Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matej Zimic

Medplatformski razvoj mobilnih aplikacij

diplomsko delo

visokošolski strokovni študijski program prve stopnje

računalništvo in informatika

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

Ljubljana, 2012

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: 00209/2012
Datum: 02.04.2012

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

Kandidat: **MATEJ ZIMIC**

Naslov: **MEDPLATFORMSKI RAZVOJ MOBILNIH APLIKACIJ**
CROSS-PLATFORM DEVELOPMENT OF MOBILE APPLICATIONS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Preučite orodja za medplatformski razvoj mobilnih aplikacij. Na podlagi argumentacije izberite eno izmed njih za razvoj aplikacije za spremljanje delovanja fotonapetostnih elektrarn, ki so priključene na sistem daljinskega nadzora. Aplikacija mora teči ekvivalentno dobro vsaj na operacijskih sistemih iOS in Android.

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 Matej Zimic, z vpisno številko **63070338**, sem avtor diplomskega dela z naslovom:

Medplatformski razvoj mobilnih aplikacij

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, dne 10. junija 2012 Podpis avtorja:

Diploma je nastala pod mentorstvom doc. dr. Petra Peera in as. Jerneja Buleta, ki se jima iskreno zahvaljujem za vse dragocene nasvete in pripombe.

Za pomoč se zahvaljujem tudi mentorju v podjetju MIEL, d.o.o. mag. Andreju Rotovniku ter sodelavcu Maticu Tovšak.

Posebna zahvala gre Heleni Kosec, Špeli Zimic, Matjažu Tauses, Hani-Tii Tauses, Ani Motnikar, Andreju Jurjevcu, Andreju Bokaliču, Meri Omrzel, Urošu Brdniku, Mateji Novak, Danielu Vrbcu, Aleksandru Petroviču in vsem drugim, ki sem jih nehote izpustil. Hvala za vso vašo pomoč, potrpežljivost in spodbude.

Posebej bi se zahvalil punci Barbari Zemljič, ki mi je stala ob strani v času mojega študija in svetovala pri diplomi.

Svoji dragi Barbari.

Kazalo

Seznam uporabljenih kratic in simbolov

API (*Application Programing Interface*) – vmesnik za komunikacijo med različnimi programskimi deli aplikacije
CSS – Cascading Style Sheets
IDE – Integrated Development Environment
iOS – iPhone Operating System
LUA – hiter in preprost skriptni jezik
NFC – Near Field Communication
MVC – Model-View-Controller
OpenGL ES – Open Graphics Library for Embedded Systems
OS – Operating System
PDA – Personal Digital Assistant
SDK (*Software Development Kit*) – komplet orodij za razvijanje programske opreme
SVG – Scalable Vector Graphics

Povzetek

Hiter tehnološki razvoj pametnih mobilnih naprav povzroča hiter razvoj številnih novih aplikacij za pametne mobilne naprave. Ker deleži proizvajalcev pametnih mobilnih naprav občutno nihajo skozi leta in ker je število operacijskih sistemov nameščenih na pametnih mobilnih naprav vedno večje, so se na trgu pametnih naprav začela pojavljati ogrodja in knjižnice za medplatformski razvoj. V diplomskem delu smo preučili večino trenutno na trgu prisotnih ogrodij in knjižnic. Izpostavili smo tudi prednosti in slabosti izbranih najboljših medplatformskih mobilnih orodij. Cilj diplomskega dela je bila izdelava mobilne aplikacije za prikazovanje podatkov o proizvedeni energiji sončnih elektrarn, v določnem časovnem obdobju, s pomočjo grafičnih prikazov. Mobilna aplikacije je bila izdelana z ogrodjem Appcelerator Titanium, ki predstavlja najboljšo izbiro medplatformskega ogrodja za izdelavo aplikacije. Skozi uporabo ogrodja so se razkrile tudi številne prednosti in pomanjkljivosti rabe ogrodja Appcelerator Titanium.

Ključne besede:

mobilne pametne naprave, medplatformski mobilni razvoj, medplaformska ogrodja in knjižnice, mobilna aplikacija

Abstract

Rapid technological development of smart mobile devices causes rapid development of numerous new applications for smart mobile devices. Since the shares of the manufacturers of smart mobile devices are fluctuating considerably through the years and the number of operating systems installed on smart mobile devices is increasing, several frameworks and libraries for cross-platform mobile development have emerged on the market. This diploma is studying most of the frameworks and libraries which are currently present on the market. We also highlight the advantages and disadvantages of some selected top-rated cross-platform mobile development tools. The goal of this diploma was also development of mobile application for presenting information about the energy output produced by solar power plants, in the certain time and with the help of graphical presentation. Mobile application was developed with Appcelerator Titanium, which represents the best choice for cross-platform mobile development. Through the use of this cross-platform mobile development tool several benefits and deficiencies were revealed.

