Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Uroš Jenko

# Medplatformski razvoj mobilne aplikacije s podporo v oblaku

diplomsko delo na študijskem programu
računalništva in informatike

Mentor: doc. dr. Peter Peer
Asistent: as. Jernej Bule

Ljubljana, 2013

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

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: 01937/2013
Datum: 02.09.2013

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

Kandidat: **UROŠ JENKO**

Naslov: **MEDPLATFORMSKI RAZVOJ MOBILNE APLIKACIJE S PODPORO V OBLAKU**

**CROSS-PLATFORM DEVELOPMENT OF MOBILE APPLICATION WITH CLOUD SUPPORT**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Pisanje potnih nalogov je večini zaposlenih predvsem zoprno in zamudno delo, vendar je potni nalog in njegova pripadajoča dokumentacija zelo pomemben dokument. Zelo pomembno je, da se potne naloge pravočasno in pravilno sestavlja. V okviru diplome preglejte področje obstoječega stanja spletnih in mobilnih aplikacij, ki omogočajo upravljanje s potnimi nalogi. Preglejte tudi vsa obstoječa orodja za medplatformski razvoj mobilnih aplikacij in z najbolj primernim razvijte mobilno aplikacijo Potni nalogi s podporo spletne aplikacije, ki teče v oblaku. Aplikacija naj bo razvita za oba najbolj popularna mobilna operacijska sistema Android in iOS. V mobilni aplikaciji mora biti omogočeno sprotno kreiranje potnih nalogov, ki so sinhronizirani z bazo in aplikacijo v oblaku. V spletni aplikaciji mora biti poleg upravljanja potnih nalogov omogočena tudi vsa administracija.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Uroš Jenko, z vpisno številko **63070036**, sem avtor diplomskega dela z naslovom:

*Medplatformski razvoj mobilne aplikacije s podporo v oblaku*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Petra Peera in as. Jerneja Buleta,
- 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, 24. septembra 2013 Podpis avtorja:

*Rad bi se zahvalil svojemu mentorju doc. dr. Petru Peer in as. Jerneju Buletu za strokovno svetovanje, potrpežljivost in spodbudo pri nastajanju diplomskega dela.*

*Hvala tudi tebi Ema, ki me sprejemaš takšnega kot sem. V vseh mojih vzponih in padcih si verjela vame, me optimistično spodbujala ter mi nesebično pomagala.*

*Iskrena hvala moji družini za vso podporo in finančno pomoč pri študiju.*

*Hvala vsem ostalim, ki ste mi vsa ta leta stali ob strani.*

Svoji dragi Emi.

# Kazalo

# Povzetek

Uporabniki od aplikacije zahtevajo, da je dostopna kjerkoli in kadarkoli, kar jim omogoča rešitev v oblaku. Rešitev v oblaku je dostopna preko spletnega brskalnika ali spletne storitve in ni omejena z operacijskim sistemom uporabnika. Neodvisnost od mobilnega operacijskega sistema se želi doseči tudi pri izdelavi mobilnih aplikacij. Cilj medplatformskega razvoja je razviti največ skupne kode za različne platforme, medtem ko se za vsako platformo ločeno zagotavlja avtohton izgled in edinstveno izkušnjo. V diplomskem delu so opisane arhitekturne možnosti mobilnih aplikacij, mobilnih storitev v oblaku in aplikacij v oblaku. Podan je pregled ogrodij za medplatformski razvoj in opis nekaj najbolj uporabljenih. Cilj diplomskega dela je bila izdelava medplatformske mobilne aplikacije za vnašanje potnih nalogov s podporo mobilne storitve v oblaku in aplikacije v oblaku. Za medplatformski mobilni razvoj se je uporabilo ogrodje Titanium, za mobilno storitev ter aplikacijo v oblaku pa ogrodje Visual Studio. Rešitev upošteva zadnje smernice arhitekture in tehnologije s strani Microsofta, Appceleratorja, Androida in Appla.

## Ključne besede

rešitev v oblaku, neodvisnost od mobilnega operacijskega sistema, medplatformskega razvoja, avtohton izgled, arhitekturne možnosti, mobilnih aplikacij, mobilnih storitev v oblaku, aplikacij v oblaku, ogrodij za medplatformski razvoj

# Abstract

Users demand applications to be accessed anytime and anywhere, this is what offers cloud solutions. The solution in the cloud can be accessed via a web browser or web services and is not limited by the user's operating system. Independence of the mobile operating system is also goal in the development of mobile applications. The objective of cross platform development is to develop most common code for different platforms while providing native appearance and unique experience for each platform separately. In diploma is described the architectural options of mobile applications, mobile cloud services and cloud applications, review of frameworks for cross platform development and described some of the most used. The goal of the diploma was to develop mobile application to enter expense reports with support mobile cloud services and application in the cloud. The cross platform mobile application was developed with Titanium framework. Service and application in the cloud was developed with Visual Studio framework. The solution is based on the latest architectures and technology guidelines from Microsoft, Appcelerator, Android and Apple.

## Keywords

cloud solutions, independance of the mobile operating system, cross platform development, native appearance, architectural options, mobile applications, mobile cloud services, cloud applications, frameworks for cross platform development

# Poglavje 1

# Uvod

Hitri razvoj spleta [17] in spletnega programiranja je povzročil veliko razširjenost aplikacij v oblaku. Sodobni uporabnik spleta le tega ne uporablja zgolj za pregledovanje spletnih strani, temveč uporablja tudi veliko število aplikacij v oblaku. Razlog za to je, da so aplikacije v oblaku dostopne kjerkoli in kadarkoli ter so neodvisne od operacijskega sistema. Aplikacija je dostopna preko spletnega brskalnika in ne potrebuje namestitve, poleg tega so uporabniški podatki varni pred okvarami uporabniške strojne opreme.

Sodobni uporabniki spleta so postali zelo mobilni, saj ima večina pametne mobilne naprave z možnostjo internetne povezave ne glede na lokacijo, kar je povzročilo napredek pri razvoju mobilnih aplikacij in ponudbo mobilnih storitev v oblaku. Namen medplatformskega razvoja mobilnih aplikacij je odpraviti nekompatibilnost med mobilnimi operacijskimi sistemi.

V diplomskem delu so opisane arhitekturne možnosti mobilne aplikacije in aplikacije v oblaku, pregled ogrodij za medplatformski razvoj in opis nekaj najbolj uporabljenih. Cilj diplomskega dela je bila izdelava medplatformske mobilne aplikacije za vnašanje potnih nalogov s podporo mobilne storitve v oblaku in aplikacije v oblaku.

Drugo poglavje opisuje možne arhitekture za mobilne aplikacije in aplikacije v oblaku, pri tem so podane prednosti in slabosti ter primernost uporabe. Opredeljen je medplatformski razvoj in opisane so specifične značilnosti An-

droid in iOS platforme. Podrobno so opisani nivoji arhitekture v oblaku in mobilna storitev v oblaku. Na koncu poglavja je obrazložena izbira arhitekture za rešitev Potni nalogi.

Tretje poglavje podrobneje opisuje medplatformski razvoj (arhitekturo, komponente in možne rešitve ogrodij za medplatformski razvoj). Poglavje se zaključi z obrazložitvijo izbire ogrodja za razvoj mobilne aplikacije Potni nalogi.

Četrto poglavje opisuje okolje Titanium studio. Opisan je postopek namestitve iOS in Android podpore v okolje ter pregled sestave Titanium ogrodja in njegove podpore mobilnim storitvam v oblaku MBaaS (ang. mobile backend as a service).

V petem poglavju je pregled obstoječih rešitev za upravljanje s potnimi nalogi. Predstavljeni sta dve spletni in tri mobilne rešitve.

Šesto poglavje opisuje razvoj rešitve Potni nalogi. Podroben opis spletne aplikacije vsebuje opis arhitekture (večuporabniška in 3-slojna arhitektura), implementacije (uporabljene tehnologije in podroben opis ASP.NET MVC 4), strukture rešitve in navodila za namestitev. Poglavje se nadaljuje z opisom funkcionalnosti spletne aplikacije, opisom mobilne storitve v oblaku in opisom mobilne aplikacije ter njene funkcionalnosti. Poglavje se zaključi s primerom uporabe celotne rešitve in primerjavo rešitve z obstoječimi možnostmi za upravljanje potnih nalogov.

Sedmo poglavje vsebuje sklepne ugotovitve in zaključek.

# Poglavje 2

# Možnosti arhitekture

Za razvoj mobilne aplikacije in rešitve v oblaku je na voljo več arhitekturnih rešitev. Mobilna aplikacija je lahko avtohtona, spletna ali hibridna, medtem ko računalništvo v oblaku vsebuje več nivojev.

## 2.1 Vrste mobilnih rešitev

### 2.1.1 Izdelava avtohtonih mobilnih aplikacij

Avtohtona mobilna aplikacija [46] je aplikacija, ki je napisana v določenem programskem jeziku. Za Android je določen programski jezik Java, medtem ko se pri iOS uporablja Objective-C.

#### Prednosti

Avtohtone aplikacije [46], [9] zagotavljajo visoko zmogljivost in zanesljivost ter imajo dostop do funkcionalnosti naprave npr. kamera, imenik, GPS, kompas itn. Avtohtone aplikacije imajo polni dostop do bogatih in fleksibilnih avtohtonih uporabniških vmesnikov. Uporabniki so navajeni in navezani na svoje naprave in pričakujejo izkušnjo, ki jim jo ponuja avtohtona aplikacija. Nekatere avtohtone aplikacije je možno uporabljati brez internetne povezave.

## Slabosti

Razvoj avtohtonih aplikacij [46], [9] je drag, ker je vezan na eno platformo. Za podproro več platform je potrebno razviti verzijo, ki deluje na specifični platformi. Za namestitev aplikacije je potrebno uporabljati interno aplikacijsko trgovino AppStore za iOS aplikacije in Google Play za Android aplikacije.

#### 2.1.1.1 Pregled Android platforme

Posebnosti pri Androidu [13], [1], [3], zahtevajo večjo pozornost. Poleg razlik v uporabniškem vmesniku, so razlike tudi v vmesnikih na nižjem nivoju. Ogrodje za medplatformski razvoj opravi velik delež k abstrakciji teh podrobnosti, vendar je pomembno, da se jih razvijalec zaveda.

#### Uporabniški vmesnik

Posebnost Androida [13] je, da je izgled uporabniškega vmesnika odvisen od naprave. Obstaja standardiziran uporabniški vmesnik, ki se imenuje Vanilla, vendar je njegova uporaba manj pogosta. Razlog temu je, da je Android [1] odprtokodni in s tem razširljiv. Številni proizvajalci, kot so Motorola, HTC, Samsung in drugi, prilagodijo uporabniške vmesnike.

#### Gumbi

Android naprave [13] imajo štiri gumbe:

- Nazaj, preusmeri uporabnika na predhodno aktivnost. Če predhodna aktivnost ne obstaja, prikaže uporabniku domačo stran.
- Meni, prikaže meni aktivnosti.
- Domov, prikaže domačo stran.
- Iskanje, vrne možnost iskanja.

Od naprave je odvisno, ali so gumbi fizični ali na dotik in kakšna je postavitev gumbov.

#### Velikost zaslona in gostota pik

Android naprave [13], [30] se močno razlikujejo v velikosti in gostoti zaslona. Android razdeli velikosti v štiri skupine: majhni, običajni, veliki in zelo veliki. Znotraj skupine pa se zasloni razlikujejo še v gostoti pik in razmerju med širino in višino zaslona. Zaradi velikega števila različnih zaslonov mora razvijalec aplikacije testirati na velikem številu različnih naprav.

#### Komponente aplikacije

Android aplikacije [13] so sestavljene iz:

- Aktivnosti.
- Storitev.
- Obvestil.

Aplikacija [13], [19], [1] je sestavljena iz ene ali več aktivnosti. Aktivnost predstavlja en pogled z uporabniškim vmesnikom. Skupina aktivnosti v aplikaciji predstavlja funkcionalnost aplikacije. Moč aktivnosti je, da lahko pokličejo ostale aktivnosti, katere so lahko razvili drugi. Če želimo na primer posneti fotografijo, pokličemo aktivnost aplikacije za kamero. Prav tako lahko naredimo aktivnosti, katero pokličejo drugi.

Storitve [13], [56] so aplikacije, ki tečejo dalj časa in brez interakcije z uporabniki.

Obvestila [13], [35] so obveščevalni objekti, ki se izmenjujejo med aktivnostmi. Obvestila omogočajo aplikaciji interakcijo med aktivnostmi, ki so na voljo na napravi. Pri tem ni potrebno vedeti, katere aplikacije so nameščene.

### 2.1.1.2 Pregled iOS platforme

iOS se razlikuje od ostalih mobilnih operacijskih sistemov po istem proizvajalcu programske in strojne opreme, prav zaradi tega je povezava med njima močnejša. Osnovni principi [13] iOS platforme so:

- Oblikovno in dosledno izdelan uporabniški vmesnik.
- Minimalistični zunanji izgled.
- Upoštevanje Applovih smernic vmesnika.
- Uporaba Cocoa Touch [25]. Cocoa Touch je abstrakcija iOS operacijskega sistema in je namenjena pomoči razvijalcem.
- Kontroliran distribucijski sistem aplikacij. Pred prihodom aplikacije v Applovo trgovino App Store, se aplikacija pošlje v pregled.

### Uporabniški vmesnik

iOS je skrbno načrtovan in oblikovan operacijski sistem. Kljub različnim napravam iOS uporablja skoraj identičen uporabniški vmesnik. Razvijalci lahko upoštevajo smernice Apple-a ali ustvarijo popolnoma edinstven uporabniški vmesnik. Pri poslovnih aplikacijah [13] je priporočeno uporabiti smernice Apple-a, medtem ko je pri igrah potrebno ustvariti svoj izgled.

### Minimalen zunanji izgled

iOS je poznan po svojem izgledu z enim gumbom. Torej je poleg gumba za vklop in uravnavanje glasnosti samo en gumb, kateri preusmeri na začetno okno. S tem morajo biti kontrole vključene v aplikacijo.

### Applove smernice

Apple [13], [50] priporoča, da se upošteva smernice, kar omogoča ustvariti kakovosten uporabniški vmesnik in uporabniško izkušnjo z aplikacijo. Spodaj je naštetih nekaj smernic:

- Prikaz je najpomembnejši.
- Usmerjenost naprave se lahko spremeni.
- Aplikacije se odzivajo na poteze in ne na klike.

- Ljudje upravljajo eno aplikacijo naenkrat.
- Nastavitve so na voljo v skupnem razdelku Nastavitve.
- Aplikacija ima eno okno.
- Ne obstajajo fizični gumbi.

### 2.1.2 Izdelava spletnih mobilnih aplikacij

Mobilne spletne aplikacije [9] so nameščene in tečejo na strežniku. Razvite so s HTML5 (ang. hyper text markup language), za katerega obstaja veliko različnih ogrodij in orodij za razvoj. Za spletne mobilne aplikacije sta značilna tudi omejen dostop do funkcionalnosti naprav in velika odvisnost od internetne povezave.

#### Prednosti

Prednost uporabe spletnega pristopa [9] za aplikacijo je, da obstoječe spletne aplikacije večinoma že delujejo na mobilnih napravah. Poleg tega je enostavno prilagoditi večino spletnih aplikacij za uporabniku prijaznejši uporabniški vmesnik za mobilno napravo. Učni proces učenja spletnega mobilnega razvoja je kratek, ker se lahko uporabi večino znanj iz klasičnega spletnega razvoja, kjer se uporablja HTML5 in CSS3 (ang. cascading style sheets).

