UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dejan Dakić

MOBILNA APLIKACIJA ZA SPREMLJANJE VOZNIH REDOV IN ZAMUD ZA MESTNI AVTOBUSNI PROMET IN ŽELEZNICE

DIPLOMSKO DELO
NA VISOKOŠOLSKEM ŠTUDIJU PRVE STOPNJE

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2011

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: 00081/2011
Datum: 01.04.2011

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

Kandidat: **DEJAN DAKIĆ**

Naslov: **MOBILNA APLIKACIJA ZA SPREMLJANJE VOZNIH REDOV IN ZAMUD ZA MESTNI AVTOBUSNI PROMET IN ŽELEZNICE**

MOBILE APPLICATION TO MONITOR TIMETABLES AND DELAYS FOR CITY BUS NETWORK AND RAILWAYS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Razvijte mobilno aplikacijo, ki bo popotnikom v Sloveniji omogočila pregledovanje voznih redov in zamud za Slovenske železnice in za ljubljanski mestni potniški promet. Aplikacija naj deluje na več-nivojski arhitekturi. Uporabite tudi tehnologijo GPS in vse razpoložljive internetne vire, ki vam bodo omogočili, da v aplikaciji glede na trenutno lokacijo uporabnika prikažete za uporabnika čim več koristnih informacij.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

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 **Dejan Dakić**,

z vpisno številko **63060561**,

sem avtor diplomskega dela z naslovom:

MOBILNA APLIKACIJA ZA SPREMLJANJE VOZNIH REDOV IN ZAMUD ZA MESTNI AVTOBUSNI PROMET IN ŽELEZNICE

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 **22.09.2011** Podpis avtorja:

Zahvala

Za pomoč pri izdelavi diplomske naloge se zahvaljujem mentorju doc. Dr. Roku Rupniku.

Zahvaljujem se staršem in sestri Lani za lektoriranje ter spodbudo in pomoč v času pisanja diplomske naloge.

Zahvaljujem se tudi prijatelju Darku za vso naučeno znanje, brez katerega izdelava diplomske naloge ne bi bila mogoča, Luku, Roku in Žanu za predloge in vsem sodelavcem za pomoč in podporo.

Kazalo

Seznam uporabljenih kratic in simbolov

WCF	Windows Communication Foundation
EF	EntityFramework
Regex	Regular expressions
IIS	Internet Information Services
XML	Extensible Markup Language
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
AES	Advanced Encryption Standard
SQL	Structured Query Language
LINQ	Language Integrated Query
HTML	Hyper Text Markup Language
GPS	Global Positioning System
IP	Internet Protocol
SVN	Subversion
LPP	Ljubljanski potniški promet
SŽ	Slovenske železnice
POST	Metoda za HTTP zahtevo
WAP	Wireless Application Protocol
WVGA	Wide VGA (800×480) ločljivost

Povzetek

Kot uporabnik javnega potniškega prometa sem večkrat imel težave z voznimi redi, saj podatki o javnem prometu še vedno niso urejeni in enostavno dostopni.

Mobilne tehnologije so v razmahu, s tem pa pa narašča tudi število uporabnikov pametnih telefonov. Razvoj mobilnih tehnologij je prinesel enostaven dostop do spleta in s tem poenostavil dostop do spletnih aplikacij.

Odločil sem se narediti mobilno aplikacijo za pametne telefone, s katero lahko uporabnik na hiter in enostaven način dobi natančne informacije o javnem potniškem prometu.

Pri izdelavi rešitve sem se odločil za uporabo uporabniškega in strežniškega dela, s čimer sem prihranil na prenosu podatkov in se izognil težavam, ki bi nastale pri posodobitvah izvornih strani. Strežniški del je narejen tako, da ga lahko uporabljajo tudi aplikacije za ostale mobilne platforme, kot so Windows Mobile, Windows Phone, Android in druge.

Aplikacija omogoča hiter in enostaven pregled informacij o železniškem in Ljubljanskem mestnem potniškem prometu. Aplikacija s pomočjo sistema GPS omogoča prikaz najbližjih postaj in navodila za prihod do postaje v grafičnem in besedilnem načinu.

Izdelana aplikacija je privlačnega izgleda in enostavna za uporabo. Dnevno jo uporablja približno dvajset ljudi.

Ključne besede: Windows Mobile, Pametni telefoni, Potniški promet, Spletne storitve, Podatkovne baze

Abstract

As a regular public transportation user I had several problems with timetables since the available data on public transport is still not sufficiently ordered and accessible.

Associated with the rise of mobile technologies is the increasing number of smart phone users. With the development of mobile technologies, the access to the world wide web and web applications has become much easier.

I have decided to develop a mobile application for smart phones that enables users to more easily get the accurate up-to-date information on the public transportation. To minimize the amount of transferred data and to allow the application to be seamlessly updated in case of any source data format changes, I have chosen to separate the solution to client and server parts. The server part is created in a way that allows it to be used by other mobile platforms, such as Windows Mobile, Windows Phone, Android and others.

The application allows quick and simple view of the information regarding public railway and Ljubljana bus transport. By facilitating the use of the GPS system, it provides the user with access to the list and directions of the geographically closest stations in both graphical and text modes.

The application is intuitively designed and simple to use. It is being used daily by approximately twenty users.

Key words: Windows Mobile, Smart Phones, Public transportation, Web Service, Database

1 Uvod

Diplomsko delo je mobilna aplikacija z imenom Potnik, ki nam omogoča hiter in enostaven dostop do informacij, kot so prihodi avtobusov in vlakov ter lokacijo najbližjih postaj. Namen aplikacije je, da pred odhodom na postajo preverimo vozni red in se tako izognemo nepotrebnemu čakanju na prihod prevoznega sredstva. Če smo na neznanem mestu in ne vemo kje se postaja nahaja, pa lahko aplikacijo, z uporabo sistema GPS uporabimo kot vodič k najbližji postaji.

Aplikacija je narejena za mobilne telefone z operacijskim sistemom Windows Mobile 6.5, narejena je v programskem ogrodju .NET Compact 3.5 in programskim jezikom C#. Za izvedbo sem uporabil programski paket Visual Studio, v katerem sem moral narediti dva projekta. Aplikacijo ki teče na telefonu in strežniški del, ki teče na strežniku. Z uporabo strežnika se izognemo nepotrebni obdelavi podatkov na mobilnem telefonu. Aplikacija pošlje zahtevo na strežnik, ta pa nato dostopa do spletnih strani in z uporabo regularnih izrazov sintaktično analizira podatke, jih združi skupaj v urejeno celoto in jih nato posreduje nazaj na mobilno napravo, kjer mobilna aplikacija poskrbi za njihov prikaz. Tako pospešimo celotno aplikacijo in privarčujemo na življenjski dobi naprave, saj se zahtevno procesiranje izvaja na strežniku. V naslednjih poglavjih so podrobneje opisane uporabljene tehnologije, arhitektura programa, strežniški del programa in aplikacija.

Aplikacijo sem naredil na željo prijateljev, za lastne potrebe in kasneje tudi z namenom udeležbe na tekmovanju Dnevi mobilne informatike, ki je potekalo na FRI-ju oktobra leta 2010. Z aplikacijo Potnik sem dosegel prvo mesto in bila je zmagovalna aplikacija na tem tekmovanju.

2 Uporabljene tehnologije

Tekom študija smo zelo malo uporabljali Microsoftove tehnologije. Menim, da je Microsoft vodilen v razvoju računalniške programske opreme in s tem tudi v razvoju programskih jezikov. Odločil sem se, da bom uporabljal Microsoftove tehnologije, saj mi omogočajo velik razpon prenosljivosti z uporabo istih orodij.

V razvojen okolju Visual Studio lahko razvijamo namizne, spletne in mobilne aplikacije na povsem isti način. Lahko tudi razvijamo veliko drugih reči, obenem pa programske knjižnice in strežniške aplikacije, katero sem razvil jaz, ki jo uporablja aplikacija Potnik. Razvijal sem v programskem ogrodju .NET in jeziku C#. Mobilni del sem razvijal v ogrodju .NET Compact Framework 3.5, saj operacijski sistem Windows Mobile ne podpira novejše verzije ogrodja. Za uporabo strežniškega dela pa sem uporabil najnovejšo ogrodje in principe, ki jih ponuja Microsoft.

