UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Ceglar Simon

Prenova spletnega portala za prodajna mesta z uporabo tehnologije Silverlight

DIPLOMSKO DELO NA
VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU
PRVE STOPNJE

Mentor: višji pred. dr. Igor Rožanc

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00084/2011
Datum: 01.04.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **SIMON CEGLAR**

Naslov: **PRENOVA SPLETNEGA PORTALA ZA PRODAJNA MESTA Z UPORABO TEHNOLOGIJE SILVERLIGHT**

THE RENOVATION OF THE POS WEB PORTAL USING SILVERLIGHT TECHNOLOGY

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi predstavite prenovo spletnega portala za prodajna mesta v podjetju Intesa Sanpaolo Card s poudarkom na uporabi tehnologije Silverlight.

V ta namen najprej predstavite tehnologijo Silverlight in uporabljena orodja. Temu naj sledi sistematičen prikaz razvoja portala na podlagi definiranih zahtev. Pri izvedbi izpostavite predvsem prednosti novega pristopa. Na koncu predstavite izgled in uporabo novega portala.

Mentor:

viš. pred. dr. Igor Rožanc

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Simon Ceglar,

z vpisno številko 63030198,

sem avtor/-ica diplomskega dela z naslovom:

Prenova spletnega portala za prodajna mesta z uporabo tehnologije Silverlight

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  viš. pred. dr. Igorja Rožanca

  in somentorstvom (naziv, ime in priimek)

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____ Podpis avtorja/-ice: _____

Zahvala

Zahvaljujem se mentorju dr. Igorju Rožancu za mentorstvo, za nasvete in za pomoč pri nastanku diplomske naloge.

Zahvaljujem se svojim staršem za podporo in potrpežljivost v času študija.

Iskrena hvala tudi Tjaši, ki me je vseskozi podpirala in mi vedno stala ob strani.

Kazalo

KAZALO SLIK

Seznam uporabljenih kratic

XAML	Deklarativni jezik, ki je zasnovan na osnovi XML-ja in se uporablja za ustvarjanje in urejanje grafičnega uporabniškega vmesnika (ang. eXtensible Application Markup Language)
C#	Microsoftov programski jezik
.NET	Microsoftova tehnologija za razvijanje dinamičnih spletnih strani
Visual Basic	Programski jezik tretje generacije z integriranim razvojnim okoljem
BCL	Standardna knjižnica za vse programske jezike, ki uporabljajo .NET ogrodje (ang. Base Class Library)
WPF	Grafični podsistem za upravljanje uporabniških vmesnikov v Windows aplikacijah (ang. Windows Presentation Foundation)
Silverlight	Ogrodje za pisanje in vodenje poslovnih aplikacij (bogatih spletnih aplikacij)
CSS	Slogovni jezik za predstavitev dokumenta, ki je napisan v označevalnem jeziku (ang. Cascading Style Sheets)
DHTML	Dinamični HTML
JavaScript	Objektni skriptni programski jezik, ki ga je razvil Netscape z namenom, da pomaga spletnim programerjem pri ustvarjanju interaktivnih spletnih strani
AJAX	Skupina med seboj povezanih metod na odjemalčevi strani za razvoj interaktivnih spletnih aplikacij (ang. Active Javascript and XML)
Expression Design	Orodje za vektorske ilustracije in grafično oblikovanja spletnih slik
Expression Blend	Orodje za kreiranje grafičnega vmesnika za spletne in namizne aplikacije
Plug-in	Programska oprema, ki se lahko enostavno namesti in uporablja kot del spletnega brskalnika
API	Programski vmesnik (ang. Aplication Programming Interface)
CLS	Odprto kodna komponenta ki jo je razvil Microsoft in opisuje izvedljivo kodo in izvedeno okolje (ang. Common Language Specification)
HTML	Standardni jezik za razvoj spletnih strani. Z jezikom HTML označujemo in določamo lastnosti besedila (ang. HyperText Markup Language)
XML	Programski jezik, podoben HTML-ju, za opis strukture podatkov in njihovo izmenjavo med več omrežji (ang. Extensible Markup Language)
WAS	Proces za aktivacijo mehanizma z IIS (ang. Windows Activation Service)
IIS	Spletni strežnik za uporabo na Microsoft Windows platformi (ang. Internet

	Information Service)
HTTP	Komunikacijski protokol med odjemalci in strežniki. Protokol je namenjen objavljanju in prejemanju informacij na spletu preko HTML strani (ang. HyperText Transfer Protocol).
MSMQ	Protokol za sporočila, ki omogoča da aplikacije delujejo na ločenih strežnikih (ang. Microsoft Message Queuing)
SOAP	Protokol za izmenjavo strukturiranih podatkov pri izvajanju spletnih storitev v računalniških omrežjih (ang. Simple Object Access Protocol).
OTP čitalec	Čitalec, ki generira enkratno geslo za dostop do portala (ang. One Time Password)
POS terminal	Terminal na prodajnem mestu (ang. Point of Sale terminal)
ISP	Združenje bank Intesa SanPaolo
PAN	Unikatna številka kartice (ang. Primary Account Number)

POVZETEK

Cilj diplomske naloge je opis prenove obstoječega portala z uporabo novih tehnologij za razvoj spletnih aplikacij. V diplomski nalogi želimo predstaviti prenovo portala, nove prednosti, ki jih s seboj prinaša nov portal ter primerjavo med starim in novim portalom. Pri razvoju smo uporabljali Microsoftove produkte: Silverlight in ASP.NET. Kot glavni razvojni jezik smo uporabljali programski jezik C#. V nalogi predstavimo tehnologije, načine razvoja aplikacije, njihovo uporabo in integracijo v novi aplikaciji. V nadaljevanju opišemo tudi obstoječi portal, pokažemo, kaj je namen prenove portala ter na koncu analiziramo in primerjamo rešitvi.

Zahteva po predelavi obstoječega portala je bila prisotna že kar nekaj časa. Predvsem zaradi zastarelosti obstoječega portala in omejitev pri razvoju novih funkcionalnosti je bila predelava nujna.

ASP.NET je namenjen razvoju dinamičnih spletnih strani in spletnih web servisov. ASP.NET je podprt z ogrodjem .NET. Silverlight je .NET tehnologija namenjena razvoju in izdaji nove generacije bogatih spletnih aplikacij. Nudi bogato podporo multimedijskim predstavitvam in številne module za gradnjo aplikacij. XAML je opisni jezik, ki zagotavlja izgradnjo bogatega uporabniškega vmesnika in hkrati predstavlja most med oblikovalskim delom in programsko logiko aplikacij, ki so izdelane v Silverlightu.

V zaključku diplomske naloge predstavimo izgled portala po prenovi in analiziramo uporabljene rešitve tehnologije Silverlight in ASP.NET.

Ključne besede: Silverlight, ASP.NET, XAML, WCF, WPF, spletni portal

ABSTRACT

The goal of this project is renovation of the existing portal, using new technologies for developing web applications. We want to present the renewal of Portal, the new benefits in a new portal and a comparison between the old and the new portal. In the development we used Microsoft products: Silverlight and ASP.NET. The main development language we used the programming language C#. In thesis we present technology, methods of application development, their use and integration into new application. Further we describe the existing portal, show the main changes of renewal old. At the end we analyze and compare both solutions.

The need for renovation of an existing portal has been present for quite a while. The new development was neccesary because of the obsolescence of the existing portal and due to it's restrictions in the development of new functionalities. In this thesis we present a detailed development of a new portal, the introduction of Silverlight and its components in the application and the operation of the new portal.

ASP.NET is tehnology for developing dynamic web pages and online web services. ASP.NET is supported by the Framework.NET. Silverlight is .NET technology for developing and issuing a new generation of Rich Internet Applications. It offers support for multimedia presentations and a number of modules for building applications. XAML is a descriptive language used for construction of rich user interface. It is a bridge between work design and programming logic of the applications made in Silverlightu.

In conclusion we present the use of new portal after renovation and analyze used solutions of Silverlight and ASP.NET technology.

Key words: Silverlight, ASP.NET, XAML, WCF, WPF, Web portal

1. UVOD

V diplomski nalogi predstavimo Activa Portal, ki je namenjen vsem prodajnim mestom, ki uporabljajo POS terminale in so v lasti sistema Activa. Na portalu uporabniki lahko pregledujejo transakcije, provizije in analize za izbrano določeno obdobje. Zahteve prodajnih mest po novih funkcionalnostih so bile vedno večje, zato se je pojavila potreba po razvoju novega portala, ki bo prodajnim mestom ponudil več podatkov in funkcionalnosti in tako povečal število uporabnikov. Razvoj novih funcionalnosti v starem portalu je bil skorajda nemogoč, saj je zahteval veliko časa in je imel zelo veliko omejitev. Predstavili bomo glavni namen prenove portala, ki je bil zagotoviti, da se bodo vsi podatki o kartičnem poslovanju hranili na spletnem okolju, do katerega bodo preko spletne strani dostopala prodajna mesta. Razvoj aplikacije mora biti zelo dinamičen, kar nam bo omogočalo razvoj novih zahtev s strani prodajnih mest. Po končanem razvoju mora biti aplikacija tudi prijazna do uporabnika, saj mora na zelo preprost način uporabniku nuditi možnost pridobivanja informacij o prodajnem mestu in želenih transakcijah.

Pri predelavi portala smo se srečali s problemom izbire tehnologije. Odločili smo se za Microsoftovo tehnologijo Silverlight, saj nam le-ta dopušča veliko prostora pri razvoju aplikacije in podpira veliko komponent, ki jih ponuja Microsoft in so prijazne do uporabnikov, ki so vajeni Microsoftovih orodij.