Upravljanje spletne mobilne aplikacije je na enem mestu, kar je velika prednost pred razvojem avtohtonih mobilnih aplikacij. Ker je aplikacija v oblaku, se ni potrebno ukvarjati s strategijo nameščanja in z upravljanjem naprav. Poleg tega imamo zagotovljeno večplatformsko podporo. HTML5 zagotavlja zadovoljivo izkušnjo uporabnika in omogoča pridobitev lokacije in shranjevanje podatkov, ki omogoča delovanje aplikacije brez internetne povezave.

### Slabosti

Spletni razvoj mobilnih aplikacij [9], [45] ima kljub izredno hitremu napredku veliko slabosti v primerjavi z avtohtonimi aplikacijami.

Brez dostopa do funkcionalnosti naprave uporabniku ne moremo zagotoviti popolne izkušnje. Če potrebujemo od naprave več kot samo lokacijo, moramo uporabiti avtohton ali hibridni pristop. Čeprav HTML5 omogoča podporo aplikacijam brez povezave, je namenjen predvsem zmanjšanju porabe sredstev, tako da je odvisnost od internetne povezave zelo težko odpraviti. Kljub temu da je možno s HTML5 in CSS-om ustvariti bogate uporabniške komponente, je uporabnik navajen na avtohton izgled aplikacije.

V primerjavi z avtohtonimi aplikacijami je pri spletnih mobilnih aplikacijah težje pridobiti denar od uporabnikov, saj so aplikacije odvisne zgolj od oglaševanja in uporabniške naročnine. Prav tako uporabniki težje najdejo spletne aplikacije, ker niso v trgovini z aplikacijami.

## 2.1.3 Izdelava hibridnih mobilnih aplikacij

Hibridna aplikacija [9], [11] je tista, ki uporablja avtohtone in spletne mobilne pristope. Pristop je uporaben, ko značilnosti aplikacije zahtevajo in omogočajo vmesni pristop. Pogosto se uporablja izraz, da hibridna aplikacija ponuja najboljše iz obeh svetov, vendar vsebujejo tudi nekaj slabosti obeh svetov.

### Prednosti

S hibridnim pristopom ponujamo aplikacijo, ki uporablja spletno tehnologijo in avtohtone komponente. Iz spletnega mobilnega programiranja se pogosto privzame centralni pristop. Na centralnem strežniku zagotavljamo del aplikacije, do katere dostopamo s spletno tehnologijo. Za izgradnjo izgleda in dostopa do funkcionalnosti naprav je v uporabi avtohton pristop.

### Slabosti

Hibridne aplikacije je potrebno namestiti na napravo, kar poveča stroške vzdrževanja. Zaradi uporabe avtohtonega in spletnega pristopa, se lahko zmanjša zmogljivost in se ne more primerjati z izkušnjo z avtohtono aplikacijo.

## 2.1.4 Izdelava medplatformskih mobilnih aplikacij

Z medplatformskim razvojem se lahko razvijejo avtohtone in hibridne aplikacije. Cilj medplatformskega razvoja [9] je razviti največ skupne kode za različne platforme, medtem ko se za vsako platformo ločeno zagotavlja avtohton izgled in edinstveno izkušnjo. Ker je vsaka platforma edinstvena, je izdelava skupne kode velik izziv. Medplatformski razvoj abstrahira podrobnosti posamezne platforme, da lahko zagotavlja skupne vmesnike. Za razvoj dobre aplikacije mora razvijalec razumeti značilnosti in komponente posamezne platforme.

### Prednosti

Prednosti medplatformskega razvoja:

- Ponovna uporaba kode: razvijalec lahko uporabi isto kodo za različne projekte in platforme.
- Dodatki: večina ogrodij vsebuje dodatke, ki integrirajo storitve in orodja.
- Enostavost za spletne razvijalce: večina ogrodij uporablja skriptni programski jezik ali HTML označevalni jezik.
- Zmanjšana cena razvoja: za razvoj aplikacij ni potrebno poznati vseh podrobnosti specifične platforme.
- Enostavna namestitev: večina ogrodij omogoča neposredno namestitev.

- Podpora oblačni arhitekturi: večina ogrodij vsebuje dodatke, kateri omogočajo neposredno integracijo s storitvami v oblaku.

### Slabosti

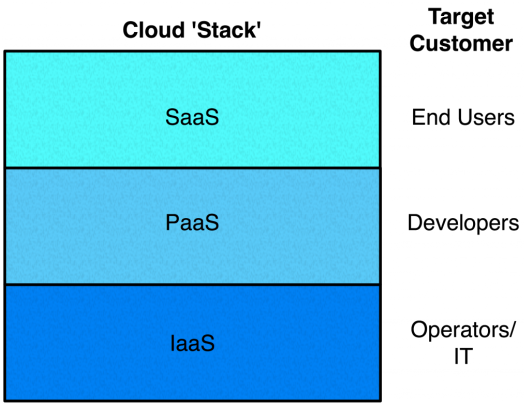
Slabosti medplatformskega razvoja:

- Ogrodja ne podpirajo vseh funcionalnosti: v primeru nove funkcionalosti je potrebno ogrodje posodobiti.
- Ogrodja ne podpirajo obstoječih razvojnih orodij: razvijalec se mora naučiti novih razvojnih orodij, z izjemo PhoneGapa, ki se ga lahko integrira v obstoječa okolja (Xcode, Eclipse, Visual studio).
- Aplikacija je počasnejša: medplatformski proces prevoda aplikacije je včasih počasnejši od avtohtonih orodij in klicev aplikacije.
- Podpora naprednejši grafiki in 3D grafiki je omejena.
- Neskladna koda med ponudniki medplatformskih ogrodij: večina ogrodij vsebuje lastne vmesnike, kar onemogoča enostavnejši prehod med ogrodji.

## 2.2 Računalništvo v oblaku - arhitektura

Računalništvo v oblaku [14] omogoča, da se tehnologijo uporablja takrat, ko se jo potrebuje. Koncept računalništva v oblaku [24] je razdeljen na več nivojev, kar prikazuje slika 2.1. Nivoji se medseboj dopolnjujejo:

- Infrastruktura kot storitev (IaaS, ang. infrastructure as a service) je prvi nivo in pomeni, da ima uporabnik dostop do strojne opreme, ki ima internetno povezavo. Oprema je lahko fizična ali virtualna. Za namestitev in posodabljanje operacijskega sistema in aplikacij je zadolžen uporabnik. IaaS storitve ponuja več ponudnikov, kot so Amazon [20],




Slika 2.1: Nivoji računalništva v oblaku [24].

Memset [39], Google [32], Windows [55] in drugi. Če je strojna oprema v oblaku, ni potrebno skrbeti za [14]:

- načrtovanje kapacitet in ukrepanje v primeru pomanjkanja,
- skrbeti za redundanco v primeru težav,
- primer naravne katastrofe,
- kaj storiti s strojno opremo, ko se je ne potrebuje več in
- račun za elektriko.

- Platforma kot storitev (PaaS, ang. platform as a service) je drugi nivo in ponuja večjo podporo uporabniku, saj vključuje elemente, kot so: operacijski sistem, podatkovna baza in spletni strežnik. Največji prednosti sta, da se uporabnik osredotoči na svojo aplikacijo in da skrbnik oblaka upravlja vire samodejno. Večina ponudnikov IaaS ponuja tudi PaaS.

- Programska oprema kot storitev (SaaS, ang. software as a service) je zadnji nivo računalništva v oblaku in omogoča, da se uporabnik osredotoči na stranke, saj ni potrebno upravljati infrastrukture ter platforme, na kateri je nameščena aplikacija. Primer SaaS so elektronska pošta, spletni urejevalniki besedila in rešitev diplomskega dela Potni nalogi. Lastnosti SaaS-a [14], [18]:

- Dostopnost preko spletnega brskalnika.
- Dostopnost na zahtevo: pred uporabo ni potrebno iti skozi nakupovalni proces. Ko je omogočen dostop, se lahko dostopa kadarkoli in kjerkoli.
- Minimalne informacijske zahteve: SaaS zahteva minimalna informacijska znanja. Običajno je potrebno pravilno nastaviti konfiguracijo.

SaaS običajno omogoča večuporabniško arhitekturo, ki več strankam omogoča uporabo iste instance programske opreme, kar ima veliko prednosti za lastnika programske opreme v oblaku, kot sta npr.:

- manj strojne opreme,
- hitrejše in lažje posodabljanje.

Posredno imajo koristi tudi stranke, saj lahko lastniki zaradi nižjih stroškov ponudijo nižje cene.

### 2.2.1 Mobilne storitve v oblaku

Trg mobilnih aplikacij [63] se razvija zelo hitro. Pred nekaj leti le-ta še ni obstajal. Hiter razvoj in zahteve po prilagodljivosti so povzročile raznovrstno ponudbo naprav (pametni telefoni, tablični računalniki itn.), operacijskih sistemov (Android, iOS, Windows Phone 8, BlackBerry itn.) in tehnologij, ki omogočajo razvoj avtohtonih, spletnih in hibridnih aplikacij. Prva generacija mobilnih aplikacij ponuja relativno enostavne uporabniško usmerjene aplikacije, ki omogočajo mobilno podporo obstoječim spletnim aplikacijam in poslovnim podatkom. Naslednja generacija mobilnih aplikacij ponuja bogatejšo uporabniško izkušnjo. Aplikacije izkoriščajo funkcionalnosti mobilnih naprav (npr. fotografiranje in nalaganje slik), kontekst, katerega priskrbijo mobilne naprave (lokacija, zunanje storitve v oblaku itn.) in celovite programske poslovne rešitve. Podjetja želijo, da mobilne aplikacije naslednje

generacije uporabljajo njihove storitve, kar je ustvarilo novo vejo trga mobilnih aplikacij in sicer ponudbo mobilnih storitev v oblaku. Za razvoj mobilnih storitev v oblaku obstaja več pristopov:

- Samostojni razvoj in uporaba lastne infrastrukture: mobilno storitev razvije ponudnik sam in jo namesti na lastno infrastrukturo.
- Samostojni razvoj in uporaba infrastrukture v oblaku: mobilno storitev razvije ponudnik sam in jo objavi na infrastrukturi zunanjega ponudnika. Ta pristop uporablja mobilna rešitev diplome.
- MBaaS (ang. mobile backend as a service): vsebuje knjižnico pogosto uporabljenih storitev (upravljanje z uporabniki, obveščanje uporabnikov, integracija s socialnimi omrežji, kontrola dostopa itn.). Razvijalci uporabljajo knjižnico in dodatno razvijajo lastne storitve.

### 2.2.2 Primer javnega oblaka - Windows Azure

Windows Azure [36] je aplikacijska platforma Microsofta za javni oblak, ki podpira različne programske jezike, orodja in ogrodja. Primeri uporabe platforme:

- Aplikacija se nahaja v Azure oblaku, podatke shranjuje v Microsoftov podatkovni center.
- Azure se uporabi za shranjevanje podatkov aplikacije, ki se izvaja izven oblaka.
- Azure se uporabi za izdelavo virtualnih okolij za razvoj in testiranje.
- Itd.

## 2.3 Izbira arhitekture

Za izdelavo rešitve Potni nalogi je bil izbran medplatformski razvoj avtohtone aplikacije in aplikacijska platforma javnega oblaka Windows Azure. Medplat-

formski razvoj je bil izbran, ker omogoča razvoj skupne kode za Android in iOS operacijska sistema, ki pokrivata [41], [21] približno 90% trga. Avtohtona aplikacija je izbrana, ker uporabnik pričakuje avtohtoni izgled, ki ponuja največjo zmogljivost in jo je možno uporabljati brez internetne povezave.

Med različnimi ponudniki se je izbralo Windows Azure oblak, saj se je za razvoj spletne aplikacije uporabilo Microsoftova orodja za razvoj, ki imajo možnost neposredne povezave z Windows Azure oblakom. Spletna aplikacija Potni nalogi bo ponujena uporabnikom kot SaaS. Za mobilno storitev v oblaku se je uporabil samostojni razvoj in uporaba infrastrukture v oblaku, saj je cenejši kot uporaba lastne infrastrukture. Pristop z MBaaS ni bil izbran, ker je za uporabo potrebno plačevati naročnino in ker mobilna rešitev diplome ne zahteva pogosto uporabljenih storitev, kot je integracija s socialnimi omrežji, možnost ocenjevanja, nalaganje fotografij itn.

# Poglavje 3

# Medplatformski razvoj

Medplatformski razvoj mobilnih [9] aplikacij omogoča enotno skupno kodo, ki deluje na večih platformah. Z njim je možno razviti hibridne ali avtohtone aplikacije. Za medplatformski razvoj obstaja več ogrodij, katera uporabljajo različne programske jezike. Nekaj najpopularnejših [62], [65]:

- PhoneGap [47], kateri uporablja HTML, JavaScript in CSS.
- MoSync [43], kateri uporablja C/C++, JavaScript, HTML in CSS.
- Titanium [60], kateri uporablja HTML in JavaScript
- Rhodes [54], kateri uporablja HTML, JavaScript in Ruby

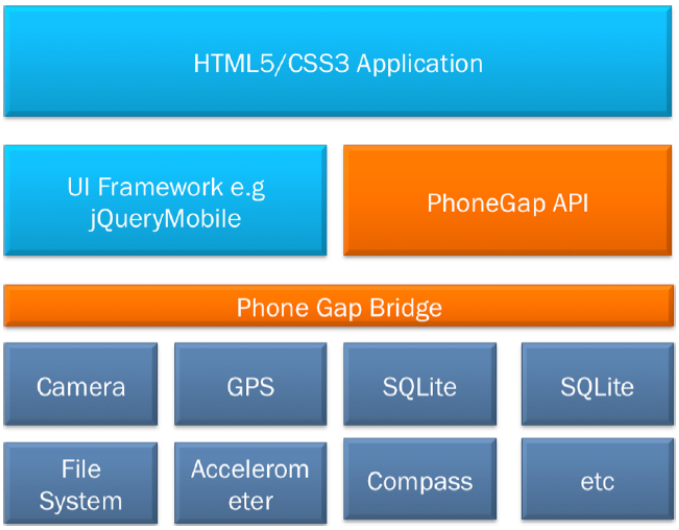
## 3.1 PhoneGap

PhoneGap [16], [6] je HTML5 aplikacijsko ogrodje, ki ga je kupilo podjetje Adobe Systems. Uporablja se za razvoj hibridnih mobilnih aplikacij preko spletnih tehnologij, kar pomeni, da razvijalec uporablja obstoječa znanja HTML-ja, CSS-a in JavaScripta-a iz spletnega programiranja. Razvite aplikacije niso popolnoma HTML/JavaScript niti niso popolnoma avtohtone. HTML/JavaScript del aplikacije vsebuje uporabniški vmesnik, aplikacijsko logiko in komunikacijo s strežnikom. Ostali deli aplikacije, ki komunicirajo

in upravljajo z napravo, so prevedeni v avtohtoni jezik naprave. PhoneGap zagotavlja JavaScript API (ang. application programming interface) povezavo med spletnimi tehnologijami in avtohtonim svetom ter mu omogoča upravljanje in dostop do naprave.

PhoneGap [48] zagotavlja podporo Androidu, BlackBerryu, iOS-u, Symbianu, WebOS-u, Windows Phone-u, Bada-u in Tizenu. PhoneGap je ogrodje in ne zagotavlja okolja za razvoj. Za Android je potrebno uporabljati Eclipse, za iOS Xcode in za ostale platforme pa ustrezno okolje s podporo.

### 3.1.1 Arhitektura ogrodja PhoneGap




Slika 3.1: Prikaz arhitekture PhoneGap-a [6].

