Slika 22*).

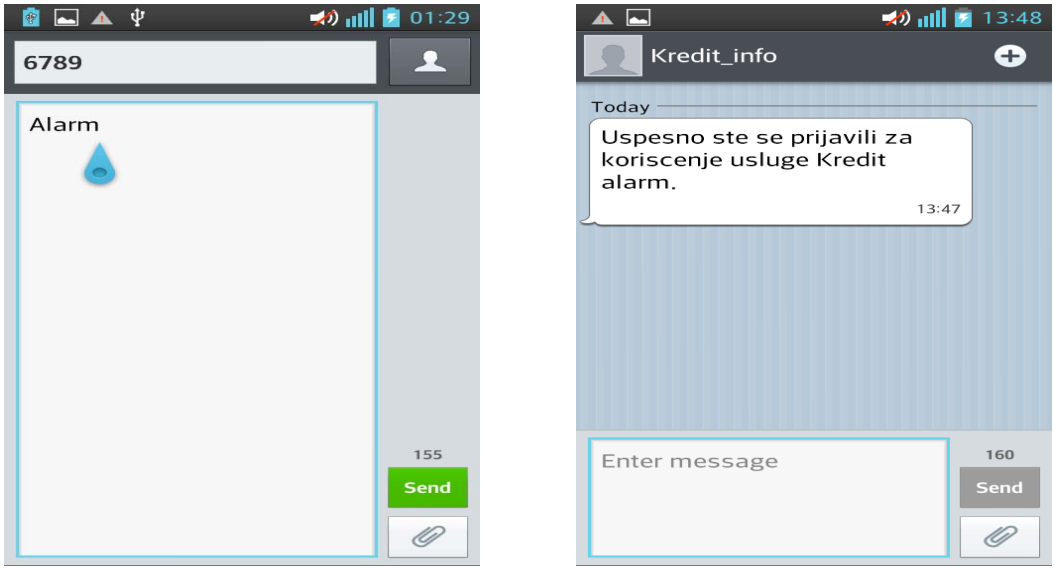
Redni broj testa	Opis testa
1	Pripejd korisnik šalje zahtev za aktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Uspesno ste se prijavili za koriscenje usluge Kredit alarm.*
2	Pripejd korisnik kojem je već aktivirana usluga ponovo šalje zahtev za aktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Vec ste prijavljeni za koriscenje usluge Kredit alarm.*
3	Korisnik koji nije pripejd šalje zahtev za aktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Ova usluga Vam nije omogucena. Za vise informacija pozovite 0800/100100.*
4	Pripejd korisnik šalje poruku sa sintaksnom greškom za aktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Vas zahtev nije prihvacen. Poslali ste poruku u pogresnom formatu. Molimo Vas da pokusate ponovo.*
5	Korisnik koji nije pripejd šalje poruku sa sintaksnom greškom za aktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Ova usluga Vam nije omogucena. Za vise informacija pozovite 0800/100100.*
6	Pripejd korisnik pokuša aktivaciju, ali je servis nedostupan zbog tehničkih problema. Rezultat - korisnik dobija SMS poruku: *Usluga trenutno nije dostupna. Molimo Vas da pokusate kasnije.*
7	Pripejd korisnik šalje zahtev za deaktivaciju usluge, ima aktivnu uslugu. Rezultat - korisnik dobija SMS poruku*: Uspesno ste odjavili koriscenje usluge Kredit alarm.*
8	Pripejd korisnik šalje zahtev za deaktivaciju usluge, a nema aktivnu uslugu. Rezultat - korisnik dobija SMS poruku: *Usluga Kredit alarm Vam nije aktivirana, tako da odjava nije moguca.*

