UDC: 004.75:004.4 Info M: str. 44-50

RAZVOJ PERSONALNOG SOFTVERSKOG SISTEMA ZA UPRAVLJANJE FOTOGRAFIJAMA NA RAZLIČITIM PLATFORMAMA
DEVELOPING PERSONAL SOFTWARE SYSTEM FOR MANAGING IMAGES ON DIFFERENT PLATFORMS

Davidović Vukašin, Saša D. Lazarević

REZIME: Brojni su uređaji koji rade na različitim platformama i sa različitim programima. Zato dolazi do potrebe da se razvija softver koji je sposoban da komunicira sa različitim aplikacijama napisanim u različitim programskim jezicima i na različitim platformama. Najbolji način da se to primeni je kreiranje veb servisa koji se zasniva na servisno orijentisanoj arhitekturi. Taj servis će komunicirati sa ostalim aplikacijama preko SOAP poruka koje su zasnovane na XML-u. Kao tehnologiju za razvoj veb servisa je korišćen Microsoft WCF (Windows Communication Foundation). Klijentske aplikacije koje taj servis opslužuje u ovom softverskom sistemu su .NET Rich Client aplikacija i Java Android aplikacija. Dakle dve aplikacije koje su pisane u različitim programskim jezicima i obe su zasnovane na drugačijim platformama.

KLJUČNE REČI: SOA, WCF, Android, Cloud, interoperabilnost, veb servis

ABSTRACT: There are many devices that run on different platforms and with different computer programs. Therefore there is a need to develop software which has ability to communicate with different applications written in different programming languages and run on different platforms. The best way to do that is to create a web application service which is based on service-oriented architecture. This service will communicate with other applications via SOAP-based XML messages. Technology used for developing Web services is WCF (Windows Communication Foundation). Client applications that this service is serving in this software system are .NET Rich Client applications and Java Android applications. Those are two applications that are written in different programming languages, and both are based on different platforms.

KEY WORDS: SOA, WCF, Android, Cloud, interoperability, web service

1. UVOD

Danas živimo u tehnološkom dobu gde uređaj koji je pre mesec dana bio primer najnovije tehnologije i koja je zadovoljavala sve korisničke zahteve, a danas samo jedan od zastarelih modela. U tom svetu se izdvajaju milijarde novčanih jedinica da bi se našao najbolji način da se privuče kupac plasiranog tehnološkog proizvoda ili korisnik softverskih usluga.

Kompanije koje su bili pioniri nekih tehnoloških inovacija a koje su iskoristile trend naglog rasta tehnoloških dostignuća su danas vodeće firme u okviru informacionih tehnologija i softverskog razvoja. One oblikuju izgled uređaja koji nas okružuju, podstiču način u kom će se tehnologija kretati kako u kratkoročnom tako i u dugoročnom vremenskom periodu. U trci sa drugim velikim firmama za deo profita koji ova industrija u zaletu nudi, često prave serije uređaja koji nisu komaptibilni sa uređajima drugih, rivalskih firmi. Upravo tu se javlja jedan od najvećih problema u tom „moru" tehnoloških proizvoda, a to je njihova neunificiranost. Ako izaberete tehnologiju ili sistem koji nudi jedna kompanija, ne mora da znači da ćete fotografije, muziku i druge mulitimedijalne formate moći da pogledate ili preslušate na platformama nekog drugog proizvođača.

Kako je to unosan posao od kog može neverovatno da se zarađuje veliki broj manjih kompanija se otvara i nude svoje proizvode kao alternativu za često skupe brendirane uređaje. Te kompanije uglavnom nemaju dovoljno kapitala da bi napravile velike investicije, pa često i njihove alternative koje kreiraju za korisnike sa ''plićim'' džepom. Te kreirane aplikacije ili uređaji imaju svoje formate ili čak totalno drugačiji tip povezivanja u odnosu na ove velike kompanije. Često ti proizvodi nemaju sve mogućnosti da se nešto napravi, da se povežu sa nekim aplikacijama ili programima koji korisnici smatraju podrazumevanim.

Takođe, postoje i *open source* platforme, programi i aplikacije koji predstavljaju tek jedan od najvećih mogućnosti za raznovrsnost. *Open source* dozvoljava korisniku skoro neograničenu slobodu da se „igra" sa tim tehnologijama i stvori nešto što ne mora da se naslanja ni na šta što je do tog trenutka ljudski intelekt smislio. Te apliakcije uglavnom ne koristi veliki broj ljudi ali upravo iz razloga što takva neka aplikacija ili program nije kompatabilan sa nekim drugim procesima koji bi je oplemenili i koji bi zainteresovali krajnjeg korisnika za nju.

