INTEGRACIJA WINDOWS PHONE7 APLIKACIJA S AZURE OBLAKOM PUTEM CSLA.NET POSLOVNE LOGIKE

INTEGRATION OF WINDOWS PHONE 7 APPLICATION WITH AZURE CLOUD USING CSLA.NET BUSINESS LOGIC

mag.inf. Zlatko Stapić, Matija Besednik, Ivan Curić, Kristijan Kralj, Kristina Samošćanec

SAŽETAK:

Funkcionalnost mobilnih aplikacija koje su namijenjene korisniku, kao i funkcionalnost poslovnih mobilnih aplikacija nove generacije u velikoj se mjeri temelji na korištenju resursa i servisa dostupnih na Internetu. Osobito je zanimljivo vidjeti kako je razvoj aplikacija koje su podržane oblakom (engl. cloud) također sve više prisutan u perspektivi mobilnih aplikacija. Ipak, jedna specifična domena u ovom području ostaje značajno nedorađena, a riječ je o arhitekturalnom izdvajanju sloja poslovne logike, te primjeni takvog pristupa u spomenutoj integraciji. Ovaj rad, kao jedan od prvih u struci, prikazuje bitne elemente integracije Windows Phone 7 aplikacija sa oblakom primjenjujući izdvojeni sloj poslovne logike implementirane u CSLA.Net okviru.

ABSTRACT:

The functionalities of mobile applications that are intended to end users and mobile business applications of the new generation are largely based on the use of resources and services available on the Internet. It is particularly interesting to see how is the development of applications that are supported by a cloud also increasingly present in the perspective of mobile applications. However, a specific domain in this area remains a significantly unspecified and unexplored, and it is about the architectural separation of business logic layer aligned with mentioned integration. This work, as one of the first in the field, shows the essential elements of the integration of Windows Phone 7 applications and the cloud backend infrastructure using separated business logic implemented in the CSLA.Net framework.

1. UVOD

Pojavom pametnih telefona (*engl. smartphone*) i zamahom koji su doživjeli u posljednjih nekoliko godina, razvoj aplikacija za mobilne uređaje je poprimio jednu posve novu dimenziju. Pametni telefoni su postali dostupni širokom broju korisnika čime se fokus svih vodećih proizvođača aplikacija za mobilne uređaje počeo pomicati od razvoja poslovnih aplikacija prema razvoju aplikacija namijenjenih običnim korisnicima i mladima. Aplikacije „zabavnog sadržaja“ kako ih često volimo zvati imaju naglasak na korisničkom sučelju i korisničkom doživljaju (*engl. user experience*) te posebno na stalnom osvježavanju sadržaja koje se najčešće realizira stalnom vezom prema Internetu.

Budući da s jedne strane, korisnici spomenutih aplikacija postaju sve zahtjevniji po pitanju kvalitete, performansi i funkcionalnosti istih, a s druge strane, korištenje pametnih telefona ne isključuje nego potiče razvoj i poslovnih aplikacija, s pravom možemo zaključiti da je već sada potrebno tražiti nove modele iskorištavanja mogućnosti koje nude pametni telefoni, ali i brzi pristup internetu.

Jedan od takvih modela je svakako i primjena računarstva u oblaku (*engl. cloud computing*) i razvoj mobilnih aplikacija podržanih ovom novom i prodornom tehnologijom.

Iako je primjena oblaka kao aplikacijskih sustava već u punom zamahu, izuzetno je zanimljivo primijetiti kako većina postojećih mobilnih aplikacija i aplikacija koje su integrirane u oblaku nema arhitekturalno izdvojen sloj poslovne logike, što naravno rezultira smanjenom modularnošću i fleksibilnošću dorade i održavanja istih.

Jedan od rijetkih okvira za razvoj arhitekturalno izdvojenog sloja poslovne logike od slojeva korisničkog sučelja i baze podataka je CSLA.Net okvir razvijen od [3]. Primjena „*komponentno bazirane i scalabilne logičke arhitekture*“ (*engl. Component-based Scalable Logical Architecutre*) znači primjenu razvojnog okvira koji pomaže razvoju snažnog i održivog sloja poslovne logike za različite programske proizvode, uključujući i mobilne aplikacije [3]. Ipak, pretraga istraživanja na temu primjene CSLA.Net okvira i infrastrukture oblaka u jednoj aplikaciji nije dala rezultata.

Stoga, ovaj rad prikazuje integraciju WP7 aplikacije i Azure oblaka, ali primjenom CSLA.Net poslovne logike. Sloj poslovne logike je dizajniran na način da iz oblaka iskoristi sve prednosti, uključujući znatne potrebne resurse za obradu nad složenim podatkovnim strukturama, te velikom količinom informacija. S druge strane, spomenuti sloj poslovne logike, podatke pakira i dostavlja aplikaciji na WP7 uređaju u obliku prikladnom za jednostavan i brz prikaz.

Nakon uvodnog razmatranja, ovaj rad kratko predstavlja Windows Phone 7 platformu, prikazuje detaljno arhitekturu mobilne aplikacije sa izdvojenim slojem poslovne logike, diskutira o primjeni oblaka u razvoju mobilnih aplikacija, te uvodi čitatelja u osnovne koncepte CSLA.Net okvira.

Konačno, rad prikazuje bitne elemente integracije ovakvog sloja poslovne logike s jedne strane na WP7 Silverlight aplikaciju, a s druge strane na infrastrukturu web servisa smještenu u oblaku.