Keywords:

smart mobile devices, cross-platform mobile development, cross-platform frameworks and libraries, mobile application

Poglavje 1

Uvod

Na trgu so danes prisotne številne pametne mobilne naprave, ki imajo nameščene različne operacijske sisteme različnih proizvajalcev. Vse te mobilne naprave omogočajo uporabo aplikacij, ki si jih uporabniki lahko prenesejo s spletnih trgovin z aplikacijami (*npr. App Store, Google Play, BlackBerry App World, Ovi Store, Windows Phone Marketplace*) in namestijo na svoje mobilne naprave. Razvijanje aplikacij za tako veliko število različnih mobilnih platform je vse prej kot enostavno. Razvijalci morajo biti seznanjeni z razvojem na vsaki platformi posebej, če želijo ponuditi na trg delujočo aplikacijo na vseh mobilnih napravah. Veliko podjetij je zato pričelo razvijati ogrodja, ki omogočajo razvoj mobilnih aplikacij za več platform hkrati.

V prvem poglavju bomo predstavili zgodovino nastanka in razvoj pametnih mobilnih naprav ter razvoj vedno večjega števila operacijskih sistemov za mobilne naprave. Na ta način lahko sklepamo, kaj se lahko dogaja v prihodnosti na trgu pametnih mobilnih naprav, kar je eden ključnih dejavnikov zakaj izbrati medplatformsko ogrodje za razvoj aplikacij. Ogledali si bomo izvoren razvoj aplikacij za nekaj vodilnih platform na trgu. Tako bomo lahko primerjali razvoj aplikacij na izvornen način in razvoj z uporabo medplatformskih ogrodij.

V drugem poglavju si bomo s pomočjo primerjalne tabele ogledali, koliko ogrodij že obstaja na trgu. Podrobneje bo predstavljenih nekaj izbranih medplatformskih ogrodij, izpostavljena pa bo predvsem razlika med ogrodji, s pomočjo katerih razvijalci razvijajo aplikacije z različnimi programskimi jeziki. Razvidno bo tudi, kakšne vrste aplikacij dobimo, če za razvoj izberemo medplatformsko ogrodje in ali te aplikacije omogočajo vse funkcionalnosti, ki jih pametne mobilne naprave ponujajo.

V tretjem poglavju bomo prikazali razvoj mobilne aplikacije "Sončne elektrarne" z medplatformskim ogrodjem Appceletator Titanium. Aplikacija je bila izdelana za podjetje MIEL, d.o.o., ki želi svojim strankam, poleg že obstoječe spletne aplikacije, ponuditi še mobilno aplikacijo. Le ta mora delovati na iOS in Android platformah. Glavni namen mobilne aplikacije je prikazovanje podatkov o proizvedeni energiji v določnem časovnem obdobju s pomočjo grafičnih prikazov.

Definirane so bile tudi dodatne zahteve:

- aplikacija se mora povezovati na strežnik in od tam pridobivati vse podatke;
- prikazani podatki morajo biti konsistentni s podatki prikazanimi na spletni aplikaciji, ki jo podjetje že uporablja;
- uporabnikovo geslo mora biti v kriptirani obliki in shranjeno na strežniku;
- aplikacija mora biti dvojezična (*v slovenskem in angleškem jeziku*);
- prikazati se mora sporočilo ob povezovanju na strežnik;
- prikazati se mora čas zadnje posodobitve podatkov;
- podatki morajo biti prikazani s pomočjo (*dnevnih, mesečnih, letnih*) grafičnih prikazov;
- aplikacija mora biti oblikovno skladna, dinamična in uporabniku prijazna.

V zadnjem poglavju smo predstavili sklepne ugotovitve smotrnosti uporabe medplatformskih ogrodij. Predstavili smo tudi prednosti in pomankljivosti mobilne aplikacije “Sončne elektrarne” ter podali napotke za nadaljni razvoj.