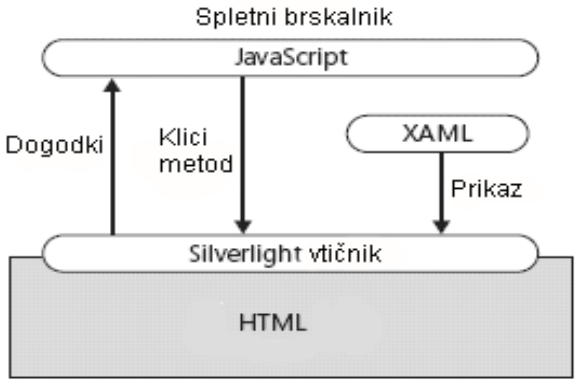
Namen in cilji diplomske naloge so: opisati in predstaviti uporabljene tehnologije in orodja, prikazati razvoj uporabne spletne aplikacije za pridobivanje podatkov ter ponazoriti glavne prednosti te aplikacije.

2. UPORABLJENE TEHNOLOGIJE IN ORODJA

Poglavje je namenjeno predstavitvi tehnologij ter orodij, s pomočjo katerih je nastala aplikacija. Predstavili bomo tehnologijo Silverlight in komponente, kot so ASP.NET članstvo in ASP.NET profili. Pri razvoju smo uporabljali Microsoftove produkte, kot so WindowsCommunicationFoundation in WindowsPresentationFoundation. Opisali bomo tudi orodje Microsoft Visual Studio 2008, ki smo uporabljali pri samem razvoju.

2.1 Tehnologija Silverlight

Silverlight je spletno zasnovana platforma, ki teče v obliki vtičnika (ang. plug-in) znotraj spletnega brskalnika [7]. Vtičnik spletnega brskalnika ima nalogo, da prebere XAML in prikaže podatke ter zagotavlja programski model. Ta je lahko brskalniško usmerjen, kar dosežemo z uporabo ogrodja .NET in jezikov C# ali Visual Basic. Slika 1 prikazuje primer aplikacije, ki teče v brskalniku z uporabo vtičnika Silverlight.

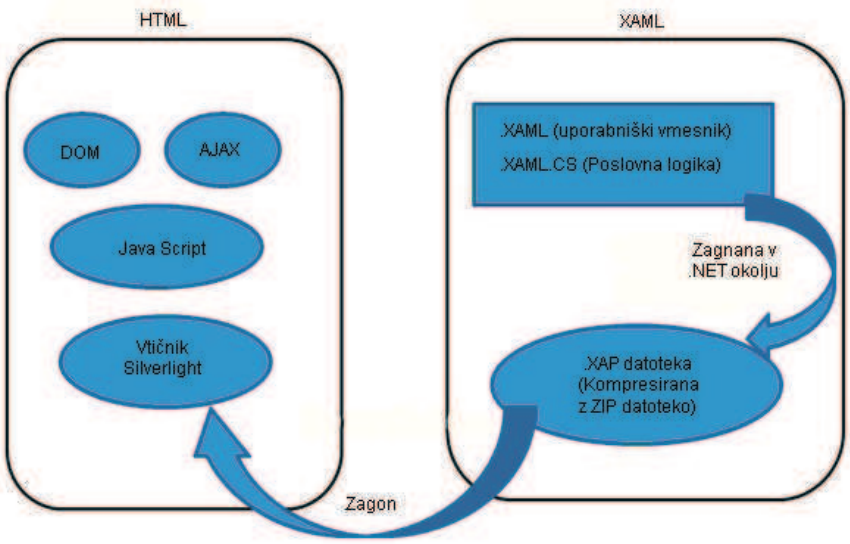


Slika 1: Arhitektura aplikacije v Silverlight

Cilj tehnologije Silverlight je postaviti spletne aplikacije na enak kakovostni nivo, kot jo najdemo v uporabniških vmesnikih namiznih aplikacij. Zasnovana je tako, da premosti ovire, ki so trenutno prisotne med oblikovalci in programerji na način, da obema taboroma ponudi enoten jezik za izgradnjo vmesnika, imenovan XAML. Več o XAML-u je v podpoglavju 2.2. Poleg enotnega jezika XAML, s pomočjo katerega lahko oblikovalci zgradijo dovršen prezentacijski model, pa Silverlight vključuje tudi dobršen del komponent BCL (ang. Base Class Library - BCL) in WPF (ang. Windows Presentation Foundation - WPF), ki se uporabljajo pri razvoju namiznih aplikacij [5]. Zaradi te sorodnosti med WPF in Silverlight lahko rečemo tudi, da je WPF »starejši brat« tehnologije Silverlight. Dobršen del pravimo zato, ker vnos celotnih knjižnic BCL iz WPF v tehnologijo Silverlight ni smiseln, saj so

zaradi drugačne arhitekture v delovanju nekatere knjižnice odvečne. Primer takšne knjižnice bi bila `Security`, ki v Silverlightu ni smiselna, saj celotna aplikacija teče v peskovniku (ang. sand-box) znotraj brskalnika in je tako varnost že zagotovljena s strani spletnega brskalnika. Kljub sorodnosti med Silverlight in WPF, pa knjižnice le niso povsem identične. Nova je namreč skrbno napisana na novo ter optimalno prirejena za rabo v Silverlightu, na način kot smo ga vajeni pri WPF. WPF je torej služil kot vzorec pri snovanju Silverlighta. Podmnožica knjižnic iz BCL-ja, uporabljenih v Silverlightu, lahko poimenujemo tudi jedrne knjižnice BCL (ang. Core BCL). Z vnosom jedrnih knjižnic BCL v Silverlight programerji pridobimo na moči, ki jo nudi ogrodje .NET: izolirano shrambo (ang. Isolated Storage), omrežno povezovanje, bogat nabor kontrol in podobno.

Kljub vključitvi knjižnic .NET pa Silverlight vseeno ostaja kompatibilen z različnimi brskalniki in različnimi operacijskimi sistemi, saj vtičnik, ki ga namestimo na lokalni računalnik že vsebuje vse potrebno za prevajanje in izvajanje programske kode. Silverlight tako omogoča izgradnjo izrazitih uporabniških vmesnikov s pomočjo jezika XAML, CSS in DHTML na predstavitvenem nivoju in hkrati razširja programski nivo JavaScript-a in AJAX-a z močjo platforme .NET, kot kaže slika 2. S to strategijo tehnologija Silverlight dosega globalno razširljivost, združeno z močjo namiznih aplikacij.



Slika 2: Silverlight API platforma

2.2 Jezik XAML

Kratica XAML (ang. eXtensible Application Markup Language) predstavlja jezik, zasnovan po pravilih jezika XML Uporabljamo ga za definiranje vizualnega dela naše aplikacije

Silverlight. Sprva je bil predstavljen za potrebe WPF, namizno usmerjene tehnologije, ki je del .NET ogrodja 3.0. Kasneje pa se je ta jezik prenesel tudi v tehnologijo Silverlight in predstavlja most med razvijalci in oblikovalci [16].

Ker je XAML zasnovan na XML, pomeni, da za opis uporabniškega vmesnika uporablja elemente XML. Dokument XML mora imeti natanko en korenski element in pri XAML-u je uporabniška kontrola (ang. `UserControl`), ki vsebuje deklaracijo xml imenskega prostora (ang. `namespace`), ki ga potrebuje Silverlight in element, ki je ponavadi predstavnik enega izmed vsebovalnikov (ang. `container`), kot je npr. platno (ang. `StackPanel`), znotraj katerega lahko zgradimo ali narišemo uporabniški vmesnik kot kaže slika 3.

```
<StackPanel>
  <Button Content="Click Me"/>
</StackPanel>
```

Slika 3: Preprost primer XAML

Elementi XAML so objekti, ki se v času izvajanja preslikajo v primerek razreda, atributi pa v lastnosti tega primerka. Tako bi lahko za primer iz slike 4, kjer smo deklarirali `Border` v XAML, zapisali ekvivalent izključno v programski kodi, tako da bi ustvarili nov primerek razreda `Button` in mu določili lastnost ozadja črk z vrednostjo `Brushes.Red`.

```
<Border>
  <TextBox Width="300"/>
</Border>
<!--explicit equivalent-->
<Border>
  <Border.Child>
    <TextBox Width="300"/>
  </Border.Child>
</Border>
```

Slika 4: Primer dodelitve lastnosti TextBoxu

Lahko pa bi za dosego enakega rezultata uporabili tudi mešanico obojega, torej jezika XAML in C#. Pri tem je pogoj, da v jeziku XAML ustvarimo ime v razširjenem imenskem prostoru (`x:Name`), katero bo služilo kot referenca za naslavljanje pravkar ustvarjenega primerka v jeziku XAML tudi v programski kodi jezika C#, kot kaže slika 5.

```
namespace ExampleNamespace
{
  public partial class ExamplePage
  {
    void Button_Click(object sender, RoutedEventArgs e)
    {
      Button b = e.Source as Button;
      b.Foreground = Brushes.Red;
    }
  }
}
```

Slika 5: Primer dodelitve lastnosti s programsko kodo

Najpogostejši primer uporabe razširjenega imenskega prostora je ravno uporaba atributa `x:Name`, s katerim elementu določimo ime, na katero se lahko kasneje sklicujemo v naši programski kodi. Namreč popolnoma vsak dokument XML nosi s sabo pripeto tudi istoimensko datoteko s končnico `.cs` ali `.vb`, ki ta dokument razširja s programsko kodo jezika C# ali Visual Basic. Vsak element, ki ga ustvarimo v jeziku XAML (npr. v datoteki `Domov.xaml`) in mu določimo ime v razširjenem imenskem prostoru (`x:Name='ime'`), bo pod tem imenom viden in dosegljiv tudi v programski kodi pripete datoteke (npr. v `Domov.xaml.cs`).

