Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Luka Žgur

Razvoj mobilne aplikacije mFri

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00207/2012
Datum: 02.04.2012

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

Kandidat: **LUKA ŽGUR**

Naslov: **RAZVOJ MOBILNE APLIKACIJE MFRI**
THE DEVELOPMENT OF MFRI MOBILE APPLICATION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvijte mobilno aplikacijo mFRI, ki študentom Fakultete za računalništvo in informatiko omogoča uporabo storitev spletne Učilnice na platformi Moodle. Mobilna aplikacija naj poleg tega študentom omogoča tudi vpogled v ponudbo za študentske bone fakulteti bližnjih restavracij ter vozni red avtobusom mestnega prometa na bližnjih postajah. Mobilno aplikacijo razvijte na platformi Windows Mobile.

Mentor:

doc. dr. Rok Rupnik

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Luka Žgur, z vpisno številko **63050136**, sem avtor diplomskega dela z naslovom:

Razvoj mobilne aplikacije mFri

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Rok Rupnik,
- 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 5. maja 2012 Podpis avtorja:

Zahvala

Zahvaljujem se staršem, ki so mi omogočili študij, in vsem, ki so mi kakorkoli pomagali pri diplomskem delu.

Kazalo

Povzetek

Diplomsko delo predstavlja razvoj aplikacije za mobilno platformo Windows phone 7, ki je namenjena študentom Fakultete za računalništvo in informatiko v Ljubljani. V osnovi aplikacija študentom ponuja informacije o obveznostih in urniku, ki se nahajajo na študentskem portalu Spletna učilnica Fakultete za računalništvo in informatiko. Nadalje aplikacija omogoča tudi vpogled v ponudbo restavracij s študentsko prehrano, pregled avtobusnih voznih redov Ljubljanskega potniškega prometa ter pregled filmov, ki se trenutno predvajajo v kinodvoranah. V diplomskem delu je podrobneje opisan razvoj in delovanje aplikacije ter pripadajočega strežnika ter opis orodij in tehnologij, ki so bila uporabljena za razvoj celotne rešitve.

Ključne besede:

Windows phone 7, mobilna aplikacija, mFri, Spletna učilnica

Abstract

The thesis describes the development of mobile application for Windows phone 7 platform, designed for students of Faculty of computer and information science in Ljubljana. Main purpose of application is to provide students the information about their tasks and schedule from student portal of Faculty of computer and information science named Spletna učilnica. Application also provide information about restaurants which offer meals with student vouchers, bus schedules of Ljubljana city public transport and movies currenty playing in cinemas in Ljubljana. The development and how mobile application and corresponding web server work and also usage of tools and technologies used, is described in details through the whole thesis.

Keywords:

Windows phone 7, mobile application, mFri, student web portal Spletna učilnica

Seznam uporabljenih simbolov in kratic

LMS - learning management system; sistem za upravljanje z znanjem
WP7 - Windows phone 7; operacijski sistem Windows phone 7
XAML - extensible application markup language; deklarativen jezik za shranjevanje vrednosti
XML - extensible markup language; deklarativen jezik za shranjevanje vrednosti
CLR - common language runtime; skupno izvajalno okolje
BCL - base class library; knjižnjica razredov v .NET
SOA - service-oriented architecture; storitveno usmerjena arhitektura
SOAP - simple object access protocol; protokol za izmenjavo informacij
JSON - javascript object notation; objektna notacija javascript
HTML - hyper text markup language; označevalni jezik
SQL - structured query language; sestavljeni jezik za poizvedbe
DOM - data object model; pravila za prikaz in interakcijo z HTML objekti
UML - unified modeling language; poenoteni jezik za modeliranje

Poglavje 1

Uvod

Danes je brez tehnologije skoraj nemogoče živeti, saj nas spremlja na vsakem koraku. Najboljši primer naprav, ki so se je v zadnjih tridesetih letih najbolj tehnološko razvile, so zagotovo računalniki. Ti postajajo vse zmogljivejši, manjši in energijsko manj potratni. Prav zaradi teh lastnosti so se računalniki preselili tudi v mobilne telefone, ki jih zaradi vgrajenega računalnika imenujemo kar pametni mobilni telefoni. Na voljo so nam šele dobro desetletje in njihova prodaja še vedno strmo narašča. Njihove zmogljivosti dosegajo zmogljivosti osebnih računalnikov, zato področje uporabe ni omejeno samo na telefonske pogovore in pošiljanje kratkih sporočil, ki so nam jih ponujali navadni telefoni. Pametne telefone lahko uporabljamo za nešteto stvari, kot so na primer brskanje po svetovnem spletu, prejemanje in pošiljanje elektronske pošte, kot fotoaparat za zajem videoposnetkov in fotografij in drugo. Število možnosti uporabe je ogromno in je omejeno samo s številom idej, ki jih prenesemo v obliki aplikacij na telefon.

Raziskava, opravljena med marcem in julijem leta 2011, je pokazala, da je odstotek pametnih telefonov v Evropi med 18 in 33 odstotkov [1]. V Sloveniji naj bi bil delež pametnih telefonov malo pod 30 odstotkov [2]. Po raziskavi, ki jo je izvedel Nilsen v februarju 2012, je odstotek uporabnikov pametnih telefonov v ZDA že 49 odstotkov, medtem ko je bil še leto poprej delež samo 36 odstotkov [3]. Vidimo, da je rast deleža pametnih telefonov izjemna.

Na trgu mobilnih naprav imamo več operacijskih sistemov. Po Gartnerjevih statističnih podatkih, objavljenih za zadnjo četrtino leta 2011, vidimo, da je najbolj priljubljen operacijski sistem Android z 50.9 % tržnega deleža. Sledijo mu Applov iOS (23.8 %), Symbian (11,7 %), Research in motion (8,8 %), Bada (2,1 %), združena Microsoftova Windows Mobile in Windows Phone 7 (1,9 %) [4].

Diplomsko delo se osredotoča na razvoj mobilne aplikacije mFri, ki študentom s telefoni, ki imajo nameščen operacijski sistem Windows Phone 7, omogoča pregled obveznosti in urnika na fakulteti, pregled restavracij s študentsko prehrano, avtobusnih voznih redov Ljubljanskega potniškega prometa ter filmov, ki se trenutno predvajajo v kinodvoranah. V drugem poglavju je predstavljen sistem Spletne učilnice FRI in njegove značilnosti. V naslednjem poglavju spoznamo orodja in tehnologije, ki so bile uporabljene za razvoj rešitve. V četrtem poglavju pa je predstavljeno, kako rešitev deluje in kako smo jo razvili.

Poglavje 2

Spletna učilnica

2.1 Kaj je Moodle?

Moodle je odprtokodni programski paket za upravljanje učnih vsebin na spletu [5]. Uvrščamo ga med sisteme za upravljanje s poučevanjem (LMS). Preveden je tudi v slovenščino in uporablja ga veliko izobraževalnih ustanov po Sloveniji, med drugim tudi Fakulteta za računalništvo in informatiko v Ljubljani. V slovenščini se za Moodle pogosto uporablja izraz spletna učilnica. Beseda Moodle je akronim za Modular Object-Oriented Dynamic Learning Environment (Modularno objektno-orientirano dinamično učno okolje). Moodle namestimo na spletni strežnik, ki ima podporo skriptnemu jeziku PHP in SQL podakovnim bazam. Izvajalcem izobraževanj omogoča nalaganje učnih gradiv, datotek, izvedbo kvizov (preverjanje znanja), dodajanje domačih nalog, izvedbo anket... Učenci lahko oddajajo domače naloge, pregledujejo ocene, pregledujejo učna gradiva. Udeleženci med seboj ter izvajalci komunicirajo preko vgrajenega foruma, komentarjev ali pa direktnih sporočil.

2.2 Spletna učilnica FRI

Za uporabo Spletne učilnice Fakultete za računalništvo in informatiko (FRI) potrebujemo spletni brskalnik s povezavo v internet. Študent se v sistem vpiše z digitalno identiteto. Digitalna identiteta je kombinacija uporabniškega imena in gesla, ki jo študent pridobi ob vpisu na Univerzo v Ljubljani in ga enolično opisuje. Na začetni strani so povezave s predmeti, ki so razdeljeni po stopnjah in po letnikih študija. Predmeti so v učilnici predstavljeni kot mreža spletnih strani s povezavami na podstrani. Na straneh predmetov izvajalci navedejo osnovne podatke o predmetu, kot so urnik predavanj in vaj, obveznosti, ki jih imajo študenti pri predmetu, govorilne ure, izpitni roki ter gradiva predavanj in vaj. Učilnica omogoča, da študenje aktivno sodelujejo pri predmetu, tako da opravljajo domače naloge in se udeležujejo elektronskega preverjanja znanj. Pri nekaterih predmetih se preverjanja vaj izvajajo izključno v spletni učilnici, tako da študenje rešijo elektronski kviz iz katerega prejmejo oceno. Študent ima direkten vpogled v trenutno dogajanje na podstrani Moj dom. Tu so za vsak predmet, v katerega je prijavljen, kronološko prikazane obveznosti in pomembna obvestila. Trenutno dogajanje je prikazano tudi v koledarju, na katerem so dnevi, ki vsebujejo obveznost obarvani, da jih študent lažje vidi.

Poglavje 3

Orodja in tehnologije uporabljena pri razvoju aplikacije

3.1 Windows phone 7

3.1.1 Osnovni podatki

Windows Phone 7 (WP7) je najnovejši Microsoftov operacijski sistem za mobilne platforme, ki ga je Microsoft prvič predstavil v začetku leta 2010. Njegov predhodnik Windows Mobile 6 je v primerjavi z uspešnimi mobilnimi operacijskimi sistemi, kot sta Googlov Android in Applov iOS, že precej zaostajal. Microsoft je naslednika Windows Mobile 6, ki naj bi se imenoval Windows Mobile 7, začel razvijati že leta 2004, vendar se je zaradi slabih odločitev pri razvoju odločil opustiti razvoj le-tega in na novo razviti operacijski sistem. Leto in pol po izzidu so prodajni rezultati precej skromni. V zadnjem četrtletju leta 2011 je imel Microsoft samo 1.9% delež vseh prodanih telefonov po svetu [4]. Velike upe Microsoft polaga v sodelovanje s finsko Nokio, s katero so sklenili ekskluzivno pogodbo o sodelovanju. Večkrat so se tudi pojavile govorice, da bo Microsoft prevzel mobilni del Nokie [6].