Poglavje 2

Pametna mobilna naprava in operacijski sistemi

2.1 Pametna mobilna naprava

Pametna mobilna naprava [52], [63], [75] je narejena na mobilni računalniški arhitekturi in ima naprednejše računalniške ter povezovalne sposobnosti kot sodobni osnovno-funkcijski telefoni. Največ tovrstnih naprav je bilo na začetku pravzaprav pametnih mobilnih telefonov, vendar je danes vedno več pametnih naprav, ki nimajo funkcije telefona. Pametni telefoni sicer združujejo telefonijo in uporabo interneta v eni sami napravi, vsebujejo pa tudi različne funkcije, kot so prejemanje elektronske pošte, koledar opravil, medijski predvajalnik, digitalno kamero, GPS navigacijske storitve, povezljivost z Wi-Fi omrežjem, kompas, pospeškometer, žiroskop ter seveda še veliko drugih funkcij. Običajno imajo pametne mobilne naprave še zaslon občutljiv na dotik, ki lahko poenostavi uporabo nekaterih opravil (*npr. brskanje po spletnih straneh*). Na pametne mobilne naprave lahko uporabnik po svojih željah namesti številne aplikacije in si napravo prilagodi po svojih potrebah. Aplikacije lahko izdeluje tudi sam oz. s pomočjo strokovne ekipe. Če povzamemo, pametne mobilne naprave danes praviloma ponujajo [72]:

- zmogljiv operacijski sistem,

Slika 2.1: Pametni telefoni različnih proizvajalcev.

- brezžično povezavo,
- popolno podporo elektronski pošti,
- sinhronizacijo s pošto, kontakti in koledarjem,
- zunanjo ali virtualno QWERTY tipkovnico,
- podporo različnim dokumentom,
- na dotik občutljiv zaslon,
- medijski prevajalnik,
- digitalno kamero in fotoaparat,
- senzorje kot so GPS navigacijska naprava, kompas, žiroskop, ipd.

Na sliki 2.1 so prikazane različne pametne mobilne naprave.

V pametnih mobilnih napravah so nameščeni različni operacijski sistemi različnih proizvajalcev, kar je ključno pri razvoju aplikacij za mobilne naprave. Najbolj razširjeni operacijski sistemi so Android podjetja Google, iOS podjetja Apple, Windows Phone in Windows Mobile podjetja Microsoft, Symbian podjetja Symbian v povezavi z Nokio, BlackBerry OS podjetja RIM (*Research In Motion*), Maemo podjetja Nokia, Bada OS podjetja Samsung, webOS podjetja Palm ter MeeGo, ki je nastal s skupnim sodelovanjem podjetij Nokia, Intel in Linux fundacijo.

Slika 2.2: Prvi pametni telefon podjeta IBM, imenovan Simon.

2.1.1 Zgodovina pametnih mobilnih naprav

Razvoj pametnih telefonov je relativno kratek in je pravzaprav posledica združitve funkcij, ki jih ponujajo dlančniki in mobilni telefoni. Zgodovina pametnih mobilnih naprav oz. telefonov [34], [72] sega v leto 1992, ko je podjetje IBM predstavilo prvi pametni telefon, imenovan Simon. Skupaj z običajnimi funkcijami mobilnega telefona je vseboval še koledar, imenik, svetovno uro, računalo, beležko, branje elektronske pošte in možnost prejemanja ter pošiljanja sporočil preko telefaksa. Simon, prikazan na sliki 2.2 , je imel zaslon občutljiv na dotik, s pomočjo katerega je bilo mogoče s prsti ali s posebnim pisalom dodajati in urejati tekst.

V letu 1996 je podjetje Nokia predstavilo Nokia komunikator, pametni telefon z imenom Nokia 9000. Vseboval je fizično QWERTY tipkovnico in zaslon z ločljivostjo 640 x 200 slikovnih točk, uporabljal je operacijski sistem GEOS V3.0. Kasneje sta sledili še Nokia 9110 (*leta 1998*) in Nokia 9110i (*leta 2000*) z izboljšano podporo spletnega brskanja.

Pametnim telefonom se je leta 2000 pridružil R380, podjetja Ericsson, ki je prvi uporabil operacijski sistem Symbian. Združeval je lastnosti mobil-

nega telefona in dlančnika (*angl. Personal Digital Assistant, krajše PDA*). Dve leti kasneje se mu je pridružil model P800, ki je prvi imel tudi kamero. Sočasno z modelom P800 je na področje pametnih telefonov vstopil še Microsoft z operacijskim sistemom Windows CE, iz katerega se je kasneje razvil Windows Mobile za pametne telefone. To je bil prvi operacijski sistem za mobilne naprave, ki je bil namenjen telefonom različnih proizvajalcev.

BlackBerry je na trg pametnih telefonov vstopil v letu 2002 in je prvi uporabljal brezžična omrežja za spletno pošto. Tri leta kasneje je Nokia predstavila pametni telefon Nokia N95 z GPS navigacijsko napravo, kamero s samodejnim fokusom in bliskavico, 3G omrežjem in brezžično povezavo.

Prelomni leti na področju razvoja mobilnih naprav predstavljata leto 2007, ko je podjetje Apple predstavilo pametni telefon iPhone z iOS, in leto 2008, ko je izšel odprtokodni operacijski sistem [1] Android podjetja Google. Tako iOS kot Android sta danes vodilna mobilna operacijska sistema na svetovnih trgih.