U zaključku autori sumiraju predstavljene koncepte te kratko diskutiraju o prednostima ali i nedostacima ovakve integracije.

Rad će svakako biti interesantan razvojnim inženjerima koji se intenzivno bave razvojem višeslojnih aplikacija, a osobito razvojem višeslojnih mobilnih aplikacija, budući da je poseban fokus dat upravo na kontekst razvoja za mobilne uređaje.

2. WINDOWS PHONE 7

Windows Phone 7 je novi operacijski sustav za pametne mobilne uređaje razvijen od strane Microsofta. Da bi proizvođači mobilnih uređaja mogli koristiti ovaj OS, Microsoft je u svrhu podizanja razine korisničkog iskustva specificirao i minimalne karakteristike koje svaki uređaj mora imati [4]. Time je Microsoft lansirajući operacijski sustav WP7 na tržište, stvorio temelje za priključak razvoju aplikacija nove generacije. Naravno, riječ je o aplikacijama koje su prvenstveno namijenjene korisniku, a koje su kako brojna istraživanja pokazuju pojavom pametnih telefona postale izuzetno popularne. Brojne značajke Windows Phone 7 uređaja, kao što su obavještavanje korisnika putem iniciranih obavijesti (*engl. push notification*), olakšana navigacija aplikacijom korištenjem panoramnih ili stupčanih stranica (*engl. panorama & pivot page*), podrška za Bing mape, pamćenje podataka u izoliranom spremniku aplikacije (*engl. isolated storage*) te posebni alat za dizajn sučelja (Microsoft Expression Blend), omogućuju stvaranje bogatog korisničkog iskustva.

2.1 Arhitektura WP7 aplikacije koja sadrži poslovnu logiku

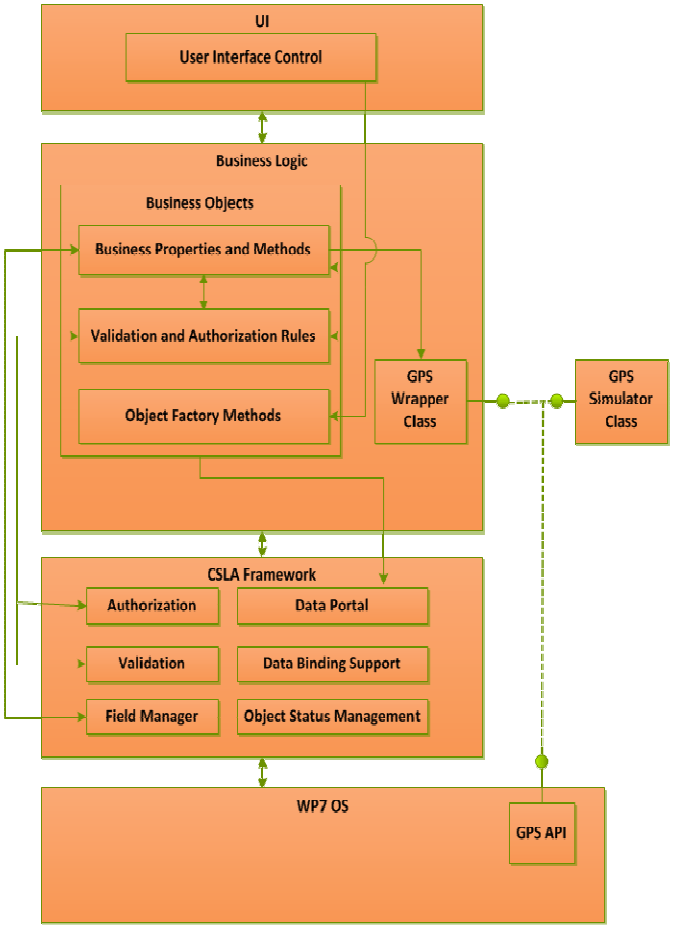
Kako se operativni sustav Windows Phone 7 uređaja temelji na Silverlight 3 platformi, tako pruža iznimno dobru podlogu za razvijanje aplikacije korištenjem višeslojne arhitekture (slika 1). Taj pristup nam omogućuje razdvajanje prezentacijskog dijela aplikacije (korisničko sučelje koje sadrži sve nabrojane WP7 značajke) i dijela odgovornog za poslovnu logiku aplikacije što je ključno kada želimo imati mogućnost nadograđivanja i prilagođavanja korisničkog sučelja.

Sloj poslovne logike sadrži implementaciju poslovnih objekata korištenjem CSLA.NET razvojnog okvira koji implementira njegova svojstva i metode, validaciju i autorizacijska pravila i koristi CSLA WPF (*engl. Windows Presentation Foundation*) infrastrukturu koja komunicira sa WPF web servisom poslovne logike (*engl. Business Logic Web Service*). To znači da se dizajneri korisničkog sučelja (*engl. UI developers*) mogu fokusirati na kreiranje korisničkog sučelja koji će sadržavati bogato korisničko iskustvo povezujući objekte sučelja sa poslovnom logikom. Ostatak programiranja, dakle razvoj poslovne logike, je prepušten programerima.

Poslovna logika, oslanjajući se na CSLA razvojni okvir, obavlja poslove:

- validacije korisničkog unosa,
- kreiranje izvještaja o prekršenim pravilima sučelju aplikacije,
- autoriziranje korisnika da mogu pristupiti podacima,
- komuniciranje sa servisom
- ...