V začetku decembra 2011 je bilo v Windows Phone Marketplace na voljo 40.000 aplikacij. Glede na to, da jih ima Applov App Store 300.000, Android Market pa 200.000, je to majhna številka, vendar pa je WP7 star šele dobro leto, tako da to niti ni tako majhna številka.

3.1.2 Uporabniški vmesnik Metro

V WP7 je Microsoft razvil nov uporabniški vmesnik, ki je baziran na oblikovalskem jeziku, poimenovanem Metro. Vmesnik izgleda zelo moderno in ima velik poudarek na tipografiji ter uporablja velik tekst, ki uporabniku hitro pade v oči. Namesto ikon, ki jih uporabljata sistema iOS in Android, so tu tako imenovane »žive ploščice«, ki lahko služijo kot bližnjice do aplikacij ali pa dinamično predstavljajo podatke v realnem času. Zanimiva lastnost uporabniškega vmesnika je tudi t.i. panorama, to je širjenje vsebine čez meje fizičnega zaslona, do katere dostopamo z drsanjem vsebine levo ali desno po zaslonu. Meniji v sistemu so zasnovani tako, da se lahko do vsake nastavitve dostopa s kar najmanj kliki.

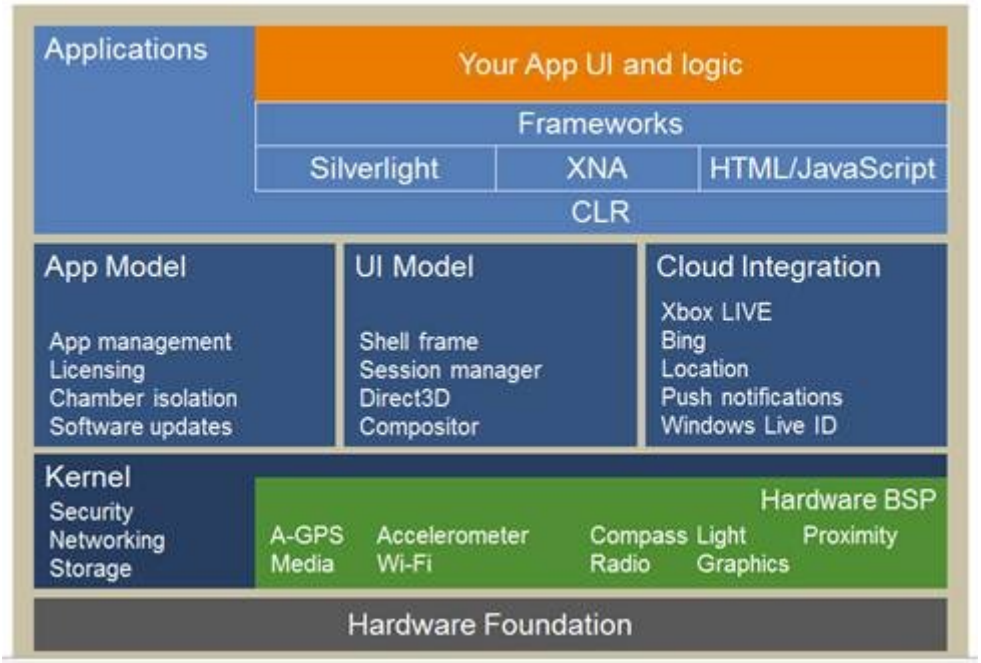
3.1.3 Strojna oprema

Zaradi zagotovljanja tekočega delovanja na vseh mobilnih telefonih se je Microsoft odločil, da bo imel v WP7 večjo vlogo pri določanju, kakšna strojna oprema bo poganjala njihov operacijski sistem, zato so postavili stroge minimalne zahteve. V drugi polovici leta 2011 so minimalne strojne zahteve nekoliko znižali predvsem zaradi cenovne dostopnosti telefonov ter uporabe v poslovne namene, kjer kamere niso vedno zaželjene.

3.1.4 Arhitektura

WP7 je zgrajen na jedru Windows Embedded Compact 6, operacijskem sistemu prilagojenem za delovanje na mobilnih napravah in vgrajenih sistemih. Kot je razvidno z diagrama na sliki 3.1, WP7 sestavljata dve plasti programskih komponent: uporabniške plasti ter jedra, ki tečeta na vrhu strojne

plasti.



Slika 3.1: Arhitektura Windows phone 7

Jedro (angl. Kernel)

Vsebuje komponente, kot so datotečni sistem, mreženje, varnost ter gonilnike za strojno opremo, ki jih v večini Microsoft sam napiše, tako da je operacijski sistem lažje prenosljiv na vse strojne platforme.

Uporabniška plast

- Aplikacije (angl. Applications)

  Osnovne aplikacije, kot so na primer aplikacije za klicanje, pošiljanje kratkih sporočil, imenik, brskalnik, so že prednaložene, ostale pa si lahko uporabnik sam naloži na mobilni telefon.

- Aplikacijski model (angl. App Model)

  V aplikacijskem modelu se nahajajo komponente za licenciranje aplikacij, oddaljeno posodobitev sistema, model za namestitev aplikacij na telefon, varnostni paket.

- Uporabniški vmesnik (angl. UI Model)

  Vsebuje komponente, kot so grafična lupina, izrisovanje tridimenzionalne grafike, prehajanje med stranmi vmesnika, upravljanje prehodov med stranmi vmesnika.

- Oblačne storitve (angl. Cloud Integration)

  Integracija z Microsoftovimi storitvami kot so Xbox Live, Bing, Windows Live ID, lokacijske storitve, potisna sporočila.

3.1.5 Razvoj aplikacij

Microsoft za razvoj aplikacij na WP7 ni razvijal novih programskih jezikov, ampak je za ogrodje uporabil in prilagodil svoje .NET Framework razvijalsko okolje, tako da aplikacije pišemo v jeziku C#. Pri oblikovanju uporabniškega vmesnika ima razvijalec možnost uporabe dveh ogrodij, Silverlight za mobilne telefone in XNA. Silverlight je podobno kot Adobe Flash vtičnik v spletnem brskalniku, ki omogoča izdelavo strani z bogato vsebino. Uporablja deklarativen jezik XAML, razširjen jezik XML, ki je zelo uporaben za izgradnjo uporabniškega vmesnika. Pomembna lastnost Silverlighta je, da ne omogoča dostopa do operacijskega sistema, temveč samo do podatkov, ki mu jih damo na voljo, zato je primeren za razvijanje poslovnih aplikacij. XNA je razvijalsko okolje in orodja, ki so namenjena izdelavi iger na platformah Xbox, Windows Phone, Zune, zatorej je prva izbira, če želimo razviti igro za WP7.

3.2 .NET

Ogrodje .NET (.NET Framework) je programska platforma, ki je ogrodje za aplikacije in orodja napisana v .NET okolju. V operacijski sistem se ga namesti kot dodatek in omogoča lažjo izdelavo aplikacij ter izvajanje leteh. Nudi podporo za večje število programskih jezikov in daje možnost, da so deli programske kode lahko napisani v različnih jezikih. Zgrajen je iz dveh glavnih komponent: skupnega izvajalnega okolja (CLR), v katerem se izvajajo napisani programi in knjižnjice razredov .NET (BCL).

Skupno izvajalno okolje je programski vmesnik, ki upravlja s programsko kodo in jo prevaja v času izvajanja ter poskrbi za pomembne servise, kot so upravljanje s pomnilnikom, varnost, upravljanje niti in drugo. Izvorna koda se ne prevede direktno v strojno kodo, ampak v vmesni jezik (angl. intermediate language), ki je enak za vse programske jezike, ki se uporabljajo v .NET okolju. To v teoriji pomeni, da lahko uporabimo katerikoli programski jezik. Kodo, pretvorjeno v vmesni jezik, v strojno kodo prevede JIT prevajalnik (angl. just in time).

Knjižnjica razredov .NET (BCL) je skupek objektno orientiranih razredov, ki programerju olajšajo in pohitrijo pisanje programske kode. Večina razredov je tako napisana, da jih lahko razširimo s svojo kodo. Vsebuje razrede za povezovanje na bazo podatkov in jemanje podatkov iz baze, uporabniški vmesnik, mrežno komunikacijo, upravljanje z datotekami, izgradnjo spletnih aplikacij.

Z .NET ogrodjem lahko razvijamo različne tipe aplikacij:

- konzolne aplikacije,
- aplikacije z uporabniškim vmesnikom (Windows Forms),
- aplikacije z uporabniškim vmesnikom, ki temeljijo na Windows Presentation Foundation (WPF),
- spletne aplikacije v ASP.NET,

- spletne servise,
- windows servise,
- servisno orientirane aplikacije, ki temeljijo na Windows Communication Foundation (WCF).

3.2.1 Windows Communication Foundation

WCF (Windows Communication Foundation) je model, ki ga je Microsoft izdal za gradnjo storitveno usmerjenih aplikacij [9]. Storitveno usmerjena arhitektura (SOA) je programski pristop, v katerem je aplikacija organizirana v funkcionalne dele programske kode z določenim obnašanjem, ki jih imenujemo storitve. Storitve so locirane na spletnih strežnikih, tako da se odjemalci storitev povezujejo do njih in jih uporabljajo preko omrežja. WCF je zato ogrodje za ustvarjanje porazdeljenih povezanih aplikacij. Za komunikacijo med klientom in strežnikom uporablja protokol SOAP (angl. simple object access protocol) in dokumente XML, ki poskrbijo za hierarhičen prenos podatkov. Microsoft pri prenosu omogoča tudi protokol JSON, saj je njegova uporaba pri asinhronih zahtevah po podatkih na strežnikih postala zelo pogosta. Za opis storitve uporablja WSDL (angl. web service description language), xml dokument, ki pove, kje se storitev nahaja ter katere metode oziroma operacije storitev ponuja. Ker WCF uporablja protokole, katerih opis je prosto dostopen, omogoča komunikacijo tudi z drugimi platformami in tehnologijami v različnih operacijskih sistemih.