Ena izmed novosti v novem programskem ogrodju .NET 4.0 je Windows Communication Foundation – vmesnik za izdelavo strežniških aplikacij, ki ima zelo enostavno in varno komunikacijo z ostalimi aplikacijami narejenimi v ogrodju .NET. Omogoča tudi uporabo formatov kot so JSON, XML in ostale. S podporo ostalih formatov lahko uporabljamo strežniške storitve tudi na drugih operacijskih sistemih in poljubnih jezikih.

Za komunikacijo z bazo sem uporabil brezplačni in privzeti relacijski bazni strežnik, ki ga ponuja Microsoft, Windows Server 2008 Express. Za izdelavo podatkovnega modela sem uporabil brezplačen program Microsoft SQL Server Management Studio Express. Strežniška storitev komunicira z bazo prek vmesnika Entity Framework. Ta vmesnik nam omogoča zelo enostavno integracijo podatkovnega modela v aplikacijo tako, da generira podatkovne tipe in objekte, ki obdelujemo v toku delovanja aplikacije. Omogoča tudi nam tudi poizvedovanje v bazo s pomočjo LINQ. Tehnologija LINQ nam poenostavi pisanje kode, manipulacijo nad objekti in tudi poizvedovanje po bazi, brez uporabe SQL stavkov.

Za izločanje podatkov iz HTML dokumenta sem se odločil za uporabo regularnih izrazov, saj se mi zdijo regularni izrazi zelo pomembni za znanje razvijalca programske opreme. Ker pred tem projektom še nisem imel izkušenj z regularnimi izrazi, sem se odločil, da se jih naučim in uporabim v aplikaciji. Pri pisanju regularnih izrazov sem si pomagal s programom RegexBuddy, ki poenostavi pisanje izrazov z barvno označitvijo rezultatov. V mobilni aplikaciji sem želel uporabiti čim več tehnologij. Tako sem uporabil sistem GPS za zaznavo trenutne lokacije in Googlov program za zemljevide z imenom Google Maps.

V naslednjih poglavjih so vse omenjene tehnologije podrobneje opisane.

2.1 Visual Studio

Visual Studio je programsko okolje, katerega začetki segajo že v leto 1997. Omogoča nam pisanje konzolnih, spletnih in grafičnih aplikacij v sistemu Windows. V okolju Visual Studio programiramo v ogrodju .NET. Izbiramo lahko med paleto Microsoftovih programskih jezikov kot so C#, C++, F#, Visual Basic in ostalimi. Izbiramo pa lahko tudi med programskimi jeziki, ki niso Microsoftovi, npr. Pyhron, Ruby, JavaScript ter mnogi ostali. Okolje nam nudi napredne funkcije kot so razhroščevanje v realnem času, uporaba več monitorjev za preglednejši nadzor kode, samodejno dokončanje kode in ostale.

Pri razvoju sem uporabil Visual Studio 2008, za mobilno aplikacijo in Visual Studio 2010 za strežniško aplikacijo.

2.1.1 Programsko ogrodje .NET

Programsko ogrodje .NET je skupek vmesnikov in knjižnic, ki nam omogočajo lažje razvijanje aplikacij. Ogrodje nam nudi knjižnice za grajenje uporabniških vmesnikov, spletnih komunikacij, dostopov do različnih podatkovnih baz, kriptografijo in mnogo drugih. Dobra stran ogrodja .NET je, da uporabljamo isto ogrodje pri več različnih programskih jezikih. To pomeni, da lahko knjižnico, ki je napisana v programskem jeziku Visual Basic, uporabimo pri razvoju v jeziku C#. Ogrodje nam nudi tudi določeno mero prenosljivosti. Aplikacije napisane v ogrodju .NET lahko namreč uporabljamo na sistemih Windows, Mac in Linux, a imata slednja določene omejitve.

2.1.1.1 .NET Compact Framework 3.5

Mobilna aplikacija uporablja ogrodje z imenom .NET Compact Framework 3.5. Ogrodje je namenjeno kompaktnim napravam kot so mobilni aparati, POS terminali in ostale manjše računalniške naprave. Ta različica ima sicer veliko omejitev, zato sem moral večino stvari razviti sam. Tako sem razvil gradnik za samodejno dokončanje vnosa in prikaz možnosti, gradnik gumb, ki podpira transparentnost, modul za avtomatično posodabljanje verzije in še nekaj ostalih. Te gradnike sem naredil tako, da sem uporabil gradnike ki jih ponuja ogrodje in jih sestavil skupaj v nove. Ogrodje omogoča nitenje aplikacije. S tem, da aplikacija teče na več nitih, sem dosegel, da je program odziven v vsakem trenutku, tudi, ko se povezuje s strežnikom, ali pa, ko izrisuje podatke na zaslon. Za razvoj mobilne aplikacije v sistemu Windows Mobile 6.5 je potrebna uporaba Visual Studia 2008.

2.1.1.2 .Net 4.0

Strežniški del sem napisal v ogrodju .NET 4.0. To je najnovejše ogrodje, ki je izšlo skupaj s programskim paketom Visual Studio 2010. V tej verziji so pri Microsoftu izboljšali podporo paralelnosti in več jedrnim procesorjem. LINQ je dobil podoro za parelelnost z imenom PLINQ in v programskih jezikih je možnost pisanja LINQ stavkov z lambda izrazom. Za novejšo verzijo ogrodja sem se odločil zaradi uporabe novega načina spletnih strežniških aplikacij WCF in tudi novega, prijaznejšega okolja.

2.1.2 Programski jezik C#

Programski jezik C# je mlajši programski jezik, ki je nastal leta 2002 kot sestavni del ogrodja .NET Framework 1.0. C# je objektno orientiran funkcionalni programski jezik, ki ga je razvil Microsoft, kot odgovor na programski jezik Java, ki ga je razvilo podjetje Sun. Uporaba programskega jezika C# je primerna za razvoj aplikacij v ogrodju .NET. Sintaktično je zelo podoben programskemu jeziku C++ in Javi. Za razliko pri programiranju v C++, se nam v C# ni treba ukvarjati z nastavljanjem pomnilniških naslovov in alokacijo spomina, saj to počne prevajalnik sam. Program s podobno kodo se bo hitreje izvršil v C++ kot v C#, vendar je ta razlika skoraj neopazna, ko razvijamo splošne aplikacije.

Pri razvoju strežniške aplikacije sem uporabljal C# verzije 4.0, pri mobilni aplikaciji sem moral uporabiti C# 3.0.

2.1.3 Windows Communication Foundation

Windows Communication Foundation (WCF) je ogrodje za grajenje strežniških aplikacij. Z uporabo WCF lahko pošiljamo različne podatke več odjemalcem naenkrat. WCF je lahko samostojna aplikacija, lahko pa teče v okviru IIS spletnega strežnika. Pošiljamo lahko enostavne tipe, denimo cela števila ali nize znakov, ali pa zahtevnejše tipe, ki jih sestavimo sami[1]. Pošiljamo lahko tudi XML ali pa binarne podatke. Deluje lahko na različnih protokolih kot so SOAP, HTTP in drugi. Omogoča zelo enostavno integracijo s C# projekti. Komunikacija je v projekt vgrajena na zelo preprost način: prenesejo se podatkovni tipi, metode in ostale stvari, potrebne za delovanje strežnika. WCF lahko uporabimo tudi pri gradnji aplikacij za druge operacijske sisteme. WCF lahko na zahteve odgovarja v formatu JSON, kar je priročno za aplikacije, ki tečejo v Linux sistemu ali drugih operacijskimi sistemih in spletnih brskalnikih.

2.1.4 Windows SQL Server 2008 Express

Windows SQL Server 2008 je Microsoftova relacijska podatkovna baza. Obstaja več različic, jaz sem se odločil za brezplačno različico Express, ker ima zelo dobro podporo pri izdelavi .NET aplikacij v Visual Studiu. Baza je zelo varna, omogoča 128-bitno in 256-bitno AES zaščito. Brezplačna različica nam ponuja bazo veliko do dveh gigabajtov[2]. Novejša različica SQL strežnika ima precej novosti v primerjavi s starejšo, je tudi bolj zahtevna pri strojni opremi in operacijskem sistemu.

2.1.5 Microsoft SQL Server Management Studio Express

Program Microsoft SQL Server Managment Studio Express je brezplačen program za upravljanje s podatkovno bazo. Uporabil sem različico 2008, saj sem uporabljal SQL Server 2008. S tem orodjem lahko izdelamo podatkovni model, pišemo SQL poizvedbe, ga uporabimo tudi za izris različnih diagramov. Povežemo se lahko na več različnih strežnikov in izvajamo poizvedovanja med njimi. Bazo, ki jo uporablja strežni del, sem v celoti razvil s tem orodjem. Ko sem na prenosnem računalniku aplikacijo nameščal iz razvojnega okolja na strežnik, sem tudi s pomočjo orodja na zelo preprost način prenesel celotno strukturo baze in podatke na strežnik. Program omogoča tudi razhroščevanje med izvajanjem SQL poizvedbe, kar je še posebej priročno pri zahtevnejših poizvedbah.