Ovakva arhitektura nam dopušta da gradimo visoko modularne sustave koji omogućavaju kreiranje dodataka (*engl. plugins*) za Windows Phone 7 aplikaciju. Ako Windows Phone 7 aplikacija koristi lokaciju kao jedan od elemenata, potrebno je u razvojno okruženje dodati klasu koja će simulirati trenutnu lokaciju uređaja i koja će biti samo tijekom razvoja. Tada se kao dodatak infrastrukturi koristi prilagođena GPS klasa za objedinjavanje (engl. *Global Positioning System Wrapper Class*) i GPS simulator da bi simulirali kompleksno kretanje korisnika i scenarije korisničke interakcije. Naravno, ove zadnje dvije spomenute klase se koriste samo kod razvoja sustava, dok u samom produkcijskom okruženju podaci koje dobivamo od njih se zamjenjuju onim pravim, dobivenim od samog Windows Phone 7 uređaja.



Slika 1. Arhitektura WP7 aplikacije integrirane s CSLA poslovnom logikom

2.2 Povezanost WP7 i oblaka

Krajem siječnja 2011. godine Microsoft Research su izdali alat za razvoj softvera (*engl. SDK – Software Development Kit*) pod nazivom „Windows Phone 7 + Cloud Services SDK“ koji će služiti za objedinjavanje dvije Microsoftove nove tehnologije: Windows Phone 7 i Windows Azure [6]. Konkretno, ovaj SDK omogućuje kreiranje Windows Phone 7 aplikacija koje koriste servise koji se pokreću na oblaku (*engl. cloud*). Na ovaj način mogu se razvijati aplikacije koje će biti proširene dodatnim servisima koji se obavljaju na oblaku, a WP7 uređaj će postati dijelom terminal. Naime, pri izvršavanju aplikacije koja ima podršku oblaka, korisnik može slati složene upite servisima na oblaku. Ako u tu priču ubacimo i podršku za višezadaćnost (*engl. multitasking*), koja je najavljena kao jedna od ključnih funkcionalnosti u jednom od slijedećih izdanja WP7 OS-a, dolazimo do platforme koja će zaista u pozadini slati upite, dok korisnik pritom obavlja neke druge operacije. Na taj način, sve operacije obrade upita koje bi se odvijale na WP7 uređaju i trošile procesorsku snagu i bateriju,

odvijaju se na oblaku, a korisniku se zatim prezentiraju rezultati dobiveni upitom.

Također treba istaknuti da spomenuti SDK služi kao potpora Projektu Hawaii, studentske inicijative za istraživanje kako se servisi koji se zasnivaju na oblaku mogu koristiti da bi poboljšali iskustvo korištenja Windows Phone 7 uređaja. Njihova trenutna platforma se sastoji od Windows Phone 7 pametnog mobilnog uređaja i nekoliko servisa na oblaku, kao što su Windows Azure koji služi za proračune i pohranu podataka, Bing mape za servis s mapama i Windows Live ID za identifikaciju korisnika [5].

3. PRIMJENA OBLAKA

Sam pojam računalstvo u oblaku (*engl. cloud computing*) definirao je Nacionalni Institut za standarde i tehnologiju (NIST) [2]. Definiran je kao model koji omogućava pristup mreži na zahtjev korisnika, točnije pristup računalnim resursima poput pristupa serverima, aplikacijama i servisima. Sama riječ oblak se koristi kao metafora za Internet. Upotrebom oblaka smanjena je potreba za kupnjom novog sklopovlja i programa, a pristup Internetu je preduvjet jer se aplikacijama i dokumentima pristupa putem web preglednika.

Osim toga, NIST je definirao pet ključnih karakteristika računalstva u oblaku [2]: *usluge na zahtjev korisnika, širok pristup mreži, udruživanje resursa, elastičnost i servisi koji se mogu mjeriti*. Širok mrežni pristup znači da se mogućnosti koje su dostupne putem mreže mogu koristiti uporabom različitih platformi, a te se platforme odnose na prijenosna računala, mobilne uređaje i slično. Sljedeća karakteristika, brza elastičnost koja označava korištenje usluga od strane klijenata kojemu korištenje tih usluga može izgledati kao da ne postoji nikakvo ograničenje. Ipak, korištene usluge su mjerljive o čemu govori sljedeća karakteristika, odnosno mjerljiva usluga. Klijent može koristiti usluge oblaka kada želi, a jedino što mu je potrebno jest pristup Internetu. Osim toga, resursi se udružuju kako bi se dinamički dodijelili ili uklonili prema zahtjevima korisnika.

Što se tiče računalstva u oblaku, razlikujemo implementacije oblaka i servisne modele oblaka. Oblak može biti implementiran na sljedeće načine [2]:

- *Privatni oblak* – realizacija oblaka koji organizacija ima za sebe. Može biti kupljen od treće strane ili ga organizacija može sama uspostaviti.
- *Zajednički oblak* – infrastruktura oblaka koju dijeli par organizacija, te koje imaju zajedničku misiju, viziju, sigurnosne zahtjeve, politiku i sl.
- *Javni oblak* – oblak koji je u posjedu organizacije koja se bavi prodajom usluga oblaka, te njega koristi šira javnost ili pak velika industrijska grupa
- *Hibridni oblak* – oblak koji je nastao kao posljedica združivanja dva ili više oblaka, bilo da se radi o privatnom, zajedničkom ili javnom oblaku.