3.2.2 ASP.NET

ASP.NET je Microsoftova tehnologija za razvoj sodobnih spletnih strani, spletnih storitev in spletnih aplikacij. S prihodom .NET ogrodja je zamenjal predhodnika, skripti jezik ASP (angl. active server pages). Ker temelji na ogrodju .NET, lahko uporablja vse dobre lastnosti ogrodja .NET, kot sta na primer uporaba knjižnjice razreda .NET (BCL) in številnih podprtih

programskih jezikov. Za razliko od svojega predhodnika je ASP.NET ločil prezentacijsko od izvajalne programske kode. Ideja ASP.NET je, da lahko programerji tako kot v razvoju za Windows okolje enostavno nizajo spletne kontrole na spletno stran, urejajo njihove lastnosti in se v izvajalni kodi v ozadju dinamično odzivajo na dogodke, ki jih sprožijo kontrole. Prav zato Microsoft in tudi drugi ponudniki za razvoj v ASP.NET ponujajo velik nabor zmogljivih kontrol. Življenjski cikel ASP.NET strani je točno določen vrstni red izvajanja dogodkov in kontrol na strani. ASP.NET spletne strani so datoteke, v katerih se nahaja HTML ter spletne kontrole in imajo končnico .aspx. Izvajalna koda v ozadju se nahaja v datotekah s končnico, ki ustrezajo uporabljenemu programskemu jeziku.

Spletni brskalnik zna prikazati samo navaden HTML, ne pa tudi ASP.NET strani. V primeru ASP.NET spletne strani spletni brskalnik na IIS strežnik pošlje zahtevo in prepozna, da je zahtevana stran napisana v ASP.NET, zato jo preda v ASP.NET izvajalno okolje. Izvajalno okolje strežniške kontrole, ki se nahajajo na ASP.NET strani obdela in odjemalcu vrne stran v čistem HTML, ki ga zna spletni brskalnik prikazati.

3.2.3 Html Agility Pack

Html Agility Pack je zastonjska .NET knjižnjica, ki naredi brisanje, pisanje in razčlenjevanje HTML dokumentov enostavnejše [8]. Omogoča pridobivanje HTML dokumenta s spletne strani in nato izgradnjo celotnega DOM (angl. document object model) za pridobljeni HTML dokument. Knjižnjica je zelo tolerantna do nepravilno napisanih HTML dokumentov. Ponuja podobne funkcionalnosti, kot jih ponujajo XML razredi v .NET ogrodju, vendar za HTML dokumente. Omogoča enostavno premikanje med vozlišči zgrajenega dokumenta, za njihovo iskanje pa uporablja jezika Xpath in XSLT, ki se uporabljata tudi v dokumentih XML. V knjižnjici lahko uporabljamo tudi jezik LINQ to objects, ki zelo poenostavi pisanje poizvedb nad kolekcijami podatkov, kot so Html Agility Pack vozlišča. Html Agility Pack v projektih uporabimo tako, da dodamo HtmlAgilityPack.dll knjižnjico in nanjo v

projektu dodamo referenco.

3.3 Orodja

3.3.1 Visual Studio 2010

Visual Studio je Microsoftovo razvojno okolje, namenjeno razvoju aplikacij. V njem se razvija aplikacije, ki podpirajo .NET ogrodje, operacijski sistem Windows, Microsoftove mobilne operacijske sisteme, Silverlight in drugo.

Glavne komponente razvojnega okolja VS 2010 so:

- urejevalnik programske kode s podporo za samodejno zaključevanje kode (angl. intellisense) ter barvanje programske kode glede na tip vnosa (angl. syntax highlighting),
- orodje za načrtovanje programskih vmesnikov. Podpora za spletne, windows in WPF aplikacije ter urejevalnik za podatkovne baze,
- orodjarna,
- razhroščevalnik, s katerim sledimo izvajanju programske kode, za lažje iskanje napak v kodi. Deluje z vsemi programskimi jeziki, podprtimi v VS 2010,
- pregledovalnik dokumentov, objektov,
- prevajalnik.

Visual Studio omogoča tudi dodajanje vtičnikov in dodatkov, ki programerjem olajšajo in pospešijo razvoj aplikacij.

3.3.2 Microsoft Windows Server 2008

Microsoft Windows Server je operacijski sistem, namenjen strežnikom. Sistem je zasnovan modularno. Določimo mu lahko eno ali več strežniških vlog

(angl. roles), glede na katere se v sistem naloži vse potrebno za delovanje izbranih vlog. Vloge določajo, čemu bo strežnik služil, npr. spletni, domenski ali terminalski strežnik.

Microsoft ponuja veliko število različic Windows Serverja, glede na velikost podjetja, števila uporabnikov sistema ali namembnosti.

3.3.3 SQL Server Management Studio

SQL Server Management Studio je razvojno okolje, namenjeno dostopanju, konfiguriranju, administraciji in razvoju vseh komponent v SQL Server. Vsebuje različna grafična orodja in urejevalnike besedil, v katerih se piše SQL skripte. Obstaja tudi SQL Server Management Studio Express različica, ki jo Microsoft ponuja zastonj.

3.3.4 SQL Server

Microsoft SQL server je Microsoftov strežnik za upravljanje relacijskih podatkovnih baz. Njegova osnovna naloga je shranjevanje in pridobivanje podatkov iz podatkovne baze, ki se nato uporabijo v različnih aplikacijah. Za pisanje poizvedb po podatkovni bazi uporablja izpeljanko jezika SQL, poimenovano T-SQL (Transact SQL). Največ se uporablja v manjših in srednje velikih podjetjih, predvsem zaradi cenovne ugodnosti v primerjavi z večjimi ponudniki relacijskih podatkovnih baz.

Za izdelavo manjših projektov in učenje je Microsoft ponudil zastonjsko različico Microsoft SQL Server Express. V primerjavi z SQL Server ima precej tehničnih omejitev, ki onemogočajo uporabo za večje projekte: podatkovna baza je omejena na 4GB, delovanje procesorja samo z enim jedrom, maksimalna uporabljena velikost pomnilnika je omejena na 1GB in drugo.

3.3.5 Apache Subversion (SVN)

Apache Subversion je sistem za nadzor različic dokumentov. Ker je izdan pod odprtokodno licenco, je njegova uporaba brezplačna. Uporablja se za

upravljanje datotek skozi čas, tako da si zapomni vse spremembe, ki se zgodijo na datotekah. To omogoča, da lahko pridobite starejšo verzijo datoteke, če jo pomotoma pobrišemo.

Poglavje 4

Razvoj aplikacije

4.1 Načrtovanje in arhitektura

4.1.1 Načrtovanje razvojnih procesov

Načrtovanje razvojnih procesov pri razvoju aplikacij je pomemben del razvoja, če želimo ustvariti kakovosten produkt v doglednem času. Programerjem in projektnim vodjem zagotavlja stabilnost, kontrolo in organizacijo nad razvojem. Poznamo več različnih modelov procesov razvoja programske opreme, da lahko natančno opredelimo, kako bo potekalo razvijanje produkta.

Pri razvoju smo se odločili za tehniko razvoja programske opreme s prototipi. Do take odločitve je prišlo, ker nam niso bili znani vsi detajli kako bo aplikacija delovala in izgledala, pa tudi ker v razvoju mobilnih aplikacij za Windows phone 7 nismo imeli nobenih predhodnih izkušenj in smo nekaj tehnologij, uporabljenih v aplikaciji, šele prvič uporabljali v praksi. Kot prvo funkcionalnost smo oblikovali začetni grafični vmesnik, ki nam je služil kot podlaga ter nato na tem gradili vse ostale funkcionalnosti. Aplikacijo smo skozi tok sprememb, izboljšav in novih funkcionalnosti razvili v končni produkt.

4.1.2 Arhitektura

Aplikacija mFri je zgrajena po principu odjemalca in strežnika. Strežnik je sestavljen iz spletnega strežnika, podatkovne baze, spletnega servisa, odjemalec pa sta mobilna aplikacija za mobilni operacijski sistem Windows phone 7 ter spletna aplikacija za vnos reklamnih sporočil v mobilno aplikacijo. Komunikacija med odjemalcem in strežnikom poteka preko storitveno usmerjene arhitekture v primeru mobilne aplikacije, medtem ko je pri spletni aplikaciji uporabljena trinivojska arhitektura. Če uporabnik pri uporabi mobilne ali spletne aplikacije želi določene podatke, mu jih, če so na voljo, prikaže, v nasprotnem primeru pa pošlje zahtevo po podatkih na strežnik. Strežnik zahtevo obdela in v primeru, da odjemalcu lahko zagotovi podatke, mu jih pošlje in aplikacija uporabniku primerno prikaže podatke.

4.2 Strežnik

Zajemanje podatkov iz spletnih strani in manipuliranje s podatki iz podatkovne baze je procesorsko za odjemalca, kot je mobilna naprava, zelo zahtevno. Zato smo vse procesorsko zahtevne naloge opravili na strežniku, na mobilni napravi pa prikazovali podatke. Za strežnik smo želeli, da bi podatke lahko serviral tudi napravam, ki ne temeljijo na Microsoft tehnologijah. Uporabili smo ogrodje WCF, ki implementira storitveno usmerjeno arhitekturo. Za potrebe naše mobilne aplikacije, napisane za operacijski sistem Windows phone 7 smo razvili storitve, ki temeljijo na SOAP protokolu. Soap za komunikacijo med strežnikom in mobilno uporablja sporočila v obliki dokumentov XML.

4.2.1 Delovanje

Strežnik za delovanje potrebuje operacijski sistem Windows z nameščenim podatkovnim strežnikom Microsoft SQL 2008. Strežniška programska koda je napisana v programskem jeziku C#, zato mora biti na strežniku nameščeno

ogrodje .NET verzije 4.

Strežnik

Glavna naloga strežnika je, da iz podatkovne baze, spletne učilnice in spletne strani s podatki o avtobusnih prihodih pridobiva zahtevane podatke in jih preko storitev vrača odjemalcu. Urnik in tekoče obveznosti študenta strežnik na zahtevo odjemalca v realnem času pridobi iz spletne učilnice.

Podatki o študentski prehrani in kino predstavah se preberejo iz podatkovne baze, kamor jih shrani v nadaljevanju opisani servis. Nato jih strežnik preko zahtevane storitve posreduje odjemalcu. Avtobusne prihode za izbrano postajališče strežnik pridobi na zahtevo odjemalca iz spletne strani http://wbus.telargo.com. V primerni obliki urejene podatke pošlje odjemalcu. Strežnik odjemalcu na zahtevo pošilja tudi reklame, ki jih prebere iz podatkovne baze, v katero jih shranimo preko spletne aplikacije, opisane v poglavju 4.3.1.