PhoneGap arhitekturo določa JavaScript knjižnica, ki omogoča HTML, JavaScript aplikacijam dostop do funkcij naprave. Slika 3.1 prikazuje pregled PhoneGapove arhitekture. Zgornji del slike prikazuje tehnologije, ki jih uporablja razvijalec, spodnji del pa skupine funkcionalnosti naprave. Vmesni del služi kot povezava med JavaScript kodo razvijalca in funkcionalnostmi naprave. Kodo aplikacije zgrajene s PhoneGapom lahko razdelimo na dva dela:

- JavaScript poslovno logiko, katera upravlja uporabniški vmesnik in njegovo funkcionalnost.
- JavaScript del, ki dostopa in upravlja z napravo.

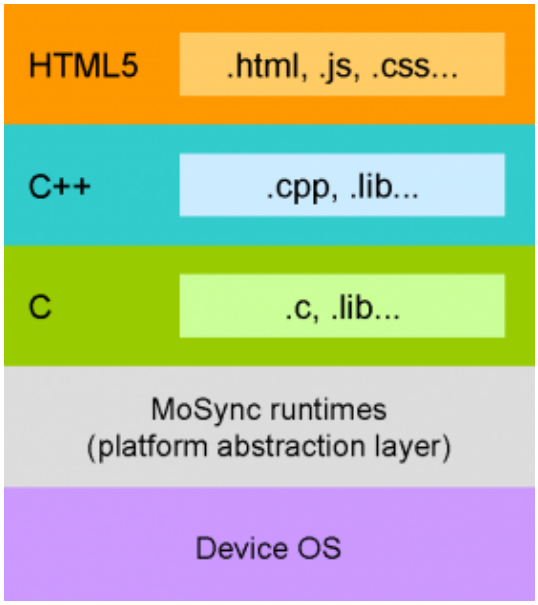
### 3.1.2 Prednosti in slabosti ogrodja PhoneGap

Osnovna prednost ogrodja PhoneGap [8] je, da omogoča uporabo dolgoletnih izkušenj programiranja z uporabo spletnih standardov. Kljub dobrim znanjem HTML-ja, CSS-a in JavaScript-a uporabnik naleti na težave pri uporabi specifičnih PhoneGap vmesnikov, ki jih mora osvojiti. PhoneGap je dober v zagotavljanju povezave med spletnimi standardi in funkcionalnostmi naprav. Omogoča lahek in hiter dostop do kamere, kontaktov, GPS-a itd. PhoneGap omogoča enostavno povezavo s spletnimi storitvami preko JavaScript jezika, vendar ne zagotavlja, da aplikacija deluje na vseh platformah, če sočasno deluje že na eni. Za prenos na druge platforme je potrebno testiranje, spreminjanje in optimiziranje. Poleg tega je za izdelavo aplikacije posameznih platform potrebno uporabiti različna okolja, kar pomeni veliko namestitev, nastavljanja in kopiranja.

## 3.2 MoSync

MoSync [42] je odprtokodno ogrodje za razvoj mobilnih aplikacij in je vgrajeno v okolje Eclipse. Ogrodje omogoča izdelavo avtohtonih aplikacij za več mobilnih platform z uporabo C/C++, JavaScript programskega jezika in HTML5 ter CSS. Mobilno aplikacijo je možno razviti samo z uporabo JavaScript programskega jezika, saj JavaScript vmesniki samodejno kličejo nižji C/C++ sloj. MoSync podpira Android, iOS, Windows Phone, Symbian, Java Mobile in Moblin.

MoSync ogrodje dostopa do avtohtonih delov uporabniškega vmesnika. Tehnologija Wormhole [53] povezuje JavaScript klice z nižjim C slojem. Slika 3.2 [29] prikazuje sloje MoSync arhitekture. Zgornji sloj predstavlja spletne




Slika 3.2: Prikaz arhitekture MoSync [29].

tehnologije s katerimi upravlja razvijalec. C/C++ sloj razvijalec lahko uporablja neposredno. MoSync zagotovi, da JavaScript vmesniki kličejo ustrezne dele C/C++ sloja. Spodnja sloja predstavljata napravo in sloj, ki skrbi za prevod v avtohtono aplikacijo. Ogrodje vsebuje veliko C funkcij, ki so sprogramirana na zelo nizkem nivoju in se imenujejo "syscalls". Wormhole programerju zakrije podrobnosti in lahko uporablja JavaScript vmesnike. MoSync je kompatibilen s PhoneGap funkcijami, kar omogoča enostavni uvoz PhoneGap aplikacije v MoSync.

## 3.3 Titanium

Titanium [13] je razvojna platforma podjetja Appcelerator, ki omogoča razvoj avtohtonih mobilnih aplikacij v programskem jeziku JavaScript. Platforma omogoča izdelavo, poganjanje in ustvarjanje paketa, ki ga lahko naložimo na mobilno napravo. Omogočen je razvoj za iOS, Android in BlackBerry platformo. Titanium mobilne aplikacije tečejo na samostojnem JavaScript pogonu, ki komunicira z avtohtonim API, tako so s Titaniumom razvite av-

tohtone mobilne aplikacije. Sestava Titaniuma:

- Titanium SDK: programska orodja [13] so sestavljena iz Python skript, ki sodelujejo z avtohtonim SDK (ang. software development kit). Titanium SDK povezuje JavaScript kodo razvijalca, JavaScript prevajalnik in statična sredstva v aplikacijsko binarno datoteko, ki jo lahko namestimo na napravo ali emulator. Postopek povezovanja je avtomatiziran in razvijalcu ni potrebno skrbeti zanj.

- Titanium Mobile APIs: Titanium aplikacijski programski vmesniki [13] so narejeni v programskem jeziku JavaScript in omogočajo dostop do avtohtonih komponent. Zaradi velikega števila komponent so vmesniki razdeljeni na več imenskih prostorov.

- Titanium Studio: Titanium Studio [13], [2] je integrirano razvijalno okolje. Uporablja se za izdelavo, testiranje in razhroščevanje Titanium mobilnih aplikacij. Titanium Studio vsebuje več predlog in primerov aplikacij, ki olajšajo delo. Okolje skrbi tudi za posodobitve Titanium SDK, analitiko in uporabo modulov.

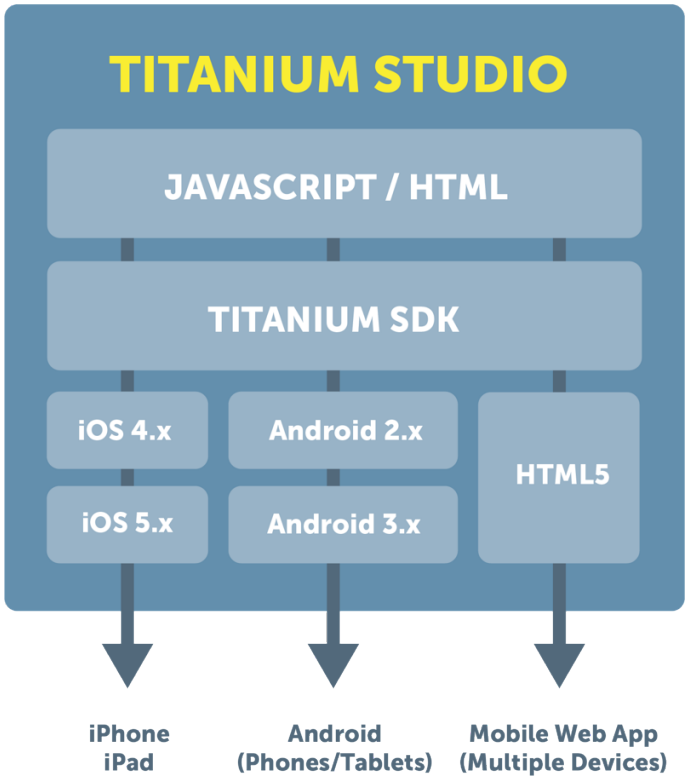
- Modules: Titanium [13] je zgrajen iz modulov, kar pomeni, da so vsi vmesniki moduli. Poleg osnovnih modulov se lahko dodaja module, ki se jih najde na spletu. Module posameznik lahko naredi sam in jih objavi.

- Appcelerator cloud services: Appcelerator [13] omogoča različne storitve vključno z analitiko, ki omogoča osnovne informacije, kako pogosto je uporabljena aplikacija in na kateri platformi. Možno je nastaviti tudi analitiko po meri, če uporabnika na primer zanima, kolikokrat je bil pritisnjen točno določeni gumb.

### 3.3.1 Arhitektura Titaniuma

Titanium [13] deluje kot povezava med avtohtonim operacijskim sistemom in kodo razvijalca. Slika 3.3 prikazuje to arhitekturo. Vrh slike prikazuje

uporabljeno tehnologijo razvijalca, dno uporabniške operacijske sisteme in sredina slike Titanium, kot povezavo med njimi.




Slika 3.3: Prikaz arhitekture Titaniuma [13].

Prednost Titaniuma [2] je razvoj aplikacije v programskem jeziku JavaScript in prevod v avtohtone aplikacije različnih platform. Razvijalec v kodi kliče Titanium vmesnike, od katerih zahteva na primer izris gumba, odpiranje okna, prikaz kamere itn. Del Titaniumovega ogrodja, ki se imenuje Kroll, prevede klic v avtohtoni ekvivalent in s tem je dosežen cilj Titaniuma, da aplikacije delujejo kot avtohtone.

Medplatformska skladnost je mogoča zaradi veliko podobnosti med platformo Android in iOS. Razlik ni mogoče medplatformsko podpreti in je potrebna ločena obravnava za vsako platformo. Titanium skrije veliko značilnosti platforme in razvijalcu ni potrebno poznati vseh podrobnosti.

Titaniuma ni priporočeno uporabljati, če razvijamo hibridno ali spletno

mobilno aplikacijo, kjer uporabljamo HTML5 in CSS in ne izkoristimo prednosti prevoda v avtohtono aplikacijo.

# Poglavje 4

# Razvoj s Titanium Studiom

Za razvoj mobilne aplikacije v diplomskem delu je izbrano ogrodje Titanium, ki v primerjavi z MoSync in PhoneGap vsebuje svoje okolje za razvoj, imenuje se Titanium Studio ter je najlažje in najhitreje namestljivo. PhoneGap je ogrodje, ki potrebuje okolje, ki podpira razvojno platformo. Za Android je to Eclipse in za iOS Xcode, kar pomeni veliko nepotrebnega kopiranja kode in učenje več razvojnih okolij. Pri MoSync je potrebno namestiti podporo v Eclipse. PhoneGap je namenjen izdelavi hibridnih aplikacij, medtem ko se s Titaniumum razvije hitrejše in uporabniku prijaznejše avtohtone aplikacije. Literatura podrobno opisuje ogrodji Titanium in PhoneGap, medtem ko je literatura za MoSync omejena na uporabniku neprijazno dokumentacijo in spletne članke. Iz naštetega je torej izbira Titanium ogrodja najprimernejša odločitev za medplatformski mobilni razvoj.

## 4.1 Pregled

### 4.1.1 Titanium Studio

Titanium Studio je razvojno okolje za razvoj mobilnih aplikacij. Omogoča upravljanje s projekti, zagon aplikacije na simulatorju ali napravi in pošiljanje aplikacije v App Store ali Android Market.

Prav tako okolje omogoča [2] avtomatično preverjanje sintakse, samodejno

dokončanje kode in razhroščevanje.

### 4.1.2 Titanium ogrodje

Titanium ogrodje skrbi za prevod JavaScript kode v nativni projekt. Poleg tega skrbi tudi za slike Android emulatorja in iPhone simulator. Zaradi razlik v specifikacijah Android naprav je priporočljivo imeti več različnih slik emulatorja.

Titanium ogrodje [22] je zelo obsežno in zato porazdeljeno po funkcionalnostih. Tako imenski prostor UI vsebuje komponente za izgradnjo uporabniškega vmesnika.

Nekaj ostalih pomembnejših imenskih prostorov:

- Network upravljanja s internetno povezavo.
- API zagotavlja možnost izpisovanja v konzolo.
- APP se uporablja za dostop do informacij aplikacije med izvajanjem in za poslušanje in proženje sistemskih dogodkov.
- Calendar za upravljanje s koledarjem.
- Filesystem za dostop do datotek in direktorijev sistema.
- Geolocation za dostop do informacij o lokaciji.
- Platform za dostop do podatkov in specifičnih funkcionalnosti naprav.
- Database za kreiranje in dostopanje do SQLite podatkovne baze.
- Modules so moduli zunanjih aplikacij kot so Facebook, Map, Newsstand, Nfc itd.
- itd.

Priporočena arhitektura mobilne aplikacije [28], [12] je modularna arhitektura s CommonJS moduli. CommonJS so [26], [27] neodvisni gradbeni

bloki, ki izničijo probleme z globalnim imenskim prostorom in imenskimi konflikti. CommonJS je projekt s ciljem vzpostaviti ogrodje za JavaScript izven spletnega brskalnika. Projekt se je začel januarja 2009 in je imel prvotno ime ServerJS.

#### 4.1.2.1 Titanium mobilne storitve v oblaku

Titanium ogrodje podpira pristop MBaaS za mobilne storitve v oblaku. Podjetje Accelerator imenuje pristop ACS (ang. Appcelerator cloud services). MBaaS vsebuje knjižnico obstoječih storitev [61]:

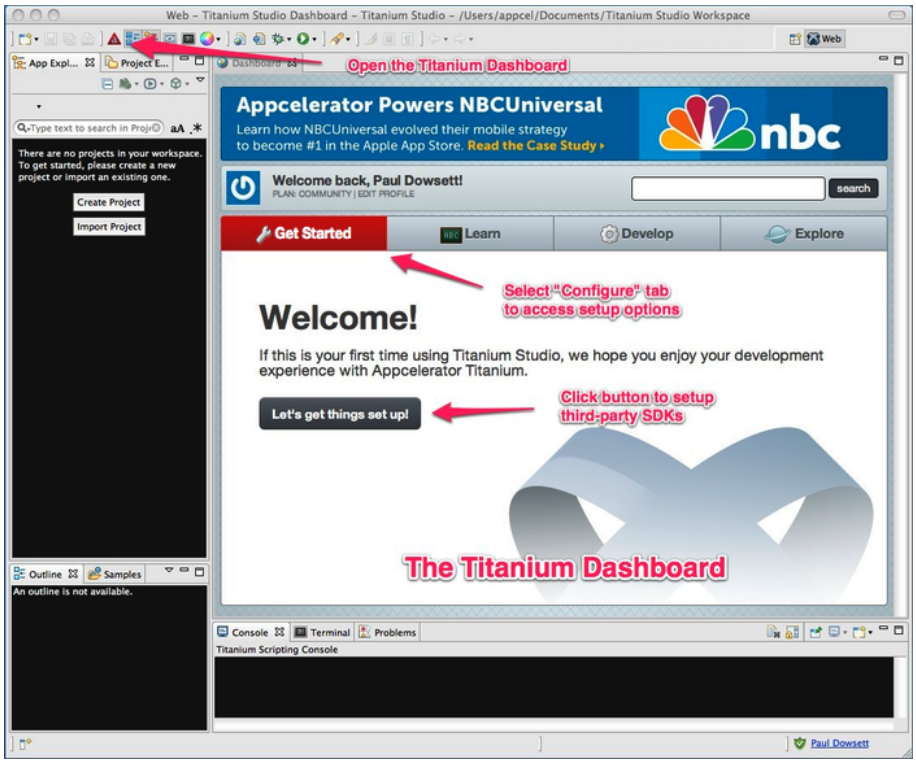
- Chats: storitev omogoča dostop do sporočil pogovora.
- Checkins: storitev omogoča izdelavo fotografij z oznakami.
- Clients: storitev omogoča poizvedbo lokacije odjemalca.
- Emails: storitev omogoča pošiljanje elektronske pošte preko ACS Email storitve.
- Events: storitev omogoča izdelavo, urejanje in poizvedovanje dogodkov.
- Files: storitev omogoča izdelavo in urejanje datotek.
- SocialIntegrations: storitev omogoča integracijo s socialnim omrežjem Facebook.
- Itd.