Promatrajući danu definiciju oblaka, te karakteristike računalstva u oblaku, možemo zaključiti da su svi preduvjeti korištenja oblaka od strane aplikacija razvijenih za mobilne uređaje ispunjeni. Također, nije važno na koji način je oblak implementiran, budući da sve navedene implementacije podrazumijevaju pristup oblaku sa udaljenih računala, što svakako uključuje i mobilne uređaje.

S druge strane, postoje različiti modeli pružanja usluga u oblaku. Tako prema [2] razlikujemo:

- Softver kao servis (engl. SaaS – Software as a Service)
- Platforma kao servis (engl. PaaS – Platform as a Service)
- Infrastruktura kao servis (engl. Iaas – Infrastructure as a Service)

Ako spomenute modele stavimo u kontekst mobilnih aplikacija, tada možemo zaključiti da već sada, bez posebnih i značajnih dorada, oblake možemo koristiti i na mobilnim uređajima u obliku SaaS modela, za što je potrebno samo korištenje web preglednika sa određenim specifičnim mogućnostima. S druge strane, primjena oblaka u obliku PaaS i IaaS modela na mobilnim uređajima još uvijek nije dovoljno zaživjela, a ovaj rad ima za cilj prikazati jedno od mogućih načina primjene oblaka na mobilnim uređajima u obliku PaaS i IaaS modela.

Naravno, treba uzeti u obzir i činjenicu da je korištenje oblaka kao platforme ili infrastrukture uvjetovano i plaćanjem. Kada se usporede troškovi uporabe oblaka i troškovi proizašli iz korištenja vlastite infrastrukture možemo reći da su troškovi korištenja oblaka vrlo mali [7], što prikazuje i tablica 1. Iako postoji više organizacija koje pružaju usluge oblaka, mi ćemo prikazati cijene Microsoftovih usluga budući da smo se odlučili za usluge koje pruža Microsoft [7].

Tablica 1. Cijene Microsoftovih usluga računalstva u oblaku

Resurs	Jedinica	Cijena u kn
Premještanje podataka na poslužitelj	GB	0,52
Premještanje podataka s poslužitelja	GB	0,73
Vrijeme CPU	Sati rada CPU	0,65
Pohrana podataka	GB mjesečno	0,78

Promatrajući ekonomski aspekt primjene oblaka u svijetu mobilnih aplikacija, te uzimajući u obzir i istraživanja provedena na ovu temu zaključujemo da postoji veliki broj poslovnih scenarija u kojima primjena oblaka umjesto vlastite infrastrukture znatno smanjuje troškove produkcije aplikacije, te ubrzavanja procesa prilagodbe mogućem velikom broju potencijalnih korisnika.

Zaključujući razmatranje o primjeni oblaka u kontekstu mobilnih aplikacija, navest ćemo da postoji velika opravdanost istog, te da na novim platformama kao što je WP7 još uvijek ima dosta nedorečenosti u načinu integracije aplikacija s oblakom, osobito u scenariju smještanja podataka i poslovne logike na oblak, a zadržavanja sloja korisničkog sučelja i same aplikacije na mobilnom uređaju.

4. CSLA.NET OKVIR

CSLA.NET okvir služi lakšem i bržem razvoju poslovnog sloja aplikacije. Razlog razvoju ovakvog okvira leži u tome što Microsoft .NET sadrži obilje alata i tehnologija za izradu korisničkog sučelja te pristupa podacima, ali jako malo tehnologija za izradu ključnog sloja modularne ili distribuirane aplikacije: sloja poslovne logike.

Razvoj poslovnog okvira je iznimno skup i dugotrajan proces, te se pri izradi velikog broja aplikacija takva arhitektura izbjegava te se aplikacije razvijaju vezanjem izvora podataka direktno na kontrole korisničkog sučelja, ili na poslovni okvir iznimno male funkcionalnosti.

Takva arhitektura sustava pruža malo mogućnosti za ponovnu upotrebu koda te otežava daljnji razvoj i održavanje aplikacije. Još veći problemi javljaju se kod razvoja nove aplikacije, na drugačijoj platformi, gdje se poslovna pravila moraju nanovo implementirati.

CSLA.NET okvir pruža sljedeće funkcionalnosti koje je bez njega potrebno posebno implementirati [3]:

- Validacijska pravila implementiraju se unutar samih poslovnih objekata
- Poslovni objekti automatski održavaju listu prekršenih pravila
- Poslovni objekti prate svoje stanje (da li su mijenjani)
- Autorizacijska pravila implementiraju se unutar objekata te se definiraju na razini objekta i na razini atributa
- Uređeni odnosi agregacije i kompozicije među objektima
- Mogućnost vraćanja stanja objekta uključujući objekte koje sadrži sam objekt čije se stanje vraća
- Jednostavan i apstraktan model za programera korisničkog sučelja
- Puna podrška za povezivanje podataka (*engl. Data Binding*) u WPF, Windows Forme, Web Forme, Silverlight i Windows Phone 7 sučelja
- Čitanje i spremanje podataka u izvor podataka
- Vlastiti sustav autentifikacije

Bitno je napomenuti kako poslovni objekti razvijeni CSLA.NET okvirom zadovoljavaju osnovne koncepte objektno orijentiranog programiranja: *apstrakciju, začahurivanje, višeobličnost te nasljeđivanje*.