Redni broj testa	Opis testa
9	Korisnik koji nije pripejd šalje zahtev za deaktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Ova usluga Vam nije omogucena. Za vise informacija pozovite 0800/100100.*
10	Pripejd korisnik šalje poruku sa sintaksnom greškom za deaktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Vas zahtev nije prihvacen. Poslali ste poruku u pogresnom formatu. Molimo Vas da pokusate ponovo.*
11	Korisnik koji nije pripejd šalje poruku sa sintaksnom greškom za deaktivaciju usluge. Rezultat - korisnik dobija SMS poruku: *Ova usluga Vam nije omogucena. Za vise informacija pozovite 0800/100100.*
12	Pripejd korisnik pokuša deaktivaciju, ali je servis nedostupan zbog tehničkih problema. Rezultat - korisnik dobija SMS poruku: *Usluga trenutno nije dostupna. Molimo Vas da pokusate kasnije.*
13	Pripejd korisniku iznos kredita pada ispod 50,00 din. Potrebno je da u narednih 3-5 min dobije SMS notifikaciju. Rezultat - korisnik dobija SMS poruku*: Obavestavamo Vas da imate manje od 50 dinara kredita. mts Kredit alarm*
14	Nakon dobijene SMS notifikacije korisniku je iznos pripejd kredita još niži, ali ne treba da dobije novo obaveštenje. Rezultat - korisnik ne dobija SMS poruku.
15	Nakon dobijene notifikacije da mu je kredit pao ispod 50,00 din korisnik dopuni kredit i ubrzo mu kredit opet pada ispod 50,00 din. Potrebno je da u narednih 3-5 min dobije SMS notifikaciju. Rezultat - korisnik dobija SMS poruku*: Obavestavamo Vas da imate manje od 50 dinara kredita. mts Kredit alarm*

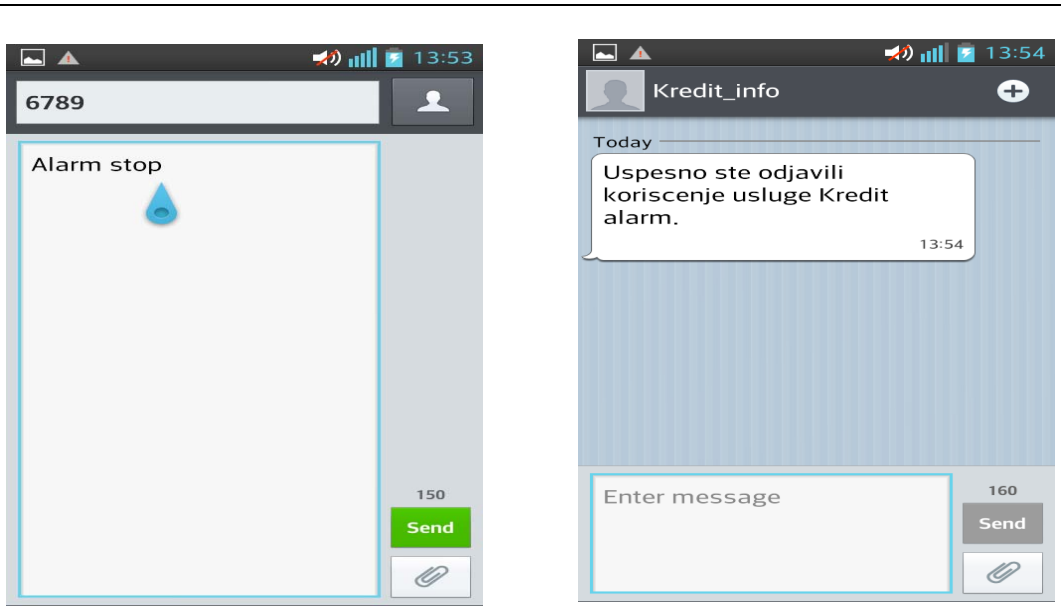
Slika 22. Spisak testova

5 Primeri korišćenja usluge „Kredit alarm“

Primeri korišćenja usluge „Kredit alarm“ putem SMS poruka prikazani su na slikama koje slede (*Slika 23* – *Slika 26*). Slike predstavljaju prikaz ekrana mobilnog telefona korisnika u trenucima slanja zahteva ka sistemu i primanja odgovora od sistema.