Servis

Za zajemanje podatkov o študentski prehrani in kino predstavah smo napisali aplikacijo, ki iz spletnih strani zajema podatke o restavracijah in filmih, ki se trenutno predvajajo v kinu. Za zajem podatkov iz spletne strani moramo spletno stran razčleniti in podatke obdelati ter shraniti v čimbolj pregledni in čitljivi obliki. Razčlenjevanje spletnih strani, na katerih se nahaja zelo veliko podatkov, kot je v našem primeru, je na splošno zelo kompleksen in zamuden proces. Zato smo se odločili, da podatkov ne bomo pridobivali v realnem času, ker bi uporabnik čakal preveč časa, preden bi mu bili podatki prikazani. Servis pridobi podatke s spletnih strani s študentsko prehrano www.studentska-prehrana.si in kino predstavami na www.kolosej.si. Za študenstko prehrano pridobimo podatke samo o restavracijah v Ljubljani in Sežani, kjer se nahaja Fakulteta za računalništvo in informatiko. Podatke o kino predstavah pridobivamo samo za Ljubljano. Servis deluje tako, da se poganja znova in znova na izbran časovni interval. Ob vsakem zagonu prebere

spletni strani, ju razčleni in podatke v pregledni obliki shrani v podatkovno bazo.

4.2.2 Arhitektura

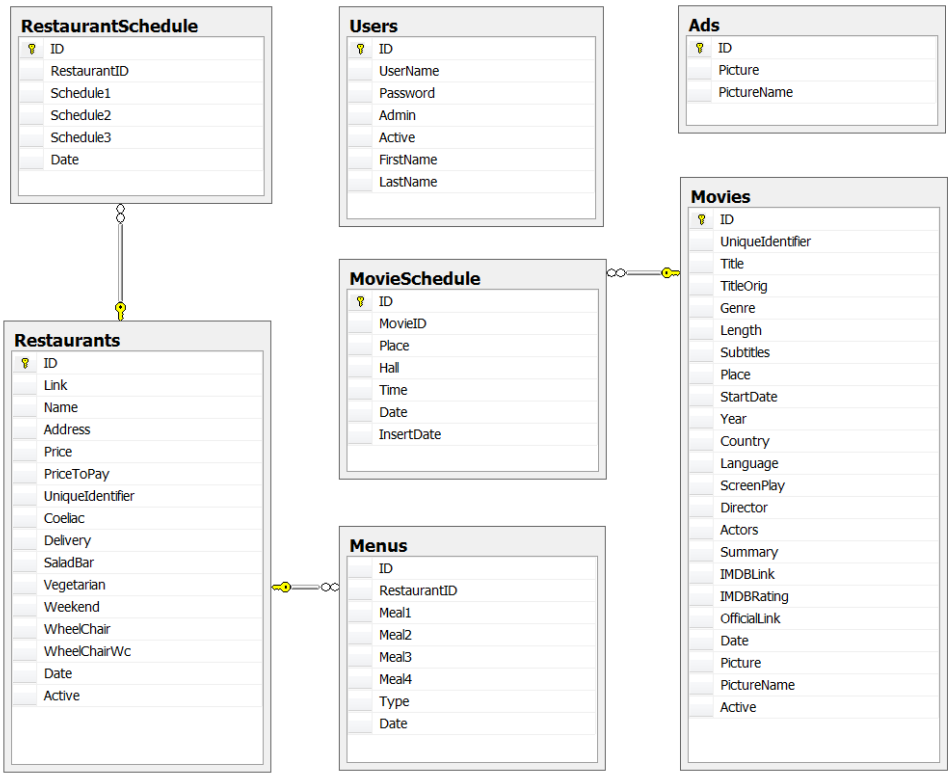
Podatkovna baza

Podatkovna baza, ki smo jo zgradili (slika 4.1), je sestavljena iz sedmih tabel. Tabela *Users* hrani podatke o uporabnikih spletne aplikacije za dodajanje reklam, in sicer enolični identifikator, ime, priimek, uporabniško ime, geslo in polji ali je aktiven ter ali je administrator. Tabela, ki vsebuje reklame, se imenuje *Ads* in vsebuje identifikator, ime reklame in sliko reklame.

V tabeli *Restaurants* hranimo podatke o restavracijah, in sicer enolični identifikator, ime restavracije, povezavo do restavracije na spletni strani Študentske prehrane, naslov, cena obroka, subvencionirana cena obroka, datum vnosa v podatkovno bazo, polja, ki označujejo ali restavracija ponuja hrano za vegetarijance in ljudi s celiakijo, dostava hrane na dom, solatni bar, odprtost ob vikendih, dostop za invalide in ali je restavracija aktivna. Urnik delovanja za posamezno restavracijo smo shranili v tabelo, imenovano *RestaurantSchedule*. V njej hranimo enolični identifikator, identifikator restavracije, datum vnosa, urnik ob delavnikih, urnik ob sobotah ter urnik ob nedeljah. Za vsako restavracijo hranimo podatke o dnevnem in stalnem meniju v tabeli *Menus*. Vsak vnos v tabeli predstavlja celoten obrok, ki je sestavljen iz različnih hodov. Podatki, ki jih tabela vsebuje, so enolični identifikator, identifikator restavracije, datum vnosa in štiri polja za vnos hodov obroka.

Podatke o filmih, ki se trenutno predvajajo v kinu, hranimo v tabeli *Movies*, in sicer enolični identifikator, identifikator filma, slovensko in originalno ime filma, žanr, dolžina, podatke o podnapisih, čas in kraj začetka sporeda, leto in kraj izida, jezik, scenarist, režiser, igralci, opis, povezava na film na spletni strani Imdb, Imdb ocena filma, povezava na stran filma, naslovnica filma, ime naslovnice filma, datum vnosa in ali je film aktiven. Za vsak

film hranimo urnik predvajanja filma, kjer vsaka predstava predstavlja en zapis v podatkovni bazi. Podatki, ki jih hranimo v tabeli *MovieSchedule*, so enolični identifikator, identifikator posameznega filma, kino v katerem se film predvaja, čas in datum predvajanja in datum vnosa.



Slika 4.1: Struktura podatkovne baze

Objekti

Za dostop do podatkovne baze smo napisali razred *Database*. Vsebuje metode za povezavo do podatkovne baze in manipulacijo podatkov. Dostop do podatkovne baze potrebujemo v strežniku, servisu za zajemanje podatkov ter v spletni aplikaciji, zato je pomembno, da imamo enake funkcionalnosti na enem mestu brez podvajanja programske kode. Ko ustvarimo novo instanco razreda *Database*, avtomatsko ustvarimo povezavo na izbrano podatkovno

bazo. Za dostopanje do podatkov iz podatkovne baze, imamo dve možnosti. Lahko uporabimo metodo, ki ji podamo SQL poizvedbo ali pa metodo, ki ji podamo ime shranjene procedure in parametre shranjene procedure. Podatke pridobimo v objektu *DataTable*.

Pri načrtovanju smo se odločili, da bomo ločili med razredi, ki dostopajo in manipulirajo s podatkovno bazo, in razredi, ki opisujejo objekte v naši aplikaciji. Razrede, ki dostopajo do podatkovne baze, smo poimenovali s končnico dao (angl. data access object), medtem ko smo razrede, ki opisujejo objekte, poimenovali s končnico dto (angl. data transfer object). Dao razrede uporabljamo kot vmesno točko med podatkovno bazo in dto razredi. V njih smo spisali metode, ki vračajo in manipulirajo s podatki iz podatkovne baze. Povezovanje na podatkovno bazo v metodah se vrši preko razreda za dostop do podatkovne baze *Database*. Razredi dto predstavljajo vrstico v podatkovni bazi. Sestavljeni so iz lastnosti, ki posamezen objekt opisujejo. Ko želimo na primer iz podatkovne baze zajeti podatke izbrane restavracije, metoda iz razreda dto pokliče metodo za pridobitev podatkov o restavraciji v razredu dao. Ta se poveže na podatkovno bazo, pridobi podatke in jih nazaj posreduje razredu dto, ki s podatki iz podatkovne baze napolni lastnosti restavracije v objekt dto.

Strežnik

Strežnik smo zgradili po pravilih gradnje aplikacije ogrodja WCF. Storitvena pogodba (angl. service contract) je implementirana z vmesnikom *IService*, v kateri imamo navedene operacije, ki jih storitev ponuja. Vmesnik *IService* implementiramo z razredom *Service*, ki deduje iz storitvene pogodbe. Razred *Service* predstavlja implementacijo naše storitve in funkcionalnosti vseh operacij navedenih v storitveni pogodbi.

Nastavitve našega strežnika imamo v datoteki *Web.config*. V njej smo nastavili podatke o dostopu do podatkovne baze na podatkovnem strežniku ter nastavitve naše storitve. V nastavitvah storitve nastavimo ime storitve, naslov, na kateri se nahaja naša storitev, preko katerega protokola bo potekala

komunikacija in drugo.

Urnik in obveznosti se na zahtevo odjemalca v realnem času pridobijo iz spletne učilnice. Pomembna razreda, ki smo ju napisali, sta *RetrieveData* in *Parser*. *RetrieveData* vsebuje metode, ki na vmesnik spletne učilnice pošljejo zahtevo po podatkih in jih v obliki objekta JSON posredujejo razredu *Parser*. V razredu *Parser* JSON podatke, ki jih dobimo, preberemo in z njimi napolnimo objekte tipa *CourseItemDto*, ki opisuje predmet na našem urniku. Da vemo, ali imamo na urniku predavanje ali vaje, imamo v objektu *CourseItemDto* lastnost, ki nam omogoča, da na odjemalcu ločimo med njimi. Podobno kot za urnik pridobimo podatke o študentovih obveznostih in s podatki napolnimo objekte tipa *EventItemDto*. Za ločevanje med končanimi in nedokončanimi obveznostmi imamo lastnost, s katero na odjemalcu povemo, katera obveznost je končana oziroma nedokončana. Za pridobitev urnika imamo na strežniku operacijo `List<CourseItemDto> GetUrnik(int day, string username, string password, string locale)`, za študentove obveznosti pa operacijo `List<EventItemDto> GetEvents(string username, string password, string locale)`.

Podatke o prihodih avtobusov pridobimo iz spletne strani http://wbus.telargo.com. Na spletni strani se nahaja obrazec, v katerega vnesemo ime ali številko postaje. V razredu *BusDto* imamo metodo `string GetStationRequest(string station)`, ki izpolni obrazec, ga pošlje in nazaj prejme HTML dokument, v katerem se nahaja lista prihodov. Vrnjena stran je zelo enostavne oblike, zato smo uporabili t.i. razčlenjevanje spletne strani s pomočjo podnizov. Urejene podatke o prihodih smo shranili v listo objektov *BusDto*. Na strežniku imamo operacijo `ObservableCollection <BusDto> GetStationData(string station)`, ki odjemalcu vrne listo prihodov za izbrano postajo. Odjemalcu lahko vrnemo tudi seznam vseh postaj v Ljubljanskem potniškem prometu. V datoteki na strežniku imamo shranjene vse postaje s podatki o imenu, številki in lokaciji, ki jih nato preberemo in jih zapišemo v listo objektov *StationDto*. Operacija, ki vrne seznam vseh postaj, je `ObservableCollection <StationDto> GetStations()`.