Jedan od najvećih problema kod razvoja suvremenih višeslojnih aplikacija je upravo problem razdvajanja prezentacijskog sloja i sloja poslovne logike. Naime, većinom je riječ o web aplikacijama ili o aplikacijama baziranim na servisno-orijentiranim arhitekturama (*engl. SOA – Service Oriented Arhitecture*), u kojima se podaci koje korisnik unosi obrađuju na udaljenom aplikacijskom serveru. Problem leži u tome što, kod suvremenih aplikacija, korisnici žele interaktivno sučelje koje će ih odmah izvijestiti o pogreškama u unosu ili prezentirati podatke vezane za njihov unos. Na primjer, kada se korisnik registrira za korištenje naših web stranica, želi vidjeti da li je korisničko ime koje je odabrao već zauzeto ili kada unese željenu lozinku, želi dobiti informaciju o tome koliko je lozinka jaka i da li zadovoljava sigurnosnu politiku našeg sustava (da li zadovoljava postavljene uvjete). Iz tog razloga provjeru korisničkog unosa je potrebno obaviti na strani korisnika. Nadalje, provjeravamo li te podatke isključivo na strani korisnika dobivamo veliki sigurnosni problem: ne možemo garantirati da su podaci koje je server zaprimio valjani te je provjeru valjanosti i vjerodostojnosti potrebno obaviti i na serverskoj strani.

CSLA.NET okvir taj problem rješava korištenjem koncepta mobilnih objekata. Isti objekt koristi se i u klijentskoj aplikaciji i na serveru. Naime, u komunikaciji između klijentske i serverske aplikacije objekti se serijaliziraju (binarno kodiraju) te se takvi šalju mrežom. Na taj način se uz prijenos samih podataka prenosi i ponašanje samog objekta, njegov globalni kontekst, te podaci o korisniku koji je uputio zahtjev. CSLA.NET okvir sadrži gotovu infrastrukturu za jednostavnu implementaciju i konfiguriranje servisa u .NET-u.

Najveća prednost CSLA.NET okvira leži u tome što je razvijen tako da podržava korištenje u kombinaciji sa generatorima koda. Za razne tipove objekata poslovnog okvira, te za njihova polja i metode, razvijeni su predlošci za generatore koda. Njihovim korištenjem drastično se smanjuje količina vremena potrebnog za razvoj sloja poslovne logike.

4.1 Način integracije tehnologija

CSLA.NET okvir, uz osnovnu .NET biblioteku, sadrži i biblioteke za Silverlight, WPF i Windows Phone 7. Prostor imena (*engl. namespace*) jednak je bez obzira na platformu, odnosno biblioteku, koja se koristi kod razvoja poslovne logike. Time je dodatno olakšan razvoj jedinstvenog sloja poslovne logike za aplikacije koje koriste različite tehnologije. Naime, svaka od platformi (Silverlight, ASP.NET i Windows Phone 7) traži korištenje biblioteke prilagođene toj platformi.

Taj problem rješava se tako da se za svaku platformu kreira biblioteka. Jedna od platformi se odabere kao izvorišna (najčešće .NET platforma) te se za nju generiraju same klase. U ostale biblioteke klase, odnosno datoteke,se dodaju kao veze.

Način implementacije nekih detalja te potrebna funkcionalnost objekta razlikuju se među platformama. Na primjer, dio koda za pristup bazi podataka nije potreban u klijentskim aplikacijama već je potreban na aplikacijskom serveru. Taj dio možemo riješiti korištenjem pretprocesorskih naredbi (na primjer `#if` ASP), ili pisanjem klasa kao djelomičnih (*engl. partial*) i odvajanjem specifičnosti u zasebne datoteke.

4.2 Specifičnosti implementacije za Windows Phone 7 platformu

Prva osobina koju svaki poslovni objekt u Windows Phone 7 okruženju mora imati je javni (*engl. public*) konstruktor bez parametara. Razlog je korištenje refleksije (generičko pozivanje metoda i instanciranje objekata) kod implementacije CSLA.NET okvira. Naime, komunikacija sa serverom se odvija preko četiri osnovne metode CSLA.NET WcfPortal servisa: `Create, Fetch, Update i Delete`. To su jedine metode koje servis sadrži i generalizirane su za sve objekte koji su implementirani korištenjem CSLA.NET okvira, te se nalaze u poslovnom okviru. Kada se zaprimi zahtjev na serveru, ili odgovor servera u klijentskoj aplikaciji, potrebno je instancirati objekt određenog tipa koji je poslan u zahtjevu, odnosno odgovoru. To se obavlja refleksijom. Nažalost Silverlight ne podržava instanciranje objekata koji nemaju javni konstruktor bez parametara. Kako se Windows Phone 7 okruženje temelji na Silverlight-u, potrebno je implementirati javni konstruktor bez parametara te ga je preporučljivo „sakriti“ u editoru kako ga programer korisničkog sučelja ne bi mogao koristiti (za instanciranje objekata koristi se `Create()` metoda):

```
[System.ComponentModel.EditorBrowsable(System
.ComponentModel.EditorBrowsableState.Never)]
public MojaKlasa()
{

}
```

Komunikacija sa servisom u CSLA.NET okviru odvija se preko generičke klase `DataPortal<T>` koja je implementirana u CSLA.NET okviru. T označava tip poslovnog objekta s kojeg se šalje. Klasa `DataPortal` odgovorna je za komunikaciju sa WcfPortal servisom spomenutim ranije te se pozivom jedne od njenih metoda odvijaju sve potrebne radnje kako bi se objekt serijalizirao, poslao, deserijalizirao na serveru, pozvala

odgovarajuća metoda, te se na isti način poslao s rezultatom nazad klijentu.