Oblikovalci aplikacij uporabljajo drugačna orodja kot razvijalci aplikacij. Microsoft je za oblikovalce razvil specifična orodja, kot so npr. Expression Design in Expression Blend, v katerih oblikovalci gradijo uporabniški vmesnik na vizualni način, medtem ko ta orodja v ozadju gradijo drevo elementov XAML, njihove lastnosti in dogodke. Ker se da jezik XAML preslikati v množico razredov v programski kodi, lahko programerji istočasno XAML oblikovalcev oživijo in podprejo s programsko logiko, razvito v poljubnem jeziku .NET.

2.3 Orodje Microsoft Visual Studio 2008

Microsoftov razvojni sistem Visual Studio je zbirka razvojnih orodij, ki razvijalcem programske opreme pomaga premagovati zapletene izzive in ustvarjati inovativne rešitve. Vloga zbirke Visual Studio je izboljšati razvojni proces ter omogočiti preprostejše uresničevanje inovacij [11]. Razvijalci se danes soočajo z izzivom razvoja aplikacij, ki so usmerjene k široki paleti različnih platform, ki morajo zagotavljati stabilno in enostavno vzdrževanje, hkrati pa privabijo k uporabi aplikacije veliko število uporabnikov. Integrirane oblikovalske in jezikovne možnosti, ki jih nudi Visual Studio, razvijalcem omogočajo razvoj

povezanih aplikacij, kakršne zahtevajo današnja podjetja. Hkrati pa razvijalci lahko izkoriščajo tudi prednosti ogrodja .NET Framework 3.5, ki omogoča skrajšan čas razvoja.

2.4 Ogrodje .NET 3.5

Microsoft .Net Framework je programski paket, ki vsebuje različne knjižnice, namenjene reševanju večine znanih programskih problemov in izvajalno okolje za programe, napisane za to ogrodje [14]. Glavna knjižnica (ang. Base Class Library - BCL) je celovita zbirka objektno usmerjenih in ponovno uporabljenih primerov za razvijanje aplikacij. Ponuja široko vrsto funkcionalnosti, med katere sodijo: uporabniški vmesnik, dostop do podatkov, povezava na podatkovno bazo, kriptografija, razvijanje spletnih aplikacij, numerični algoritmi ter mrežne komunikacije. Knjižnica je torej namenjena razvijalcem, ki kombinirajo njene komponente in jih združujejo s svojimi ter tako gradijo učinkovite rešitve [1]. BCL določa razrede, ki zajemajo veliko število skupnih nalog, vključno z branjem in pisanjem v datoteko, grafičnega prikaza in upravljanje XML dokumenta [10].

Izvajalno okolje (ang. Common Language Runtime - CLR) predstavlja temelj ogrodja .NET. Predstavljamo si ga lahko kot nekakšnega agenta, ki skrbi za kodo med njenim izvajanjem. Odgovoren je za osnovne naloge, kot so upravljanje s pomnilnikom in nadzor nad nitmi, obenem pa skrbi za pravilno, varno in robustno izvajanje programa [14].

Ogrodje .NET 3.5 sestavljajo štiri komponente: Windows Presentation Foundation, Windows Communication Foundation, Windows Workflow Foundation in Windows Card Space. Razporeditve komponent ogrodja prikazuje slika 6.

Slika 6: Shema ogrodja .NET 3.5

2.5 Jezik XML

XML (ang. Extensible Markup Language) [17] je označevalni jezik, ki je podoben jeziku HTML. Obstaja pa velika razlika, saj XML ni namenjen prikazovanju podatkov, ampak njihovemu prenašanju in shranjevanju, s poudarkom na tem, kaj ti podatki sploh so in kaj predstavljajo. XML tako ne naredi ničesar, ampak samo strukturira, shranjuje in prenaša informacijo na samoopisen način.

Dokumentu XML pravimo, da je dobro oblikovan (ustreza sintaksi), če zadostuje naslednjim pogojem:

- Dokument XML lahko vsebuje samo en korenski element. Vsa vsebina, ki je del dokumenta samega, je vsebovana v korenskem elementu.
- Elemente XML lahko gnezdimo
- Vsak element XML vsebuje začetno in končno oznako.
- Imena začetnih in končnih oznak morajo biti enaka.
- Elementi XML se ne smejo prekrivati med seboj
- Znotraj besedila v dokumentu XML se ne smejo pojavljati naslednji znaki: < > &. To so znaki, ki imajo določen pomen za razčlenjevalnik XML.

XML dokument je veljaven, če je dobro oblikovan in če se lahko prilagodi določenemu naboru omejitev, ki so definirane v shemi XML.

Shema XML je formalna specifikacija ki je ravno tako napisana v jeziku XML in določa strukturo dokumenta XML, imena elementov in obogatenih podatkovnih tipov, kateri elementi se lahko prikažejo v kombinaciji in kateri atributi so na voljo za posamezen element.

XML dokument je sestavljen iz dveh glavnih delov: uvoda in korenskega elementa. Uvod vsebuje deklaracijo XML. Ta navaja, da je to dokument XML, vsebuje navodila za obdelavo, obravnavo dokumenta in deklaracije shem XML za preverjanje veljavnosti dokumenta.

Korenski del elementa je glavni del dokumenta XML. Informacije v korenskem elementu so shranjenje v elementih in atributih. Elementi so glavni gradniki dokumenta XML in se uporabljajo za predstavitev strukture dokumenta XML. Začetna oznaka elementa pa lahko vsebuje tudi atribute, ki se uporabljajo za opis določenih lastnosti tega elementa kot kaže slika 7.

```
<?xml version="1.0" encoding="utf-8" ?>
<ManagerReaderForm>
  <components>
    <component
          id="ACQOnUs"
          service="Exact.BLPripomocki.Vmesniki.IReader`1, Exact.BLPripomocki"
          type="Exact.BLPripomocki.ManagerReader`1, Exact.BLPripomocki">
      <parameters>
        <FilePattern>ISS_${yyyyMMdd}_${HHmmss}.XML</FilePattern>
        <Split>true</Split>
      </parameters>
    </component>
  </components>
</ManagerReaderForm>
```

Slika 7: Primer dokumenta XML

2.6 ASP.NET članstvo

ASP.NET članstvo (ang. `ASP.NET membership`) [4] je vgrajen način za preverjanje in shranjevanje uporabniških računov. ASP.NET članstvo vam zato pomaga upravljati overjanje uporabnika na vaši spletni strani. ASP.NET membership lahko uporabite z avtentikacijo ASP.NET aplikacij za uporabo ASP.NET prijavnih kontrol za oblikovanje celotnega sistema za avtentikacijo uporabnikov.

Nekaj funkcionalnosti ASP.net članstva:

- ustvarjanje novih uporabnikov in gesel
- shranjevanje podatkov (uporabniška imena, gesla, in vse druge informacije) v Microsoft SQL Server, Active Directory, ali druge podatkovne baze
- avtentikacija uporabnikov, ki obiščejo vašo spletno stran. Pristnost uporabnikov lahko preverjamo programsko, ali pa z uporabo ASP.NET prijavne kontrole za nadzor
- upravljanje gesel, ki vključuje oblikovanje, spreminjanje in ponastavljanje. Glede na možnost članstva izberete ponastavitev gesla ali pa uporabimo varnostno vprašanje z odgovorom.
- izpostavljanje enotne identifikacije prijavljenih uporabnikov, ki jih lahko uporabite v lastnih aplikacijah in obstaja možnost, da se tudi združi s personalizacijo ASP.NET in vlogo upravljanja sistemov.
- določanje članstva po meri, ki omogoča, da zamenjamo svojo kodo za upravljanje in vzdrževanje članstva v podatkovnem modelu po meri.

2.7 ASP.NET profil

V številnih aplikacij želimo shraniti in uporabiti podatke ki so specifični za določenega uporabnika. Ko uporabnik obišče vašo spletno stran, lahko uporabimo informacije, ki smo jih prej shranili in predstavimo uporabniku prilagojeno različico naše spletne aplikacije. Prilagajanje aplikacije zahteva več elementov: podatke moramo hraniti z edinstveno identifikacijsko oznako uporabnika, da jih bomo lahko prepoznali ko bomo aplikacijo ponovno obiskali, nato pa poberemo tiste podatke o uporabniku, ki so potrebni. Da bi poenostavili naše aplikacije, lahko uporabimo ASP.NET nastavitev [3], ki lahko opravlja vse te naloge namesto nas.

ASP.NET profil deli informacije o posameznem uporabniku in jih trajno hrani. Profili vam omogočajo, da upravljate z informacijami o uporabniku, ne da bi aplikacija od vas zahtevala, da ponovno vzpostavljamo in vzdržujemo svoje baze podatkov. Poleg tega ASP.NET profil zagotavlja da so informacije o uporabniku na voljo kjerkoli v naši aplikaciji.

Z ASP.NET profilom lahko shranjujemo katerekoli objekte. Zagotavlja tudi splošno shranjevanje, ki omogoča, da vzdržujemo vse vrste podatkov, medtem ko so vsi podatki na voljo v varnem načinu.

Uporabo profilov omogočimo tako, da spremenimo konfiguracijsko datoteko za našo spletno stran. Ta del konfiguracije, ki smo jo določili ponudnikom profila z osnovnim razredom, ki opravlja nizko stopnjo nalog za shranjevanje in pridobivanje podatkov profilov. Lahko uporabimo profil, ki je ponujen v .NET Frameworku in shranjuje podatke o profilu v SQL Server, ali pa ustvarimo in uporabimo svojega ponudnika profi. Določimo lahko na primer število instanc, ki se povežejo z zbirko podatkov, lahko pa uporabimo privzeto instanco `SqlProfileProvider`, ki shranjuje podatke na lokalni ravni [3].