Razvijalci uporabljajo storitve iz knjižnice in s tem razširijo svojo aplikacijo na relativno enostaven način. Ustvarijo lahko svojo storitev in jo dodajo v knjižnico. Smiselno je dodati lastno storitev v knjižnico, ko se le ta neprestano uporablja skozi več projektov (npr. dostop do poslovnega sistema ali podatkovnih virov). Podjetja imajo možnost objave lastnih storitev v javnem ali zasebnem oblaku podjetja Appcelerator ali v zasebnem oblaku v zasebni lasti. Mobilne storitve v oblaku podjetja Appcelerator so brezplačne za razvijalce, vendar je potrebno skleniti naročnino v primeru poslovne uporabe.

## 4.2 Namestitev

Namestitev [13], [4] se razlikuje glede na operacijski sistem, vendar poteka brez posebnosti. Namestitveno datoteko se prenese iz spletne strani, pri tem se sledi korakom namestitve.

Po uspešni namestitvi se odpre Titanium studio, prikaže se nadzorna plošča, ki jo prikazuje slika 4.1 in preko katere se nastavi razvojno okolje.




Slika 4.1: Nadzorna plošča orodja Titanium Studio [57].

### 4.2.1 Minimalne zahteve

Sistem mora zagotavljati minimalne zahteve, da lahko razvijalec nemoteno razvija aplikacije. Osnovno pravilo [52], [4] je, da je 4 GB (ang. gigabyte) spomina dovolj za celotno Titanium ogrodje, vendar so minimalne zahteve nižje in se razlikujejo glede na operacijski sistem. Operacijski sistem Windows za

normalno delovanje zadnjega Android SDK potrebuje 1 GB spomina, medtem ko Linuxu in OS X-u zadošča 1,5 GB spomina. Za razvoj z Blackberry SDK pa je potrebno pri vseh operacijskih sistemih minimalno 4 GB spomina. Titanium podpira naslednje operacijske sisteme:

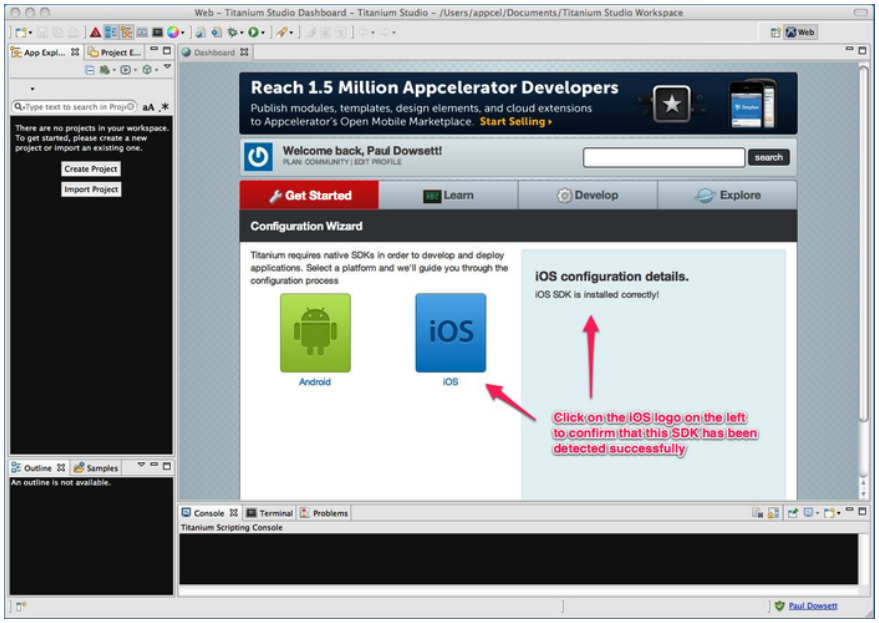
- Apple Mac OS X
  - 10.7 - Lion
  - 10.8 - Mountain Lion
- Windows
  - Windows 7
  - Windows 8
- Linux Desktop
  - Ubuntu 12.04 LTS - Precise Pangolin

Titanium razvojno okolje dodatno potrebuje Java Development Kit 6 podjetja Oracle in Node.js.

Android aplikacije se lahko razvijajo na Windows, Linux in OS X, medtem ko iOS aplikacije le na OS X operacijskem sistemu.

### 4.2.2 Namestitev iOS SDK

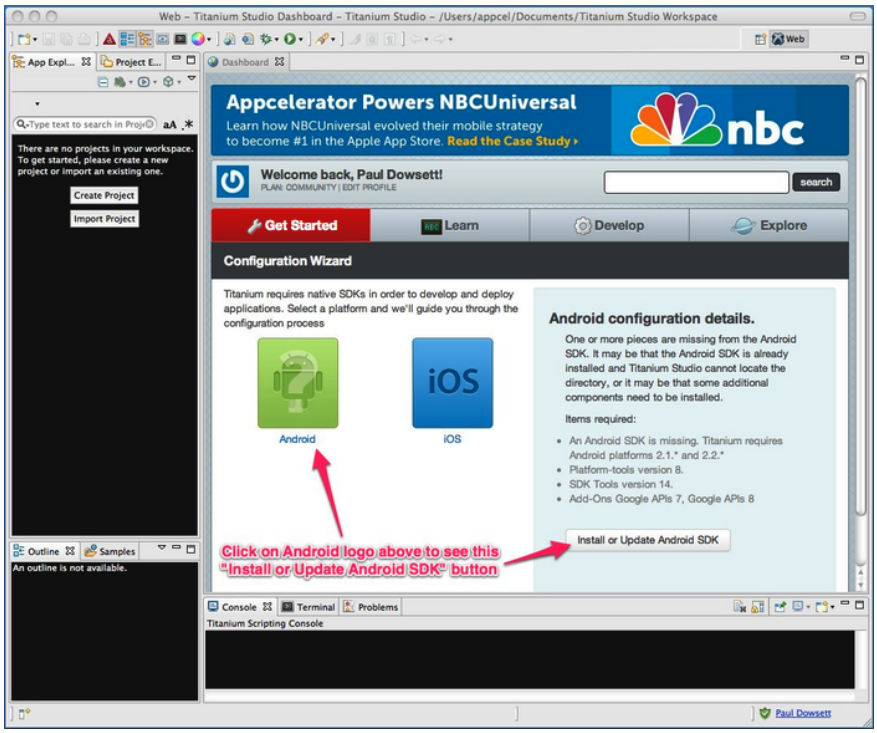
Če se želi razviti iOS aplikacije, mora biti nameščeno Xcode razvojno okolje in iOS SDK. V Titanium Studio [13], [4] nadzorni plošči se klikne iOS ikona, ki preusmeri na namestitveno stran predhodno omenjenih komponent. Po uspešni namestitvi se prikaže obvestilo o uspehu v Titanium Studio nadzorni plošči, ki jo prikazuje slika 4.2.




Slika 4.2: Namestitev iOS SDK preko nadzorne plošče [57].

### 4.2.3 Namestitev Android

Preko Titanium Studio [13], [4] nadzorne plošče je potrebno namesti Android SDK, kot prikazuje slika 4.3. Med namestitvijo se izbere željene verzije Android SDK.




Slika 4.3: Namestitev Android SDK preko nadzorne plošče [57].

# Poglavje 5

# Pregled obstoječih rešitev za potne naloge

Obstaja veliko spletnih rešitev za potne naloge, medtem ko funkcionalnih mobilnih rešitev ni.

## 5.1 Spletne rešitve

Na spletu obstaja nekaj spletnih rešitev za potne naloge, pri katerih je večina plačljiva z visoko ceno. Opisani sta dve brezplačni rešitvi, kateri je možno nadgraditi v plačljivo različico.

### 5.1.1 Spletna rešitev Potni nalog

Spletna rešitev Potni nalog [51] je brezplačna spletna aplikacija, ki je namenjena vsem podjetjem, ki obračunavajo potne naloge. Prednosti aplikacije so poenostavljeno vnašanje, obračunavanje, izpisovanje in analiziranje službenih poti, dostopnost aplikacije kjerkoli in kadarkoli ter neodvisnost od operacijskega sistema. Prav tako aplikacija samodejno posodablja ceno dnevnic in kilometrino določeno z zakonom, dodajanje zaposlenih in dodajanje ter pregledovanje potnih nalogov. Potnemu nalogu določimo:

- Datum in uro začetka potovanja.
- Datum in uro konca potovanja.
- Možnost obračuna dnevnice.
- Podatke o zaposlenem.
- Opis poti.
  - Vozilo.
  - Število prevoženih kilometrov.
  - Ceno kilometra.
  - Cilj potovanja.
  - Namen potovanja.
  - Opombe.
  - Izplačilo predujema.
- Stroške potovanja.
- Poročilo o opravljeni poti.

Aplikacijo je možno nagraditi v plačljivo verzijo, kjer dobimo dodatne možnosti filtriranja, tiskanja in urejanja potnih nalogov. Odstranijo se oglasi.

Slabosti aplikacije:

- Prijavlja se lahko samo administrator podjetja, kar pomeni, da mora vsak zaposleni dodati potni nalog preko njega. Možnost prijave zaposlenih v sistem je omogočena samo v plačljivi verziji.
- Brezplačna verzija prikazuje oglase, kateri se zelo hitro spreminjajo in so moteči.
- Potnemu nalogu se ne more dodati lokacij in strank.
- Uporabnik ne more izpisati potnih nalogov niti za daljše obdobje niti več označenih potnih nalogov.

### 5.1.2 Spletna rešitev pisarna.biz

Aplikacija pisarna.biz [49] ponuja hitro in enostavno izdelavo potnih nalogov, omogoča dodajanje šifrantov prevoznih sredstev, voznikov, namenov potovanj in relacij potovanj. Potni nalog vsebuje:

- Številko naloga.
- Ime in priimek voznika.
- Naslov voznika.
- Prevozno sredstvo.
- Začetno stanje števca kilometrov.
- Končno stanje števca kilometrov.
- Datum potovanja.
- Zaključek potovanja.
- Kraj potovanja.
- Datum in znesek predujema.
- Datum in znesek izplačila.
- Relacije potovanja.
  - Zaporedna številka.
  - Datum.
  - Relacija.
  - Namen.
  - Kilometri.
  - Cena kilometra.
  - Znesek.

- Dnevnice.
- Ostali stroški.

Slabosti:

- Aplikacija ni posodobila cene dnevnice, ki je bila spremenjena z zakonom in prikazuje neažurne vrednosti.
- Uporabnik mora sam vnesti število prevoženih kilometrov.
- Pregled potnih nalogov je nepregleden, filtri so nesmiselno označeni s simbolom vprašaja.
- Pri dodajanju novega potnega naloga je potrebno dvojno vnašanje končne relacije v polje kraj potovanja in razdelek relacije potovanja.
- Relacijam potovanja ni možno vnesti imena stranke.
- Možnost prijave ima samo administrator podjetja, kar pomeni, da mora vsak zaposleni dodati potni nalog preko njega.

## 5.2 Mobilne rešitve

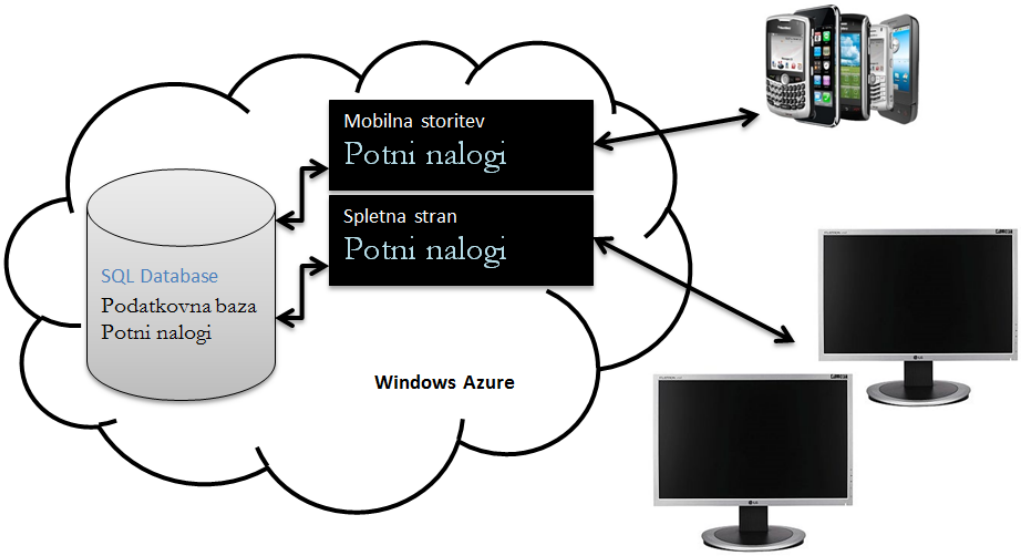
Ponudba dobrih mobilnih aplikacij za potne naloge je zelo slaba, oziroma je ni. Za iOS je možna le ena plačljiva aplikacija Potni nalog [23]. Za prenos končanih potnih nalogov je potrebno končni dokument poslati na elektronsko pošto ali jih natisniti z AirPrint tiskalnikom. Vsebina aplikacije Potni nalog je zelo okrnjena, saj vsebuje le nekaj osnovnih možnosti vnosa.

Za Android napravo sta na voljo dve aplikacije. Aplikacija Moj Potni Nalog [33] obljublja veliko, vendar temu ni tako. V opisu aplikacije je na voljo tudi iOS različica, katera ne obstaja. Prav tako opisuje možnost prenosa potnega naloga na strežnik in urejanje potnega naloga preko spletne strani. Opisana spletna stran ne obstaja in tudi v aplikaciji ni možnosti, ki bi omogočale prenos potnega naloga na strežnik. Potni nalogi so izolirani na

mobilni napravi. Aplikacije ni možno uporabljati brez internetne povezave in pogled se ne prilagaja položaju zaslona. Poleg tega je uporabnik v spletni trgovini opisal probleme delovanja na tabličnem računalniku. Aplikacija ima dobro zamišljen uporabniški vmesnik, vendar zaradi vseh pomanjkljivosti ni primerna za uporabo, razen za shranjene potne naloge na mobilni napravi, ki jih ni možno uporabiti. Druga aplikacija PaNdroid [34] je sestavljena samo iz enega okna, kjer se lahko vnese datum odhoda, uro odhoda, uro prihoda, relacijo in kilometre. Aplikacija je brez gumbov in je popolnoma nerazumljiva. Tudi spletna stran razvijalca je zelo nepregledna.

# Poglavje 6

# Razvoj aplikacije Potni nalogi




Slika 6.1: Shema rešitve Potni nalogi.

Rešitev Potni nalogi je sestavljena iz mobilne aplikacije, mobilne storitve v oblaku in aplikacije v oblaku. Mobilna aplikacija komunicira s storitvijo v oblaku, kot prikazuje slika 6.1. Do aplikacije v oblaku lahko uporabnik dostopa preko spletnega brskalnika. Za razvoj mobilne aplikacije se je uporabilo ogrodje Titanium, medtem ko aplikacija v oblaku uporablja 3-slojno večuporabniško arhitekturo razvito v ogrodju Visual studio, v katerem je

bila prav tako razvita mobilna storitev v oblaku. Podatki mobilne aplikacije in aplikacije v oblaku so sinhronizirani, saj uporabljajo skupno podatkovno bazo v oblaku.