2.1.6 Entity Framework

Entity Framework (EF) ogrodje za objektno-relacijsko preslikovanje baze za .NET ogrodje. Z uporabo EF sem poenostavil komunikacijo med aplikacijo na strežnikom in bazo. V fazi razvoja ni bilo treba skrbeti za preslikovanje modela ob spremembah, saj se je model ob spremembi v bazi samodejno posodobil. Z uporabo EF lahko poizvedujemo po bazi na SQL način, lahko pa pišemo LINQ poizvedbe, ki omogočajo zlaganje ukazov in ima ostale prednosti [4]. V aplikaciji sem uporabljal LINQ. Z uporabo Entity Framework lahko v projekt preslikamo tudi druge podatkovne baze kot so DB2, Oracle, MySQL ter mnoge druge.

2.2 LINQ – Language Integrated Query

LINQ, ki se izgovori »link«, je Microsoftovo ogrodje za poizvedovanje po različnih virih podatkov. LINQ lahko uporabimo pri poizvedbah z bazo, kar sem uporabil tudi v strežniški aplikaciji. Lahko ga uporabimo pri standardnih oz. lastnih kompleksnih podatkovnih tipih, kar nam omogoča, da se izognemo pisanju odvečnih kod, namesto tega pa uporabimo enostaven LINQ izraz [3].

Z uporabo LINQ postane programska koda bolj berljiva, lažje je tudi njeno popravljanje je lažje. Z uporabo LINQ lahko poizvedujemo tudi po XML strukturah, kar je danes zelo uporabno, saj je velik del podatkov shranjenih v XML strukturi. LINQ je tudi mnogo hitrejši kot poizvedovanje na običajen način, pisanje navadnih zank oz. ForEach zank in odstotek hitrosti, pa je odvisen od primera. LINQ ne izvrši poizvedbe takoj, temveč uporablja sistem zlaganja poizvedb.

Zlaganje poizvedb je sicer zelo zanimiv pristop. Pri poizvedovanju s SQL se za vsako poizvedbo povežemo z bazo, kar pa zahteva svoj čas. LINQ shranjuje oz. zlaga ukaze na sklad in jih izvrši naenkrat. Tudi, če je več zaporednih ukazov za poizvedbo na bazo, bo LINQ sam ugotovil, da je bolje, da se poveže le enkrat in izvrši vse ukaze naenkrat. Z ogrodjem .NET 4.0 je prišel tudi PLINQ, kar pomeni paralelni LINQ in omogoča poizvedovanje v paralelnem načinu, kar je pa zelo uporabno pri današnjih večjedrnih procesorjih.

2.3 Regularni izrazi

Regularne izraze pogosto poimenujemo kar regex. Regularni izrazi se uporabljajo za iskanje in zamenjavo besedila. So izjemno zahtevni za uporabo, saj imajo zelo obsežen seznam pravil. Navadni računalniški uporabniki in neizkušeni programerji se jih pogosto izogibajo ravno zaradi njihove kompleksnosti. Regularni izrazi niso enaki za vse programske jezike: obstaja razlika med izrazom napisanim za programski jezik Java in .NET[5]. Te razlike pri enostavnih izrazih sicer niso vidne, postanejo pa zelo opazne pri zahtevnejših izrazih.

V mojem strežnem delu aplikacija komunicira s spletnimi stranmi, ki so napisane v jeziku HTML. HTML je urejena struktura podatkov, zato so regularni izrazi primerno orodje. Če bi se spletna stran spremenila, tudi moj strežni del ne bi deloval več, saj bi nastala drugačna struktura, ki pa je trenutni regularni izraz ne bi ujel. V aplikaciji bi lahko uporabil HTML razčlenjevalnik, ampak sem se zaradi želje po znanju regularnih izrazov odločil za uporabo le-teh. Medtem ko bi ob spremembi spletne strani moral spremeniti kodo, pa moram z uporabo regularnih izrazov spremeniti le izraz.

2.3.1 Program RegexBuddy

Pri pisanju regularnih izrazov sem si pomagal s programom RegexBuddy, sicer je plačljivo orodje, a je na voljo tudi brezplačna različica, ki jo lahko uporabljamo deset dni. V program sem vnesel naslov spletne strani in nato začel s pisanjem regularnih izrazov. Program podpira skoraj vse programske jezike, kjer je podprta uporaba regularnih izrazov. Je nepogrešljiv pripomoček, ko v programski rešitvi uporabljamo regularne izraze, uporabimo pa ga lahko tudi kot urejevalnik besedil in z izrazi iščemo in menjamo dele besedila. Zelo zanimiva

funkcija programa je barvanje zadetkov v realnem času. Ko pišemo izraz, nam program samodejno pobarva del besedila, ki se ujame z določenim izrazom. Tako lahko hitro preverimo, ali smo napisali pravilen regularni izraz.

Regularni izrazi sicer vsebujejo veliko poševnic in narekovajev, ki so v mnogih programskih jezikih obravnavani kot posebni znaki. Z njimi je treba biti precej pazljiv, zato nam program RegexBuddy omogoča, da si izraz v pravilni obliki shranimo v odložišče in ga nato enostavno prilepimo v programsko kodo.

2.4 Mobilni operacijski sistem Windows Phone 6.5

Windows Mobile je mobilni operacijski sistem, ki ga je razvil Microsoft. Uporablja se na pametnih telefonih in mobilnih napravah. 6.5 je zadnja različica saj je oktobra leta 2010 prišla v celoti obnovljena nova različica mobilnega operacijskega sistema Windows Phone 7. Različica 6.1 temelji na jedru sistema Windows CE 5.2. Različica je 6.1 je prišla kot posodobitev za sistem Windows Mobile 6.1. Po različici 6.5 so na trg prihajale novejše različice sistema 6.5 kot so 6.5.1, 6.5.3 in 6.5.5. Vsaka od teh različic je vsebovala novosti, s katerimi je Windows Mobile konkuriral mobilnim sistemom na trgu. Windows Mobile je izjemno robusten sistem in je skoraj kot namizna verzija Windowsov, saj imamo na voljo programski paket Office 2010, program za komunikacijo z Exchange strežnikom Outlook in kopico ostalih zastonjskih programov. Prednost Windows Mobile pred mobilnimi sistemi je izjemno dobra integracija z namiznimi različicami sistema Windows.

Razvijalci in programerji imajo na sistemu Windows Mobile zelo proste roke kar se tiče dostopov do vmesnikov, kar je vodilo do eksplozije aplikacij in igric za ta sistem. V novejši različici Windows Phone 7 so stvari drugačne, bolj zaklenjene, npr. dostop do vtičnic ni mogoč, medtem ko Windows Mobile 6.5 takšnih omejitev nima.

Za razvoj v Windows Mobile 6.5 sem se odločil na podlagi preteklih izkušenj v pisanju aplikacij za sistem 6.5. Poznal sem vmesnike in strukturo in imel telefon s operacijskim sistemom Windows Mobile 6.5, tako da sem lahko izdelek testiral v pravem okolju in ne samo v simulatorju. Testiranje na dejanski napravi mi je omogočilo, da aplikacijo optimiziram kar se da najbolje z uporabniškim vmesnikom kot performančnimi operacijami, saj je razlika med simulatorjem in napravo precejšna. Ko sem začel z razvojem aplikacije sploh še ni bilo mobilnih aparatov z Windows Phone 7. Zdaj pa je že napisana različica Potnika za Windows Phone 7 z imenom Sedmi Potnik.

2.5 Sistem globalnega določanja lege (GPS)

GPS je sistem za globalno določanje položaja. Naredili so ga v obrambnem ministrstvu ZDA za potrebe ameriške vojske. GPS uporablja satelite, ki krožijo okoli zemlje, sistem pa sestavlja 24 satelitov.

Pri aplikaciji Potnik z uporabo GPS dobimo podatke o bližnjih postajah. Podatki so shranjeni v bazi na strežniku. Zapisani sta dolžina in širina. Mobilna aplikacija pošlje le svoje trenutne koordinate na strežnik, strežnik pa poišče najbližje postaje in na podlagi formule izračuna zračno razdaljo med našo trenutno pozicijo in pozicijo postaje.