Profil nastavimo s funkcijami, ki jih opredeljuje naš seznam lastnosti, katere vrednosti želimo ohraniti. Na primer, morda želimo shraniti poštno številko uporabnika, da lahko naš program ponudi specifične informacije, ki so značilne za določeno pošto, kot je recimo vremenska napoved. V konfiguracijski datoteki opredelimo za naš profil lastnost z imenom `PostalCode`. Konfiguracijska datoteka za naš profil naj izgleda kot je prikazano na sliki 8.

```
<profile>
  <properties>
    <add name="PostalCode" />
  </properties>
</profile>
```

Slika 8: Konfiguracijska datoteka profila po meri

2.8 Komponenta WCF

(ang. Windows Communication Foundation – WCF) je programski vmesnik za razvoj povezljivih, storitveno orientiranih aplikacij. Je izvajalno okolje za aplikacije, ki pošiljajo sporočila med storitvami in odjemalci. Infrastruktura in paket API vmesnikov je venomer enak, pri komunikaciji nad storitvami na enem računalniku ali med več storitvami, porazdeljenimi preko interneta. Komponenta WCF je zasnovana na sporočilni zgradbi, kar pomeni, da se vse lahko modelira kot sporočilo (npr. HTTP zahteva ali MSMQ sporočilo) in se predstavi v programskem modelu. To omogoča enoten vmesnik med različnimi transportnimi sistemi. V modelu srečamo dve vlogi: odjemalce (ang. client), ki sprožijo komunikacijo, in storitve (ang. service), ki čakajo na odjemalce in jim posredujejo sporočila. Aplikacija je lahko naenkrat v vlogi odjemalca in storitve. Sporočila se pošiljajo in prejemajo preko končnih točk (ang. endpoint), v katerih so zabeleženi vsi potrebni podatki za izmenjavo sporočil. Končna točka vsebuje informacije o naslovniku sporočila, kako naj bo poslano in kako je sporočilo oblikovano [18]. Storitve, napisane v WCF, so sposobne sodelovati s storitvami konkurenčnih podjetij (IBM, BEA in Novell) in so dovolj razširljive, da ostanejo v koraku s hitro spremenljivimi industrijskimi standardi [14]. Ključ do uspeha je večslojna arhitektura, prikazana na sliki 9 [18]. Dogovori (ang. Contracts) določajo različne poglede sporočilnega sistema: podatkovne (ang. data contracts), ki opisujejo parametre za generiranje sporočila, sporočilne (ang. message contracts), ki skrbijo za pravilnost sporočil s SOAP protokolom, ter storitvene (ang. service contracts), ki so v vlogi vmesnika.

Storitveno izvajalno okolje (ang. Service runtime) vsebuje različne dogodke, ki se lahko pojavijo pri komunikaciji, npr. število sporočil, ali je prišlo do napake, kaj narediti v primeru napake in podobno.

Slika 9: Predstavitev WCF modela

Sporočanje (ang. `Messaging`) vsebuje transportne in protokolne kanale, ki pregledujejo glavo sporočila ki za razliko od storitvenega izvajalnega okolja skrbi za vsebino.

Storitve je potrebno izvajati in jih objaviti, za kar skrbi sloj Aktivacija in gostitev (ang. `Activation&hosting`). Za izvajanje znotraj izvajalnega okolja skrbi zunanji posrednik, kot je npr. IIS (ang. `Internet Information Service - IIS`) ali WAS (ang. `Windows Activation Service - WAS`), ki poskrbi za avtomatsko aktivacijo storitve ob namestitvi [18].

2.9 Komponenta WPF

Skozi razvoj .NET ogrodja je Microsoft predstavil veliko različnih paketov orodij za oblikovanje grafičnih vmesnikov (C/C++/Win32 API development, Visual Basic 6 oz. VB6, Microsoft Foundation Class oz. MFC, itd.). Vsak od teh vmesnikov je vseboval osnovne oblike prikaza (npr. okno, dialog, meni, gumb in ostale podrobnosti). S prvim izidom ogrodja .NET je postal Windows Forms API eden izmed najbolj uporabljenih vmesnikov za oblikovanje GUI, primarno zaradi svoje enostavnosti in obenem bogate funkcionalnosti. S prihodom WPF so se vsa ta orodja povezala v eno celoto.

Cilj razvijalcev WPF je ponuditi bogat paket funkcionalnosti na področju predstavitve podatkov: od osnovnih vektorskih grafik, gradientov in točkastih slik do naprednejših 3D

animacij, pretočnih vsebin in tipografij. Druga pomembna lastnost je bila združiti vse to v platformo, kar omogoča razvoj bogatejših aplikacij in celo razširitev na povsem nova področja delovanja. Naslednji korak je poenotenje razvijanja grafičnega vmesnika namiznih in spletnih aplikacij, kar je vpeljal WPF. Z uporabo jezika XAML (ang. Extensible Application Markup Language - XAML) lahko enostavno kreiramo elemente uporabniškega vmesnika, dogodke in podatkovne povezave, saj jezik temelji na formatu XML. Za spletne aplikacije skrbi ogrodje Microsoft Silverlight, ki je nekoliko okleščeno v primerjavi z WPF. Tako so mu odstranili 3D funkcionalnost, obdržali pa 2D animacije in ostale prednosti WPF. Kompatibilen je z večino operacijskih sistemov, za odprtokodne pa so poskrbeli pri podjetju Novell z izdajo ogrodja Moolight [15].

Slika 10: Arhitektura ogrodja WPF

3. OPIS OBSTOJEČEGA PORTALA

Portal je namenjen bankam in prodajnim mestom za pregled transakcij in za prevzemanje datotek, ki jih izdela aplikacija na osnovi nastavitev v portalu. Poleg ročnega prevzema datotek ima uporabnik možnost uporabe spletne storitve in tako avtomatizirati prenos med Portalom in sistemom uporabnika. Portal pridobiva podatke iz aplikacij Activa, Plkar, Access, Exact, Tandem in avtorizacijske baze. Spletne storitve lahko uporabljajo samo odjemalci, ki ustrezajo izbrani varnostni politiki podjetja.

V sistem se prodajno mesto prijavi prek zaslonske maske z uporabniškim imenom in geslom ki ga dobijo po podpisu pogodbe v poslovni enoti kar vidimo na sliki 11.

Slika 11: Zaslonska maska prijave v Portal

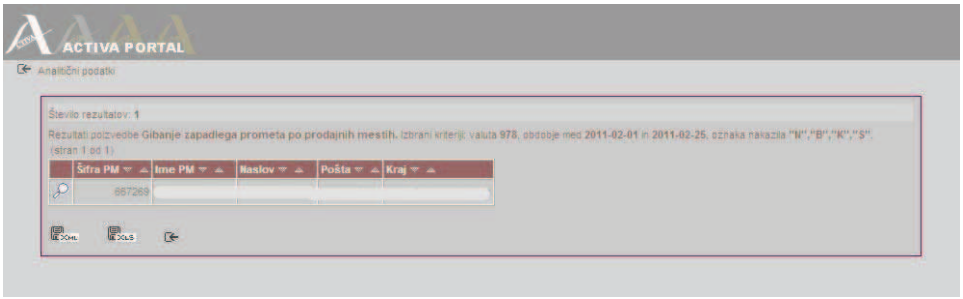
Slika 12: Uvodna stran po uspešni prijavi v Portal

Po uspešni prijavi se uporabniku prikaže meni, kar vidimo na sliki 12, kjer lahko izbira med naslednjimi možnostmi:

- Paketi in transakcije, kjer ima uporabnik možnost pregledovanja posameznih transakcij v izbranem obdobju, ki so bile izvedene na prodajnem mestu in paketov, ki so skupek posameznih transakcij, združenih po različnih kriterijih (tip kartice, datum transakcije), kar je prikazano na sliki 13.

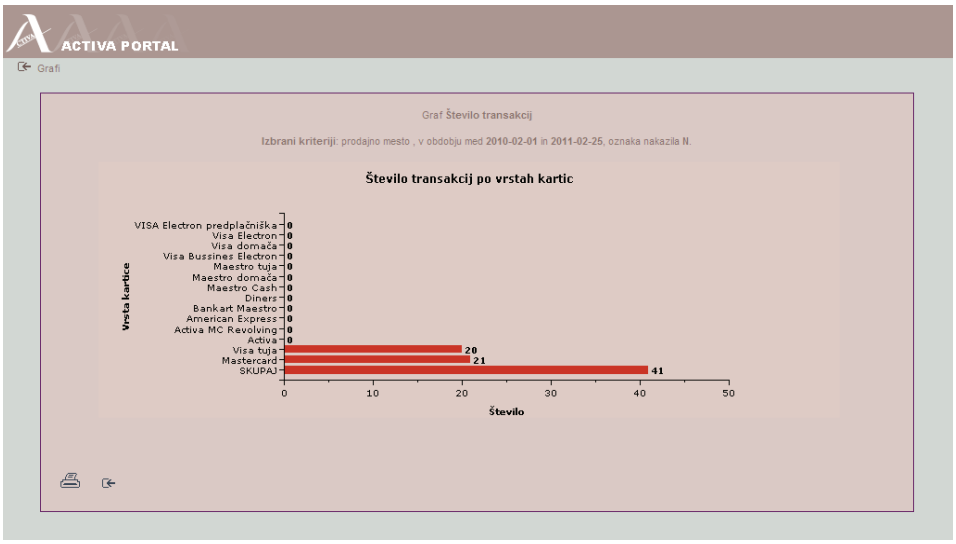
Slika 13: Primer iskanja transakcij v določenem obdobju

- Analitični so podatki kjer imamo možnost pregleda gibanja zapadlega prometa v določenem obdobju, kar vidimo na sliki 14.