## 6.1 Opis spletne aplikacije

### 6.1.1 Arhitektura rešitve

#### Več uporabniška arhitektura

Spletna aplikacija ima večuporabniško arhitekturo, ki pomeni [58] več uporabnikov, ki so običajno neodvisni in uporabljajo skupne vire. Aplikacija uporablja pristop deljene baze in sheme [44], saj uporabniki uporabljajo isto bazo in tabele, vsak podatek pa je vezan na ustreznega uporabnika. Ta pristop omogoča najnižje stroške strojne opreme in varnostnega kopiranja, vendar vključuje dodatno delo za zagotavljanje varnosti, da uporabnik v primeru napake ali napada ne pridobi podatkov drugih uporabnikov. Ker aplikacija upošteva več uporabniško arhitekturo in se nahaja v oblačni platformi, jo lahko klasificiramo [15] z oznako programska oprema kot storitev ali s kratico SaaS.

#### 3-slojna arhitektura

Aplikacija upošteva 3-slojno arhitekturo in je sestavljena iz podatkovnega, poslovnega in predstavitvenega sloja. Glavne prednosti večslojne arhitekture [7] so:

- Vzdrževanje: sloji so med seboj neodvisni, v primeru sprememb ni potrebno spreminjati aplikacije kot celote.
- Razširljivost: posamezeni sloj se lahko obravnava ločeno, kar olajša delo.
- Prilagodljivost: večja prilagodljivost zaradi ločenega vzdrževanja in razširjanja posamezenega sloja.

- Razpoložljivost: razširljivost komponent zaradi večslojne arhitekture.

Naloga podatkovnega sloja je komunikacija s podatkovno bazo. Ločen podatkovni sloj omogoča, da se v primeru zamenjave podatkovne baze popravi povezava samo na tem sloju. Poslovni sloj skrbi za logiko aplikacije in je vmesni sloj med podatkovnim in predstavitvenim slojem. Njegova naloga je posredovanje ustrezno formatiranih podatkov iz podatkovnega v predstavitveni sloj in obratno. V predstavitvenem sloju se definira uporabniške vmesnike za predstavitev podatkov.

### 6.1.2 Implementacija

Rešitev Potni nalogi je bila razvita v ogrodju Visual Studio 2012. Na strežniški strani se uporablja C#, medtem ko se na uporabniški strani uporablja JavaScript programski jezik. Za prikaz skrbita HTML5 označevalni jezik in CSS3. Rešitev je realizirana v 3-slojni arhitekturi, zato je projekt razdeljen v tri sloje.

Podatkovni sloj skrbi za komunikacijo z bazo, za kar skrbi ogrodje Entity Framework, ki je priporočena tehnologija [31] za podatkovni dostop novih aplikacij s strani Microsofta. Podatkovni sloj je implementiran generično, kar pomeni, da aplikacija deluje s katerokoli podatkovno bazo, ki ima ustrezno strukturo. Za poizvedovanje se uporablja poizvedovalni jezik LINQ (ang. language integrated query) [38], ki je bil predstavljen z Visual Studiom 2008 in omogoča poizvedovalne možnosti programskemu jeziku C# in Visual Basicu. LINQ je mogoče povezati s katerokoli podatkovno strukturo, njegov cilj je zapolniti luknjo med podatki in objekti.

Poslovni sloj skrbi za komunikacijo podatkovnega in predstavitvenega sloja. Predstavitveni sloj je narejen v ASP.NET MVC 4 ogrodju, ki je namenjeno [5] izdelavi spletnih aplikacij, katere upoštevajo vzorec Model-View-Controller. Ogrodje predstavlja alternativo [10] Microsoftovem ogrodju Web Forms na platformi .NET. Prva verzija ASP.NET MVC je nastala novembra leta 2007, trenutno je v uporabi četrta izdaja. Največje prednosti so:

- Popolna kontrola nad generiranjem HTML-ja.
- Popolna kontrola URL naslovov.
- Omogoča neodvisno obravnavo plasti aplikacije.
- Razširljivost.
- Omogoča lažje testiranje.

Vzorec Model-View-Controller (MVC) je bil aktualen že leta 1979. Takrat se je imenoval Thing-Model-View-Editor in so ga kasneje preimenovali. Največja moč MVC-ja je v neodvisni obravnavi plasti aplikacije, posledica le te je manjše povečanje kompleksnosti arhitekture aplikacije, vendar velika pridobitev skozi čas razvoja. Vzorec MVC je sestavljen iz treh plasti:

- Črka M predstavlja model in je skupek razredov, ki predstavljajo podatke in poslovna pravila, ki določajo, kako so uporabljeni podatki.
- Črka V predstavlja pogled in definira izgled uporabniškega vmesnika.
- Črka C predstavlja upravljanje in je skupek razredov, ki upravljajo s komunikacijo z uporabnikom, tokom aplikacije in specifično logiko aplikacije.

Za izdelavo poročil se uporablja iText [37], to je knjižnica, ki omogoča izdelavo in urejanje pdf dokumentov. Knjižnica je na voljo za programski jezik Java in C#. Za prikaz podatkov v tabelah se uporablja DataTables, ki je dodatek za jQuery JavaScript knjižnico.

Večuporabniška arhitektura je implementirana v poslovni logiki in v strukturi podatkovne baze. Na nivoju poslovne logike je implementirana varnostna zaščita, ki zagotavlja, da uporabniki ne morejo dostopati do podatkov drugih uporabnikov. Na nivoju podatkovne baze so podatki posredno ali neposredno povezani s podjetjem, kateremu pripadajo, do njih ni možen dostop uporabnikom drugih podjetij.

### 6.1.3 Namestitev aplikacije

Celotno namestitev je potrebno opraviti z uporabniškim dostopom, ki ima administratorske pravice.

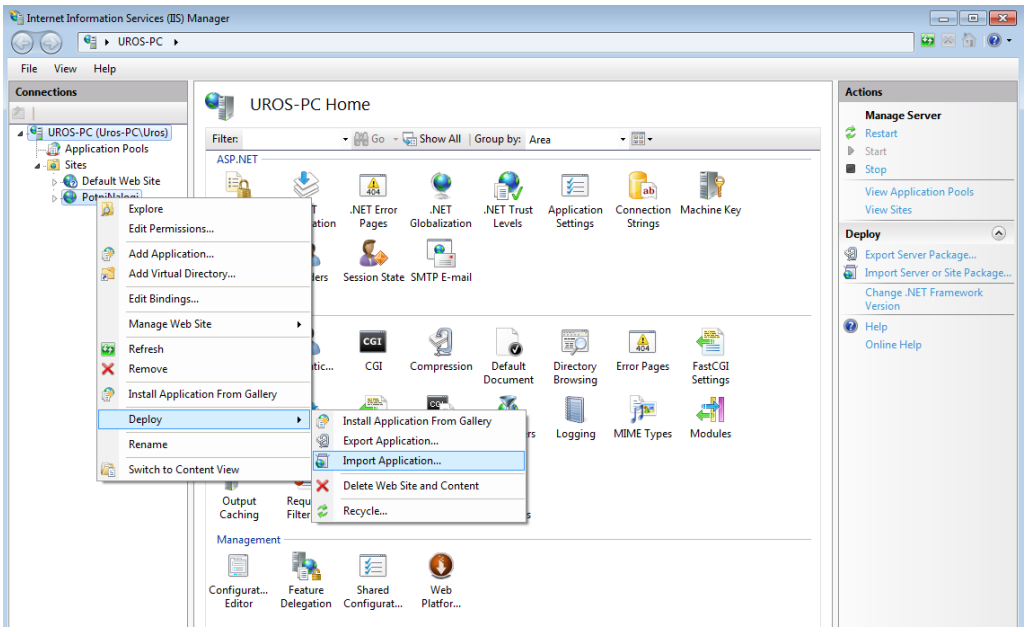
#### Pred namestitvijo

Pred samo namestitvijo je potrebno preveriti, če je nameščen .NET Framework 4.5 in "Internet Information Services" (IIS). V primeru nenameščenega .NET Framework 4.5, se ga prenese z Microsoftove strani [40] in namesti. IIS se namešča preko okna "Vklop ali izklop funkcij sistema Windows" pod zavihkom "Internet Information Services". Za lažjo namestitev aplikacije na IIS je potrebno namestiti dodatek "Web Deploy Tool", ki je na voljo na uradni strani IIS [64]. Potreben je tudi strežnik, na katerem bo nameščena podatkovna baza.

#### Namestitev spletne aplikacije

Spletna aplikacija Potni nalogi se namesti z "Internet Information Services (IIS) Manager". Pod zavihkom Sites se lahko pregleduje obstoječe strani na strežniku. Aplikacijo se lahko namesti na privzeto spletno stran ali se ustvari novo. Stran mora imeti aplikacijski bazen nastavljen na ASP.NET v4.0, ki se preveri v polju "Application Pool", ki se nahaja pod naprednimi nastavitvami. Aplikacija se namesti z desnim klikom strani in izbiro "Deploy/Import Applicationž menija, kar omogoča dodatek "Web Deploy Tool", kot prikazuje slika 6.2.

Odpre se pogovorno okno, ki zahteva namestitveni paket aplikacije, ki se ga ustvari z Visual Studiom. Sledimo namestitvenim korakom do pogovornega okna, kjer je potrebno nastaviti aplikacijsko pot. V primeru samostojne strani v IIS se lahko aplikacijska pot pusti prazna, v nasprotnem primeru se običanjo nastavi aplikacijsko pot na ime aplikacije. Tu je tudi možnost nastaviti povezavo do podatkovne baze. V primeru že ustvarjene podatkovne baze se nastavi dostop do instance baze, izbere katalog in nastavi uporabniško ime




Slika 6.2: Prikaz menija preko katerega namestimo spletno aplikacijo.

ter geslo. V nasprotnem primeru se povezava uredi v datoteki "Web.config". Namestitev se zaključi s klikom naprej.

#### Namestitev podatkovne baze

Na podatkovnem strežniku poženemo namestitveno skripto podatkovne baze, ki ustvari novo podatkovno bazo. V skripti je možno popraviti ime podatkovne baze. V primeru že nameščene spletne aplikacije je potrebno popraviti datoteko Web.config s pravilno povezavo z bazo.

### 6.1.4 Struktura projekta rešitve

#### Podatkovni sloj

Podatkovni sloj je sestavljen iz dveh projektov. Projekt PotniNalogi.Data.Access je vmesnik za povezavo s podatkovno bazo. Vsi ostali razredi v tem sloju morajo dedovati od tega vmesnika, da zagotavljajo popolni dostop do podatkovne baze. Drugi projekt PotniNalogi.Data.Access.MySql je implementacija

vmesnika za dostop do podatkovne baze. Spremembe ob zamenjavi podatkovne baze niso potrebne, ker je dostop narejen generično.

### Vmesni sloj

Razdelek "Cross-cutting"je namenjen generičnosti rešitve. Ta razdelek vsebuje vse, kar se uporablja skozi več slojev (modeli, seznami in entitete). Entitete predstavljajo tabele v podatkovni bazi, ki se ustvarijo samodejno, za kar skrbi ADO.NET Entity Data Model. Ta model vsebuje datoteka "Model.edmx". V primeru spreminjanja strukture podatkovne baze, je potrebno posodobiti model. Priporočeno je, da se ob spremembi izbriše vse entitete iz modela edmx ter z menija izbere "Update model from database", kjer se doda potrebne tabele.

### Poslovni sloj

Poslovni sloj vsebuje en projekt, v njem je definirana komunikacija predstavitvenega sloja s podatkovnim slojem. Poslovni sloj je implementiran kot en razred skozi več datotek, kar omogočena večjo preglednost zaradi velikega števila metod. Metode so razdeljene v datoteke po vlogi uporabnikov, katerih funkcionalnosti implementirajo.

### Predstavitveni sloj

Predstavitveni sloj je sestavljen iz enega ASP.NET MVC 4 projekta. Ker aplikacija ustreza 3-slojnim arhitekturnim zahtevam, predstavitveni sloj nima neposrednega dostopa do podatkovnega sloja. Predstavitveni sloj pridobi podatke preko poslovnega sloja. Uporabniški vmesniki in komunikacija s poslovnim slojem je implementirana v območjih pod razdelkom "Areas", ki so razdeljena glede na vloge uporabnikov. Ta območja so:

- "Admin".
- "CompanyAdmin".

- "Employee".

Območje "Admin"vsebuje implementacijo za vlogo administrator, v "CompanyAdmin"je implementacija za vlogo administrator podjetja in "Employee"vsebuje implementacijo za vlogo zaposleni. Komunikacija z uporabnikom, ki ni vezana na vloge je implementirana izven območij in je v razdelku "Controllers"in "Views", tu je implementirana tudi prijava uporabnika in komunikacija z mobilno aplikacijo. Vsaka akcija ima nad kontrolnim razredom določeno vlogo, katera lahko dostopa, logika dostopa je implementirana pod razdelkom "Filters".

## 6.2 Funkcionalen opis spletne aplikacije

Aplikacija Potni nalogi podpira tri vloge uporabnikov:

- Administrator.
- Administrator podjetja.
- Zaposleni.

Ob prijavi uporabnika aplikacija ponudi možnosti glede na vlogo. Vsak prijavljen uporabnik lahko spreminja podatke profila in geslo za vpis. Podatki profila so:

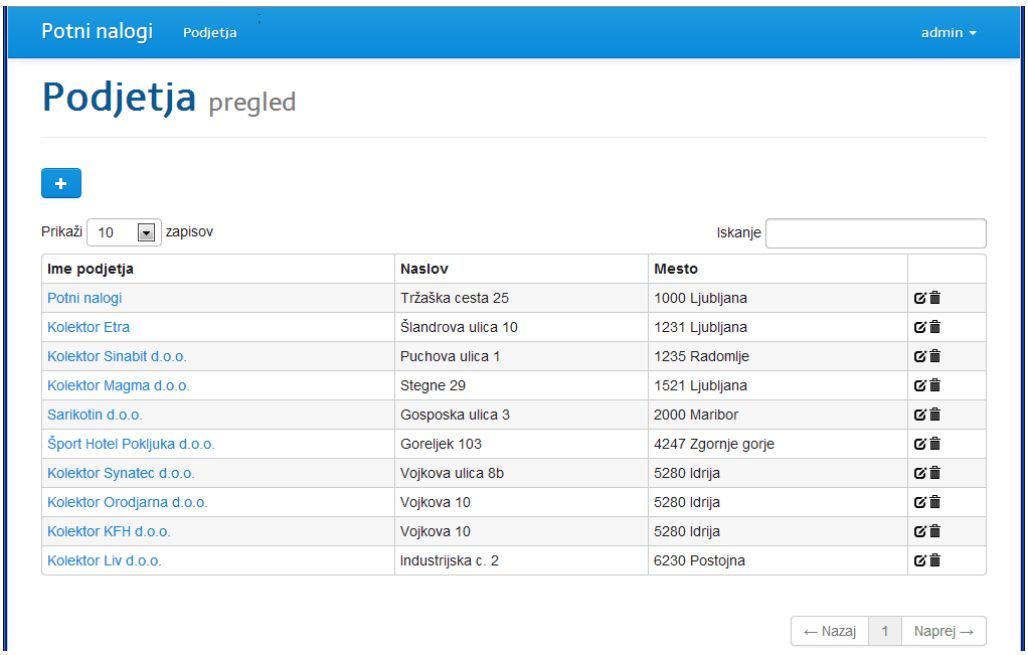
- Ime.
- Priimek.
- Telefonska številka.
- E-pošta.
- Naslov.
- Poštna številka in mesto.

### 6.2.1 Administrator

Administrator je skrbnik sistema, ki se lahko prijavi preko običajnega okna za prijavo ali uporabi okno, kjer se prijavi s prstnim odtisom. Njegova naloga je, da dodaja in odstranjuje podjetja iz sistema.