Uporaba GPS sprejemnika na mobilni napravi je precej počasna in zahteva precej energije, kar pomeni, da se baterija hitro izprazni. Aplikacija problem rešuje tako, da vsaj enkrat tedensko z uporabo IP naslova razbere približno lokacijo. GPS sprejemnik zna izračunati, katere satelite lahko pričakuje in tako hitreje razbere našo lokacijo. S tem prihranimo na času in bateriji. V aplikaciji je uporabljen tudi sistem za kratkoročno shranjevanje koordinat. To pomeni, da če poženemo aplikacijo v manj kot eni minuti, se uporabijo iste koordinate, razen če eksplicitno zahtevamo izračun novih koordinat in postaj.

2.6 Google Maps

Googlov strežnik in program Google Maps vsebuje geografske podatke in zemljevide. Obstaja več različic aplikacij, spletna, namizna in mobilna. Program je na voljo za več mobilnih sistemov. S pomočjo Google Maps si lahko ogledamo teren z različnimi oblikami zemljevidov. Tako vidimo relief površja, lahko pa uporabimo tudi satelitsko različico, ki nam sicer nudi omejen pogled reliefa, a omogoča vpogled za ceste, železniške postaje, reke in ostale značilnosti terena. V nekaterih državah je omogočen tudi t.i. ulični pogled, s katerim vidimo okolico iz vseh kotov. Ulični pogled v Sloveniji ni mogoč. Program lahko uporabimo tudi za izpis navodil iz mesta A v mesto B, nudi nam tudi izpis znamenitosti, ki so na tej poti.

V aplikaciji Potnik sem uporabil Google Maps za izris poti od naše trenutne lokacije do izbrane postaje. Če pa smo postajo izbrali ročno, pa lahko z Google Mapsom pogledamo, kje se ta postaja nahaja. Aplikacija nam ponudi tudi izpis navodil do postaje v tekstovni in grafični obliki.

2.7 Sistem za kontrolo različic datotek (Subversion)

Subversion pod kratico SVN, je sistem za kontrolo različic. Je brezplačen, koda je dostopa vsakomur – odprtokoden. SVN upravlja z datotekami in mapami ter njihovimi različicami. Glavna struktura datotek se nahaja v t.i. repozitoriju. Ta se nahaja na strežniku, ima možnost da si zapomni vse spremembe, ki so se zgodile kdajkoli na datotekah oz. mapah. S tem lahko vrnemo projekt na prvotno verzijo ali pa samo določeno datoteko. Lahko pogledamo tudi razliko med delovno različico datoteke in različico na strežniku.

Z uporabo SVN sem lahko razvijal več funkcionalnosti hkrati, ne da bi pri tem ogrozil delovanje glavne različice programa. Po koncu razvoja sem združil razvito funkcionalnost v glavni del.

3 Analiza in načrtovanje

V naslednjem poglavju bom predstavil način, kako sem analiziral problem in načrtoval arhitekturo sistema.

Pri analizi sem si pomagal z znanjem, pridobljenim med študijem. Prvi korak je bil izdelava diagrama primera uporabe, v katerem sem zajel vse zahteve in načrtovane funkcionalnosti. Po dokončanem diagramu primera uporabe sem se lotil načrtovanja arhitekture.

Tega se nisem naučil na fakulteti, temveč med delom, ki sem ga izvajal med študijem. Odločil sem se za navadno obliko tri-nivojske arhitekture s prezentacijskim nivojem na mobilni napravi.

Razlog za to odločitev je centraliziran logični nivo, ki nam omogoča enostavno urejanje kode. S tem je močno poenostavljeno posodabljanje aplikacije [6]. Uporabnikom ni treba prenesti nove različice programa, če se denimo spremeni spletna stran iz katere zajemamo podatke, odpravljajo napake na strežniškem delu, spremenijo podatki v bazi itd. Podatkovna baza je samo ena, dostop do baze pa je omogočen samo prek strežnika. Če pride do spremembe na bazi, nam ni treba skrbeti za posodobitev sprememb na vseh napravah, ki uporabljajo aplikacijo.

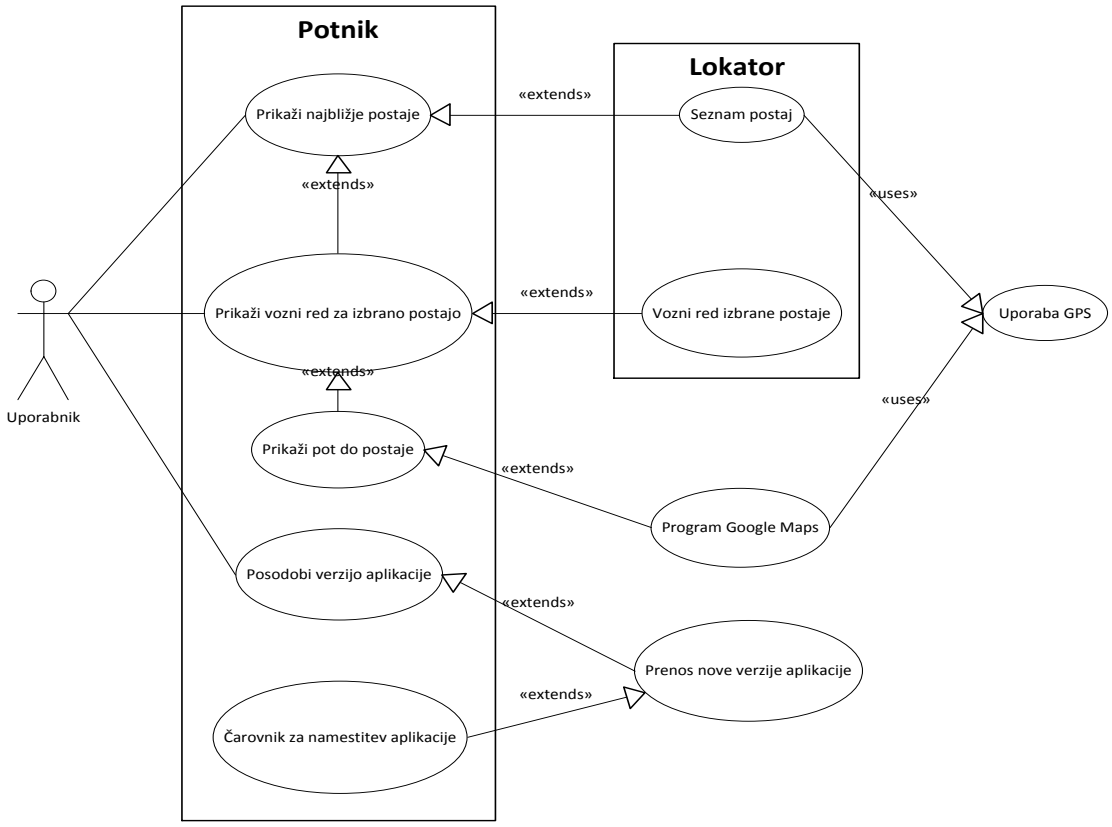
Aplikacija komunicira samo s strežnikom tam, kjer je postavljen strežniški del in zato nudi večjo anonimnost, saj do strani z informacijami ne dostopamo neposredno iz naprave temveč za to poskrbi strežniški del. Podatki se prenašajo prek interneta, vsak prenesen podatek pa pomeni večji strošek delovanja storitve. Če bi se odločil in izpustil strežniški del, bi bila celotna storitev precej dražja. Strežniški del opravi poizvedovanje po spletnih straneh s podatki, iz njih izlušči potrebne informacije, jih zloži v urejeno množico podatkov in pošlje na mobilno napravo.

Mobilna aplikacija je zelo optimizirana, skrbi le za izpis podatkov na zaslon. Razlog za centraliziran strežniški del ni samo v ceni storitve, temveč tudi v hitrosti. Mobilne naprave imajo počasne centralno-procesorske enote in delujejo z baterijo. V primeru, da mobilna aplikacija deluje brez strežnika, to pomeni, da mora vse opraviti sama. Operacije kot so povezovanje z zunanjimi strežniki nam prihranijo čas in prenos podatkov. Sintaktično analiziranje podatkov je zelo zahtevna računalniška operacija, za katero mobilna centralna-procesorska enota potrebuje precej več časa kot za isto operacijo potrebuje strežnik, ki ima na voljo večja strojna sredstva. Aplikacija je zasnovana za mobilne aparate, z zasloni na dotik. Ti

so navadno večje velikosti kot navadni mobilni zasloni, zato se hitreje izprazni baterija. To je tudi eden izmed razlogov, da procesiranje opravi strežnik, saj vsaka operacija, ki bi bila izvedena na aparatu, zahtevala procesorsko moč. Večja procesorska moč pa ima tudi večjo porabo baterije. S centraliziranim strežniškim delom prihranimo tudi na življenjski dobi aparata.