Slika 23. Prijava za korišćenje usluge i obaveštenje o uspešnoj prijavi



Slika 24. Odjava korišćenja usluge i obaveštenje o uspešnoj odjavi

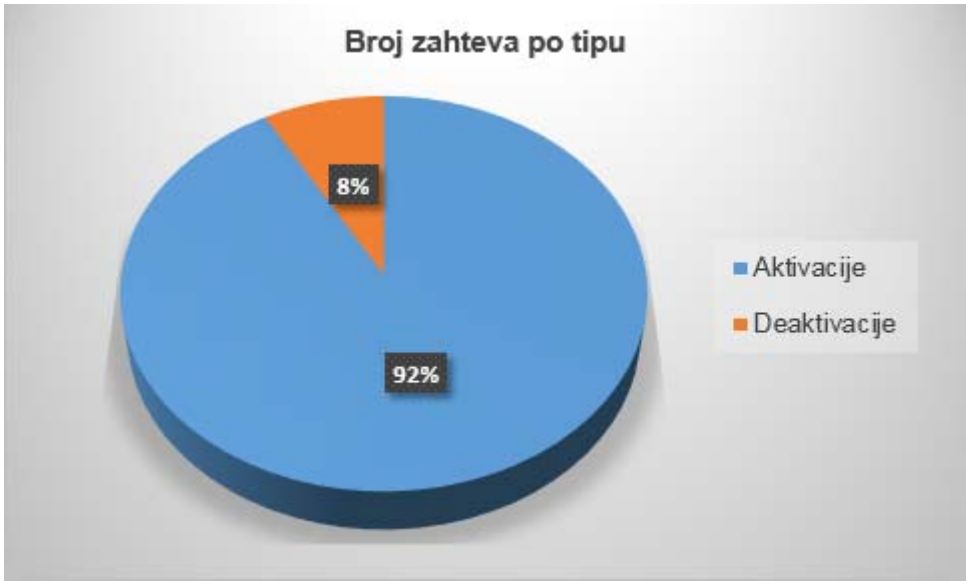
Slika 25. Obaveštenje o padu kredita ispod dogovorenog limita

Slika 26. Obaveštenje u slučaju sistemske greške

6 Statistički podaci vezani za uslugu „Kredit alarm“

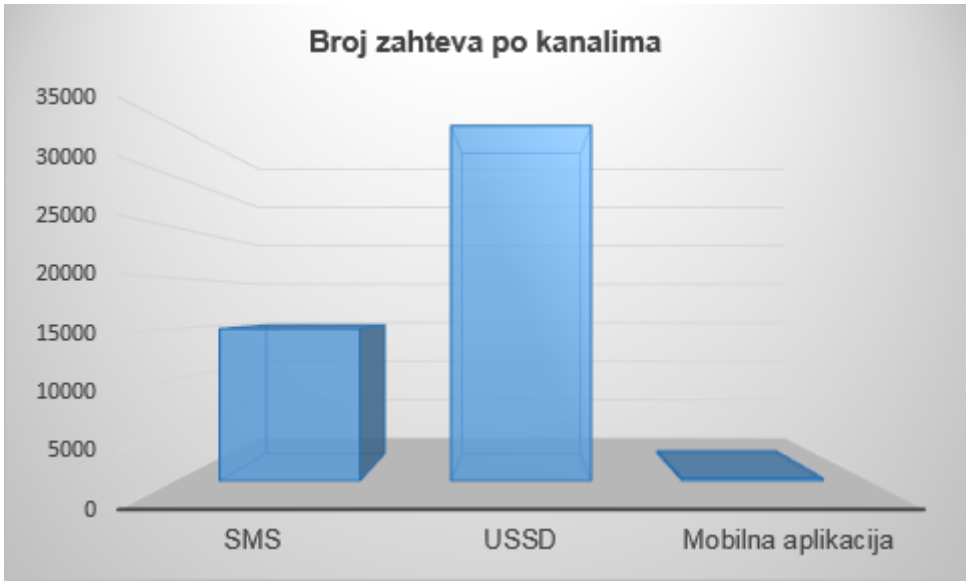
Servis „Kredit alarm“ je razvijen u toku 2015. godine i od tada je u upotrebi kod lokalnog operatera mobilne telefonije (pogledati [7], [8]). Trenutno ima preko 40.000 aktivnih korisnika, a broj je u stalnom porastu. Ukoliko se ovakav trend nastavi, korisnici usluge činiće značajan procenat ukupne pripejd populacije.