Jedna od velikih mana neunificiranosti tehnologija ne utiče samo na krajnjeg korisnika koji kupuje željeni prozvod ili potrebnu uslugu. Ona takođe utiče i na ljude koji direktno stvaraju te programe. Danas je pravo pitanje kako najbolje izabrati neku tehnologiju ili platformu? Čega se krajnji korisnik odriče? Da li je bolje imati dobar proizvod koji nema mogućnost komunikacije sa svim uređajuma ili je bolje napraviti ili nabaviti rešenje koje zadovoljava sve uređaje i platforme ali samim tim nije toliko efikasno.

U ovom radu će biti predstavljena ova tematika, kako povezati više uređaja sa različitm tehnologijama, sa totalno različitim platformama, koristeći različite programske jezike u jednu celinu i dati im mogućnost da se i dalje integrišu sa nekim novim rešenjima. Primer povezivanja je prikazan preko transfera fotografija. Fotografije su danas rasprostranjene na sve strane, svaki pametni telefon je u mogućnosti da napravi stotine fotografija različitih formata. Te fotografije danas često imaju nazive koji nemaju veze sa samim sadržajem same foto-

grafije. Zato je predstavljena aplikacija koja tim fotografijama preko baze podataka pridružuje meta podatke i omogućava da se one kasnije lakše pretraže.

U našem okruženju postoje različiti uređaji: mobilni telefon, stacionirani računar, laptop, itd. Svi ti uređaji imaju svoje fotografije i nije baš jednostavno da se te fotografije stalno prebacuju sa jednog uređaja na drugi. Zbog toga je kao primer napravljen veb servis u koji može jednostavno da se poveže sa nekim uređajem ili nekom aplikacijom. Zajednički format preko koga se šalju fotografije je *XML* i njega svaka aplikacija može da prihvati i pročita. Takođe velika većina programskih jezika danas ima mogućnost da enkriptuje i dektiptuje fotografiju kao tekst i da je tako prenese i kasnije pročita iz sopstvene aplikacije.

Dakle, napravljena je centralna aplikacija, web servis i ostale aplikacije sa različitm platformama koje fotografije napravljene na jednom uređaju u vrlo kratkom vremenskom roku mogu da pogledaju na nekom drugom uređaju ili preko neke druge aplikacije. Jedino što je neophodno je dobra internet konekcija, što je danas sasvim normalna stvar.

2. SERVISNO ORIJENTISNA ARHITEKTURA - SOA (SERVICE-ORIENTED ARCHITECTURE)

U današnjem IT svetu veliki izazov predstavlja implementacija distribuiranih sistema koji podržavaju poslovnu logiku na pouzdan način. Kada kreiramo nedistributivne sisteme većinu karakteristika uzimamo zdravo za gotovo – a te karakteristike mogu postati vrlo važne kada se radi sa distribuiranim sistemima. Iako su neki od izazova na koje se nailazi očigledni (npr. gubitak konekcije ili gubljenje i korupcija samih podataka) postoje i drugi aspekti, kod čvrsto zavisnih sistema, veza između komponenti sistema je takva da je cena izmene prevelika u odnosu na korist koju dobija poslovanje tom promenom. Poslovni procesi su često u velikoj zavisnosti od sistema koji koji rade na drugim platformama i sa drugačijim tehnologijama, kako unutar tako i izvan organizacije. Servisno orijentisna arhitektura je mehanizam koji omogućava organizaciji da olakša komunikaciju između sistema koji rade na drugim platformama.

Tokom protekle decenije je bilo dosta istraživanja na polju distribuiranih sistema. *Microsoft* i ostali vodeći proizvođači su napravili dosta tehnologija koje podržavaju distribuciju programa i aplikacija. Svaka od tih tehnologija olakšava kreiranje bogatih aplikacija i smanjuje troškove kreiranja takvog softvera. Jedna od tehnologija je i *Microsoft*-ov *Windows Communication Foundation* (*WCF*) koji omogućava uniforman način razvoja distribuiranih aplikacija pružajući servisno orijentisan programski model.

Sa *WCF*-om (nekad poznat kao Indigo) od verzije *.NET* 3.x i novije je moguće napraviti aplikacije sa širokim spektrom mogućnosti kroz njegov pojednostavljen model. Zasnovan na ideji servisa, *WCF* sadrži najbolje karakterisitike današnjih distribuiranih tehnologija za razvoj novih i povezivanje sa već postojećim sistemima.