Ločen strežniški del nam tudi omogoča enostavno razširitev storitve na druga okolja. Treba je napisati le program za poljuben operacijski sistem, ki komunicira s strežnikom in izpisuje pridobljene podatke. Programi ki uporabljajo strežniški del Lokator, so trenutno podprti za štiri sisteme: Windows Mobile 6.5, Windows Phone 7, Windows 7 in Andorid. Slaba stvar ločenega strežniškega dela pa je dodaten strošek strojne opreme in internetne povezave, brez katere storitev ne bi mogla delovati.

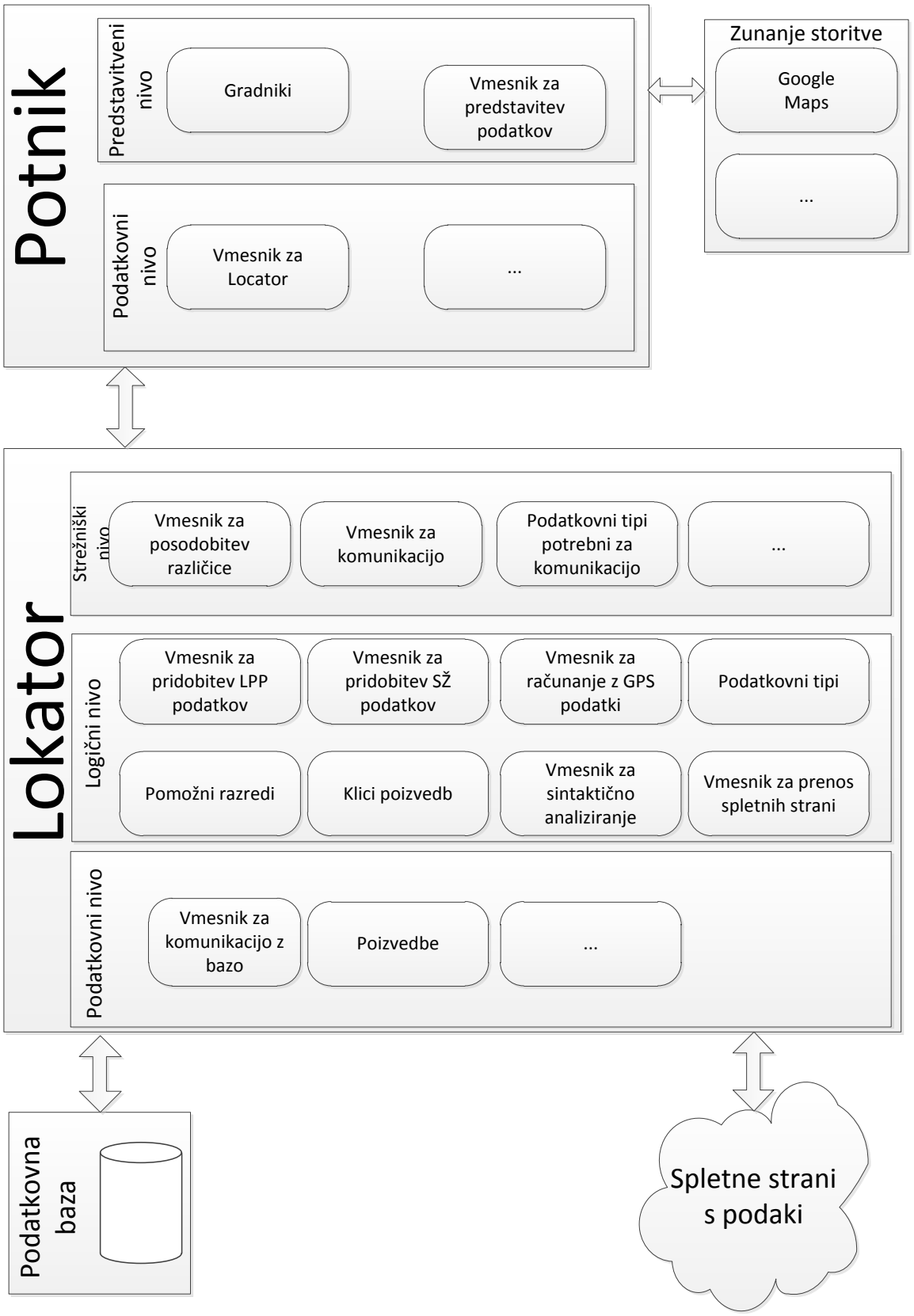
3.1 Diagram primera uporabe



Slika 1. Diagram primera uporabe.

Diagram primera uporabe je shema, s katero na grafični način prikažemo in zajamemo funkcionalnosti, ki jih sistem omogoča oz podpira. Glavni namen diagrama poteka je prikaz vseh funkcij, ki jih lahko izvrši akter in v kakšnem zaporedju jih lahko izvršuje. V mojem primeru je akter samo uporabnik.

3.2 Arhitektura



Slika 2. Model arhitekture aplikacije.

Arhitektura aplikacije je sestavljena iz treh delov. Prvi del je mobilna aplikacija Potnik, drugi je strežniški del imenovan Lokator, ki je povezan s tretjim delom, podatkovno bazo. V naslednjih treh poglavjih so opisane strukture delov, ki sestavljajo aplikacijo. Na sliki modela arhitekture aplikacije je narisana groba skica arhitekture in vmesnikov, ki so implementirani na določenih nivojih.

3.3 Podatkovni model

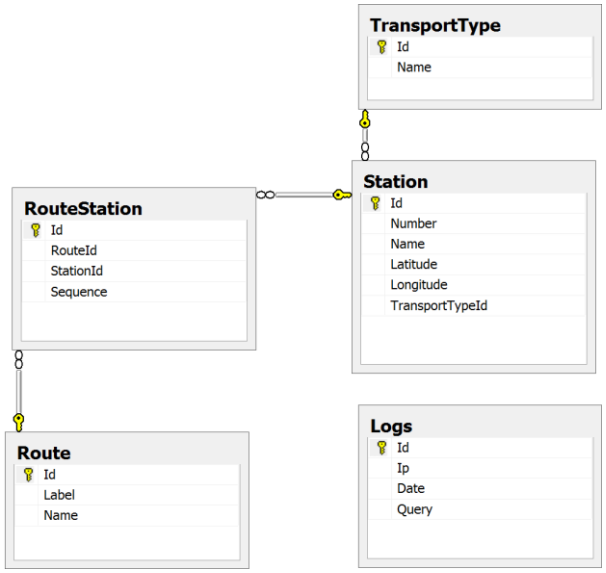
Podatkovni model je precej enostaven. Sestavljen je le iz petih entitet, ki pa povsem zadoščajo mojemu informacijskemu sistemu. Za arhiviranje dostopa do aplikacije je zadolžena entiteta Logs. Vsak dostop do strežnika se zapiše: beleži se IP naslov, točen čas in tudi, kakšna je bila poizvedba. Entiteto Logs sem sprva uporabljal le za testiranje, ko je pa aplikacija postajala vedno bolj priljubljena, sem z njeno pomočjo naredil statistiko uporabe aplikacije in funkcij, ki se največkrat uporabijo.

V entiteti TransportType so shranjeni tipi prevozov. Sedaj sta podprta le dva tipa prevozov, avtobus in vlak. V fazi razvoja pa je tudi funkcionalnost, ki bi podpirala dostop do telefonskih številk taksijev v določenem kraju.

V Station so shranjene vse postaje. Vsaka postaja je določena s tipom prevoza, da vemo ali gre za železniško postajo ali za avtobusno postajališče. Shranjeno je ime in številka postaje. Številka postaje pri železniških postajah se ujema z identifikacijsko številko postaje na spletni strani Slovenskih železnic. V tej entiteti sta shranjena tudi parametra za dolžino in širino, ki skupaj določata položaj postaje. S tema podatkoma lahko izračunam najbližje postaje okoli trenutnega položaja mobilne naprave, uporabim ju tudi, ko na zemljevidu izrisujem pot do postaje.

Route hrani vse mestne povezave – proge, kot so npr. »ČRNUČE – DOLGI MOST«. Hrani tudi številko postaje.

V RouteStation so shranjeni ID-ji vseh postaj, ki so na določeni progi.



Slika 3. Podatkovni model.