Drugi uvjet koji poslovni objekti moraju zadovoljavati jest da se ranije spomenute metode pozivaju asinkrono. Naime, sinkroni poziv zaključava dretvu dok se ne zaprimi odgovor, dok se kod asinkronog poziva dretva nastavlja, a odgovor se obrađuje preko događaja (*engl. event*).

Budući da Silverlight, a posljedično i Windows Phone 7 platforma, ne podržava sinkrone pozive servisa, potrebno je implementirati asinkrone pozive servisa, to jest `DataPortal` metodu:

```
Public static void GetMojaKlasa(Guid id,
EventHandler<DataPortalResult<MojaKlasa>>
callback)
{
  var dp = new DataPortal<MojaKlasa>();
  dp.FetchCompleted += callback;
  dp.BeginFetch(new SingleCriteria
    <MojaKlasa, Guid>(id));
}
```

Kao što je vidljivo u primjeru, instancira se `DataPortal` objekt za specifičnu klasu, `EventHandler` u parametru se poveže na odgovarajući event te se pozove odgovarajuća metoda parametrima koji su potrebni. Kod asinkronog poziva odgovarajuće metode, potrebno je obraditi odgovor servera:

```
MojaKlasa.GetMojaKlasa(Csla.ApplicationContex
t.User.Identity.Name, (o, e1) => {
  if(e1.Error==null && e1.Object!=null)
  {
    mojaKlasa = e1.Object;
    LbMojaLista.ItemsSource = locations;
  }
});
```

U prethodnom primjeru dohvaćamo podatke za određenu klasu prema korisničkom imenu trenutno prijavljenog korisnika. Kao drugi parametar prosljeđujemo lambda izraz u kojem obrađujemo odgovor servera. Provjeravamo da li je upit bio uspješan i u slučaju da jest pridjeljujemo vrijednost rezultata varijabli mojaKlasa tipa MojaKlasa. Zadnji korak je prikazivanje podataka u objektu sučelja LbMojaLista.

Treći uvjet je da se atributi definiraju korištenjem interne strukture podataka CSLA.NET okvira za pohranu podataka, to jest bez korištenja privatnih polja za pohranu. Takav način implementacije atributa je povoljniji za korištenje generatora koda, te zbog ranije opisane refleksije u Silverlightu nije moguće implementirati atribute korištenjem privatnih polja.

```
Public static readonly PropertyInfo<string>
NameProperty=RegisterProperty<string>(c =>
c.Name, RelationshipTypes.None);
public string Name
{
  get
  {
    Return GetProperty(NameProperty);
  }
  set
  {
    SetProperty(NameProperty, value);
  }
}
```

4.3 Specifičnosti implementacije za Azure platformu

Kako je Azure oblak zapravo ASP.NET platforma, implementacija poslovnih objekata nema posebne uvjete već se koriste klasični načini implementacije poslovnih objekata korištenjem CSLA.NET okvira (naravno, kada se radi o integraciji sa Windows Phone 7 platformom, uzimaju se u obzir specifičnosti implementacije za Windows Phone 7 platformu).

Zbog razlika između ASP.NET i Silverlighta, pa tako i Windows Phone 7 platforme, CSLA.NET okvir sadrži dvije implementacije WcfPortal servisa: jednu za .NET i ASP.NET klijente, drugu za Silverlight i Windows Phone 7 klijente. Kako se radi o integraciji Azure oblaka sa Windows Phone 7 platformom, koristi se Silverlight verzija servisa.

Kako bi smo definirali koja klasa unutar CSLA.Net okvira predstavlja kôd servisa, konfigurirat ćemo *WcfPortal.svc* datoteku i podesiti servis na sljedeći način:

```
<%@ ServiceHost language=c# Debug="true" Serv
ice="Csla.Server.Hosts.Silverlight.WcfPortal"
 CodeBehind="Csla.Server.Hosts.Silverlight.Wc
fPortal" %>
```

Isto tako, da bi smo konfigurirali krajnju točku našeg WCF servisa, u *Web.config* datoteci potrebno je napraviti sljedeće bitne promjene i definirati ponašanje web servisa, to jest definirati protokol komuniciranja klijenta i servisa:

1. U element *behaviors* dodajemo definiciju ponašanja servisa u nepredviđenim *(engl. exception)* situacijama:

```
<behavior name="WcfPortalBehavior">
  <serviceMetadata httpGetEnabled=
    "true"/>
  <serviceDebug
    includeExceptionDetailInFaults=
    "true"/>
</behavior>
```

2. U element *services* dodajemo definiciju protokola komuniciranja klijenta i servisa:

```
<service behaviorConfiguration=
  "WcfPortalBehavior" name=
  "Csla.Server.Hosts.Silverlight.
  WcfPortal">
  <endpoint address="" binding=
    "basicHttpBinding" contract=
    "Csla.Server.Hosts.Silverlight.
    IWcfPortal">
    <identity>
      <dns value="localhost"/>
    </identity>
  </endpoint>
  <endpoint address="mex" binding=
    "mexHttpBinding" contract=
    "IMetadataExchange"/>
</service>
```

3. U slučaju da koristimo CSLA.Net sustav autentifikacije u element appSettings također dodajemo odgovarajuću definiciju:

```
<add key="CslaAuthentication"
value="Csla" />
```

Bitno je primijetiti da je sučelje `IWcfPortal` (implementirano u samom CSLA.NET okviru) ugovor (*engl. contract*) za naš servis. Dok varijabla `name` označava ime klase koja pruža kôd u pozadini servisa.