2.1. *Detaljnije o SOA*

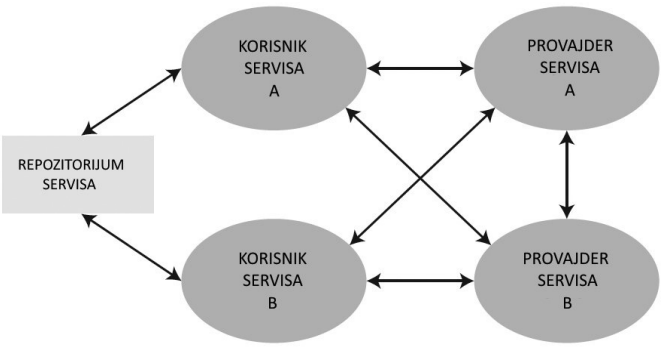
U današnjim multinacionalnim kompanijama nije praktično napraviti jednodelni (sistem od jedne centralne aplikacije i korišćenjem jedne tehnologije) sistem. Ovakvim sistemima je obično potrebno dosta vremena, čak i godine da se implementira i obično je zadužen za uzak dijapazon potreba poslovanja. Današnji poslovni procesi uz sebe traže agilnu i adaptivnu tehnologiju koja može jednostavno da prati promene u biznisu. Upravo je *SOA* jedan od principa koji se zasniva na toj ideji jednostavnosti. Iako dosta implementacija može proći kao *SOA* implementacije korišćenjem web servisa – one to nisu.

U stvarnosti, *SOA* predstavlja kolekciju dobro definisanih servisa. Servis je autonoman (poslovni) sistem koji prihvata jedan ili više zahteva (request) i vraća jedan ili više odgovora (response) preko skupa javnih i dobro definisanih interfejsa. Svaki od individualnih servisa može biti modifikovan nezavisno od ostalih servisa. Upravo je to ono što je potrebno modernom biznisu i evoluirajućem tržištu, brza i agilna promena.

Za razliku od tradicionalne, uzajamno vezane arhitekture, *SOA* implementira arhitekturu koji ne pravi direktnu zavisnost između komponenti a kolektivno rešavaju zajedničke probleme i koračaju ka istom cilju. Pored toga, pošto su detalji skriveni od krajnjeg korisnika, promene u implementaciji neće imati nikakve posledice na korisnika dok god se ne menja ugovor. Upravo ovo omogućava sistemima zasnovanim na *SOA* arhitekturi da se brzo adaptiraju na nove potrebe bez prevelikog uticaja na poslovanje i sa mnogo manje novčanih sredstava.

Iako su neki od aspekata slični razvoju zasnovanom na komponentama (*Component-based*) a koji je zasnovan na striktnim interfejsima postoji ključna razlika, *SOA* ima pristup koji radi sa otvorenim standardima i generičkim porukama koje nisu usko vezane za specifičnu platformu ili određen programski jezik i upravo zbog toga je moguće raditi sa različitim sistemima i omogućiti njihovu komunikaciju.

Vrlo je važno da se servis ne svede na skup interfejsa, jer su oni ključ komunikacije između pružalaca usluga (*provajdera*, *provider*) i korisnika usluga (*consumer*) servisa. U tradicionalnoj klijent-server arhitekturi, pružalac usluga je server a korinsik usluga je klijent.



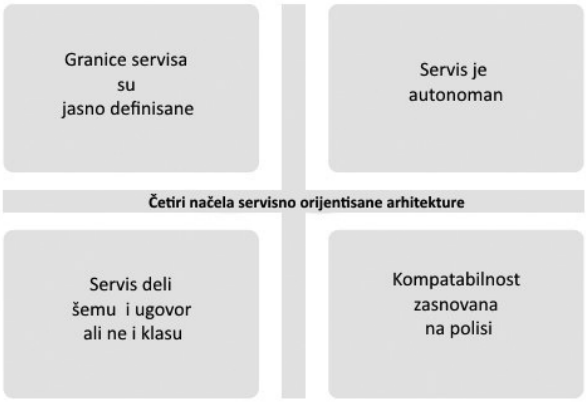
Slika 1 - *Interakcija komponenti u SOA arhitekturi*

U servisnoj orijentaciji, kako je opisana ranije, sve se svodi na servise i poruke. Na slici broj 2 se vidi primer kako pružalac

usluiga i korisnik istih mogu da koegzistiraju sa repozitorijumom koji sadrži formu zasnovanoj na SOA arhitekturi. Provajderi servisa (service providers) su komponente koje izvršavaju deo poslovne logike zasnovane na predefinisanim ulazima i izlazima i pružaju uslugu preko SOA implementacije. Korisnik (consumer), sa druge strane, je set komponenti zainteresovanih da koriste jedan ili više servisa koje nudi provajder. Repozirorijum (repository) sadrži opis servisa, odnosno mesto gde provajder registruje svoje servise a korisnik nalazi ponuđene servise.

2.2. *Načela*

Servisna orijentacija je strategija vođena poslovanjem koja definiše poslovne funkcionalnosti kroz uslove slabe povezanosti autonomnog poslovnog sistema koji razmenjuje informacije zasnovane na porukama. Kao što je gore navedeno, termin servis se koristi u različitim kontekstima, ali u smislu servisno orijentisane arhitekture, servis je zasnovan na četiri fundamentalna načela.