4 Strežniški del Lokator

Strežniški del se imenuje Lokator. To ime je dobil, ker je bila to prva funkcionalnost, ki jo je strežnik ponujal. Nato sta sledili še funkcionalnosti za zajem podatkov iz spletnih strežnikov podjetij LPP in SŽ. Strežniški del skrbi za oskrbo mobilnih aparatov s podatki in dostop do podatkovne baze. Strežnik nam lahko odgovori na vprašanja kot so, kje so najbližje postaje, kdaj odpeljejo določeni prevozi in številna druga.

Strežnik je narejen tako, da se lahko pogovarja prek različnih protokolov. Strežnik si beleži dostope do metod, da lahko lažje izmerimo, koliko ljudi uporablja aplikacijo.

Strežnik nam ponuja razrede in metode potrebne za komunikacijo in obdelavo podatkov. Na aplikacijski strani je treba uporabiti ponujene razrede, saj strežnik pošilja podatke v specifični obliki. Podatkovni tipi, ki se uporabljajo na strežniškem delu, so različni od tistih, ki jih strežnik pošlje v aplikacijo. S tem sem dosegel visoko optimizacijo ob procesiranju in urejeno logično celoto ob prenosu podatkov. Enostavna struktura na mobilni aplikaciji omogoča, prikaz podatkov na brez zahtevnega procesiranja.

Dostop do baze poteka prek Entity Framework. Z uporabo EntityFrameworka sem lahko uporabljal LINQ poizvedbe neposredno nad objekti baze, ki sem jih združil v skupni razred. Tako sem dosegel centralizacijo vseh poizvedb, kar mi je omogočilo, da sem v programsko kodo vgradil ustrezne varnostne mehanizme le na enem mestu.

Vmesniki, ki so implementirani na strežniku, so narejeni tako, da se lahko uporabijo tudi pri drugih projektih. Vsak vmesnik je umeščen v svojo knjižnico. Strežnik ob zagonu prebere vse knjižnice in s tem vse vmesnike, ki jih ima na voljo in jih uporabi v delovanju.

Strežnik sem razvijal približno petdeset ur. V ta čas ni vključen čas učenja in branja dokumentacije. Z razvojem strežniškega dela sem se naučil veliko o samem delovanju programskih jezikov in o gradnji skalabilnih aplikacij. Spoznal sem tudi ozadje in filozofijo programskega jezika C#, saj sem se trudil, da sem sledil vsem programskim standardom.

4.1 Lokator

Lokator skrbi za obdelavo podatkov, ki so potrebni za pozicioniranje podatkov uporabnika aplikacije. Lokator ima implementirano logiko za računanje zračne razdalje med dvema točkama na zemljevidu. To počne s posebno matematično formulo.

$$\boldsymbol{haversin}\left(\frac{\boldsymbol{d}}{\boldsymbol{R}}\right) = \boldsymbol{harvesin}(\boldsymbol{\varphi_2} - \boldsymbol{\varphi_1}) + \mathbf{cos}(\boldsymbol{\varphi_1}) * \mathbf{cos}(\boldsymbol{\varphi_2}) * \boldsymbol{harvesin}(\Delta\boldsymbol{\lambda}) \quad (1)$$

S Haversinovo formulo sem lahko izračunal približno razdaljo med napravo in izbrano postajo. K razdalji sem dodal še sto metrov zaradi možnih ovir na poti. V fazi testiranja so uporabniki dobivali zelo dobre rezultate.

Lokator ima tudi shranjene koordinate postaj, ki nam pomagajo pri prikazu relevantnih podatkov pri izbiri z uporabo standarda GPS. Če se nahajamo nekje na Viču, nas ne zanima kateri avtobusi vozijo okoli Bežigrada oz. katere železniške postaje so v bližini Ljutomera. Ko izberemo možnost prikaza poti do postaje s programom Google Maps, steče komunikacija preko Lokatorjevega vmesnika, da dobi koordinate, ki jih nato uporabi kot vstopne parametre za program Google Maps.

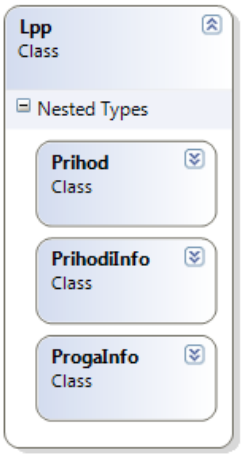
4.2 Modul za Ljubljanski potniški promet

Podatke, potrebne za obdelavo in prikaze mestnega potniškega prometa v Ljubljani, sem dobil na spletni strani podjetja Telargo na naslovu http://bus.talktrack.com. Zaradi uporabe POST metode na Telargovem spletnem strežniku, sem moral napisati svoj vmesnik za lažje vzdrževanje kode. Spletna stran je imela zaščito, ki jo je bilo treba prebiti, da bi omogočil avtomatiziran način zajemanja informacij. Metoda, ki zajema podatke iz Telargove spletne strani, prvič dostopa do strani za pridobitev piškotka (ang. Cookie). V drugi zahtevi uporabimo pridobljen piškotek in uporabimo pravilno poizvedbo, na katero nam strežnik odgovori s HTML datoteko, ki vsebuje informacije. Z uporabo regularnih izrazov sintaktično analiziramo pridobljeno HTML datoteko. Po končani analizi se izluščijo vsi pomembni podatki. Pridobljeni podatki se zapakirajo v urejeno množico podatkov, ki jih pošljemo na mobilno napravo, da prikaže zahtevane podatke.

Podatki na Telargovi spletni strani so natančni. Avtobusi imajo nameščen GPS oddajnik, ki komunicira s strežnikom. S podatki, ki jih pošljejo avtobusi, na Telargu natančno izračunajo prihod avtobusa na postajo. Pri tem upoštevajo semaforje in mesta, kjer je pogosta prometna konica. Na spletni strani se podatki posodabljajo iz minute v minuto. Zaradi posodobljenih podatkov na Telargovi spletni strani imamo tudi v aplikaciji Potnik vedno sveže podatke. Za vsako poizvedbo v aplikaciji Potnik, Locatorjev vmesnik za LPP naredi poizvedbo in nam vrne posodobljene podatke.

Če bi bile GPS koordinate, ki jih pošiljajo avtobusi, javne, bi lahko naredili uporabniku prijazen vmesnik, na katerem bi se po zemljevidu premikal avtobus in s pritiskom nanj prikazal čas, ki ga bo porabil za prihod do postaje.

Trenutno so v Sloveniji ti podatki javno dostopni samo v Kopru. Aplikacija Potnik tega območja ne podpira, saj v času razvoja aplikacije, ti podatki niso bili javni. Implementacija koprskega območja je načrtovana v letu 2011.

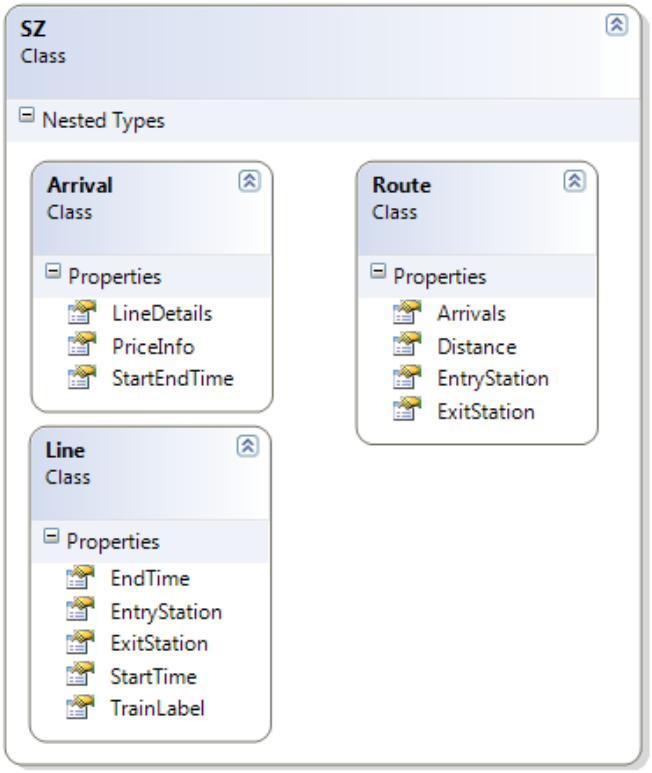


Slika 4. Objektni model za podatkovno strukturo LPP.