5. ZAKLJUČAK

Windows Phone 7 je više od samo nove inačice mobilnog operativnog sustava, riječ je o posve novoj platformi koja je fleksibilna, proširiva i s odličnim skupom razvojnih i dizajnerskih alata na raspolaganju. Povezivanje Windows Phone 7 aplikacija i servisa u oblaku je trenutno uobičajena praksa, štoviše takva integracija je i dio arhitekture Windows Phone 7 platforme, jer se iskorištavaju prednosti oblaka što daje aplikaciji bolje performanse i skalabilnost. Oblak se odlično uklapa u strategiju distribuirane platforme i uređaja s jedne strane, no što se događa kada se tu „umiješa“ arhitekturalno izdvojen sloj poslovne logike?

Budući da CSLA implementira poslovnu logiku aplikacije, prednosti i nedostaci koji se odnose općenito na sloj poslovne logike mogu se primijeniti i ovdje. Kao najveću prednost izdvojili bi rasterećenje klijentskog sloja aplikacije, jer CSLA poslovna logika implementira sva poslovna pravila te pristup do baze podataka, te na programeru klijentskog sloja ostaje samo da poziva metode poslovne logike te se brine o kvalitetnom korisničkom sučelju. Ovakav model integracije i pristupa višeslojne aplikacije mogao bi se primijeniti na razvojne timove koji bi imali dizajnere zadužene za cjelokupni klijentski sloj, budući da su rasterećeni programiranja, te mogu iskoristiti sve prednosti Microsoft Expression Blend alata za dizajn korisničkog sučelja.

Poslovna logika može donijeti prednosti kod današnjeg načina upravljanja verzijama aplikacije. Promjena poslovne logike ne znači nužno i promjenu u klijentskom dijelu aplikacije, tako da se promjena poslovne logike može obaviti centralizirano, bez da korisnici aplikacije moraju ažurirati trenutnu verziju aplikacije na Windows Phone 7 uređaju. Ukoliko s vremenom obrada aplikacije postane spora zbog količine podataka koja se razmjenjuje, može se povećati snaga središnjeg računala odnosno broj instanci Windows Azure oblaka. To su prednosti koje pruža oblak tehnologija, koja može riješiti nedostatak velikog opterećenja centralnog poslužitelja, te uvelike smanjiti troškove održavanja.

Poslovna logika implementirana putem CSLA okvira postiže visoke rezultate u održavanju koda. Korištenjem alata *Code Analysis* unutar razvojnog okruženja Visual Studio 2010 dobivaju se izrazito visoki rezultati za održavanje koda CSLA poslovne logike, što potvrđuje kvalitetu samog koncepta [1].

Kao najveći nedostatak ovakve integracije nameće se složeni razvoj poslovne logike na čiji se razvoj mogu utrošiti veliki vremenski i ljudski resursi što naravno generira određene troškove. Naravno sve ovisi o namjeni aplikacije i njenoj kompleksnosti. Razvojni tim treba procijeniti dali ima smisla implementirati poslovnu logiku. Scenariji u kojima bi bilo korisno implementirati poslovnu logiku su aplikacije koje imaju zahtjevne izračune poput genetskih algoritama za izračunavanje optimalnih ruta, zatim aplikacije koje upravljaju interakcijom velikog broja korisnika, te ostalo. Današnje mobilne aplikacije su većinom jednostavne, no razvojem mobilnih platformi, sve većim mogućnostima pametnih telefona, te sve većim zahtjevima njihovih korisnika aplikacije će morati pružati nove mogućnosti, a tu je upravo prilika za iskorištenje mogućnosti CSLA poslovne logike.

Ovaj rad pokazuje da je poštivanjem određenih bitnih elemenata u pristupu integraciji oblaka i Silverlight Windows Phone 7 aplikacije, a koji su navedeni u ovom radu, moguće iskoristiti sve prednosti izdvajanja poslovne logike u poseban arhitekturalni sloj, koji sve zahtjevne i podatkovno intenzivne operacije izvršava u oblaku. Time se mobilni uređaj oslobađa značajnog udjela obrade i pohrane podataka, a isti resursi se mogu iskoristiti u poboljšavanje korisničkog sučelja i podizanje razine korisničkog iskustva.

Literatura:

1 Besednik M., Primjena CSLA.NET okvira pri izradi aplikacija, Završni rad, Fakultet organizacije i informatike, Varaždin, 2010.

2 Grance T., Mell P.: The NIST Definition of Cloud Computing, 2011. Izvor: http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf (učitano 28.4.2011.)

3 Lhotka, R.: Expert C# 2008 BusinessObjects, Apress, New York, 2009.

4 Microsoft Corp., Minimum Specs for Windows Phone 7, 2011. Izvor: http://www.windowsmobile7.com/minimum-specs-for-windows-phone-7 (učitano 29.4.2011.)