Na grafikonu na *Slici 27* dat je prikaz dosadašnjeg broja aktivacija u odnosu na broj deaktivacija usluge.



Slika 27. Uporedni prikaz broja zahteva po tipu

Na S*lici 28* dat je dijagram sa uporednim prikazom broja zahteva po kanalima. Pokazuje se da korisnici u najvećoj meri koriste USSD meni, sledi SMS kanal sa približno polovinom prometa u odnosu na USSD, dok je mobilna aplikacija najmanje zastupljena, do 100 puta manje u odnosu na USSD kanal.



Slika 28. Uporedni prikaz broja zahteva po kanalima

7 Zaključak

U ovom radu predstavljen je način na koji je implementirana SMS usluga mobilne telefonije. SMS usluga podrazumeva slanje SMS poruka od strane korisnika na unapred definisane kratke brojeve. U radu je opisan proces koji se odvija u pozadini jedne SMS usluge. Objašnjeno je kako se programski vrši obrada pristiglih SMS poruka i kako korisnik od strane sistema biva obavešten povratnom SMS porukom. Predstavljen je kompletan razvojni ciklus aplikacije – od arhitekture sistema i baze podataka, pa sve do izveštavanja. Dat je i pregled procesa testiranja, a na kraju pokazan i praktičan primer korišćenja usluge sa korisničke strane.

Aplikacija koja je opisana u radu koristi se aktivno kod lokalnog operatera mobilne telefonije.

U daljem razvoju aplikacije u planu je da se aplikacija preradi kako bi koristila veb tehnologije uključene u nove verzije softvera korišćenog pri razvoju, odnosno WCF i REST servise.

8 Spisak slika

9 Pregled skraćenica

BLL – Business Layer Logic
DAL – Data Access Logic
DML – Data Manipulation Language (select, insert, update, delete operacije nad bazom podataka)
FK – Foreign Key
FTP – File Transfer Protocol
GUI – Graphical User Interface
HTTP – HyperText Transfer Protocol
MSISDN – Mobile Station International Subscriber Directory Number, broj mobilnog telefona korisnika
PK – Primary Key
RATEL – Republička agencija za elektronske komunikacije i poštanske usluge
REST – Representational State Transfer
SDP – Service Delivery Platform
SMS – Short Message Service
SOAP – Simple Object Access Protocol
SQL – Structured Query Language
URL – Uniform Resource Locator
USSD – Unstructured Supplementary Services Data
USSD meni – korisnički meni na mobilnom telefonu, dobija se pozivom kombinacije brojeva kojima prethodi zvezdica, a završavaju tarabom, npr. *100#
WCF – Windows Communication Foundation
WSDL – Web Service Definition Language

10 Literatura

[1] An Introduction to Database Systems, 8th edition, C. J. Date, 2004, Pearson
[2] Database Systems: The Complete Book, 2nd edition, Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, 2009, Pearson
[3] Osnove relacionih baza podataka, drugo izdanje, Gordana Pavlović-Lažetić, 1999, Matematički fakultet, Beograd
[4] Baze podataka – vežbe, Saša Malkov, 1997/98, Matematički fakultet, Beograd
[5] TS-SDP Service Provider Interface Specification, 2015, Ericsson
[6] RATEL, http://www.ratel.rs/trziste/pregledi_trzista.230.html
[7] mts, https://www.mts.rs/privatni/vesti/Privatni-korisnici---vesti/913
[8] mts Mondo, http://mondo.rs/a804579/mts/telekom-srbija-vesti/Kredit-alarm-da-vas-podseti.html
[9] MSDN, https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx
[10] MSDN, https://msdn.microsoft.com/en-us/library/ms190268.aspx
[11] Code Project, http://www.codeproject.com/Tips/351122/What-is-software-testing-What-are-the-different-ty