4.3 Modul za Slovenske železnice

Pridobivanje podatkov s spletnih strani Slovenskih železnic (SŽ) je bilo sprva precej zapleteno, zaradi izjemno zahtevne strukture njihove spletne strani, na katero nisem uspel napisati dobrega regularnega izraza. Zadeve sem se lotil na drugačen način. Ko sem iskal rešitev, sem naletel na WAP različico SŽ portala. Ta je bila zelo oskubljena in regularni izraz za sintaktično analizo podatkov je bil precej lahek. Zaradi enostavnosti je celotna procedura postala precej hitrejša, kot je bila z uporabo običajne spletne strani SŽ.

Pridobitev podatkov z uporabo WAP portala je bila enostavna. Veliko bolj zahtevno pa je bilo narediti urejeno množico podatkov za SŽ. V nasprotju s podatki o mestnih avtobusih in njihovih prihodih so bili ti pri Slovenskih železnicah bolj zapleteni. Upoštevati je bilo treba tudi prestope in drugačne linije kot pri mestnem prometu. Vlak uporabi isto linijo za relacijo med Celjem in Laškim in za relacijo med Celjem in Ljubljano.



Slika 5. Objektni model za podatkovno strukturo SŽ.

5 Mobilna aplikacija – Potnik

Mobilna aplikacija Potnik je narejena za mobilni operacijski sistem Windows Mobile 6. Aplikacija je na voljo tudi v sistemih Windows Phone 7 in Andorid. Narejena je za naprave, ki imajo zaslon na dotik in ločljivost zaslona 800 pikslov v višino in 480 pikslov v širino. V aplikaciji je možen tudi GPS način, kar je primerno za naprave opremljene z GPS sprejemnikom. Aplikacija dobi podatke iz interneta, zato je povezljivost s spletom pogoj za uporabo aplikacije. Količina podatkov, ki jo porabi aplikacija za pridobitev informacij, je minimalna. V prihodnosti bo aplikacija imela tudi podporo za pridobitev informacij o taksi službah v največjih mestih Slovenije.

Aplikacija ima na voljo približno 700 avtobusnih postajališč in okoli 250 železniških postaj.

Z aplikacijo Potnik si lahko olajšamo potovanja po Sloveniji. Uporabljamo jo lahko v glavnem mestu Ljubljana, da dobimo zelo natančne podatke o prihodih mestnih avtobusov na postaje. Aplikacija nam namreč do minute natančno prikaže naslednje tri prihode avtobusov na določeni postaji in progi.

Aplikacija je primerna tudi za turiste, saj je mogoča nastavitev angleškega jezika. Turisti pogosto ne vedo, kje se nahajajo avtobusne postaje, zato aplikacija z uporabo GPS omogoča, da najdemo najbližje postaje. Ob izbiri postaje lahko izberemo tudi navodila za pot do postaje. Navodila so možna v grafični in pisni obliki.

Aplikacija nam olajša tudi potovanje z vlaki. Spet imamo možnost izbire postaje ročno ali z GPS načinom. Izberemo vstopno in izstopno postajo, nato pa nam aplikacija pokaže vozni red vlakov in če pride do zamude, nas opozori tudi o tem. Aplikacija nam omogoča tudi vpogled v ceno vozovnice.

Za lažjo uporabo aplikacijo je mogoče nastaviti priljubljene postaje. Nastavimo lahko do tri priljubljene postaje, prek katerih hitreje dostopamo do želenih informacij. Ker imajo nekatere avtobusne in železniške postaje enako ime, je zraven imena postaje tudi grafična ikona, s katero lahko razločimo, kateri vrsti potniškega prevoza pripada postaja.

Ker je aplikacija v fazi razvoja je pomembno, da prihaja do rednih posodobitev. Pomembno je tudi, da so posodobitve kar se da enostavne. Aplikacija omogoča enostavno posodobitev v enem koraku. Mehanizem za posodobitev preveri različico nameščene aplikacije na sistemu

in zadnjo različico na strežniku. Če se ti različici razlikujeta, aplikacija naloži različico iz strežnika in zažene čarovnika za namestitev nove različice.

5.1 Mobilne aplikacije

5.1.1 Razvoj mobilnih aplikacij

Razvoj mobilnih aplikacij je proces razvoja programskih aplikacij za ročne mobilne naprave, kot so mobiteli, dlančniki in POS terminali. Mobilne aplikacije navadno namesti izdelovalec naprave, mobilni operaterji, ali pa so prenesene prek interneta iz različnih virov. Ti viri so lahko internetne strani avtorja programa ali pa programi, ki vsebujejo spisek aplikacij, ki so na voljo za prenos.

Pri razvoju mobilnih aplikacij je treba poskrbeti za zgled aplikacije. Mobilne naprave imajo manjši zaslon kot namizni računalniki, zato ne moremo prikazati informacij na isti način kot pri aplikacijah za namizne računalnike. Treba je najti način, kako prikazati vse potrebne informacije in tudi poskrbeti, da so te informacije prikazane na razviden in enostaven način. Pri razvoju uporabniškega vmesnika si pomagamo z enostavnimi grafičnimi ikonami in kratkim tekstom, tako da uporabnik v trenutku razume, kam ga pelje zahtevana akcija. Poleg enostavne predstavitve informacij je treba tudi poskrbeti, da je program tudi vizualno privlačen.

Z mobilno napravo običajno komuniciramo z zaslonom na dotik. Ti zasloni so manjših dimenzij, zato mora biti aplikacija zasnovana tako, da lahko vsak uporabnik z njo tekoče upravlja. Če naredimo premajhne gumbe in besedila in jih namečemo skupaj, se bo uporabnik velikokrat zmotil in pritisnil napačen gumb. Če bodo narazen, bomo zasedli preveč prostora na ekranu. Gradniki morajo biti previdno porazdeljeni po zaslonu, da zagotovijo optimalno uporabnost in vizualnost aplikacije.

Manjše specifikacije naprav nas prisilijo v izdelavo aplikacij, ki porabijo manj sistemskih sredstev. Zato sem se pri pisanju mobilne aplikacijo zelo trudil, da sem pisal optimalno kodo.

5.2 Uporabniški vmesnik in gradniki

Pri grajenju aplikacije sem sledil standardom, ki jih priporoča HTC. Glavno pravilo teh standardov je, da je možno aplikacijo uporabljati s prsti brez pripomočkov kot so (stylusi??). Ko odpremo aplikacijo, imamo na voljo tri opcije pa tudi dodatne opcije prek kontekstnega

menija. V aplikacijo sem vključil čim več enostavnih grafičnih podob, da bi bile opcije kar se da vidne in bi si lažje zapomnili vrstni red.

Slika 6. Osnovna stran aplikacije potnik.

Aplikacija ne podpira le bele barvne sheme, temveč je podprta tudi s črno shemo operacijskega sistema. Slika 7 prikazuje belo barvno shemo, kateri sem se trudil v aplikaciji približati.

Slika 7. Bela barvna tema za Windows Mobile 6.5.

Pri gradnji grafičnega vmesnika sem naletel na precej težav. Windows Mobile 6 ima slabo podporo gradnikov. Gradniki ki jih podpira so osnovni, ne podpirajo niti transparentnosti. Zaradi pomanjkljivosti v ogrodju, sem moral razviti mnogo gradnikov sam. To sem naredil

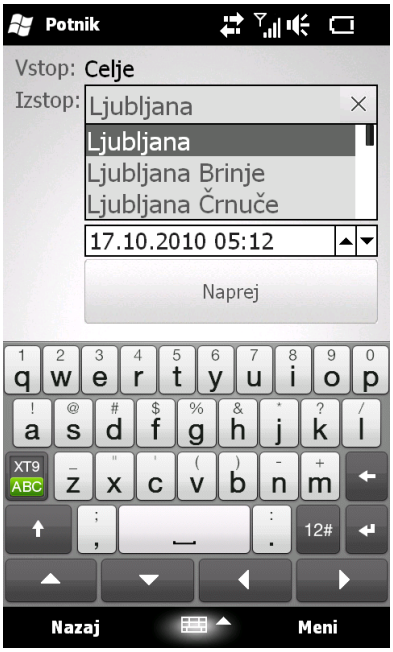
tako, da sem povezal več osnovnih gradnikov in jim spremenil lastnosti, da se obnašajo kot naprednejši gradniki.

Za potrebe aplikacije sem moral narediti tri gradnike, ki poenostavijo uporabo in polepšajo zgled aplikacije. Glavni gradniki so: samodejno dopolnjevanje besedila in izbora opcij, gumb s transparentnim besedilom in sistem za avtomatično posodobitev aplikacije ter druge manjše spremembe osnovnih gradnikov iz .NET Compact Framework 3.5.