Slika 14: Primer iskanja analitičnih podatkov

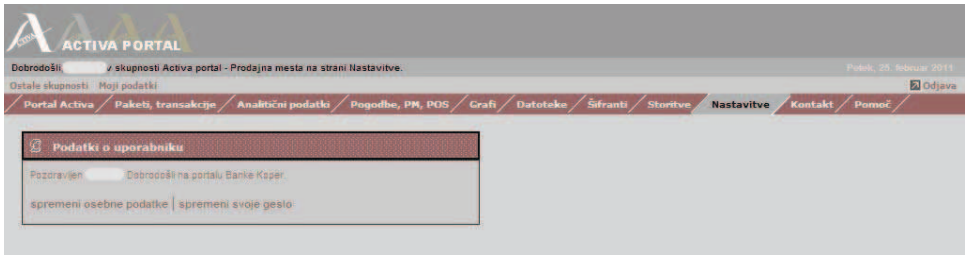
- Pogodbe, prodajna mesta in terminali kjer vidimo podatke o sklenjenih pogodbah
- Grafi prikazujejo gibanje prometa v določenem obdobju po tipu kartice, kar vidimo na sliki 15.



Slika 15: Primer grafa po številu transakcij

- Menu Datoteke je namenjen naročanju datotek
- Šifranti

- Nastavitve uporabniku omogočajo spreminjanje osebnih podatkov in spreminanje gesla, kar vidimo na sliki 16.



Slika 16: Primer menija Nastavitve

- Kontakt
- Pomoč

4. OPREDELITEV POTREB PO PRENOVI

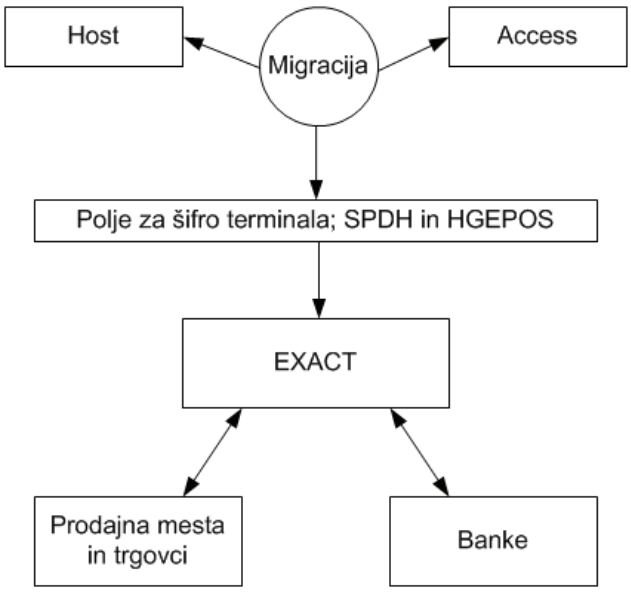
4.1 Namen prenove portala

Activa Portal so uporabljale vse večje banke iz sistema Activa in nekaj prodajnih mest, vendar je bilo zanimanje prodajnih mest vedno večje. Kot smo že omenili, Activa Portal pridobiva podatke iz aplikacij Activa, Plkar, Access, Exact, Tandem in avtorizacijske baze. Ker se podatki v portalu zbirajo iz različnih okolij obstaja možnost, da se podatki ne bi ujemali in da le ti ne bi bili ažurni.

Za uporabnike je bilo zlasti težavno, da informacije iščejo na različnih mestih. Nov portal mora omogočati dostop do vseh podatkov o kartičnem poslovanju, ki se bodo iz različnih aplikacij shranjevali v portalu.

Namen prenove je izločiti vse menije, ki se ne uporabljajo, vključiti nove ter popraviti obstoječe nedelujoče menije.

Na sliki 17 je prikaza shema prenove portala. Iz različnih virov, kot so Host in Access, prihajajo v sistem Exact podatki, ki so urejeni z enolično šifro terminala. Exact, kot glavni vir podatkov, posreduje prodajnim mestom, trgovcem in bankam podatke, ki jih potrebujejo in jih prikaže v spletnem brskalniku.



Slika 17: Shema prenove portala

4.2 Dostop do portala, prijava in varnost

Uporabniki bodo dostopali do portala preko enotnega spletnega naslova. Edini pogoj bo, da ima uporabnik dostop do interneta. Vsak uporabnik bo moral imeti možnost dostopati do portala z uporabo OTP čitalca ali certifikata. Tako bi vsako prodajno mesto, podjetje ali trgovec imelo dostop samo do svojih podatkov. Trgovec kot lastnik za vsa prodajna mesta, prodajno mesto pa za svoje POS terminale. V obstoječem portalu prodajna mesta dostopajo v portal z geslom, ki ga dobijo od banke ob sklenitvi pogodbe in imajo možnost, da ga po prvi prijavi spremenijo.

Ob sklenitvi pogodbe z banko bodo uporabniki imeli omogočen dostop in razpolaganje le s tistimi podatki, za katere so pooblaščeni in do katerih imajo pravico.

4.3 Nivo dostopanje do portala

Takoj po prijavi v sistem, uporabnik skozi menuje izbira parametre, ki mu omogočajo dostop do svojih podatkov. Banke iz drugih držav, članice skupine Intesa SanPaolo in njihova prodajna mesta že skozi prijavo določijo uporabniški jezik te države (Egipt, Romunija, Albanija, Srbija, Slovenija ....).

Za podjetje je potrebno uvesti vnos zbirnega prodajnega mesta multiračuna, ki bo določenemu trgovcu omogočil iskanje in zbiranje podatkov po vseh svojih računih oziroma prodajnih mestih.

4.4 Funkcije portala

Podatki, ki se prenašajo na portal zajemajo podatke o :

- prodajnih mestih,
- POS terminalih,
- transakcijah na POS terminalih,
- ročnih vnosih in ostalih transakcijah,
- zapadlih in nezapadlih paketih,
- transakcijah in zaključkih,
- imetnikih in njihovih karticah in
- avtorizacijah (aktivne rezervacije)

Nov portal Activa mora omogočati:

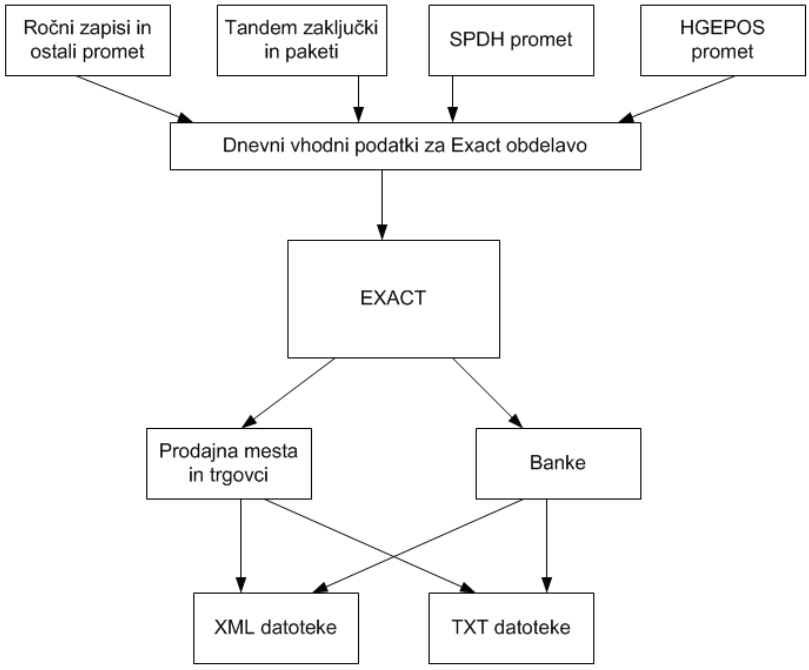
- dostop do podatkov 24ur na dan,
- pregled vseh podatkov vezanih na pogodbo z banko,
- pregled prometa v določenem časovnem obdobju,
- pregled zapadlih in nezapadlih transakcij v določenem časovnem obdobju,
- pregled paketov,
- naročanje datotek,
- prejemanje in prenos naročenih datotek in
- tabelarične in grafične prikaze.

Podatki se morajo prikazovati na različne načine:

- tabelarično ter
- grafično

Prodajna mesta, podjetja oziroma večji trgovci in banke bodo imeli možnost tudi naročanja datotek, ki jih bodo lahko uvažali v svoje sisteme. Datoteke bodo lahko prevzemali v TXT ali XML obliki.

Na sliki 18 je prikazana shema prenosa datotek v novem portalu. Exact dnevno izdeluje datoteke kot so ročni zapisi, tandem zaključki, paketi in promet terminalov. Datoteke se izdelujejo dnevno in se odložijo v mapo do katere ima portal dostop. Prodajna mesta ali banke si potem iz portala poberejo datoteke bodisi v XML bodisi v TXT obliki.



Slika 18: Shema prenašanja datotek

4.5 Uporabniki portala

Podatki portala so namenjeni naslednjim uporabnikom :

- bankam in članicam sistema Activa
- podjetjem in prodajnim mestom

4.5.1 Banke

Podatki so namenjeni vsem bankam članicam sistema Activa in v bodoče tudi bankam članicam skupine ISP v drugih državah.