5 Microsoft Corp. Research, Project Hawaii, 2011. Izvor: http://research.microsoft.com/en-us/um/redmond/projects/hawaii/ (učitano:28.4.2011.)

6 Microsoft Corp. Research, Windows Phone 7 + Cloud Services SDK, 2011. Izvor: http://research.microsoft.com/en-us/downloads/0c54f42c-84b1-4ad5-a1b3-37008f3b6bff/default.aspx?tag=mantle_skin;content (učitano: 06.05.2011.)

7 Nacionalni CERT: CloudComputing, 2010. Izvor: http://www.cert.hr/sites/default/files/NCERT-PUBDOC-2010-03-293.pdf (učitano: 11.4.2011.)

Podaci o autorima:

Zlatko Stapić, mag. inf.
tel: +385 42 390 853
fax: +385 42 213 413
e-mail: zlatko.stapic@foi.hr

Zlatko Stapić, mag. inf. je od 2006. godine asistent na Katedri za razvoj informacijskih sustava na Fakultetu organizacije i informatike u Varaždinu, te polaznik poslijediplomskog doktorskog studija Informacijske znanosti na istom fakultetu. Njegova nastavna aktivnost je prvenstveno usmjerena na kolegije koji se odnose na programsko inženjerstvo, analizu i razvoj programa, modeliranje poslovnih procesa i razvoj informacijskih sustava, te značajne napore ulaže u radu sa studentima za što je dobio i posebna priznanja.

Iz znanstvenog i stručnog rada treba izdvojiti višegodišnje voditeljstvo stručnih projekata razvoja programskih proizvoda i sudjelovanje na različitim stručnim i znanstvenim projektima iz područja razvoja, unapređenja poslovnih procesa, projektnog menadžmenta i slično.
U posljednje vrijeme se intenzivno bavi razvojem aplikacija za mobilne uređaje, što je i predmet njegovog istraživanja u okviru doktorske disertacije, a osobito je vrijedno istaknuti da razvija za skoro sve platforme, uključujući između ostalog Android, Symbian te Windows Phone 7. Zlatkov detaljniji životopis, s popisom svih radova, projekata i nagrada, te drugih važnih podataka može se pronaći na njegovoj osobnoj web stranici, na http://www.foi.hr/djelatnici/zlatko.stapic.

Zlatko Stapić is a young researcher and teaching assistant at the Faculty of Organization and Informatics working at the Information systems development department. His main areas of interests include classic and agile software engineering methodologies, software and information systems development, business processes modeling and others.

Matija Besednik, univ.bacc.inf

e-mail: matija.besednik@foi.hr

Matija Besednik, univ.bacc.inf je 2010. godine završio sveučilišni preddiplomski studij Informacijski sustavi na Fakultetu organizacije i informatike u Varaždinu, te je polaznik diplomskog studija Informacijsko i programsko inženjerstvo na istom fakultetu. Njegove glavne aktivnosti su usmjerene na razvoj informacijskih sustava. Područje interesa je razvoj višeslojnih aplikacija s naglaskom na sloj poslovne logike koristeći CSLA.NET razvojni okvir.

Ivan Curić, univ.bacc.inf

e-mail: ivan.curic@foi.hr

Ivan Curić, univ.bacc.inf je 2010. godine završio sveučilišni preddiplomski studij Informacijski sustavi na Fakultetu organizacije i informatike u Varaždinu, te je polaznik diplomskog studija Informacijsko i programsko inženjerstvo na istom fakultetu. Njegove glavne aktivnosti su usmjerene na razvoj informacijskih sustava, te na razvoj web aplikacija za što je dobio i određena priznanja. Područja razvoja s kojima se intenzivno bavi su .NET platforma za razvoj mobilnih i web aplikacija u Silverlightu, te Java platforma za razvoj web aplikacija. Ivanov detaljan životopis može se pronaći na adresi http://tiny.cc/icuric-cv.

Kristijan Kralj, univ.bacc.inf

e-mail: kristijan.kralj@foi.hr

Kristijan Kralj, univ.bacc.inf je 2010. godine završio sveučilišni preddiplomski studij, smjer Poslovni sustavi na Fakultetu organizacije i informatike u Varaždinu, te je polaznik diplomskog studija Informacijsko i programsko inženjerstvo na istom fakultetu. Njegove glavne aktivnosti su usmjerene na modeliranje poslovnih procesa i razvoj desktop i web aplikacija. Područja razvoja s kojima se bavi su Microsoft .NET razvojno okruženje za razvoj mobilnih i desktop aplikacija te razvoj web aplikacija u PHP programskom jeziku. Kristijanov detaljni životopis može se pronaći na adresi http://tiny.cc/kkralj-cv.

Kristina Samošćanec, univ.bacc.inf

e-mail: kristina.samoscanec@foi.hr

Kristina Samošćanec, univ.bacc.inf je završila sveučilišni preddiplomski studij Informacijski sustavi, na Fakultetu organizacije i informatike, 2010. godine u Varaždinu. Polaznica je prve godine diplomskog studija na Fakultetu organizacije i informatike u Varaždinu, smjer Informacijsko i programsko inženjerstvo. Interesi su joj usmjereni na sigurnost informacijskih sustava, web dizajn, te razvoj web aplikacija, kao i na Java i PHP platforme.

Fakultet organizacije i informatike

Pavlinska 2

42000 Varaždin