Podatke o restavracijah, kino predstavah in reklamah imamo shranjene v podatkovni bazi. Na strežniku podatke na zahtevo odjemalca pridobimo iz podatkovne baze. Glavne operacije za pridobitev podatkov na strežniku so `IEnumerable<RestaurantDto> GetRestaurants()`, `IEnumerable<MovieDto> GetAllMovies()` in `ObservableCollection<AdDto> GetAllAds()`.

Servis

Servis deluje tako, da se požene na izbran časovni interval, razčleni podatke s spletnih strani in podatkovno bazo posodobi z ažurnimi podatki. Naredili smo ga po predlogi, ki jo Microsoft ponuja v Visual Studio za gradnjo servisov v Windows okolju.

Razčlenjevanje ponavadi poteka tako, da iz programske kode pošljemo HTTP zahtevek na želeno spletno stran in ta nam vrne tekst v obliki HTML, ki ga nato razčlenimo in podatke poljubno obdelamo. V .NET ni vgrajene nobene knjižnice, ki bi omogočala lažje in učinkovitejše delo z razčlenjevanjem spletnih strani. Programerji zato pogosto uporabljajo manipulacije nad tekstom in regularne izraze. Odločili smo se za uporabo zastonjske .NET knjižnice Html Agility Pack, ki nam je omogočila lažje razčlenjevanje strani. *HtmlDocument* je objekt, v katerega se naloži celoten DOM (angl. document object model) vanj naložene spletne strani. *HtmlDocument* je nadalje sestavljen iz t.i html vozlišč (angl. html node), s katerimi si poenostavimo preiskovanje html dokumenta.

Spletni strani Študentske prehrane (www.studentska-prehrana.si) in Koloseja (www.kolosej.si) sta bili za razčlenjevanje precej zahtevni. Na spletni strani Študentske prehrane imamo imenik lokalov, v katerem so v seznamu prikazane vse restavracije v Sloveniji, ki ponujajo študentsko prehrano. Iz seznama lokalov smo za posamezno restavracijo pridobili povezavo na podstran restavracije. Na podstraneh restavracije smo pridobili osnovne podatke o restavraciji, delovni čas in stalni ter dnevni meni, ki ju restavracija ponuja. Na spletni strani Koloseja se nahaja seznam filmov, ki se predvajajo v kinodvoranah v Ljubljani. Iz seznama smo pridobili povezavo na naslovnico

filma in podstran posamezenga filma. Na podstrani posameznega filma smo pridobili vse podatke o filmu ter urniku predvajanja. Podatke o oceni filma smo pridobili iz spletne baze filmov Imdb (www.imdb.com).

V razredu *ParseData* imamo metodo `Run()`, ki se zažene ob zagonu servisa. V njej kličemo metode za katere želimo, da se izvedejo na izbran časovni interval: `GetRestaurantsNet()`, `GetMenusSchedules()`, `GetMoviesNet()`.

Metoda `GetResturantsNet()` nam omogoča, da pridobimo podake o restavracijah, njihovih urnikih ter menijih. V njej smo ustvarili objekt *HtmlDocument*, v katerega smo naložili spletno stran Študentske prehrane. Iz objekta smo s pomočjo jezika Xpath najprej izluščili vozlišča, ki predstavljajo posamezno restavracijo. Za vsako restavracijo smo zaradi lažjega razčlenjevanja izluščili še vozlišča, ki vsebujejo osnovne podatke o restavraciji, ceni študentskega kosila ter grafično prikazanih lastnostih restavracije. Metodo `void WriteRestaurant (HtmlNode name, HtmlNode price, HtmlNode features, int ID)` smo klicali za vsako vozlišče restavracije. V njej smo ustvarili objekt *RestaurantDto*, ki predstavlja posamezno restavracijo. Podatke iz vozlišč smo nato z manipulacijo nad tekstom in funkcijami knjižnjice HtmlAgilityPack obdelali v obliko, primerno za zapis v objekt. S podatki o restavraciji, ki smo jih pridobili z razčlenjevanjem posameznih vozlišč, smo nato napolnili objekt RestaurantDto in ga shranili v podatkovno bazo. Vsaka restavracija ima enolični identifikator, ki nam omogoča, da restavracije, ki jo že imamo v podatkovni bazi, ne dodamo na novo, temveč samo posodobimo njene podatke. Podatke o urniku za posamezno restavracijo smo pridobili z metodo `WriteSchedule(int restaurantId, string link)`. Urnik restavracije se ne nahaja na isti podstrani kot ostali podatki o restavraciji, zato smo morali internetni naslov prilagoditi. Spletno stran z urnikom smo nato naložili v objekt *HtmlDocument* in iz njega nato z razčlenjevanjem pridobili podatke o urniku restavracije. Objekt *ScheduleDto*, ki predstavlja urnik restavracije, smo napolnili s podatki in ga shranili v podatkovno bazo. Menije posamezne restavracije smo pridobili z metodo `WriteMenu(int restaurantId, sting link)`. Pri vsaki restavraciji imamo

na voljo dnevni in stalni menu. Vsak od njiju se nahaja na svoji podstrani, zato smo vsakega posebej naložili v svoj *HtmlDocument* objekt, iz katerih smo izluščili podatke o posameznem meniju. Za vsak meni smo ustvarili objekt *MenuDto*, ga napolnili s podatki in shranili v podatkovno bazo. Da menija ločimo med seboj, imamo na objektu lastnost *Type*, v katerega smo zapisali, katerega tipa je meni.

Metoda `GetMoviesNet()` služi za pridobivanje podatkov o kino predstavah. Spletno stran Koloseja smo naložili v *HtmlDocument* objekt in iz njega izluščili vozlišča, ki predstavljajo povezave na filme, ter vozlišča, ki vsebujejo povezave na slike posameznega filma. Za vsako vozlišče smo klicali metodo `void WriteMovie(HtmlDocument htmldocObject, string movieLink, string moviePicLink)`, ki smo ji kot parameter podali *HtmlDocument* objekt, ki vsebuje DOM strukturo podstrani posameznega filma. Iz podanega *HtmlDocument* objekta smo pridobili vozlišča, v katerih se nahajajo lastnosti filma in jih predelali v primerno obliko. Za vsak film smo iz spletne strani v objekt shranili tudi poster filma. Oceno filma smo pridobili iz spletne strani Imdb, ki vsebuje bazo podatkov o filmih. Ustvarili smo objekt *MovieDto*, v katerega smo zapisali vse razčlenjene podatke o posameznem filmu ter ga shranili v podatkovno bazo, če film še ni v podatkovni bazi. Urnik predvajanja smo klicali z metodo `void GetMovieSchedule(HtmlDocument htmldocObject, int movieID)`. Iz parametra funkcije *HtmlDocument* objekta smo pridobili vozlišča, v katerih se nahajajo podatki o urniku predvajanja za šest naslednjih dni. Podatke, ki smo jih razčlenili, smo vnesli v objekt *MovieScheduleDto* in podatke shranili v podatkovno bazo.

4.3 Odjemalec

V našem primeru imamo dva odjemalca. To sta mobilna aplikacija za mobilni operacijski sistem Windows phone 7 in spletna aplikacija za vnašanje oglasnih sporočil.

4.3.1 Spletna aplikacija

Spletna aplikacija je aplikacija, ki omogoča vnašanje oglasnih sporočil v podatkovno bazo. Oglasna sporočila so nato prikazana v mobilni aplikaciji. Za uporabo spletne aplikacije potrebujemo spletni brskalnik in povezavo v internet.
Napisana je v ASP.NET Microsoftovem ogrodju za pisanje spletnih aplikacij. Implementirana je s trinivojsko arhitekturo, ki sestoji iz:

1. Predstavitvenega nivoja: na predstavitvenem nivoju se izvaja uporabniški vmesnik, ki sprejema in oddaja zahteve aplikacijskemu nivoju.

2. Aplikacijskega nivoja: tu se nahaja poslovna logika, ki procesira zahteve in podatke, ki jih posredujeta ostala dva nivoja.

3. Podatkovnega nivoja: tu se nahaja podatkovni strežnik, na katerem so shranjeni podatki.

Prednosti trinivojske arhitekture so večja možnost za ponovno uporabo istih programskih komponent, boljša integriteta in varnost podatkov ter boljša razpoložljivost, ker lahko imamo podvojene aplikacijske in podatkovne strežnike.

Uporaba

Ob prihodu uporabnika na spletno stran se najprej odpre prijavna stran, na kateri se uporabnik prijavi z uporabniškim imenom in geslom. V primeru uspešne prijave je preusmerjen na glavno stran, kjer se nahaja pregled vnešenih oglasov. Podatki o oglasih so prikazani v tabeli (slika xx). Za vsak oglas je prikazano ime oglasa, oglasna slika (pasica) ter gumba za brisanje oziroma urejanje. Pod tabelo se nahaja gumb za dodajanje oglasa. Ko uporabnik klikne nanj, se pod tabelo odpre obrazec za vnos novega oglasa. Uporabnik v vnosno polje vpiše ime oglasa, izbere oglasno sliko, ki je predpisane velikosti ter nato z gumbom potrdi vnos.

Aplikacija ima dve vrsti uporabnikov, navadne uporabnike ter administratorja. Navadni uporabniki lahko samo pregledujejo in urejajo oglase,

medtem ko lahko administrator še dodaja in ureja uporabnike.

Uporabniki imajo možnost spremembe svojega gesla. S klikom na svoje uporabniško ime, ki se nahaja v zgornjem desnem kotu aplikacije, se odpre obrazec za spremembo gesla. Vpisati je potrebno staro geslo in dvakrat novo geslo.

Administrator lahko poleg pregleda oglasov v meniju izbere tudi administracijo uporabnikov. Podatki o uporabnikih so prikazani v tabeli. Vsakega uporabnika lahko administrator ureja, deaktivira ali pa mu ponastavi geslo na uporabniško ime. Pod tabelo se nahaja gumb za dodajanje novega uporabnika. S klikom na gumb se odpre obrazec za dodajanje novega uporabnika. V vnosna polja uporabnik vnese uporabniško ime, ime, priimek, ali je uporabnik administrator in ali je uporabnik aktiven. S potrditvijo se ustvari nov uporabnik.