#### Pregled podjetij

Administrator ima možnost pregleda podjetij v sistemu, kot prikazuje slika 6.3. Podjetja lahko dodaja, izbriše ali ureja. Za vsa podjetja je možen pregled osnovnih podatkov podjetja.




Slika 6.3: Prikaz maske za pregled podjetij.

#### Dodajanje podjetja

Administrator doda novo podjetje v sistem preko maske za dodajanje novega podjetja, ki jo prikazuje slika 6.4. Poleg podjetja se ustvari tudi administratorja podjetja. Vnosna polja so:

- Podatki o podjetju:
  - Ime podjetja (obvezno polje).
  - Naslov podjetja (obvezno polje).
  - Poštna številka in mesto (obvezno polje).
  - ID za DDV.
  - TRR.
- Podatki uporabnika:
  - Uporabniško ime (obvezno polje).
  - Geslo (obvezno polje).
  - Potrditev gesla (obvezno polje).
  - Ime.
  - Priimek.

#### Urejanje podjetja

Administrator lahko ureja podatke podjetja. V primerjavi z dodajanjem novega podjetja ne more spreminjati imena podjetja in podatkov uporabnika.

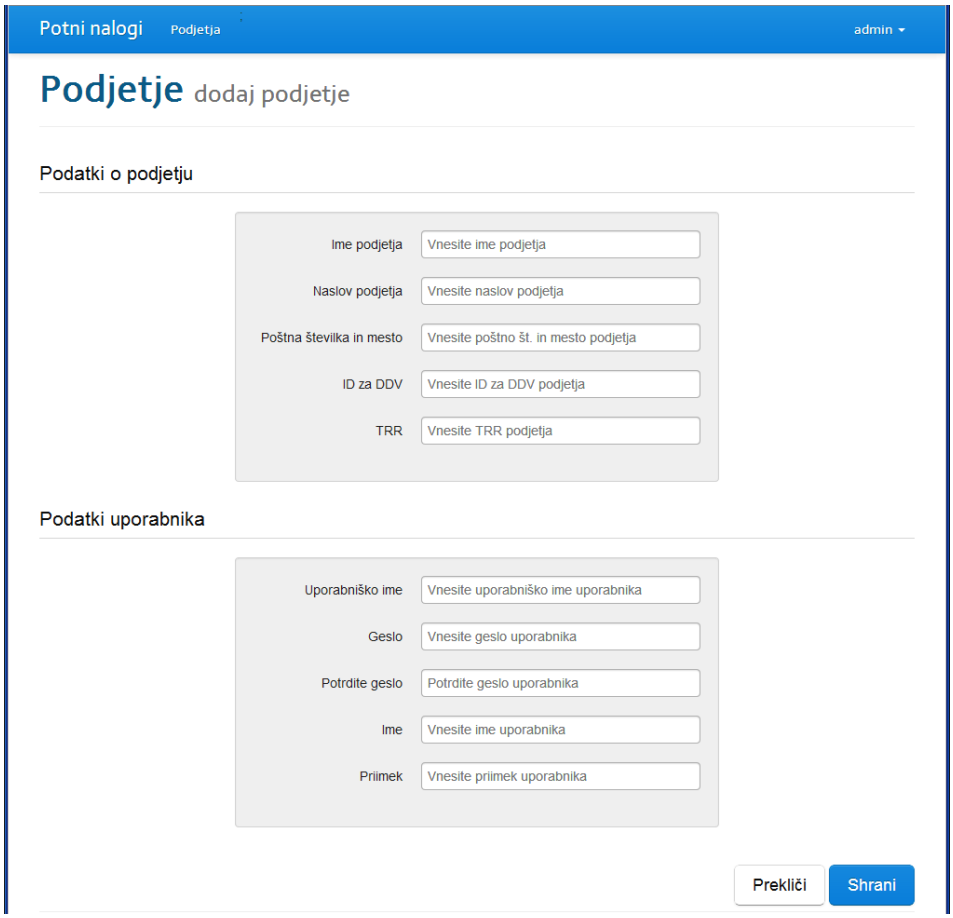
### 6.2.2 Administrator podjetja

Administrator podjetja lahko ureja podatke podjetja, zaposlenih, strank in potnih nalogov.

#### Podjetje

Administrator podjetja lahko spremeni podatke podjetja in določi ceno kilometra, kot prikazuje slika 6.5. Podatki podjetja, ki jih lahko spremeni, so:

- Naslov podjetja (obvezno polje).

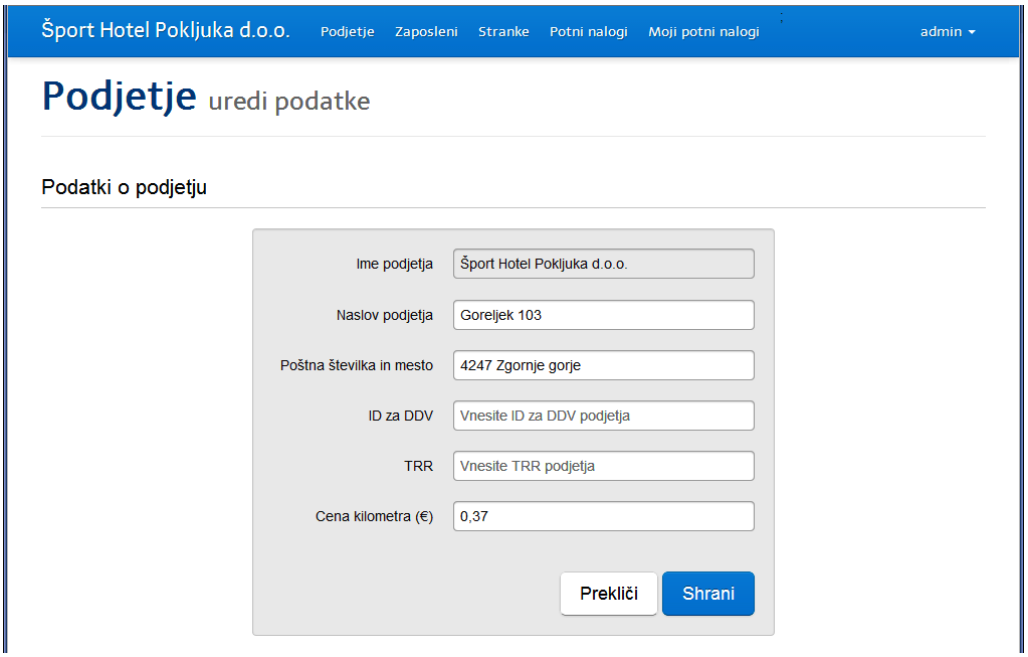



Slika 6.4: Prikaz maske za dodajanje podjetja.

- Poštna številka in mesto (obvezno polje).
- ID za DDV.
- TRR.
- Cena kilometra podjetja, katera je privzeto nastavljena na 0,37 € (obvezno polje).

### Pregled zaposlenih

Administrator podjetja lahko doda, izbriše ali ureja zaposlene. Vsak zaposleni se lahko prijavi v sistem. Pri pregledu zaposlenih so na voljo naslednji




Slika 6.5: Prikaz maske za urejanje podatkov podjetja.

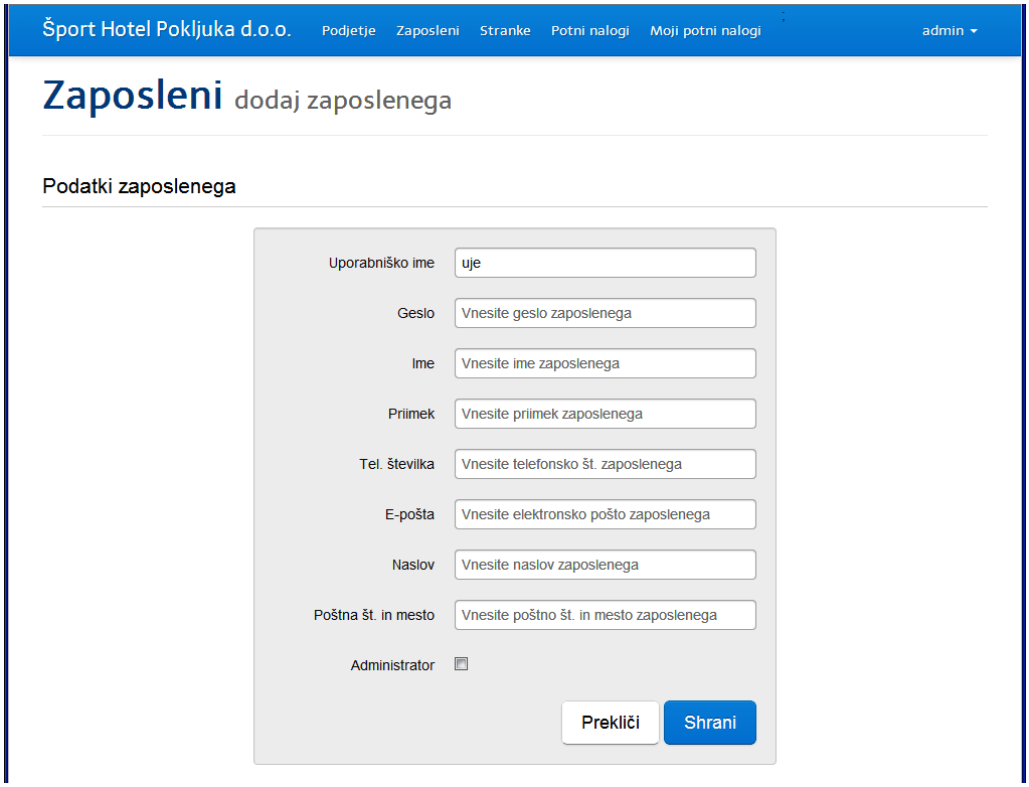
podatki:

- Uporabniško ime.
- Ime.
- Priimek.
- Naslov.
- Mesto.
- Telefonska številka.
- E-pošta.

### Dodajanje zaposlenega

Zaposlenega se doda preko maske za dodajanje zaposlenega, prikazuje jo slika 6.6. Poleg osnovnih podatkov je potrebno vpisati še geslo za vpis in določiti

ali je administrator podjetja. V primeru, da je uporabnik administrator podjetja, bo imel vse pravice le tega, s tem bo lahko urejal podatke podjetja, zaposlenih, strank in potnih nalogov. V nasprotnem primeru bo uporabnik lahko pregledoval in urejal samo svoje potne naloge.




Slika 6.6: Prikaz maske za dodajanje zaposlenega.

### Urejanje zaposlenega

Administrator podjetja lahko zaposlenemu uredi podatke z izjemo uporabniškega imena. Dodatno lahko zaposlenemu doda ali odvzame vlogo administratorja podjetja.

### Pregled strank

Administrator podjetja lahko doda, izbriše ali uredi podatke strank. Pri pregledu so na voljo podatki:

- Naziv.
- Naslov.
- Mesto.
- Telefonska številka.
- E-pošta.

### Dodajanje strank

Administrator podjetja lahko doda stranko na dva načina. Stranko lahko doda preko maske za dodajanje stranke, kjer vnese podatke stranke, kar prikazuje slika 6.7 Druga možnost je, da uvozi CSV (ang. comma separated values) datoteko s podatki o strankah. CSV datoteko se lahko ustvari preko običajnega urejevalnika besedila ali preko pogosto uporabljenega Excel Office programa. V primeru uporabe običajnega urejevalnika besedila se zapise loči s podpičjem, medtem ko Excel Office naredi to samodejno. V primeru uporabe šumnikov je potrebno uporabiti kodiranje UTF-8. Struktura zapisa stranke v CSV datoteki je določena, na prvem mestu je ime stranke, sledi mu naslov, poštna številka, mesto, elektronska pošta in telefonska številka stranke. Elektronska pošta in telefonska številka nista obvezni. V primeru uvoza stanke z imenom, ki že obstaja, bo uvoz tak zapis preskočil.

Struktura zapisa stranke v CSV datoteki:
[Ime stranke] ; [Naslov stranke] ; [Poštna številka in pošta] ; [E-pošta]* ; [Telefonska številka]*

Primer zapisa stranke v CSV datoteki:
Testno podjetje d.o.o.; Kranjska ulica 53a; 1000 Ljubljana; info@testnopodjetje.si; +38640100100

### Urejanje stranke

Administrator podjetja lahko uredi podatke o strankah. Spreminja lahko:


Slika 6.7: Prikaz maske za dodajanje stranke.

- Naziv.
- Naslov.
- Poštno številko in mesto.
- Telefonsko številko.
- E-pošto.

### Pregled potnih nalogov

Administrator podjetja lahko pregleduje, ureja in izbriše potne naloge vseh zaposlenih v podjetju. Pri pregledu ima na razpolago:

- Številko potnega naloga.
- Ime in priimek potnika.
- Namen potnega naloga.
- Odhod, ki prikazuje datum in čas prve lokacije v potnem nalogu.

- Prihod, ki prikazuje datum in čas zadnje lokacije v potnem nalogu.

- Število prevoženih kilometrov.

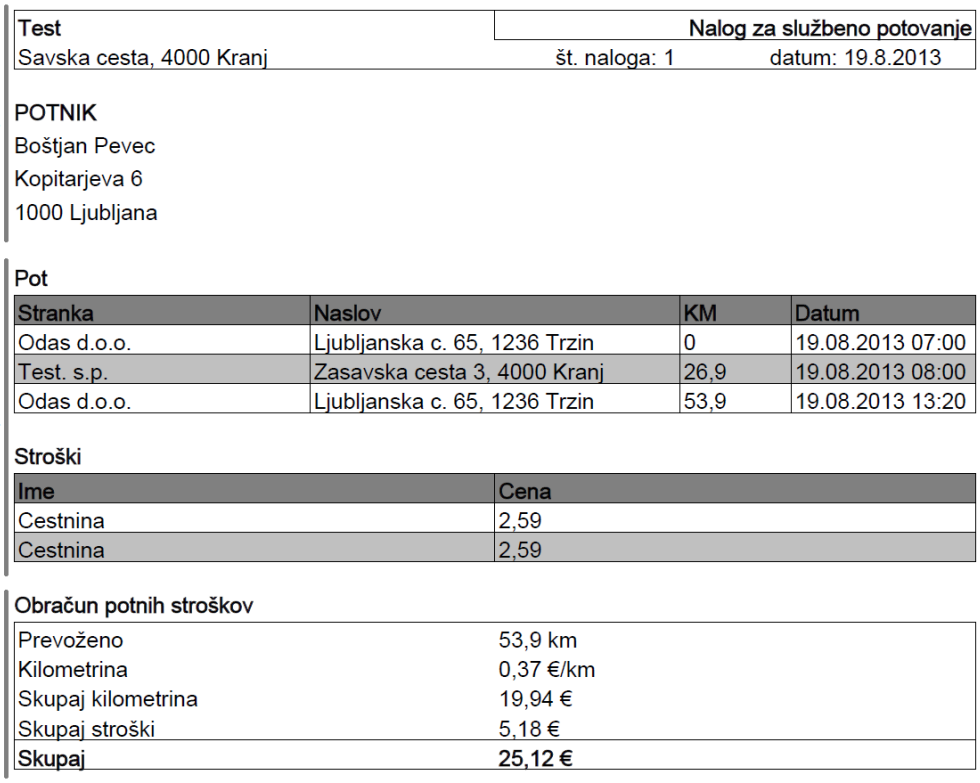

Slika 6.8: Prikaz maske za pregled postavk potnega naloga.