5.2.1 Samodejno dopolnjevanje besedila in izbora opcij

Gradnik za samodejno dopolnjevanje besedila in izbora opcij nam omogoča, da med pisanjem vstopne oz. izstopne postaje izberemo postajo še preden dokončamo ime postaje. Kontrola samodejno preverja če obstaja zadetek za vpisano besedno s šumniki. Če denimo napišemo »Crnuce«, bo kontrola samodejno prepoznala, da obstaja postaja z imenom »Črnuče« in nam jo ponudila v izbor. Znak »X« nam pobriše celotno vpisano besedilo, če se odločimo napisati čisto drugo vstopno postajo. Gradnik ni transparenten, ker bi to onemogočilo vidnost opcij.

Gradnik je sestavljen iz osnovnega gradnika za vnos besedila, gradnika za prikaz seznama in gumba za izbris besedila.



Slika 8. Gradnik za samodejno dopolnjevanje besedila in izbora opcij.

5.2.2 Gradniki transparentnim besedilom

Ogrodje .NET Compact Framework 3.5 ne vsebuje gradnikov, ki bi vsebovali transparentno barvo. Brez takšnih gradnikov bi imela aplikacija zelo osnoven zgled, podoben zgledu aplikacij iz operacijskega sistema Windows XP oz. Windows 98. S povečanjem strojnih zmogljivosti naprav, se je povišal tudi standard uporabniških vmesnikov. Transparentnost gradnikov polepša zgled aplikacije.

Slika 9. Podrobni pregled aktivne LPP proge.

Na gornji sliki (Slika 9) so vidni transparentni gradniki. Gumb “Zapri” je eden izmed takšnih gradnikov, saj je ozadje gumba večbarvna slika, besedilo pa je prilepljeno na gumb, na spodnji (Slika 10) je primer netransparetnega gumba. Gumb s transparentnim besedilom ni privzet v ogrodju. Zaradi teh omejitev sem naredil nov gradnik gumb, kateremu se lahko dodeli slika ozadja, besedilo pa se izriše na sliko, medtem ko se gradnik izrisuje na zaslon. Tako mi je uspelo narediti izgled modernega gumba, ki je uporabljen v današnjih operacijskih sistemih in aplikacijah. Naslednji transparentni gradnik je enostavno besedilo. Gradnik sem simuliral tako, da sem uporabil gradnik za besedilo iz ogrodja. Z gradnikom na formi sem dobil njegove koordinate in ob izrisu gradnikov izrisal še besedilo na te te koordinate, na podoben način kot transparenten gumb.

Pri programiranju lastnih gradnikov sem že imel precej izkušenj. Pri razvoju namiznih aplikacij sem naredil veliko gradnikov in to znanje uporabil pri diplomskem delu.

Gradnike sem zasnoval tako, da jih lahko uporabim tudi v prihodnjih projektih.

Slika 10. Primer netransparetnih gradnikov (gumb in dialog)

5.2.3 Sistem za avtomatično posodabljanje aplikacije

Med razvojem aplikacije sem porabil veliko časa za testiranje aplikacije. Pri testiranju so mi pomagali prijatelji, s pametnimi telefoni, ki strojno ustrezajo zahtevam aplikacije. Težave so nastale pri nalaganju nove različice na mobilne naprave testnih uporabnikov. Rešitev sem našel v avtomatičnem sistemu za posodobitev programa. Tehnična izvedba tega sistema je bila izjemno zahtevna. Treba je bilo razviti način, kako odstraniti trenutno različico programa in brez vednosti uporabnika namestiti novo različico. Pri iskanju rešitve sem ugotovil, da moram narediti dodaten program, ki bo poskrbel za namestitev nove različice. Zato sem naredil nov program z imenom PotnikSetup.

5.2.3.1 Projekt PotnikSetup

Projekt PotnikSetup je zadolžen za tiho namestitev nove različice programa Potnik. Namestitev nove različice aplikacije Potnik ni možna, če ne prekinemo delovanja aplikacije in poženemo čarovnika za namestitev nove različice. Zato sem razvil PotnikSetup, ki deluje tako, da preveri različico nameščene aplikacije na mobilni napravi z različico aplikacije, ki je nameščena na spletnem strežniku. Če se različici razlikujeta, PotnikSetup začne s prenosom

namestitvenih datotek z novo različico aplikacije. Ko je namestitvena datoteka prenesena na mobilno napravo, se začne postopek namestitve, ki deluje tako, da najde identifikacijsko številko procesa Potnik in prekine delovanje tega procesa, tako da mu pošlje ukaz 'KILL'. Ko aplikacija Potnik ne teče več, lahko naložimo novo različico aplikacije. Ko proces ni več v teku, so datoteke Potnika sproščene in PotnikSetup lahko pobriše obstoječe datoteke aplikacije. Ko so datoteke pobrisane, PotnikSetup zažene namestitev nove različice. Po končani namestitvi, je na voljo nova različica aplikacije Potnik.

6 Sklepne ugotovitve

V diplomski nalogi je bil cilj narediti aplikacijo, s katero bi lahko na enostaven način dostopal do voznega reda javnih prevoznih sredstev. Ker sem lastnik pametnega telefona z operacijskim sistemom Windows Mobile, sem se odločil, da bom ustvaril aplikacijo za ta operacijski sistem. V fazi razvoja sem moral odločiti, kakšno arhitekturo naj ima aplikacija in ali naj ima strežniški del ali ne, po daljšem premisleku in poizvedovanju pa sem se odločil za arhitekturo, ki vključuje strežniški del - tako sem si namreč olajšal delo pri posodobitvah, vse komponente so bile na enem mestu, hkrati pa sem želel slediti svojim smernicam, da aplikacija na mobilni napravi skrbi le za prikaz podatkov.

Ko sem zaključil s programiranjem in testiranjem strežniškega dela, sem se lotil razvoja mobilne aplikacije. Na začetku pisanja diplomske naloge še nisem poznal prav dobro mobilnega programiranja in sem se moral veliko naučiti. Med programiranjem mobilne aplikacije sem se tudi soočil s problemi malce zastarelega ogrodja, ki ni vsebovalo modernih grafičnih gradnikov, zato sem veliko gradnikov moral narediti sam, saj sem želel, da ima aplikacija simpatičen in moderen videz.

Zaradi dobre zasnove na strežniškem delu nato nisem imel težav pri prikazu informacij v aplikaciji, veliko težav pa mi je povzročal uporabniški vmesnik, ki ga je bilo potrebno zasnovati tako, da se lahko aplikacija uporablja samo s prsti.

Prvi uporabniki, ki so testirali aplikacijo, so imeli veliko težav pri posodobitvah aplikacije, saj so te prihajale vsakodnevno, zato sem naredil avtomatično posodobitev aplikacije, ki prek spleta najde novo različico aplikacije in jo naloži na mobilno napravo. Ta proces je uporabniku zelo prijazen, saj ne zahteva nobene interakcije.

Namen je bil izdelati enostavno in privlačno aplikacijo, s katero uporabniku olajšamo vsakodnevni prevoz z javnim prevozom. Menim, da je bil namen diplomske naloge dosežen, prav tako sem se med izdelavo aplikacije naučil veliko novih stvari. Za svoj izdelek sem želel dobiti tudi strokovno potrditev in sem se tudi prijavil na tekmovanje v mobilnih aplikacijah, na katerem sem s svojo aplikacijo tudi zmagal.

Slike

Literatura

[1] *(2011) Windows Communication Foundation (WCF) – MSDN – Microsoft. Dostopno na: http://msdn.microsoft.com/en-us/library/ms735119(v=vs.90).aspx*

[2] *(2011) Microsoft SQL Server Library – MSDN – Microsoft. Dostopno na: http://msdn.microsoft.com/en-us/library/bb545450.aspx*

[3] *(2011) LINQ (Language-Integrated Query) - MSDN – Microsoft. Dostopno na: http://msdn.microsoft.com/en-us/library/bb397926.aspx*

[4] *(2011) The ADO.NET Entity Framework Overview – MSDN - Microsoft. Dostopno na: http://msdn.microsoft.com/en-us/library/aa697427(v=vs.80).aspx*

[5] *(2011) Regular Expressions Reference – Basic Syntax. Dostopno na: http://www.regular-expressions.info/reference.html*

[6] *(2011) Using a Three-Tier Architecture Model (COM+). Dostopno na: http:// msdn.microsoft.com/en-us/library/ms685068(v=vs.85).aspx*