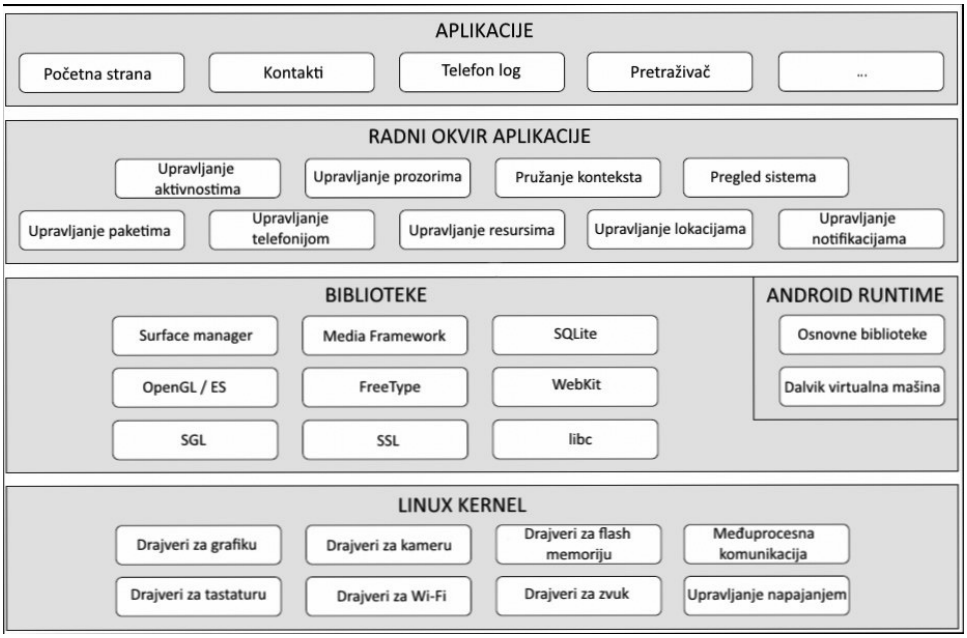


Slika 2 - *Četiri načela servisno orijentisane arhitekture*

3. ANDROID

Struktura (arhitektura) Android operativnog sistema se sastoj od pet sekcija, a koji podpadaju pod pet slojeva:

- *Linux kernel* – Ovo je kernel na kome je zasnovan *Android*. Ovaj sloj sadrži sve drajvere koji na najnižem nivou komuniciraju sa hardverskim komponentama na *Android* uređaju.
- Biblioteke – Ovaj sloj sadrži sav kod koji predtavlja glavnu karakteristiku Android operativnog sistema. Npr. *SQLite* biblioteka pruža podršku na način da aplikacija može da koristi interno skladištenje podataka. *WebKit* biblioteka omogućava krstarenje po internetu.
- *Android runtime* – U istom sloju kao i biblioteke, *Android* runtime omogućava skupom osnovnih biblioteka programerima (developerima) da razvijaju *Android* aplikacije korišćenjem Java programskog jezika. *Android* runtime takođe uključuje *Dalvik* virtualnu mašinu koja omogućava da se svaka *Android* aplikacija pokreće pod sopstvenim procesom, sa svojom instancom *Dalvik* virtulne mašine (*Android* aplikacije sekompajliraju u *Dalvik* izvršne aplikacije). *Dalvik* je specijalna virtualna mašina napravjena i dizajnirana za specifičnosti *Androida* i optimizovana za telefone koje napaja baterija sa ograničenom memorijom i snagom procesora.
- Radni okvir aplikacije (*framework*) – Otkriva različite mogućnosti Android operativnog sistema developerima aplikacija tako da te mogućnosti mogu da integrišu u svojim aplikacijama
- Aplikacije – Predstavlja najviši nivo u *Android OS* arhitekturi. I u tom sloju se nalaze sve aplikacije. One koje dolaze sa telefonom, koje se skidaju sa Android Market-a kao i one koje developeri sami razvijaju.



Slika 3 - *Arhitektura Androida*

4. IMPLEMENTACIJA

Kao što je već pomenuto u prethodnom delu rada, napravljen je veb servis koristći *WCF* kao razvojnu tehnologiju koji komunicira sa korisnicima preko *SOAP* poruka koje su struktuirane preko *XML*-a. Podaci koji se dobijaju od strane klijentskih aplikacija se čuvaju u bazi podataka *SQL Server 2014* koja se ''vrti'' na *Windows Server-u 2012*. Sve ovo čini jedan *cloud* system za čuvanje podataka. Najvažniji podaci koji se prenose preko servisa su fotografije. Korisnik mora biti registrovan da bi koristio *cloud* usluge i da bi čuvao fotografije u bazi podataka. Registracija je besplatna. Fotografije se kao i ostali tekstualni podaci prebacuju u *XML*-u preko *SOAP* poruka. Njih klijenti *encode*-uju iz niza bajtova u pravilno kreiran string koji je moguće decode-ovati na serveru ili na nekoj drugoj klijentskoj mašini.