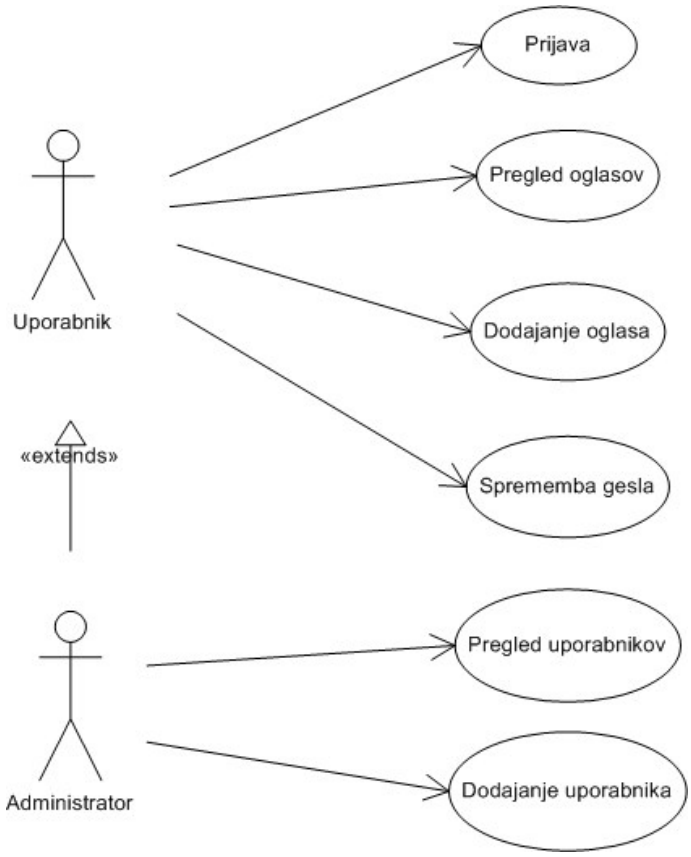
Na sliki 4.2 vidimo diagram primera uporabe (angl. Use Case Diagram), ki grafično prikazuje komunikacijo med uporabniki in sistemom.

Arhitektura

Spletna aplikacija je implementirana s tronivojsko arhitekturo. Za dostopanje do podatkovne baze smo napisali razred *Database*, ki je podrobneje opisan v poglavju 4.2.2. Predstavitveni in aplikacijski nivo sta v ASP.NET ločena tako, da sta vsak v svoji datoteki. Logika, ki predstavlja aplikacijski nivo, je v datotekah s končnico *.aspx.cs*, medtem ko je uporabniški vmesnik v datotekah s končnico *.aspx*.

Za manipulacijo z uporabniki smo napisali razreda *UserDao.cs* in *UserDto.cs*. Metode prvega omogočajo manipulacijo tabele *User* v podatkovni bazi. V drugem razredu pa uporabljamo metode razreda *UserDao*, da napolnimo objekte tipa *UserDto*. Podobno kot za uporabnike smo za oglase napisali razreda *AdDao.cs* in *AdDto.cs*.

Prijava v aplikacijo se nahaja v datotekah *Login.aspx* in *Login.aspx.cs*. Ko uporabnik klikne gumb za prijavo, validator, vezan na gumb, najprej preveri, če je v poljih za uporabniško ime in geslo sploh kaj vpisal. Nato se proži



Slika 4.2: UML diagram uporabe spletne aplikacije

metoda `LoginButton_Click(object sender, EventArgs e)`, ki preveri, če uporabnik z vpisanim uporabniškim imenom obstaja in nato še, če je vpisano geslo pravilno. Ob uspešni prijavi je uporabnik preusmerjen na stran z oglasi.

Stran z oglasi se nahaja v datotekah *Default.aspx* in *Default.aspx.cs*. Ko uporabnik pride na stran, se oglasi iz baze preko objekta *AdDto* napolnijo v ASP.NET kontrolo GridView, ki tabelarično prikaže oglase, kot vidimo na sliki 4.3. Pri kliku na gumba za brisanje ali pa urejanje oglasa se proži metoda `gridBanners_RowCommand (object sender, GridViewCommandEventArgs e)`, ki glede na pritisnjen gumb izvede primerno akcijo. V primeru brisanja pridobimo unikatno številko oglasa (id) in jo posredujemo metodi za brisanje

Slika 4.3: Spletna aplikacija

`Delete(int Id)` v razredu *AdDto*, ki oglas nato izbriše iz baze. Če pa gre za urejanje oglasa, si id oglasa zapomnimo v sejo in odpre se obrazec za urejanje oglasa. Obrazec za urejanje je enak kot obrazec za dodajanje oglasa, ki ga odpremo s klikom na gumb za dodajanje oglasa. Ko potrdimo urejanje, oziroma dodajanje se proži metoda `btSave_Click(object sender, EventArgs e)`. Metoda najprej preveri, ali gre za urejanje ali dodajanje ter ali smo sploh dodali oglasno sliko. V primeru, da so pogoji izpolnjeni, se oglas z metodo `Write()` iz razreda *AdDto* posodobi, oziroma vstavi v podatkovno bazo.

Obrazec za spreminjanje gesla se nahaja v datotekah *UserSettings.aspx* in *UserSettings.aspx.cs*. Ko uporabnik vpiše staro in novo geslo in potrdi spremembo gesla, se kliče metoda `btSave_Click(object sender, EventArgs e)`, ki najprej preveri, če je staro geslo pravilno, novega spremeni v kodirano obliko in ga z metodo `Write()` iz razreda *UserDto* shrani.

Administracija uporabnikov se nahaja v datotekah *Admin.aspx* in *Admin.aspx.cs*. V GridView se napolnijo uporabniki iz baze. S klikom na gumb za urejanje ali pa ponastavitev gesla se proži metoda `gridUsers_RowCommand (object sender, GridViewCommandEventArgs e)` in izvede izbrano akcijo.

Pri ponastavitvi gesla se iz baze na podlagi id uporabnika pridobi objekt *UserDto*, ki se mu geslo ponastavi na uporabniško ime in se ga spremenjenega shrani v bazo. V primeru urejanja se id uporabnika shrani v sejo in odpre se obrazec za urejanje uporabnika, ki je enak kot obrazec za dodajanje uporabnika. Pri potrditvi urejanja ali dodajanja validatorji, ki so vezani na spletne kontrole, najprej preverijo, če so prisotni vsi obvezni podatki, nato pa se proži metoda `btSave_Click(object sender, EventArgs e)`. V metodi se preveri, ali gre za urejanje ali dodajanje uporabnika, nato pa se objekt *UserDto* uporabnik z metodo `Write()` posodobi, oziroma vstavi v podatkovno bazo.

4.3.2 Mobilna aplikacija

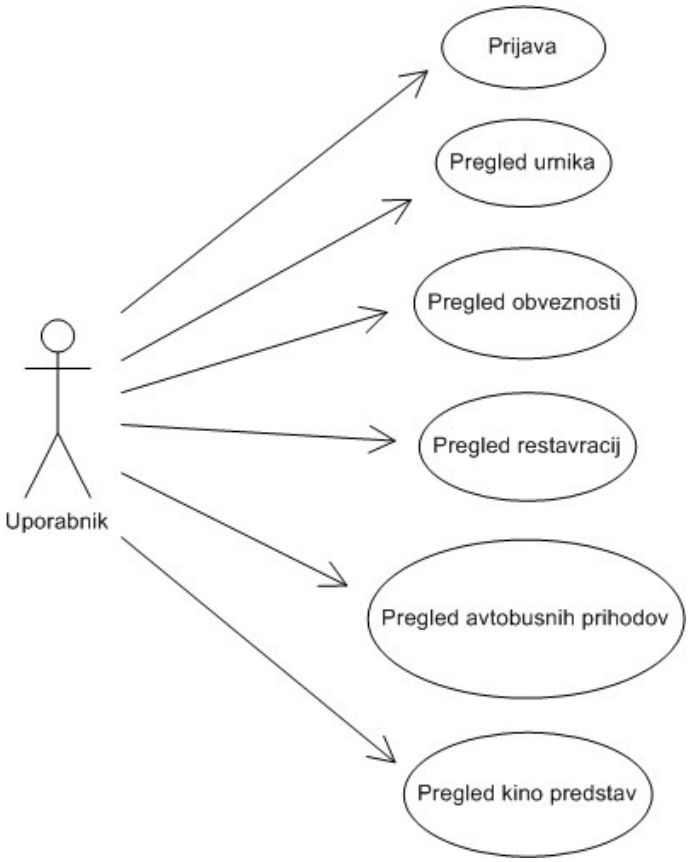
MFri je mobilna aplikacija namenjena študentom Fakultete za računalništvo in informatiko v Ljubljani. Omogoča jim, da preko mobilnega telefona pogledajo svoj urnik in šolske obveznosti, kdaj izpred fakultete odpelje naslednji avtobus, kaj bodo pojedli za kosilo ter kateri film si bodo ogledali v prostem času. Uporabljajo jo lahko študenti, ki imajo na mobilnem telefonu nameščen operacijski sistem Windows phone 7. Aplikacija za delovanje potrebuje internetno povezavo, saj podatke pridobiva iz strežnika.

Uporaba

UML diagram uporabe moiblne aplikacije lahko vidimo na sliki 4.4. Uporabnik se v aplikacijo prijavi z uporabniškim imenom in geslom, ki ju uporablja za vstop v Spletno učilnico Fakultete za računalništvo in informatiko. Ob uspešni prijavi je preusmerjen v nastavitve, kjer lahko izbere, ali bo aplikacija v slovenskem ali angleškem jeziku ter ali se telefon ob zagonu aplikacije zatrese. Ko so nastavitve shranjene, je uporabnik preusmerjen na začetno stran, na kateri se nahajata gumba za dostop do zaslonov urnika in obveznosti.

Oglasi so v aplikaciji prikazani kot manjše slike na dnu zaslonov, če je na zaslonu dovolj prostora za njihov prikaz. Na vsakih nekaj sekund se

zamenjajo.