4.5.2 Prodajna mesta

Uvedba multiračuna na pogodbi za trgovce, ki imajo več prodajnih mest. Multiračun bi moral zajemati vse račune prodajnih mest trgovca oziroma podjetja. Multiračun na pogodbo zajame vse komitente oziroma prodajna mesta trgovca. Vsako novo prodajno mesto ali ukinitev prodajnega mesta bi zajelo podatek iz pogodbe. S prijavo na portal se trgovcu omogoči preglede za vsa prodajna mesta, medtem ko prodajno mesto gleda izključno svoje podatke .

Omogočiti je bilo potrebno dostop za prodajna mesta in podjetja, ki imajo sklenjene pogodbo z banko članico skupine ISP v tujini (Romunija, Srbija, Albanija, ...).

4.6 Funkcije novega portala

- Potrebna je uvedba enolične številke, ki bi uporabnikom portala omogočila uparjanje transakcij.
- Za uparjanje transakcij bo potrebno prikazati nova polja, ki bodo morala zajemati vse kartice, tako Activine kot tuje.
- Potrebna je uvedba referenčne številke, ki bo zajemala številko paketa in šifro terminala.
- Omogočiti je potrebno tudi skupni seštevek po referenčni številki oziroma po številki paketa za terminal.
- Promet, ki prihaja na portal iz pos terminalov mora zajemati oba podatka o šifri terminala, ne glede na datum inštalacije ali ukinitve.
- Podatki o številki kartice v datotekah in portalu so zakriti tako, da se vidijo samo zadnja štiri mesta celotnega PAN-a..
- Pregled prodajnih mest in njihovih POS terminalov.
- Prikaz najemnine terminalov.
- Pregled prodajnih mest in terminalov brez prometa.
- Iskanje prometa po obdobju in vrsti kartice.
- Prikaz prometa zapadlih in nezapadlih transakcij za obdobje po tipu in vrsti kartice.

5. PRIMERJAVA TEHNOLOGIJ IN RAZLIKE V RAZVOJU

5.1 Primerjava tehnologij

Star portal je bil v celoti narejen v Accessovi bazi, ki je bila tudi glavni vir vseh podatkov. Za tehnologijo Silverlight smo se odločili iz več razlogov. Eden izmed njih je gotovo tudi podpora za najbolj razširjene spletne brskalnike kot so Internet Explorer, Firefox, Safari, Google Chrome in drugi. Prav tako nudi podporo za 3D grafiko, predvsem pa omogoča bogat in dinamičen razvoj spletnih strani.

Razlika v primerjavi s starim portalom je v tem, da si mora uporabnik pred začetkom uporabe novega portal prenesti in namestiti dodatke za brskalnik, ki ga uporablja. Silverlight podatke shranjuje v pomnilniku za kasnejše sklicevanje. Podatke lahko vnese uporabnik, lahko jih dobimo iz vira ali pa jih ustvarimo programsko.

Silverlight aplikacija se izvaja v okviru brskalnika, zato ni avtomatskih klicev na strežnik. Če je komunikacija iz aplikacije potrebna, jo moramo izvesti programsko.

V Silverlightu so vgrajene tudi varnostne omejitve, kadar imamo opravka s komunikacijo z različnimi domenami. Kadar Silvelight aplikacija komunicira z domeno, s katere izvira oziroma se je prenesla lokalno na uporabnikov računalnik potem ni nobenih omejitev. Dostop do poljubnega strežnika je prepovedan, razen če strežnik eksplicitno ne dovoli dostopa. To je urejeno s posebno datoteko, ki se nahaja na strežniku in ureja pravico dostopa.

Silverlight vsebuje okrnjeno množico XML funkcionalnosti. XML datoteke lahko programsko pošiljamo ali sprejemamo preko spleta. Uporaben je tudi pri veliki količini XML podatkov, saj se lahko z vsebovanimi razredi izognemo nalaganju vseh podatkov v pomnilnik.

Dosegli smo tudi to, da lahko na novem portalu izvažamo podatke iz zaslona v Excel datoteko.

5.2 Izvedbe in nekatere razlike v razvoju

Pri starem portalu je bil dodatni razvoj novih funkcionalnosti skorajda nemogoč, medtem ko je bila pri razvoju novega portala to ena od glavnih prioritet. Silverlight nam je omogočil, da smo uspeli med seboj povezati več aplikacij, ki tečejo na različnih sistemih in potem vse skupaj predstavili na enotni spletni strani. Razlika med obema portaloma je tudi v tem, da pri pregledu starega portala ni bilo potrebno na odjemalčev računalnik namestiti nobenih novih orodij, medtem ko je pri novem portalu obvezno, da uporabnik pred uporabo namesti na svoj računalnik komponento Silverlight, saj v nasprotnem primeru aplikacija ne bo delovala.

5.3 Ugotovitve

Stara verzija Portala je imela kar nekaj pomanjkljivosti, ki pa smo jih z razvojem novega portala poskušali odpraviti. Tu mislimo predvsem na dinamičnost portala, pregled in prenos

datotek, lepše in bolj urejeno iskanje transakcij in paketov za določeno prodajno mesto in podobno.

5.4 Uporaba rešitev v aplikaciji

V začetku razvoja novega portala smo se srečali z prijavo uporabnikov v aplikacijo. Da bi si čim bolj olajšali delo, smo se odločili za komponento ASP.net članstvo, ki jo najdemo v ogrodju .NET. Ta nam zagotavlja kreiranje novih uporabnikov, njihovo avtentikacijo, shranjevanje njihovih gesel, upravljanje z uporabniki, lahko pa se odločimo za članstvo po meri, kjer lahko sami določimo katere lastnosti so za nas pomembne. Kako smo uporabili razred `User` v naši aplikaciji, lahko vidimo na sliki 19.

```
public User()
{
    Roles = new List<string>();
    Name = Anonymous;
    IsAuthenticated = false;

    authenticationClient = ServiceAgentBase<AuthenticationServiceClient>.Create( ServiceAgentBase<AuthenticationServiceClient>.AuthenticationServiceConfigurationName, "/Services
    ServiceAgentBase<AuthenticationServiceClient>.CheckAndFixHttpsBinding(authenticationClient.Endpoint.Binding);
    authenticationClient.IsLoggedInCompleted += OnIsLoggedInCompleted;
    authenticationClient.LoginCompleted += OnLoginCompleted;
    authenticationClient.LogoutCompleted += OnLogoutCompleted;

    profileClient = ServiceAgentBase<ProfileServiceClient>.Create(ServiceAgentBase<ProfileServiceClient>.ProfileServiceConfigurationName, "/Services/ProfileService.svc");
    ServiceAgentBase<ProfileServiceClient>.CheckAndFixHttpsBinding(profileClient.Endpoint.Binding);
    profileClient.GetAllPropertiesForCurrentUserCompleted += OnGetAllPropertiesForCurrentUserCompleted;
    profileClient.SetPropertiesForCurrentUserCompleted += OnSetPropertiesForCurrentUserCompleted;

    roleClient = ServiceAgentBase<RoleServiceClient>.Create(ServiceAgentBase<RoleServiceClient>.RoleServiceConfigurationName, "/Services/RoleService.svc");
    ServiceAgentBase<RoleServiceClient>.CheckAndFixHttpsBinding(roleClient.Endpoint.Binding);
    roleClient.GetRolesForCurrentUserCompleted += OnGetRolesForCurrentUserCompleted;

    membershipClient = ServiceAgentBase<MembershipServiceClient>.Create(ServiceAgentBase<MembershipServiceClient>.MembershipServiceConfigurationName,"/Services/MembershipService
    ServiceAgentBase<MembershipServiceClient>.CheckAndFixHttpsBinding(membershipClient.Endpoint.Binding);
}
```

Slika 19: Uporaba razreda `User`

Ko je uporabnik prijavljen v sistem, smo hoteli, da bi mu poenostavili nadaljnjo uporabo aplikacije. Želeli smo doseči da se uporabniku ni več potrebno prijavljati v sistem z uporabniškim imenom in geslom. Uporabili smo komponento ASP.net profili, katera namesto nas hrani podatke o uporabniku. ASP.net profil nam omogoča tudi da za informacije o uporabniku ni potrebno več klicati baze, ampak so informacije o uporabniku dosegljive na globalni ravni, recimo ko želi uporabnik zamenjati geslo ali spremeniti nekaj osebnih nastavitev o samemu sebi. Na sliki 20 vidimo kako lahko spremenimo privzeti jezik aplikacije. Radi bi omenili še eno izmed lastnosti, ki smo jo uporabili v naši aplikaciji. Komponenta ASP.net profil nam omogoča tudi da shranimo podatke, katere si je uporabnik nazadnje ogledoval in mu le-te prikažemo, ko se naslednjič prijavi v sistem. S tem smo se izognili mučnemu klikanju uporabnika na prvih straneh, da bi čim prej prišel do želenih informacij.

```
public class ChangeLanguageCommand : ICommand
{
    private string resourceName;

    public ChangeLanguageCommand(string resourceName)
    {
        this.resourceName = resourceName;
    }
    public bool CanExecute(object parameter)
    {
        return parameter is string;
    }

    public event EventHandler CanExecuteChanged;

    public void Execute(object parameter)
    {
        string culture = (string)parameter;
        Thread.CurrentThread.CurrentCulture = new CultureInfo(culture);
        Thread.CurrentThread.CurrentUICulture = new CultureInfo(culture);
        ((PartnerResourcesBase)Application.Current.Resources[resourceName]).Refresh();
        FrameworkElement root = (FrameworkElement)Application.Current.RootVisual;
        root.Language = XmlLanguage.GetLanguage(culture);
    }
}
```