Kao primer klijentskih aplikacija koje koriste usluge *cloud*-a kao i koje poštuju ugovor veb servisa kreirana je *Android* aplikacija kao i *Rich Client* aplikacije preko *Windows* formi. *Android* aplikacija ima klasičnu izgled arhitekturu jedne aplikacije za taj sistem. Delovi koji su dodavani odnosno promenjeni je da se folder *src* koji sadrži sve kucane kodove u *Java* programskom jeziku koji se sastoje iz četiri dela:

- *Activities*
- *Util*
- *Domain*
- *WebService*

Activities sadrži sve kodove koji služe za upravljanje svake aktivnosti u aplikciji. *Util* je folder koji sadrži pomoćne metode koje se koriste u više aktivnosti kao što su delovi za upravljanje slikama ili kompoziciju i dekompoziciju *XML*-a koji se šalje ili prima od strane veb servisa. Takođe se tu nalazi i datoteka za upravljanje bazom podataka koja se nalazi u svakom *Android* sistemu a to je *SQLite* koji se koristi za čuvanje privremenih podataka. *Domain* folder je folder čiji sadržaj predstavlja izgled domenskih klasa za ovu aplikaciju. *WebService* služi da se u njemu čuvaju klase koje upravljaju povezivanjem sa veb servisom. Zasnovano je na Template method paternu gde postoji jedna centralna apsdtraktna klasa koju nasleđuju ostale. U toj apstraktnoj stoji sve što je potrebno da se definiše poziv ka veb servisu koju koriste ostale klase koje nasleđuju tu klasu i predstavljaju poziv svake metode veb servisa zasebno sastavljene zbog specifičnosti. Biblioteka koju koriste ove klase je Javina prilagođena za rad sa *SOAP* porukama *ksoap2.jar*. Izgled glavne metode u apstraktnoj klasi je sledeći:

```
   public synchronized String callWebService(final
Object[] parameters) {
     thread = new Thread() {
     public void run() {
   try {
       SoapObject request = new SoapObject(namespace,
methodName);
       addParameters(request, parameters);
       SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(
                SoapEnvelope.VER11);
       envelope.dotNet = true;
       envelope.setOutputSoapObject(request);
       HttpTransportSE androidHttpTransport = new
HttpTransportSE(
                url, timeout);
       androidHttpTransport.call(soapAction, envelope);
       SoapPrimitive response = (SoapPrimitive)
envelope.getResponse();
                        webResponse = response.
toString();
         } catch (Exception e)
           {
             e.printStackTrace();}}};
```

Rich Client aplikacija je zasnovana na proširenom MVC arhitekturalnom paternu koji nezavisno odvaja sloj poslovne logike od prezentacionog i sloja za upravljanje bazom podataka. Njima su dodati mešu slojevi kao što su kontolori ili delovi koda za zajedničku upotrebu.

Najvažniji deo koji čini jedan WCF servis je njegov *Web.config*. Sva važna podešavanja koja određuju njegovo ponašanje kao što su protokoli, povezivanja, autentifikacije, maksimalni broj karaktera u poruci, tip poruke kojim se komunicira sa veb servisom, vreme čekanja pre prekida veze – *timeout.*

Konfiguracioni fajl za ovaj rad ima sledeći izgled:

```
   <system.serviceModel>
     <bindings>
       <basicHttpBinding>
   <binding messageEncoding="Text"
   maxReceivedMessageSize="2147483647"
   maxBufferPoolSize="2147483647"
   openTimeout="00:10:00"
   closeTimeout="00:10:00"
   sendTimeout="00:10:00"
   receiveTimeout="00:10:00">

   <readerQuotas maxDepth="2147483647"
   maxStringContentLength="2147483647"
   maxArrayLength="2147483647"
   maxBytesPerRead="2147483647"
   maxNameTableCharCount="2147483647"/>
   <security mode="None"/>
    </binding>
      </basicHttpBinding>
     </bindings>
   <behaviors>
        <serviceBehaviors>
   <behavior>
     <serviceMetadata httpGetEnabled="true"/>
       <dataContractSerializer
maxItemsInObjectGraph="2147483647"/>
   </behavior>
   </serviceBehaviors>
   </behaviors>
   <services>
   <service behaviorConfiguration=""
name="ImageServiceCloudWCF.ImageServiceCloudWCF">
   <endpoint address=""
   binding="basicHttpBinding"
```

```
   bindingConfiguration=""
contract="ImageServiceCloudWCF.IImageServiceCloudWCF">
         <identity>
          <dns value="localhost"/>
         </identity>
       </endpoint>
     </service>
      </services>
     <serviceHostingEnvironment
multipleSiteBindingsEnabled="true"/>
   </system.serviceModel>
```