Slika 4.4: UML diagram uporabe mobilne aplikacije

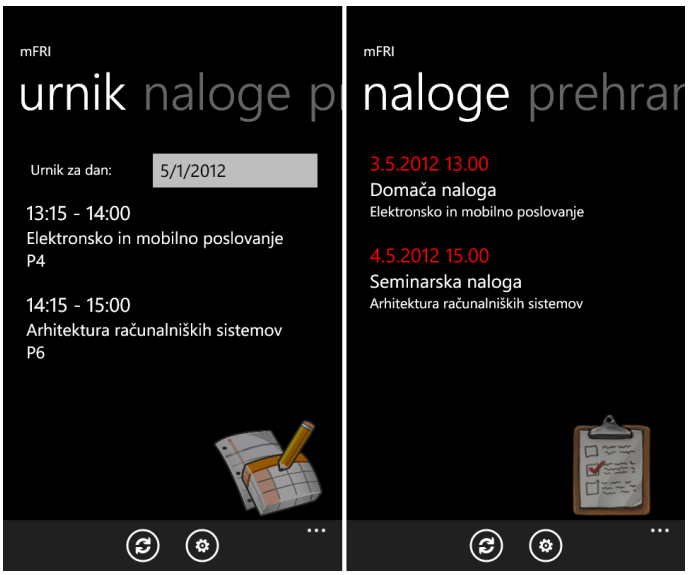
Urnik

V primeru, da si uporabnik želi pogledati svoj urnik (slika 4.5), se premakne do zaslona, na katerem se nahaja prikaz urnika. Aplikacija je prednastavljena, da prikazuje urnik za trenutni dan. Uporabnik ima možnost izbire urnika za poljuben datum. Urnik je prikazan v seznamu, ki za vsak vnos prikazuje ime predmeta, čas izvajanja in učilnico, kjer se predmet izvaja.

Obveznosti

Obveznosti, ki jih mora študent opraviti, so prikazane na naslednjem zaslonu. Prikazane so v seznamu, v katerem se za vsako obveznost prikazuje datum,

do katerega mora biti obveznost dokončana, ime naloge ter ime predmeta. Obveznosti so urejene kronološko, tako da je obveznost, ki mora biti prva dokončana, na vrhu seznama.



Slika 4.5: Zaslona z urnikom in nalogami

Prehrana

Zaslon prehrana je namenjen pregledu restavracij s študentsko prehrano (slika 4.6). Na zaslonu je šest gumbov, ki uporabnika preusmerijo na štiri izbrane restavracije, ki se nahajajo v bližini Fakultete za računalništvo in informatiko ali pa na listo vseh restavracij oziroma poljubno iskanje restavracij s študentsko prehrano v Ljubljani in Sežani.

Ko uporabnik odpre izbrano restavracijo, ga aplikacija preusmeri na nov zaslon, na katerem se nahajajo podatki o restavraciji. Za vsako restavracijo so prikazani osnovni podatki, kot so ime, naslov in urnik restavracije ter podatki o subvencionirani ceni obroka. Z ikonami je grafično prikazana dodatna ponudba, ki jo ima posamezna restavracija, na primer vegetarijanska prehrana, solatni bar, dostop za invalide ter drugo.

Za vsako restavracijo lahko uporabnik pogleda stalni in dnevni meni, ki

ga ponuja za trenutni dan. Posamezen meni je prikazan v seznamu, ki za vsak vnos prikazuje, iz katerih hodov je sestavljen obrok.

Če želi uporabnik pregledati vse restavracije s subvencionirano prehrano, je s klikom na gumb za prikaz vseh restavracij na začetnem zaslonu prehrane preusmerjen na nov zaslon, na katerem so v seznamu prikazane vse restavracije s subvencionirano prehrano. Za vsako restavracijo v seznamu je izpisano ime restavracije ter subvencionirana cena obroka. S klikom na posamezno restavracijo v seznamu je uporabnik preusmerjen na zaslon, na katerem se nahajajo podatki o restavraciji.

S klikom na gumb za iskanje restavracij na začetnem zaslonu prehrane je uporabnik preusmerjen na nov zaslon. Na njem se nahajata vnosno polje, kjer uporabnik vnese ime restavracije, in gumb za potrditev iskanja. Rezultati iskanja so prikazani v seznamu, kjer vsak vnos predstavlja podatke o restavraciji. Če uporabnik klikne na posamezno restavracijo v seznamu, je preusmerjen na zaslon s podatki o restavraciji.

Slika 4.6: Zasloni s prehrano

Avtobus

Zaslon vozni redi, na sliki 4.7 omogoča pregled odhodov avtobusov za iz-

brana postajališča Ljubljanskega potniškega prometa (LPP). Na začetnem zaslonu imamo tri gumbe. Prva dva uporabnika preusmerita na dve izbrani postajališči, ki se nahajata v bližini Fakultete za računalništvo in informatiko, tretji pa omogoča iskanje poljubnih postajališč LPP.

Ko uporabnik odpre izbrano postajališče, ga aplikacija preusmeri na nov zaslon, na katerem se nahaja seznam odhodov za izbrano postajališče. Za vsak odhod v seznamu so prikazani podatki o avtobusni linji, relacija ter času prvih treh prihodov na izbrano postajališče.

Če uporabnik želi poiskati odhode za poljubno postajališče, na začetnem enkranu avtobusa klikne na gumb za iskanje. Preusmerjen je na zaslon za iskanje, kjer se nahaja vnosno polje za vnos imena postajališča oziroma številke postajališča in gumb za potrditev iskanja. Rezultati iskanja so prikazani v seznamu, kjer vsak vnos predstavlja podatke o postajališču in ime ter številko postajališča. Ob kliku na izbrano postajališče iz seznama je uporabnik preusmerjen na zaslon s seznami odhodov z izbranega postajališča.

Slika 4.7: Zasloni z voznimi redi

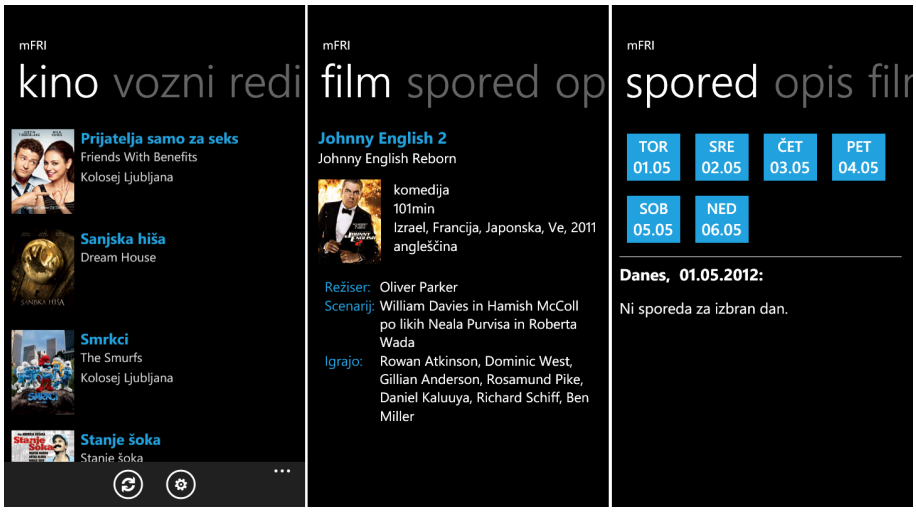
Kino

Zaslon kino omogoča pregled filmov, ki se trenutno predvajajo v Koloseju,

XpanDu ter Komuni v Ljubljani (slika 4.8). Na začetnem zaslonu se nahaja seznam filmov. Za vsak film v seznamu so prikazani podatki o imenu in kinodvorani, v kateri se film predvaja. Zraven vsakega filma je tudi slika filmske naslovnice.

Ob kliku na izbran film v seznamu filmov je uporabnik preusmerjen na zaslon s podatki o filmu. Prikazani so osnovni podatki, kot so ime filma v prevodu in originalu, žanr, dolžina, leto izida, jezik in slika naslovnice filma. Ocena filma je pridobljena iz arhiva filmov Imdb in je prikazana grafično z zvezdicami. Nadalje so prikazani še ostali podatki: režiser, scenarij in igralci. Opis filma je zaradi lažjega branja teksta večjega obsega prikazan v svojem zaslonu.

Urnik predvajanja za izbrani film si lahko uporabnik ogleda v zaslonu poleg podatkov o filmu. Na zaslonu se nahaja 6 gumbov, na katerih so datumi za šest dni vnaprej. Ko uporabnik klikne na izbran datum, se mu v seznamu prikažejo urniki predvajanja, ki prikazujejo podatke o kinu, dvorani in času predvajanja filma.



Slika 4.8: Zasloni s kino predstavami

Arhitektura

Komunikacija s strežnikom pri mobilni aplikaciji poteka preko storitveno usmerjene arthitekture. V aplikaciji imamo referenco na naš strežnik, ki nam ponuja storitve, ki jih potrebujemo za delovanje aplikacije. Predstavitveni in aplikacijski nivo sta pri razvoju za Windows phone 7 podobno kot pri razvoju za ASP.NET ločena po datotekah. V datotekah s končnico *.xaml* je predstavitveni nivo, ki je izveden z mobilno različico tehnologije Silverlight. Aplikacijski nivo se nahaja v datotekah s končnico *.xaml.cs*, kjer je uporabljen programski jezik C#. Med zasloni, ki predstavljajo funkcionalnosti aplikacije, se uporabnik premika z drsanjem vsebine po zaslonu, podobno kot da bi obračal platnice knjige. To funkcionalnost nam je omogočila v WP7 vgrajena kontrola imenovana Pivot.

Razredi, ki smo jih potrebovali za opis objektov v mobilni aplikaciji, se nahajajo na strežniku. V mobilni aplikaciji imamo samo referenco do razredov na strežniku. Čeprav nimamo fizične prisotnosti razreda, še vedno lahko delamo vse manipulacije z objekti.

V aplikaciji imamo nekaj funkcionalnosti, ki se ponavljajo na več zaslonih. Metoda `void BuildApplicationBar()` zgradi meni, do katerega lahko dostopamo iz vseh delov aplikacije in ima gumbe za posodabljanje podatkov, dostop do nastavitev ter podatkov o aplikaciji. Pri navigaciji med zasloni moramo vedeti, iz katerega zaslona smo prišli. Metoda `void OnNavigatedTo (System.Windows.Navigation.NavigationEventArgs e)` se proži, ko uporabnik pride pride z drugega zaslona in sprejme podatke, ki jih je aplikacija posredovala.

Nastavitve

Za manipulacijo globalnih nastavitev, kot so uporabniško ime, geslo in jezik, smo napisali razred *Setting*, ki implementira vzorec samskosti (angl. singleton) in drži vrednosti spremenljivk skozi čas izvajanja programa. Nastavitve smo preko razreda *Setting* hranili v Isolated Storage, prostor na mobilnem telefonu, ki je namenjen hranjenju manjših količin podatov za potrebe apli-

kacije.