Slika 20: Prikaz funkcije za zamenjavo jezika

Ker smo se pri pridobivanju podatkov iz baze želeli izogniti utripanju spletnih strani pri vsakem klicu na podatkovno bazo, smo uporabili rešitev, ki jo nudi WCF (slika 21). WCF omogoča enoten vmesnik med različnimi transportnimi sistemi in je še posebno uporaben v primerih, kjer smo na enotnem uporabniškem vmesniku želeli prikazati podatke in datoteke iz različnih sistemov.

```
private readonly DataServiceClient client = DataServiceAgent.CreateDataServiceClient();
public TransactionsByPacketReportViewModel()
{
    _sortBy = new Dictionary<string, string>();
    client.GetDictionariesCompleted += OnGetDictionariesCompleted;
    client.GetTransactionsByPacketIdCompleted += OnGetTransactionsCompleted;
    client.GetPacketsCompleted += OnGetPacketsCompleted;
    client.GetPacketsByPacketIdCompleted += OnGetPacketsByPacketIdCompleted;

    PageSize = WebContext.Current.User.GridPageSize;
    if (PageSize == 0)
    {
        PageSize = 10;
    }
    Transactions = new PagedEntityCollectionView<Transaction>(this);
    Transactions.Refreshed += OnTransactionsRefreshed;

    Packets = new ContinuousSortableCollectionView<Packet>();
    Packets.OnRefresh += (s, e) => LoadPackets(false);
}
```

Slika 21: Funkcija za osveževanje podatkov na spletni strani

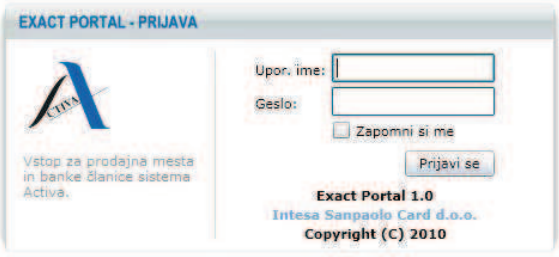
Silverlight tehnologija nam je omogočila, da smo razvili enostaven in vizualen spletni vmesnih za uporabnika, s pomočjo bogatih kontrol komponente WPF (slika 22). Izkazal se je za odlično orodje pri osveževanju spletnih strani, saj uporabnik praktično ne vidi, da se na spletni strani karkoli osvežuje, ker se vse dogaja v nekaj stotinkah sekunde. Prav tako Silverlight deluje na osnovi XML-ja in se zaradi tega dinamično prilagaja spremembah na spletnih straneh. Z pomočjo WPF-ja pa lahko v prihodnosti uporabnikom ponudimo namizno aplikacijo, ki se sploh ne bi odvijala v spletnem brskalniku, ampak bi jo uporabnik zagnal kar iz svojega namizja.

Slika 22: Prikaz uporabe WPF kontrol

6. IZGLED PORTALA PO PRENOVI

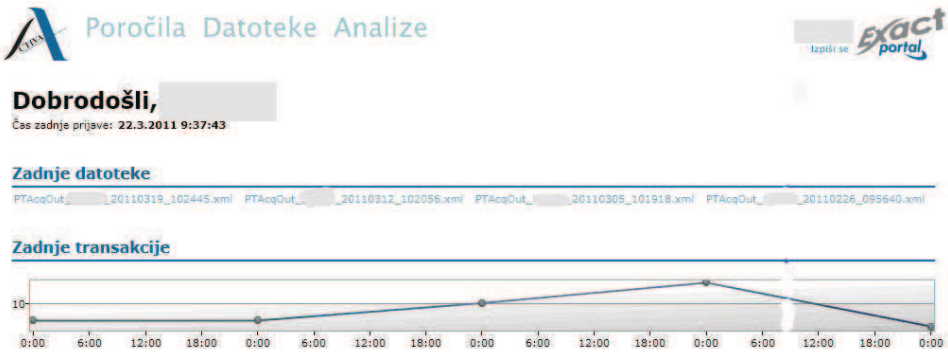
Dostop do podatkov na spletnem okolju je možen z dodeljenim uporabniškega imenom in geslom, ki ga je možno spreminjati.

Preko spletnega naslova uporabniki dostopajo na vstopno stran, kjer je obvezen vnos uporabniškega imena in gesla, kot kaže slika 23.



Slika 23: Prijava v nov Portal

Po vnosu ustreznega uporabniškega imena in gesla, se uporabnik prijavi v Portal. Tu se mu odpre osnovna ekranska maska (slika 24), kjer se izpiše pozdravno sporočilo. Pod pozdravnim sporočilom se izpišejo sveže datoteke, na katere je uporabnik naročen, ter zadnje transakcije, ki so bile opravljene na prodajnih mestih, katerih lastnik je ta trgovec oziroma prodajno mesto. Prav tako se uporabniku izpiše datum in čas zadnje uspešne prijave v sistem.



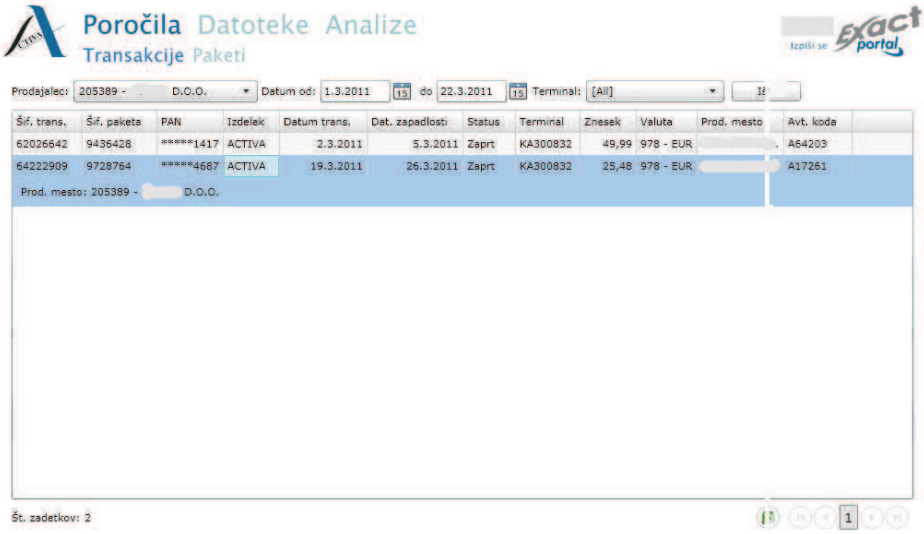
Slika 24: Osnovna ekranska maska po uspešni prijavi v sistem

V zgornjem delu ekrana so uporabniku na razpolago trije zavihki: Poročila, Datoteke in Analize.

6.1 Funkcije portala

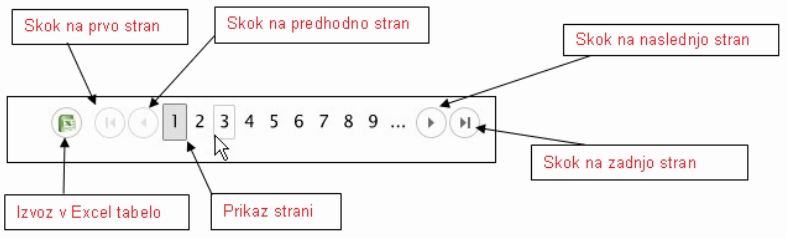
6.1.1 Zavihek poročila

Zavihek Poročila uporabniku omogoča pregledovanje transakcij in paketov. Iskanje transakcij je filtrirano z iskalnimi polji, kjer smo uporabili kontrole `TextBox` in `DropDownList`. S klikom na gumb Išči se izpiše seznam zadetkov, ki se izpišejo v kontrolo `GridView`, ki omogoča pregledovanje posameznih transakcij in njenih podrobnosti kot je vidno na sliki 25.



Slika 25: Iskanje transakcij in pregled detajlov

Pod rezultati iskanja se v levem spodnjem kotu vedno izpiše število zadetkov, v desnem kotu pa se nahajajo funkcijske tipke, ki omogočajo premikanje po straneh in izvoz prikazanih transakcij v Excelovo tabelo, kar je vidno na sliki 26.



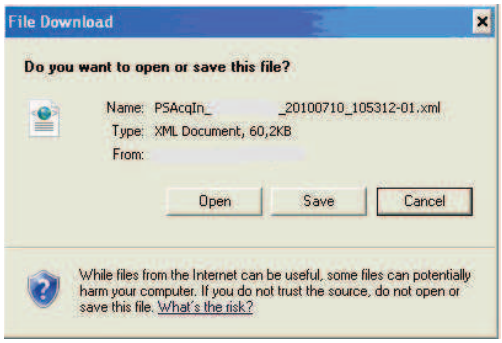
Slika 26: Navigacijski gumbi pod GridView kontrolo

6.1.2 Datoteke

S klikom na zavihek Datoteke se uporabniku izpiše seznam kreiranih datotek, ki je razvrščen po datumih v padajočem vrstnem redu. Na ekranu se izpišejo datoteke zadnjega tedna, ki se izdelujejo z avtomatiko po predhodnem naročilu in odlagajo na strežnik za poznejši pregled datotek. Iskanje datotek lahko tudi filtriramo z datumom kot je prikazano na sliki 27.