Postavke potnega naloga se pregledujejo v oknu za pregled potnega naloga, ki se prikaže s klikom na številko potnega naloga. Okno za pregled prikazuje slika 6.8 in vsebuje gumb z napisom "PDF", ki zgenerira pdf datoteko poročila. Za izpis poročila vseh potnih nalogov v določenem časovnem obdobju se izbere možnost "PDF poročilo", ki preusmeri na masko z izbiro časovno obdobje in zaposlenega, za katerega se želi poročilo. Zgenerira se PDF (ang. portable document format) datoteka, ki vsebuje potne naloge uporabnika v določenem obdobju in osnovno poročilo o številu potnih nalogov in številu prevoženih kilometrov v tem obdobju.

### Postavke potnega naloga

Potni nalog je sestavljen iz treh sklopov, kot prikazuje slika 6.9. Prvi sklop vsebuje naziv in naslov podjetja, številko potnega naloga, datum generiranja poročila, ime, priimek in naslov potnika ter začetno in končno stanje števca kilometrov, če ju je vnesel uporabnik. Srednji sklop je sestavljen iz seznama lokacij in stroškov. Posamezna lokacija vsebuje:

- Ime stranke.
- Naslov stranke.
- Razdaljo.
- Datum in čas.



| Test | Nalog za službeno potovanje |
|---|---|
| Savska cesta, 4000 Kranj | št. naloga: 1 datum: 19.8.2013 |

1

**POTNIK**
Boštjan Pevec
Kopitarjeva 6
1000 Ljubljana

**Pot**

| Stranka | Naslov | KM | Datum |
|---|---|---|---|
| Odas d.o.o. | Ljubljanska c. 65, 1236 Trzin | 0 | 19.08.2013 07:00 |
| Test. s.p. | Zasavska cesta 3, 4000 Kranj | 26,9 | 19.08.2013 08:00 |
| Odas d.o.o. | Ljubljanska c. 65, 1236 Trzin | 53,9 | 19.08.2013 13:20 |

2

**Stroški**

| Ime | Cena |
|---|---|
| Cestnina | 2,59 |
| Cestnina | 2,59 |

**Obračun potnih stroškov**

| | |
|---|---|
| Prevoženo | 53,9 km |
| Kilometrina | 0,37 €/km |
| Skupaj kilometrina | 19,94 € |
| Skupaj stroški | 5,18 € |
| **Skupaj** | **25,12 €** |

3

Slika 6.9: Potni nalog je sestavljen iz treh sklopov.

Razdalje je možno vnesti ročno ali se uporabi Google Maps Directions API spletna storitev. Spletna storitev je namenjena [59] pridobitvi podatkov zunanjim storitvam, ki potrebujejo podatke o lokacijah. Spletna storitev uporablja HTTP zahteve kot vmesnik za dostop. Parametri, ki so podani v zahtevo, služijo kot argumenti storitve. Storitev vrača rezultat kot JSON (ang. javascript object notation) ali XML (ang. extensible markup language) odgovor, odvisno od zahteve. Rešitev Potni nalogi zahteva od storitve JSON odgovor, saj ima krajši odgovor v primerjavi s XML in ker je pravilno formatiran JavaScript objekt primeren za uporabo brez obdelave. Aplikacija pošlje naslove potnega naloga storitvi, ki odgovori z JSON objektom s podatki o razdaljah med lokacijam. Aplikacija shrani podatke o razdaljah in jih prikaže na potnem nalogu. Razdalje se seštevajo in razdalja na zadnji lokaciji je seštevek celotne prevožene razdalje, kar posledično pomeni, da je razdalja prve lokacije vedno enaka nič. Posamezni strošek je sestavljen iz:

- Ime stroška.
- Cena stroška.

Zadnji sklop je povzetek in izračun potnih stroškov. Sklop je sestavljen iz:

- Število prevoženih kilometrov.
- Cena kilometra - kilometrina.
- Izračunan strošek kilometrine glede na število prevoženih kilometrov.
- Seštevek stroškov.
- Izračun potnih stroškov, kot vsota stroškov in preračunane kilometrine.

### Dodajanje potnega naloga

Nov potni nalog se ustvari preko maske za dodajanje potnega naloga. Administrator podjetja vpiše namen poti, izbere zaposlenega kateremu pripada potni nalog in doda lokacije ter stroške. Pri dodajanju posamezne lokacije

mora izpolniti ime in naslov stranke ter datum opravljene poti. Namesto vpisovanja imena in naslova stranke lahko iz izbirnega menija izbere že obstoječo stranko, katero je dodal v sistem. Opis dodajanja stranke v sistem je opisan pod naslovom Dodajanje strank.

#### Urejanje potnega naloga

Administrator podjetja lahko ureja potne naloge vseh zaposlenih. Potnemu nalogu lahko spremeni namen, pot in stroške.

### 6.2.3 Zaposleni

Uporabniki, ki imajo vlogo zaposlenega, lahko pregledujejo in urejajo svoje potne naloge.

#### Pregled potnih nalogov

Pregled potnih nalogov je popolnoma enak kot pri administratorju podjetja, a zaposleni vidi samo svoje potne naloge in nima možnosti izdelave poročila skozi daljše časovno obdobje.

#### Dodajanje potnega naloga

Zaposleni lahko doda nov potni nalog. Sistem potnemu nalogu samodejno določi potnika, to je prijavljen uporabnik, ki dodaja potni nalog. Zaposleni lahko potnemu nalogu določi namen, pot in stroške.

#### Urejanje potnega naloga

Zaposleni lahko ureja samo svoje potne naloge. Potnemu nalogu lahko spremeni namen, pot in stroške.

## 6.3 Opis mobilne storitve v oblaku

Za razvoj mobilne storitve v oblaku se je uporabil pristop samostojnega razvoja in uporabe infrastrukture v oblaku. Izmenjava podatkov med mobilno aplikacijo in mobilno storitvijo v oblaku poteka v formatu JSON (ang. JavaScript object notation). JSON je tekstovno usmerjen standard za podatkovno izmenjavo podatkov. Izpeljan je iz načina prikaza objektov v programskem jeziku JavaScript, vendar je kljub temu neodvisen od programskega jezika. Mobilna storitev v oblaku pričakuje podatke v JSON formatu, katere preoblikuje v objekte. Odgovor storitve je prav tako v JSON formatu. Mobilna storitev v oblaku vsebuje:

- Metodo LogIn: metoda je namenjena avtentikaciji uporabnika. Parametri metode so ime podjetja, uporabniško ime in geslo.

- Metodo Synchronize: metoda je namenjena sinhronizaciji podatkov mobilne aplikacije s podatki spletne aplikacije. Parameter metode so podatki potnega naloga v JSON formatu.

- Metodo getClientListByCompanyNameAndUserCredentials: metoda je namenjena pridobivanju strank podjetja. Parametri metode so ime podjetja, uporabniško ime in geslo.

## 6.4 Opis mobilne aplikacije

Mobilna aplikacija je projekt ogrodja Titanium, ki ga sestavljajo JavaScript datoteke. Uporablja se modularna arhitektura s CommonJS moduli, ki so priporočena arhitektura [28] s strani Appceleratorja. CommonJS [26], [27], [12] naredi javno viden modul in izniči JavaScript specifične probleme z globalnim imenskim prostorom in imenskimi konflikti.

### 6.4.1 Funkcionalen opis mobilne aplikacije

#### 6.4.1.1 Prijava

Začetno okno mobilne aplikacije je prijavno okno, v katerega mora uporabnik vnesti:

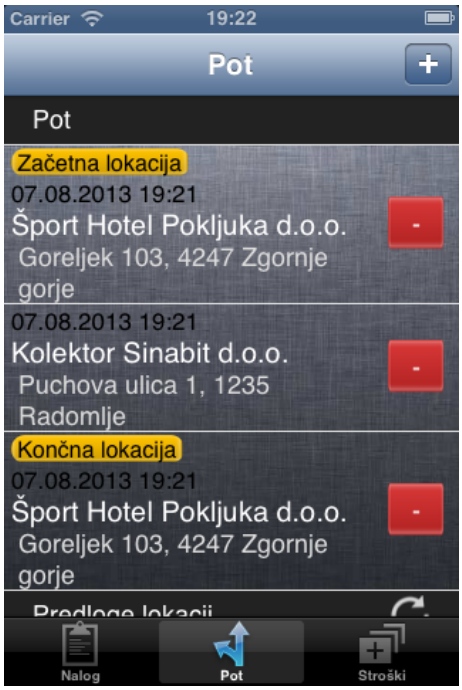
- Podjetje.
- Uporabniško ime.
- Geslo.

Uporabnik za prijavo potrebuje internetno povezavo. Po prvi uspešni prijavi si aplikacija zapomni uporabnika, kateremu se ni potrebno več prijaviti. Uporabnik se lahko odjavi in aplikacija ponovno zahteva prijavo.

#### 6.4.1.2 Izdelava potnega naloga

Uporabnik ustvari nov potni nalog s klikom na gumb "Nov potni nalog". Uporabniku se po kliku prikaže skupina zavihkov. V zavihku z imenom "Nalog"lahko konča ali prekliče potni nalog. V primeru končanja potnega naloga ga aplikacija shrani, v primeru preklica ga zavrže. Uporabnik lahko vpiše namen potnega naloga, začetno stanje števca kilometrov, končno stanje števca kilometrov, število prevoženih kilometrov ali se z gumbom "Nazaj"vrnemo na prvo okno, kjer se kasneje lahko vrne na urejanje potnega naloga. Zavihek "Pot"je namenjen določanju poti potnemu nalogu. Okno za prikaz je sestavljeno iz dveh sklopov:

- Pot, katera prikazuje pot potnega naloga.
- Predloge lokacij podjetja: predloge lokacij, ki jih ima podjetje vnešene v sistem. Podjetje vnese predloge lokacij preko spletne aplikacije pod razdelkom "Stranke", ki je opisano pod naslovom Dodajanje strank. Predloge lokacij podjetja vidijo vsi uporabniki mobilne aplikacije podjetja.



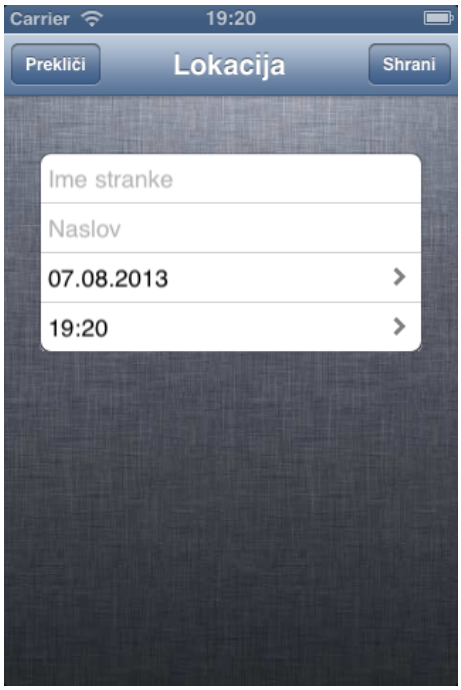

Slika 6.10: Prikaz maske izpolnjene poti potnega naloga.

Uporabnik mora izpolniti sklop "Pot", kot prikazuje slika 6.10. Lahko doda predloge lokacij podjetja ali novo lokacijo preko maske za dodajanje lokacije, ki jo prikazuje slika 6.11. Android dostopa do maske preko menija, medtem ko iPhone uporabniki preko navigacijskega menija. Pri lokaciji je potrebno vnesti:

- Ime stranke.
- Naslov stranke.
- Datum opravljene poti (privzeto se nastavi trenutni datum).
- Čas opravljene poti (privzeto se nastavi trenutni čas).

Pod zavihkom "Stroški"uporabnik vnese dodatne stroške, ki jih je imel na poti. Okno za prikaz je sestavljeno iz dveh sklopov:

- Stroški, kjer so prikazani dejanski stroški, ki jih je imel uporabnik na poti.




Slika 6.11: Prikaz maske za dodajanje lokacije v potni nalog.

- Predloge stroškov, kjer so prikazane lokalne predloge stroškov.

Uporabnik lahko stroške izpolni na dva načina in sicer doda predlogo stroška ali ustvari nov strošek. Novemu strošku je potrebno vnesiti:

- Ime stroška.
- Ceno stroška v €
- Shrani med predloge. Če uporabik izbere možnost: Shrani med predloge, se strošek shrani kot lokalna predloga stroška.

Masko za vnos novega stroška prikazuje slika 6.12.

#### 6.4.1.3 Sinhronizacija

Sinhronizacija pomeni posredovanje podatkov mobilne aplikacije strežniku. Za sinhronizacijo mora imeti uporabnik delujočo internetno povezavo. Sin-


Slika 6.12: Prikaz maske za dodajanje stroška v potni nalog.

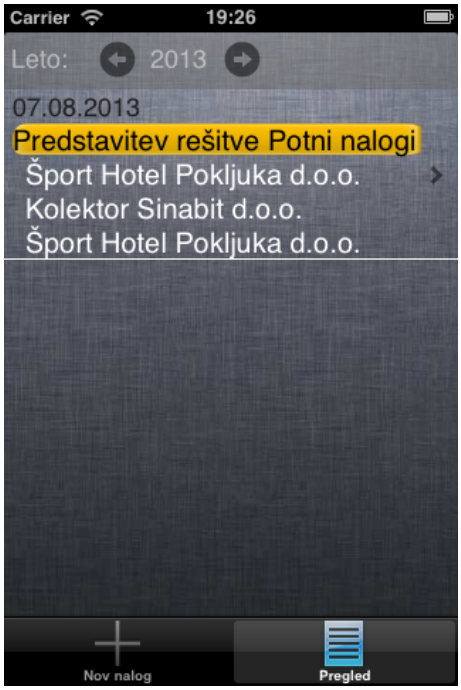
hronizacijo lahko uporabnik zahteva s klikom na gumb "Sinhroniziraj". Samodejna sinhronizacija se izvede po vsakem dodanem potnem nalogu. Samodejna sinhronizacija se izvede samo v primeru, če je vzpostavljena WIFI povezava. V primeru, če ima uporabnik vzpostavljeno podatkovno povezavo, samodejna sinhronizacija ne deluje, v izogib nenadzorovane dodatne porabe stroškov. Uporabnik lahko zahteva prenos predlog lokacij podjetja s strežnika s klikom na ikono osveži, ko ustvarja nov potni nalog pod razdelkom "Pot".

#### 6.4.1.4 Pregled potnih nalogov

Potne naloge urejene po letih se lahko pregleduje, če se v začetni skupini zavihkov izbere zavihek "Pregled", kot prikazuje slika 6.13. Prikažejo se samo tisti potni nalogi, ki so končani in imajo vsaj eno lokacijo. Zapis potnega naloga je sestavljen iz:

- Datuma začetka potnega naloga. Za datum začetka se upošteva datum prve lokacije v potnem nalogu.

- Namen potnega naloga.
- Seznam imen strank v potnem nalogu.




Slika 6.13: Prikaz maske, kjer pregledujemo končane potne naloge.

Potne naloge se lahko v Android operacijskem sistemu izbriše z daljšim pritiskom potnega naloga na zaslonu in potrditvijo izbrisa. V primeru iOS operacijskega sistema se potni nalog izbriše s povlekom od leve proti desni in izbiro gumba "Delete". S klikom na potni nalog se odpre skupina zavihkov za podrobnejši pregled in urejanje potnega naloga. V skupini zavihkov so trije razdelki:

- "Nalog": Tu se pregledujemo vse postavke potnega naloga.
- "Popravi pot": Uporabnik lahko odstrani ali doda lokacije potnega naloga.
- "Popravi stroške": Uporabnik lahko odstrani ali doda stroške potnega naloga.