Prijava v aplikacijo in urejanje nastavitev se nahajata v datotekah *Settings.xaml* in *Settings.xaml.cs*. Ko se aplikacija zažene, najprej preveri ali sta uporabniško ime in geslo že shranjena v telefonu in v nasprotnem primeru prikaže zaslon z obrazcem za prijavo. Ob kliku na gumb za prijavo se proži metoda `Login_Click(object sender, RoutedEventArgs e)`, ki s klicem storitve na strežnik preveri, ali sta uporabniško ime in geslo pravilna.

Urnik in obveznosti

Velik del programske kode aplikacije se nahaja v datotekah *MainPage.xaml* in *MainPage.xaml.cs*. Tu se nahaja grafični vmesnik, iz katerega lahko dostopamo do vseh funkcionalnosti aplikacije. Ob prijavi v aplikacijo s strežnika s klicema operacij na serverju `List<CourseItem> getUrnik(int day, String username, String password, String locale)` in `List <EventItem> getEvents (string username, string password, string locale)` pridobimo podatke o urniku in nedokončanih obveznostih. V času, dokler pridobivamo podatke, s klicem metode `void ShowPopup()` prikažemo pojavno okno (angl. popup), ki uporabnika opominja na čakanje na podatke.

Prehrana

Programska koda prehrane se nahaja v datotekah *Restaurants.xaml*, *Restaurants.xaml.cs*, *RestaurantsList.xaml* in *RestaurantsList.xaml.cs*. Za opis lastnosti restavracije, urnika obratovanja in menija smo uporabili naslednje tri razrede: *RestaurantDto*, *ScheduleDto* in *MenuDto*. Ob prihodu uporabnika na zaslon prehrane se na strežniku kličejo tri operacije. Operacija `IEnumerable<RestaurantDto> GetRestaurants()`, aplikaciji vrne listo objektov restavracij, operacija `ObservableCollection<ScheduleDto> GetAllRestaurantSchedules()` listo objektov urnikov obratovanja restavracij ter operacija `ObservableCollection<MenuDto> GetAllMenus()`, ki vrne listo objektov vseh menijev restavracij. Liste globalno shranimo, da lahko do

podatkov dostopamo tudi kasneje, brez ponovnega klica na strežnik. Če želi uporabnik podatke posodobiti, iz menija pokliče metodo `void Refresh()`, ki posodobi liste podatkov.

Podatke posamezne restavracije prikažemo tako, da najprej pridobimo objekt restavracije *RestaurantDto* iz shranjenih list ter nato podatke iz objekta prikažemo na ločenem zaslonu. Za prikaz dnevnega ali stalnega menija imamo na objektu *MenuDto* lastnost *Type*, da razlikujemo med njima:

- 0 – objekt *MenuDto* vsebuje dnevni meni
- 1 – objekt *MenuDto* vsebuje stalni meni

Ob izbiri restavracije iz seznama restavracij se proži metoda `lbRestaurant_SelectionChanged(object sender, SelectionChanged EventArgs e)`, ki pošlje podatke o izbrani restavraciji na zaslon s podatki o restavraciji, kjer podatke prestreže že zgoraj opisana metoda `void OnNavigatedTo (System.Windows.Navigation.NavigationEventArgs e)`. Pri iskanju restavracije se izvede metoda `void btnSearch_Click(object sender, RoutedEventArgs e)`, ki iz liste shranjenih restavracij pridobi restavracije z iskanim imenom in jih prikaže v seznamu. Na zaslon s podatki o iskani restavraciji uporabnika preusmeri metoda `lbSearch_SelectionChanged (object sender, SelectionChangedEventArgs e)`.

Avtobus

Večji del programske kode za avtobus se nahaja v emph Stations.xaml, *Stations.xaml.cs*, *StationsList.xaml* in *Stations.xaml.cs*. Razred *StationDto* smo uporabili za opis lastnosti posameznega avtobusnega postajališča, razred *BusDto* pa za opis lastnosti prihoda posameznega avtobusa. Z izbiro postaje na zaslonu avtobus je uporabnik preusmerjen na zaslon za prikaz odhodov. Metoda `void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)`, ki sprejme ime postaje, na strežnik pokliče operacijo `ObservableCollection <BusDto> GetBusesForStation(string station)`. Metoda vrne listo objektov *BusDto*, ki jih nato prikažemo v seznamu.

Na zaslonu za iskanje poljubnega postajališča aplikacija na strežnik pokliče operacijo `ObservableCollection<StationDto> GetStations()`, ki vrne listo vseh postaj v Ljubljani. Listo vseh postaj globalno shranimo, da ni več potrebno klicati na strežnik. Metoda `void btnSearch_Click (object sender, RoutedEventArgs e)` se proži, ko uporabnik vpiše iskano postajališče in klikne gumb za iskanje. Iz liste vseh postaj poišče postaje, ki ustrezajo imenu iskanega postajališča in jih prikaže v seznamu. Metoda `void lbSearchStation_SelectionChanged(object sender, SelectionChangedEventArgs e)` uporabnika preusmeri na zaslon za prikaz odhodov iskanega postajališča.

Kino

V datotekah *Movies.xaml* in *Movies.xaml.cs* se nahaja programska koda za kino. Za filme smo napisali razred *MovieDto*, ki vsebuje lastnosti posameznega filma, za urnik predvajanja pa razred *MovieScheduleDto*. Ob prihodu na zaslon kino se na strežnik kliče operacija `IEnumerable<MovieDto> GetAllMovies()`, ki vrne listo objektov filmov. Za poznejšo uporabo smo listo globalno shranili. Na zaslonu kino nato filme prikažemo v seznamu. Da smo lahko izvedli prikaz slik v seznamu filmov, smo napisali razred *ByteArrayToImageConverter*. Slika je na objektu *MovieDto* shranjena v binarni obliki, ki je ne moremo direktno prikazati, zato smo jo s pomočjo razreda spremenili iz binarne oblike v objekt tipa *BitmapImage*.

Ko uporabnik iz seznama izbere film, se izvede metoda `void OnNavigatedTo (System.Windows.Navigation.NavigationEventArgs e)`, ki iz shranjene liste filmov najde izbrani film in njegove podatke prikaže na zaslonu. Za prikaz ocene filma smo uporabili posebno kontrolo, ki prikaže oceno v obliki zvezdic. Metoda `void pivotTitle_LoadingPivotItem(object sender, PivotItemEventArgs e)`, se kliče na zaslonu za pregled urnika izbranega filma. Ko se izvede, na strežnik pokliče operacijo `ObservableCollection <MovieScheduleDto> GetScheduleByMovieDate(int movieID, DateTime date)`, ki vrne listo urnikov za izbran film in datum predvajanja.

Oglasi

V aplikaciji oglase prikazujemo v kontroli za prikaz slik. Ob zagonu aplikacije se kliče na strežniku operacija `ObservableCollection<AdDto> GetAllAds()`, ki vrne listo objektov oglasov. Listo globalno shranimo, da nam ni treba ponovno poslati zahteve na strežnik. Med oglasi v listi na vsakih nekaj sekund naključno izberemo oglas in ga prikažemo na zaslonu.

Poglavje 5

Sklepne ugotovitve

V sklopu diplomske naloge smo razvili celovito rešitev mFri, ki vključuje mobilno aplikacijo za uporabo na napravah z operacijskim sistemom Windows Phone 7, strežnik za komunikacijo z mobilno aplikacijo, servis za pridobivanje podatkov iz spletnih strani ter podatkovno bazo.

Uspešno smo izpolnili vse zastavljene cilje (mobilna aplikacija, ki omogoča prikaz podatkov iz spletne učilnice, pregled ponudbe restavracij s študentsko prehrano, pregled prihodov avtobusov Ljubljanskega potniškega prometa ter pregled ažurnih kino predstav).

Razvoj spletnega strežnika in postavitev podatkovne baze ter kreiranje tabel nam zaradi predhodnega znanja ni predstavljalo večjih težav. Zahtevnejši del diplomske naloge je bil razvoj mobilne aplikacije in pridobivanje podatkov s spletnih strani. Največ časa nam je vzelo spoznavanje knjižnjic in samega načina razvoja za Windows Phone 7 ter uporaba knjižnjice Html Agility Pack za razčlenjevanje in pridobivanje podatkov s spletnih strani. Rešitev zaradi tehničnih težav ni bila testirana na spletnem strežniku in mobilnih napravah, temveč samo na lokalnem računalniku, na katerem je potekal razvoj aplikacije. Tudi povezava s spletno učilnico je delovala samo del časa razvoja rešitve.

Možnosti za nadaljni razvoj aplikacije je veliko. Povezava med mobilno aplikacijo in spletno učilnico poteka preko zahteve, v kateri pošljemo upo-

rabniško ime in geslo študenta v tekstovni obliki, kar ne ustreza varnostnim standardom. V nadaljnem razvoju aplikacije bi bilo potrebno izboljšati varnost prenosa podatkov med spletno učilnico ter mobilno aplikacijo. Veliko bi bilo tudi še možnosti optimizacije programske kode, saj smo se z razvojem na mobilnih aplikacijah v Windows phone 7 operacijskem sistemu srečali prvič.

Slike

Literatura

[1] (2011) Pametni telefoni vse bolj razširjeni tudi v Sloveniji; dostopno na: http://tehnik.mobitel.si/pametni-telefoni-vse-bolj-razsirjeni/

[2] (2011) Usage of telecommunication services and Internet in Central and Eastern Europe (CEE) ; dostopno na: http://www.gfk.rs/public_relations/press/articles/008936/index.en.html

[3] (2012) Smartphones Account for Half of all Mobile Phones, Dominate New Phone Purchases in the US ; dostopno na: http://blog.nielsen.com/nielsenwire/online_mobile/smartphones-account-for-half-of-all-mobile-phones-dominate-new-phone-purchases-in-the-us

[4] (2012) Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth ; dostopno na: http://www.gartner.com/it/page.jsp?id=1924314

[5] (2012) What is Moodle? ; dostopno na: http://moodle.org/about/

[6] (2011) Bo Microsoft kupil Nokio? ; dostopno na: http://www.racunalniske-novice.com/novice/dogodki-in-obvestila/bo-microsoft-kupil-nokio.html

[7] Henry Lee, Eugene Chuvyrov: Beginning Windows Phone 7 Development, Apress, 2010, poglavje 1

[8] (2012) Html Agility Pack ; dostopno na: http://htmlagilitypack.codeplex.com/

[9] Pablo Cibraro, Kurt Claeys, Fabio Cozzolino, Johann Grabner:Professional WCF 4, Wrox, 2010, poglavje 1