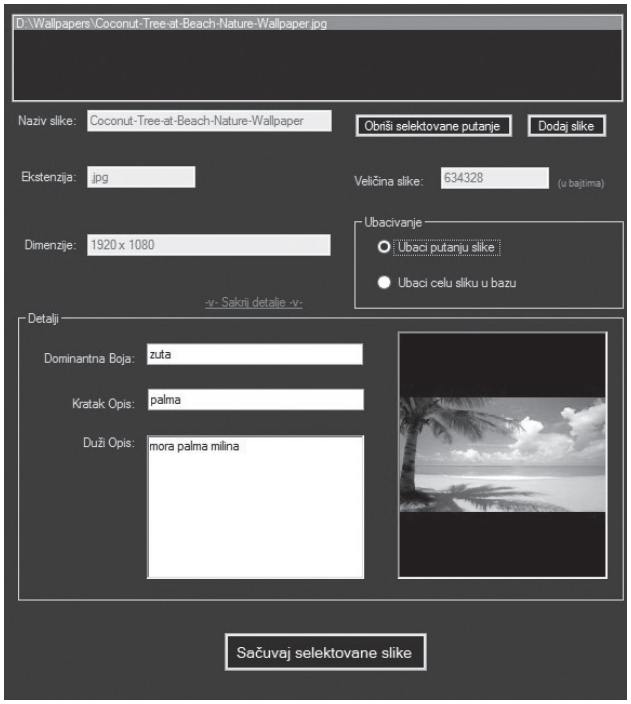
U ovom veb servisu se koristi obično *HTTP* povezivanje i svi limiti za veličinu poruka koje prima veb servis su prebačene na maksimum što je `2147483647` ili 2^{32}.

Pomenuto je da se kao sistem za upravljanje bazama podataka koristi *SQL Server 2014*. A struktura tabela unutar baze ima sledeći izgled:

Jedna od stvari koja se vidi je da se sama fotografija čuva kao string odnosno kao niz karktera. Upravo zbog ubrzavanja prebacivanja podataka sa servera na aplikaciju da se ne bi trošilo vreme na konvertovanje u niz bajtova. Na samom serveru nije potrebno da se vidi sadržaj fotografije već služi kao *storage* za prenošenje sa jedne aplikacije na drugu i nazad.

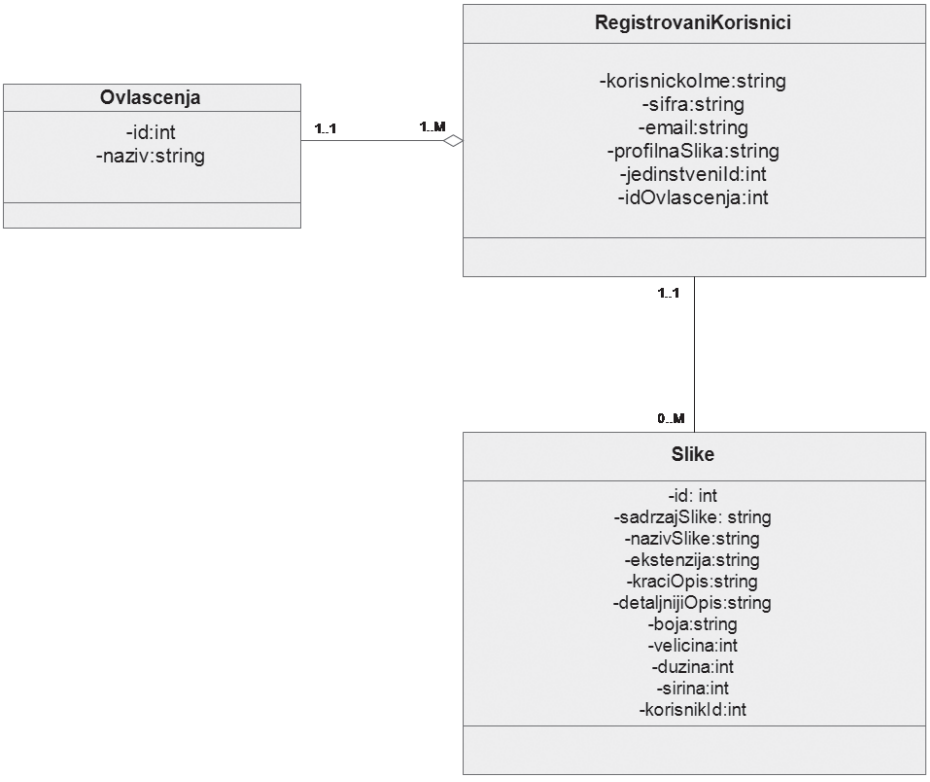
U nastavku biće prikazan i primer kako izgleda jedan proces ubacivanja fotografije i pridruženih podataka iz *Windows Form* aplikacije i njeno pretraživanje u *Android* aplikaciji istog momenta. Aplikacje se nalaze na različitim platformama i napisane su u različitim programskim jezicima. Jedna u *C#* a druga u *Java* programskom jeziku.