## 6.5 Primer uporabe aplikacije

Spletna aplikacija je nameščena in administrator doda novo podjetje. Po dodanem podjetju v aplikacijo se administrator podjetja lahko prijavi v sistem. Ta izpolni manjkajoče podatke o podjetju, doda zaposlene in stranke. Zaposleni si namesti mobilno aplikacijo na mobilno napravo, se prijavi v aplikacijo in sinhronizira podatke. Na službeni poti si uporabnik ustvari nov potni nalog v mobilni aplikaciji. Potnemu nalogu določi namen in začetno lokacijo. Vmesno in končno lokacijo doda kasneje. V primeru upravičenih dodatnih stroškov jih uporabnik vnese in zaključi potni nalog. Ob vzpostavljeni WIFI povezavi se potni nalog samodejno prenese v spletno aplikacijo. V nasprotnem primeru mora uporabnik vzpostaviti internetno povezavo in sinhronizirati podatke. Končani potni nalog lahko pregleduje v razdelku mobilne aplikacije za pregled potnega naloga. Administrator podjetja ali zaposleni se prijavi v spletno aplikacijo in natisne potni nalog.

## 6.6 Primerjava rešitve z obstoječimi

Značilnosti ostalih rešitev so opisane v poglavju Pregled obstoječih rešitev za potne naloge. Največja prednost rešitve v diplomskem delu je v mobilni aplikaciji. Mobilna aplikacija omogoča sprotno vnašanje podatkov potnega naloga in omogoča sinhronizacijo s storitvijo v oblaku. Podpira Android in iOS operacijska sistema, ki pokrivata skoraj celotni trg [41], [21]. Na področju mobilnih rešitev ni primerne aplikacije za primerjavo.

Spletna rešitev Potni nalogi omogoča, da se lahko prijavijo vsi uporabniki. Uporabniki, ki nimajo administratorskega dostopa, lahko dostopajo samo do lastnih potnih nalogov in ne morejo spreminjati potnih nalogov drugih. Rešitev omogoča vnos lokacij v potni nalog in samodejni izračun razdalje med lokacijami. Poleg tega je edina izmed pregledanih rešitev, ki poleg naslova lokacije omogoča vnos imena stranke.

Rešitev v diplomskem delu omogoča izpis več potnih nalogov za poljubnega zaposlenega in obdobje. Vsebuje šifrant strank, ki se ga doda v potni

nalog in tako ni potrebno ponovno vnašanje podatkov stranke. Poleg tega se lahko stranke uvozi preko standardne CSV datoteke. Slabost rešitve je, da ne podpira dnevnic.

Cilj rešitve je, da uporabniki uporabljajo mobilno aplikacijo za izdelavo potnega naloga, preko spletne aplikacije pa se ureja šifrante ter popravlja, pregleduje in tiska potne naloge.

# Poglavje 7

# Sklepne ugotovitve

Programska in strojna oprema v oblaku [17] je tema, ki je zadnja leta zelo popularna in je pogosto uporabljena. Ideja je, da naj bo storitev v oblaku dostopna kadarkoli in kjerkoli ter brez omejitev operacijskega sistema in da ponudniku aplikacije ni potrebno skrbeti za sredstva, dostopnost, redundanco, primer naravne katastrofe in planiranje kapacitet, katere v primeru pomanjkanja enostavno poveča. Skrbniki oblaka so specializirani za ta vprašanja. Rešitev diplomske naloge ponuja aplikacijo v oblaku in mobilno storitev v oblaku. Poleg zgoraj naštetih prednosti računalništva v oblaku se je prednost pokazala v prihranku časa, saj ni bilo potrebno ustvariti namestitvenih paketov za različne operacijske sisteme zaradi dostopnosti preko brskalnika. Prav tako ni bilo potrebno priskrbeti strojne opreme. Mobilna storitev pa je omogočila enostaven prenos podatkov iz mobilne aplikacije v aplikacijo v oblaku preko standardiziranega formata JSON.

Medplatformski razvoj je mogoč zaradi podobnosti med platformami, vendar še ni popolnoma uveljavljen in se še razvija. V diplomskem delu so ugotovljene prednosti v prihranku časa za razvoj, lažjem vzdrževanju in uporabi skupnega programskega jezika ter okolja za različne platforme. Ugotovljeno je, da je v ogrodjih veliko napak, saj se medplatformski razvoj šele uveljavlja, napake se odpravi z novejšimi verzijami. Večina napak se pojavi na operacijskem sistemu Android, kar pomeni, da je iOS priljubljenejši med

razvijalci medplatformskih ogrodij ali da je operacijski sistem Andoid kompleksnejši in ga je težje abstrahirati. Poleg tega je medplatformski razvoj v zaostanku za klasičnim razvojem, ker morajo razvijalci uskladiti kodo med več platformami in razviti vmesnike. Če uporabljamo standardne gradnike, nimamo problemov. Ogrodij za medplatformski razvoj je zelo veliko, vendar je priporočljivo uporabljati najbolj uporabljene, ker nam le to zagotavlja dobro podporo z literaturo in veliko verjetnost, da se podpora ogrodju ne bo ukinila.

Slabost medplatformskega razvoja je v velikosti namestitvenega paketa aplikacije. Aplikacija z eno osnovno masko, ki vsebuje gumb in vnosno polje je velika več MB (ang. megabyte), medtem ko je podobna avtohtona aplikacija velika nekaj KB (ang. kilobyte). Velikost pa vpliva na čas prenosa, namestitve in hitrost izvajanja aplikacije, kar pomeni, da je identična avtohtona aplikacija odzivnejša.

Rešitev diplomske naloge je mobilna in spletna aplikacija Potni nalogi. Mobilna aplikacija je razvita z ogrodjem Titanium. Ogrodje je bilo izbrano, ker le-ta podpira operacijska sistema Android in iOS, ki sta bila v specifikacijskih zahtevah. Ogrodje Titanium podpira razvoj avtohtonih mobilnih aplikacij, vsebuje lastno razvojno okolje Titanium Studio, ima veliko podporo z literaturo in je pogosto uporabljen. Medplatformski razvoj je zelo uporaben za razvoj aplikacij, ki niso grafično zahtevne. Zunanji razvijalci poleg tega razvijajo module, ki omogočajo razvoj kompleksnih mobilnih iger in v prihodnosti je pričakovati večjo grafično podporo s strani medplatformskih ogrodij. Za razvoj spletne aplikacije je bilo uporabljeno okolje Visual Studio in ogrodje .NET MVC 4, saj upošteva zadnje arhitekturne smernice in omogoča enostavno namestitev aplikacije v oblak. Za dostop do podatkovne baze je uporabljena tehnologija Entity Framework, katera je priporočena tehnologija za dostop do podatkovne baze za nove aplikacije s strani Microsofta. Izkazalo se je, da je izbira tehnologije Entity Framework zelo dobra, saj omogoča, da se v primeru sprememb dokaj enostavno spremeni podatkovno bazo aplikacije.

Izboljšave raziskave na področju potnih nalogov so možne z zajetjem in analizo plačljih aplikacij potnih nalogov. Rešitev diplomske naloge Potni nalogi se lahko razširi s podporo dnevnic.

# Literatura


[1] S. Allen, V. Graupera, L. Lundrigan, "Pro smartphone cross-platform development: iPhone, Blackberry, Windows Mobile and Android development and distribution", New York: Apress, 2010.

[2] J. Anderson, "Appcelerator Titanium: Up and Running", Kalifornija: O'Reilly Media, Inc., 2013.

[3] M. Baxter Reynolds, "Multimobile Development: Building Applications for the IPhone and Android", New York: Apress, 2010.

[4] D. Cope, "Appcelerator Titanium Application Development By Example", Birmingham: Packt Publishing Ltd., 2013.

[5] J. Galloway, P. Haack, B. Wilson, K. S. Allen, "Professional ASP.NET MVC 4", Indianapolis: John Wiley & Sons, Inc., 2012.

[6] R. Ghatol, Y. Patel, "Beginning PhoneGap", New York: Apress Media, 2012.

[7] Microsoft, "Microsoft application architecture guide", Microsoft Corporation, 2009.

[8] T. Myer, "Beginning PhoneGap", Indiana: John Wiley & Sons, Inc., 2012.

[9] S. Olson, J. Hunder, B. Horgen, K. Goers, "Professional Cross-Platform Mobile Development in C#", Indiana: John Wiley & Sons, Inc., 2012.

[10] J. Palermo, J. Bogard, E. Hexter, M. Hinze, J. Skinner, "ASP.NET MVC 4 in Action", New York: Manning, 2012.

[11] J. Pelletier, "Mobile app manual: The blueprint, How to Start Creating Mobile Apps Using jQuery Mobile and PhoneGap Build", Withinsight, 2013.

[12] B. Pollentine, T. Ward, "Appcelerator Titanium: Patterns and Best Practices", Birmingham: Packt Publishing Ltd., 2013.

[13] T. Poulsen, K. Whinnery, T. Lukasavage, P. Dowsett, 2013, "Building Mobile Applications with Titanium"[e-knjiga]. Dostopno na: http://docs.appcelerator.com/titanium/latest/#!/guide/BNAPP_ebook (avgust 2013).

[14] G. Reese, "Cloud Application Architectures", Kalifornija: O'Reilly Media, Inc., 2009.

[15] K. Roebuck, "Saas - Software as a Service", Emereo Pty Ltd, 2011.

[16] K. Shotts, "Instant PhoneGap Social App Development", Birmingham: Packt Publishing Ltd., 2013.

[17] A. T. Velte, T. J. Velte, R. Elsenpeter, "Cloud computing, A Practical Approach", New York: The McGraw-Hill Companies, 2010.

[18] B. Wilder, "Cloud Architecture Patterns", Kalifornija: O'Reilly Media, Inc., 2012.

[19] (2013) Activities. Dostopno na: http://developer.android.com/guide/components/activities.html.

[20] (2013) Amazon web services. Dostopno na: http://aws.amazon.com.

[21] (2013) Android is crushing Apple and Microsoft in the mobile device market. Dostopno na:

http://www.zdnet.com/android-is-crushing-apple-and-microsoft-in-the-mobile-device-market-7000015206/.

[22] (2013) API Documentation. Dostopno na: http://docs.appcelerator.com/titanium/latest/#!/api.

[23] (2013) App store, Potni Nalog. Dostopno na: https://itunes.apple.com/kr/app/potni-nalog/id620591089.

[24] (2013) Cloud computing: What are IaaS, PaaS & SaaS. Dostopno na: http://cloudcomputingtopics.com/2013/01/cloud-computing-what-are-iaas-paas-saas/.

[25] (2013) Cocoa Touch Frameworks. Dostopno na: https://developer.apple.com/technologies/ios/cocoa-touch.html.

[26] (2013) CommonJS. Dostopno na: http://en.wikipedia.org/wiki/CommonJS.

[27] (2013) CommonJS. Dostopno na: http://wiki.commonjs.org/wiki/CommonJS.

[28] (2013) CommonJS Modules in Titanium. Dostopno na: http://docs.appcelerator.com/titanium/latest/#/guide/CommonJS_Modules_in_Titanium.

[29] (2013) Creating html5 and c/c++ hybrid mobile apps with the mosync mobile sdk. Dostopno na: http://www.straightforward.se/storyserver/html5-javascript-css-c-and-cpp-mobile-apps-using-the-mosync-mobile-sdk.

[30] (2013) Designing for Multiple Screens. Dostopno na: http://developer.android.com/training/multiscreen/index.html.

[31] (2013) Entity Framework. Dostopno na: http://msdn.microsoft.com/en-us/data/ef.aspx.

[32] (2013) Google Cloud Platform. Dostopno na:
https://cloud.google.com/products/compute-engine.

[33] (2013) Google play, Moj Potni Nalog. Dostopno na:
https://play.google.com/store/apps/details?id=si.modrajagoda.potninalogi.

[34] (2013) Google play, PaNdroid - Potni nalogi HiSoft. Dostopno na:
https://play.google.com/store/apps/details?id=pandroid.moj.

[35] (2013) Intents and Intent Filters. Dostopno na:
http://developer.android.com/guide/components/intents-filters.html.

[36] (2013) Introducing Windows Azure. Dostopno na:
http://www.windowsazure.com/en-
us/develop/net/fundamentals/intro-to-windows-azure/.

[37] (2013) iText. Dostopno na:
http://itextpdf.com/.

[38] (2013) Linq. Dostopno na:
http://msdn.microsoft.com/en-us/library/vstudio/bb397926.aspx.

[39] (2013) Memset hosting. Dostopno na:
http://www.memset.com/cloud/compute/.

[40] (2013) Microsoft .NET Framework 4.5. Dostopno na:
http://www.microsoft.com/en-us/download/details.aspx?id=30653.

[41] (2013) Mobile/Tablet Operating System Market Share. Dostopno na:
http://www.netmarketshare.com/operating-system-market-
share.aspx?qprid=8&qpcustomd=1.

[42] (2013) MoSync. Dostopno na:
http://en.wikipedia.org/wiki/MoSync.

[43] (2013) MoSync. Dostopno na:
http://www.mosync.com/.

[44] (2013) Multi-Tenant Data Architecture. Dostopno na: http://msdn.microsoft.com/en-us/library/aa479086.aspx.

[45] (2013) Native App vs. Mobile Web App: A Quick Comparison. Dostopno na: http://sixrevisions.com/mobile/native-app-vs-mobile-web-app-comparison/.

[46] (2013) Native Mobile App. Dostopno na: http://www.techopedia.com/definition/27568/native-mobile-app.

[47] (2013) PhoneGap. Dostopno na: http://phonegap.com/.

[48] (2013) PhoneGap Documentation. Dostopno na: http://docs.phonegap.com/.

[49] (2013) PISARNA.biz - spletne rešitve, Potni nalogi. Dostopno na: https://pisarna.biz/.

[50] (2013) Platform Characteristics. Dostopno na: http://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/Characteristics/Characteristics.html.

[51] (2013) Potni nalog. Dostopno na http://www.potninalog.si/.

[52] (2013) Prerequisites. Dostopno na: http://docs.appcelerator.com/titanium/latest/#!/guide/Prerequisites.

[53] (2013) Resources for MoSync Developers. Dostopno na: http://www.mosync.com/docs/index.html.

[54] (2013) Rhodes. Dostopno na: http://www.motorolasolutions.com/US-EN/RhoMobile+Suite/Rhodes.

[55] (2013) Server and Cloud Platform. Dostopno na: http://www.microsoft.com/en-us/server-cloud/private-cloud/default.aspx.

[56] (2013) Services. Dostopno na: http://developer.android.com/guide/components/services.html.

[57] (2013) Setting up studio. Dostopno na: http://docs.appcelerator.com/titanium/latest/#!/guide/Setting_up_Studio.

[58] (2013) The Cloud View. Dostopno na: http://thecloudview.com/all-about-multi-tenancy-part-1/.

[59] (2013) The Google Directions API. Dostopno na: https://developers.google.com/maps/documentation/directions/.

[60] (2013) Titanium. Dostopno na: http://www.appcelerator.com/.

[61] (2013) Titanium.Cloud. Dostopno na: http://docs.appcelerator.com/titanium/latest/#!/api/Titanium.Cloud.

[62] (2013) Top 5 Tools for Multi-Platform Mobile App Development. Dostopno na: http://mobiledevices.about.com/od/mobileappbasics/tp/Top-5-Tools-Multi-Platform-Mobile-App-Development.htm.

[63] (2013) Using Cloud Services for Building Next Generation Mobile Apps. Dostopno na: https://pages.appcelerator.com/9.11.12WWWWhitepaper-CloudServicesforBuildingNextGenerationMobileApps.html.

[64] (2013) Web Deploy 3.0. Dostopno na: http://www.iis.net/downloads/microsoft/web-deploy.

[65] (2013) 5 Great Tools for Cross-Platform Mobile App Development. Dostopno na: http://www.techyounme.com/5-great-tools-for-cross-platform-mobile-app-development/.