Slika 27: Prikaz vsebine zavihka Datoteke

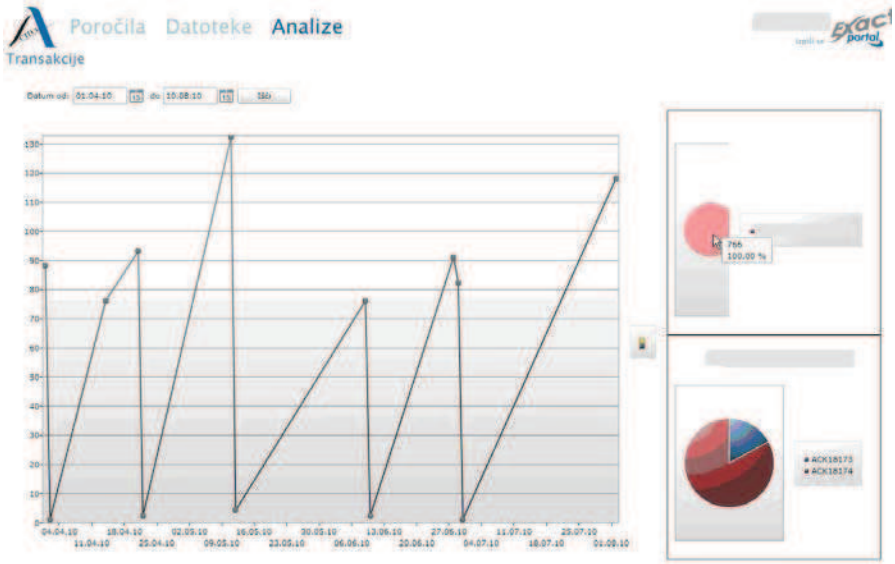
Po prikazu datotek v seznamu ima uporabnik omogočen vpogled v datoteke, ki so na strežniku shranjene v XML strukturi. Uporabnik z miško izbere datoteko, katero želi pregledovati in s klikom na levi gumb miške se mu odpre možnost pregleda ali shranjevanja te datoteke na disk (slika 28).



Slika 28: Prikaz dialoga za pregled izbrane datoteke

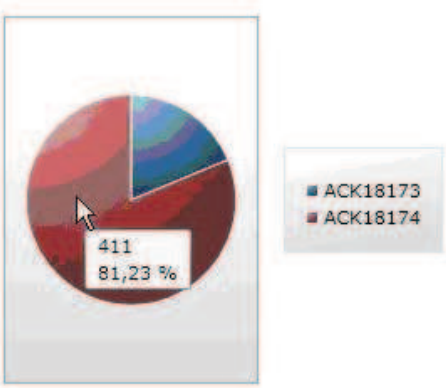
6.1.3 Analize

Zavihek Analize omogoča uporabniku grafični pregled transakcij po prodajnih mestih in terminalih. V zgornjem delu strani je filter po datumu, ki omogoča filtriranje transakcij za določeno obdobje. S klikom na gumb išči se uporabniku prikaže graf z rezultati (slika 29).



Slika 29: Prikaz zavihka Analize

Osrednji graf prikazuje število transakcij za določeno obdobje po dnevih. Leva stran linearnega grafa (na ordinatni osi) prikazuje število transakcij, medtem ko so na spodnji (abscisni) osi prikazani datumu. Na desni zgornji strani ekrana, graf prikazuje odstotek in število transakcij po prodajnih mestih. V kolikor je več prodajnih mest je ta prikazan kot tortni razrez, ki prodajna mesta z manj ali nič transakcijami združi skupaj. S klikom na izbrano prodajno mesto v grafu odpre uporabnik nov graf v spodnjem desnem kotu ekrana. Ta graf prikazuje število in odstotek transakcij po terminalih. S premikom miške na eden ali drugi terminal se prikazujejo podatki, ki ustrezajo izbranemu terminalu (slika 30).



Slika 30: Prikaz grafa

6.1.4 Nastavitve uporabnika

V zgornjem desnem kotu se uporabniku izpiše ime prodajnega mesta, s katerim se je prijavil v sistem. Ob kliku na ime se mu prikažejo uporabniške nastavitve, kjer lahko upravlja s svojimi nastavitvami kot so sprememba gesla in sprememba jezika, kar vidimo na sliki 31.

Slika 31: Upravljanje z nastavitvami uporabnika

7. ZAKLJUČEK

Cilj diplomske naloge je bila poleg primerjave obeh portalov in novih tehnologij tudi predstavitev novega portala. Predstavili smo posamezne funkcije portala, opisali uporabljene tehnologije in pokazali, kako smo tehnologije uspešno vpeljali v našo aplikacijo.

Razvoj je bil izveden v tehnologijah ASP.NET in Silverlight in bistveno lažji, predvsem zaradi hitrega razvoja, obširnega nabora uporabnih kontrol in večje dinamičnosti za končne uporabnike. Ogrodje Microsoft.NET je nudilo boljšo rešitev za vsako večjo težavo na katero smo naleteli. Tudi tehnologija Silverlight se je izkazala za dobro izbiro, saj število in zadovoljstvo uporabnikov portala narašča.

Pri izdelavi diplomske naloge smo se osredotočili na tehnologijo Silverlight, ki ima velik potencial in možnosti za uspeh. Ena izmed zanimivejših sposobnosti Silverlighta, ki sem jo opazil pri izdelavi diplomske naloge, je bil prehod spleta v namizje. Aplikacije, ki so bile razvite za splet je možno iz spleta prestaviti na namizje.

Med razvojem smo ohranjali stalno komunikacijo z končnimi uporabniki, kar se je izkazalo za uspešno. V aplikacijo smo vključili tudi tiste funkcionalnosti, ki jih star portal ni podpiral, in tiste, ki so bile pri prvotnih zahtevah precej nerazčiščene ter ne najbolj premišljene.

Razvoj aplikacije je trajal približno 4 mesece. Temu je botrovalo predvsem učenje razvoja v tehnologiji Silverlight, ki smo jo uporabili za razvoj aplikacije. Izbira omenjene tehnologije se je izkazala za odlično, saj so že v fazi testiranja uporabniki pokazali navdušenje nad enostavnostjo in izgledom novega portala, ki je preprost za uporabo ter podaja vse podatke na pregleden in vizualno privlačen način.

7.1 Analiza rešitve

Pri izdelavi diplomske naloge smo bili zadovoljni, da smo se odločili za tehnologijo Silverlight, saj le-ta ponuja izredno veliko možnosti za učenje in razvoj spletnih aplikacij. Uporabniki, ki uporabljajo aplikacijo so bili enotni, da aplikacija izpolnjuje zahteve v celoti in jih celo preseže, saj smo med razvojem dodali nekaj dodatnih izboljšav, ki jim bo nudilo boljšo in celovitejšo izkušnjo.

Hitrostne performance aplikacije so se izkazale za zelo hitre. Morda je potrebno počakati kakšno stotinko sekunde več zaradi tehnologije Silverlight, ker le-ta zahteva nekaj več pomnilnika.

Za izboljšanje aplikacije bi lahko vsekakor dodali še nekaj modulov. Eden izmed njih je pregled izpiskov in tiskanje v takšni obliki, kot jo prodajna mesta in banke prejmejo po navadni pošti. Vsekakor bi bila zanimiva funkcionalnost pošiljanje določenih analiz, pregledov in datotek po elektronski pošti.

Aplikacija trenutno deluje že kar nekaj časa. Kot zanimivost naj dodamo, da jo uporabljajo tudi v tujih državah, kot so Egipt in Albanija, največje število uporabnikov pa je še vedno v Sloveniji.

Za konec naj dodamo, da tehnologija Silverlight dobiva vse več posodobitev, in da bo Microsoft v prihodnje poslal na trg tudi novo verzijo Silverlighta, ki nam bo zagotovo ponudila več funkcionalnosti, kar bomo zagotovo izkoristili pri dodatnem razvoju aplikacije.

VIRI IN LITERATURA

[1] (2011) Arhitektura Silverlight modela. Dostopno na: http://www.codeproject.com/KB/silverlight/ASilverlightIntroduction.aspx

[2] (2011) ASP.NET MVC. Dostopno na: http://en.wikipedia.org/wiki/ASP.NET_MVC_Framework

[3] (2011) ASP.NET Profile Properties. Dostopno na: http://msdn.microsoft.com/en-us/library/2y3fs9xs.aspx

[4] (2011) Introducing to Membership. Dostopno na: http://msdn.microsoft.com/en-us/library/yh26yfzy.aspx

[5] J. Ambrose Little, Jason Beres, Grant Hinkson, Devin Rader, Joseph Croney, Silverlight 3, Prodgammer's Reference. Wrox, 2009. ISBN-13: 978-0470385401

[6] Laurence Moroney, Introducing Microsoft Silverlight 2. Microsoft press, 2008. ISBN13: 978-0735625280

[7] Laurence Moroney, Introducing Microsoft Sliverlight 3. Microsoft press, 2009. ISBN13: 978-0735625730

[8] Macdonald Matthew, Pro Silverlight 3 in C#. Apress, 2009. ISBN13: 978-1-4302-2381-8

[9] Michail Ashraf, Essential Silverlight 3. Addison-Wesley Professional, 2009. ISBN-13: 978-0321554161

[10] (2011) Microsoft .NET Framework. Dostopno na: http://en.wikipedia.org/wiki/.NET_Framework

[11] (2011) Microsoft Visual Studio 2008. Dostopno na: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[12] (2011) Profiles in ASP.NET 2.0. Dostopno na: http://odetocode.com/articles/440.aspx

[13] (2011) Sestava XML. Dostopno na: http://en.wikipedia.org/wiki/XML

[14] (2011) Shema ogrodja 3.5. Dostopno na: http://amitgpandey.com/articles/?p=13đ

[15] (2011) WPF Introduction. Dostopno na: http://www.rajneeshnoonia.com/blog/2010/03/wpf-introduction/

[16] (2011) XAML. Dostopno na: http://msdn.microsoft.com/en-us/library/ms752059.aspx

[17] (2011) XML. Dostopno na: http://office.microsoft.com/sl-si/infopath-help/xml-HP001096728.aspx

[18] (2011) Zgradba platforme WCF. Dostopno na: http://thetechjungle.blogspot.com/