Prvo je potrebno da otvorimo formu za unos podataka za fotografiju:



Slika 5 - *Forma za unos podataka o fotografijama - Windows Form aplikacija*

Kako i *Windows Form* aplikacija ima svoju lokalnu *SQL* bazu u koju se ubacuju fotografije ili putanje fotografija sa pridruženim podacima prvo je potrebno da se odradi taj proces pa onda da se pošalje na server:



Slika 4 – *Izgled i relacije tabela u bazi podataka*

Slika 6 - *Forma za prikaz i upload fotografija na cloud - Windows Form aplikacija*

Nakon nadjene fotografije u lokalnoj bazi po ključnim rečima klikom na obeleženo dugme se otvara forma za logovanje gde se unese korisničko ime i šifra i onda se fotografija automatski pošalje na server sa već pridruženim podacima iz lokalne baze.

Istog trenutka je moguće na Androidu sa istim korisnikom otići i pretražiti fotografiju kao što je prikazano:

Slika 7,8 i 9 – Izgled aktivnosti za pretragu i prikaz fotografija i pridruženih podataka u Android aplikaciji

Deo koda u Android aplikaciji koji čita fotografiju kao tekst i pretvara je u fotografiju je sledeći:

```
String base64StringImage = imagesNodes.item(0).
getAttributes().getNamedItem("content").getNodeValue();
byte[] data = Base64.decode(base64StringImage,
Base64.DEFAULT);
Bitmap img = BitmapFactory.decodeByteArray(data, 0,
data.length);
ivFullScreenImage.setImageBitmap(img);
```

U *Rich Client* aplikciji je kod za slanje sadržaja fotografije je sledeći i vrlo je jednostavan:

```
public override string GetImageContent()
        {
            return Convert.ToBase64String(content);
        }
```

5. ZAKLJUČAK

Projekat je rešio jedan od najvećih problema koji postoji u današnjem moru različitih tehnologija a to je interoperabilnost. Na jednostavan način je moguće bilo koju aplikaciju, kako novokreiranu ili postojeću, prilagoditi da komunicira sa drugim aplikacijama na različitim platformama i preko njih šalje i prima fotografije. Samo je potrebno ispratiti strukturu koju nalaže veb servis i komunikacija i uspostavljanje veze i komunikacija sa Cloud-om je moguća.

Ovaj projekat nije najbrže rešenje niti najjednostavnije rešenje ali je rešenje koje rešava problem različitih programskih jezika, operativnih sistema i platformi. Dakle trenutno postoje sistemi koji su potpuno operativni i koji su zasnovani samo na jednoj tehnologiji ili rade samo sa određenim platformama a rade sa fotografijama ali je relativno mali broj koji komuniciraju sa svim heterogenim okruženjima koji podržavaju prikazivanje multimedijalnog sadržaja.

Dalji razvoj ovog sistema je moguć u više pravaca:

- Promena načina skladištenja podataka odnosno fotografija u neku bržu i bolje prilagođenu bazu podataka za ovakve tipove fajlova.
- Uvođenje OCR-a (*Optical Character Recognition*).
- Dodavanje drugih tipova dokumenata za skladištenje.
- Povezivanje na sistem servera koji će smanjiti vreme obrade zahteva od i ka serveru.
- Dodavanje drugih protokola za razmenu podataka.

Vreme će pokazati da su ovakvi sistemi potrebni i već sada se naslućuje njihova mogućnost u ideji da se od svih uređaja napravi jedna velika globalna komunikaciona mreža. Da bi takva mreža uspela potrebno je da se različiti multimedijalni formati lako i jednostavno prenose bez obzira na programski jezik, platformu ili operativni sistem na kome se izvršava. Ovaj rad pokazuje da je moguće napraviti takav sistem i da je taj sistem stabilan i lako prilagodljiv na sve postojeće i novonastajuće aplikacije a to je glavni cilj ovog rada.

6. LITERATURA

[1] **Pro WCF 4, Practical Microsoft SOA Implementation – Second Edition**; Nishith Pathak; *Apress*; 2011

[2] ***Programming WCF Services***; Juval Löwy; *O'Reilly Media*;2010

[3] **Windows® Communication Foundation 4 Step by Step**; John Sharp; *O'Reilly Media & Microsoft Corporation*; 2010

[4] **Essential Windows Communication Foundation**; Steve Resnick, Richard Crane, Chris Bowen; *Addison-Wesley*

[5] **Windows Communication Foundation 3.5 Unleashed**; Craig McMurtry, Marc Mercuri, Nigel Watling, Matt Winkler; *Sams*;2010

[6] **Professional WCF Programming .NET Development with the Windows® Communication Foundation**, Scott Klein; *Wiley Publishing, Inc.*;2007

[7] **Professional Android Open Accessory Programming with Arduino**; Andreas Göransson, David Cuartielles Ruiz; *John Wiley & Sons, Inc.;*2013

[8] **Android Programming - The Big Nerd Ranch Guide**; Bill Phillips , Brian Hardy; *Big Nerd Ranch, Inc.*; 2013

[9] **Beginning Android 4 Application Development**; Wei-Meng Lee; *John Wiley & Sons, Inc.;*2012

[10] **Building Android Apps with HTML, CSS, and JavaScript**; Jonathan Stark; J*onathan Stark*; 2010

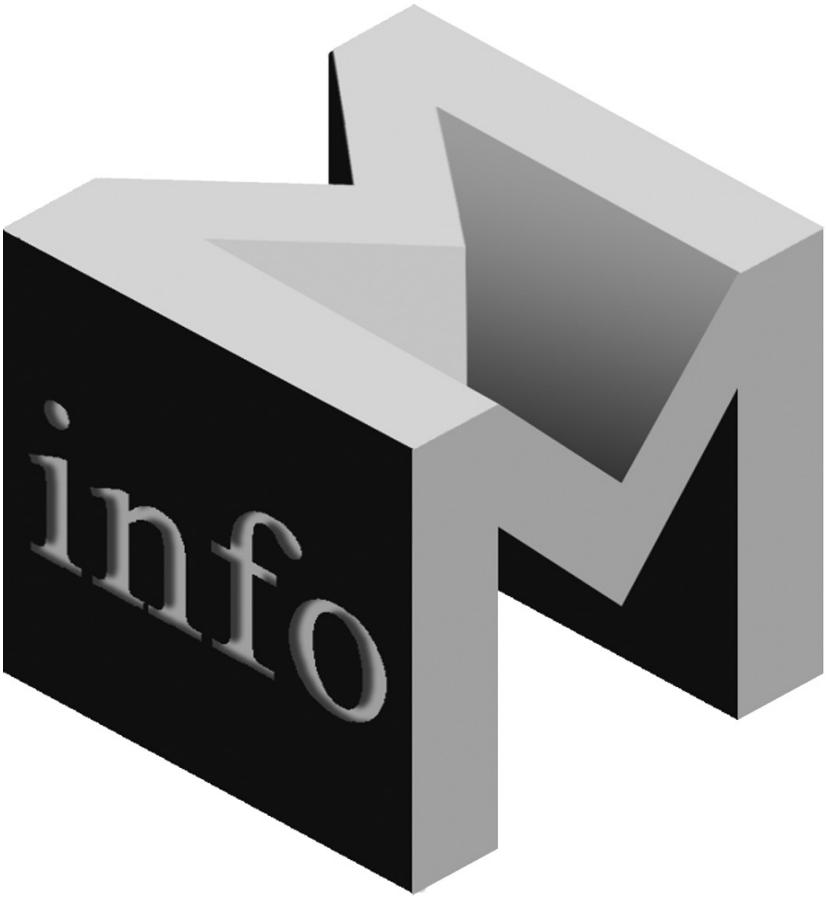
[11] **Razvoj aplikacija za operativni system Android,** *FON***,** Beograd, mart 2012

[12] **Java for Android Development- Third Edition;** Jeff Friesen; Apress; 2013

[13] **Java & XML, Second Edition;** Brett McLaughlin; *O'Reilly Media*; 2001

Vukašin Davidović, master inženjer, Fakultet organizacionih nauka.
Kontakt: vukasin.davidovic@deltaholding.rs
Oblasti interesovanja: softversko inženjerstvo, mobilne tehnologije, cloud sistemi, interoperabilna razvojna okruženja, .NET platforma, Web tehnologije

dr Saša D. Lazarević, Fakultet organizacionih nauka, Univerzitet u Beogradu.
Kontakt: slazar@fon.rs
Oblasti interesovanja: softversko inženjerstvo, informacioni sistemi, baze podataka, sistemi za upravljanje dokumentacijom, .NET platform



CIP – Каталогизација у публикацији Народна библиотека Србије, Београд 659.25

INFO M : časopis za informacionu tehnologiju i multimedijalne sisteme = journal of information technology and multimedia systems / glavni i odgovorni urednik Dejan Simić.

– Štampano izd. – God. 1, br. 1 (2002) – . – Beograd : Fakultet organizacionih nauka, 2002 – (Stara Pazova : SAVPO). – 30 cm

Tromesečno. – Je nastavak: Info Science = ISSN 1450-6254. – Drugo izdanje na drugom medijumu: Info M (CD-ROM izd.) = ISSN 1451-4435

ISSN 1451-4397 = Info M (Štampano izd.) COBISS.SR-ID 105690636