2.2 Operacijski sistemi za mobilne naprave

Operacijski sistem je sistem, ki upravlja pametno mobilno napravo. Moderni operacijski sistemi [34] združujejo lastnosti osebnega računalnika z lastnostmi mobilne naprave in dlančnika. Upravljajo s funkcijami kot so zaslon na dotik, modri zob, brezžična povezava WiFi, GPS navigacijska naprava, kamera, NFC, prepoznavanje obraza in glasu, predvajalnik za glasbo in video ter druge.

[1] Kultura odprtokodnega razvoja je prodrla na trg pametnih telefonov na več načinov. Izvajali so poskuse, da bi bili odprtokodni tako programska kot tudi strojna porema. Omembe vreden odprtokodni projekt je Neo FreeRunner pametni telefon razvit pri podjetju Openmoko. Zadnje čase je med najbolj popularnimi odprtokodnimi operacijskimi sistemi Googlov Android OS.

Do danes je znanih že veliko operacijskih sistemov različnih proizvajalcev. Najprej si oglejmo nekaj prelomnih trenutkov v zgodovini razvoja mobilnih operacijskih sistemov, ki hkrati nakazujejo možen razvoj operacijskih sistemov tudi v prihodnosti.

Leta 1996 je bil predstavljen Palm Pilot 1000 PDA s Palm mobilnim OS, istega leta pa podjetje Microsoft predstavi prvo napravo z Windows CE operacijskim sistemom. Štiri leta kasneje Symbian postane prvi moderni mobilni OS, še dve leti kasneje pa se pojavi na trgu že prvi pametni mobilni telefon podjetja Microsoft z Windows CE OS, ki se je imenoval žepni računalnik (*angl. Pocket PC*). V letu 2002 je podjetje RIM predstavilo svoj prvi pametni telefon z BlackBerry mobilnim OS. Podjetje Nokia je leta 2005 predstavilo Maemo mobilni OS na napravi N770, podjetje Apple pa leta 2007 iOS na mobilnem telefonu iPhone. Podjetje Google je naslednje leto predstavilo Android mobilni OS na napravi HTC Dream kot prvi Android mobilni telefon. Razvoj na področju operacijskih sistemov za mobilne naprave je leta 2009 nadaljevalo podjetje Palm s predstavitvijo webOS na napravi Palm Pre. Istega leta je podjetje Samsung predstavilo Bada OS na napravi Samsung S8500. Leto kasneje je bil predstavljen Windows Phone mobilni OS, ki pa ni združljiv s prejšnjim Windows Mobile OS. Leta 2011 je s skupnim sodelovanjem podjetij Nokia, Intel in Linux fundacijo izšel prvi mobilni Linux (*MeeGo mobilni OS*), predstavljen s telefonom Nokia N9. V letošnjem letu pa je podjetje Lenovo predstavilo napravo Lenovo K800, ki je prvi Intelov pametni telefon z Android OS.

Na razvoj aplikacij oziroma na odločitev o tem, za katero platformo [2] bi izdelali aplikacijo, močno vpliva tudi trenutni tržni delež mobilnega operacijskega sistema na trgu. Cilj medplatformskega razvoja je stremenje k temu,

[2]Platforma je združitev programske in strojne opreme v celoto. To definira ustreznost programske in strojne nadgradnje na tem sistemu oz., če želimo platformo nadgraditi s programsko ali strojno opremo, mora biti le-ta združljiva z platformo.

da bi izdelali aplikacijo za čim večje število platform. Veliko ogrodij ponuja podporo za samo nekaj mobilnih platform, zato je pomemben tudi zgodovinski vpogled v podatke oz. trend razdelitev deležev mobilnih OS na trgu. Na ta način lahko deloma predvidimo prihodnost in izberemo »pravilno« medplatformsko orodje.

Statistični podatki o številu prodanih pametnih mobilnih telefonov kažejo, da število le-teh hitro narašča. Leta 2006, ko Android, iOS, Windows Phone in Bada mobilnih operacijskih sistemov še ni bilo, je bilo prodanih samo 64 milijonov pametnih telefonov. Danes je ta številka tudi desetkrat večja. Po deležih na trgu so vodilni Android, iOS, Symbian, BlackBerry, Windows Phone, MeeGo in Bada. V tabeli 2.1 [34] je prikazan trend spreminjanja prodanih deležev mobilnih telefonov ob prihodu novih operacijskih sistemov na trg. V letu 2012 je delež Symbian OS močno padel, medtem ko se je delež iOS občutno povečal, vendar pa Android kljub vsemu zavzema vodilno mesto.

2.2.1 Izvorni razvoj aplikacij

V tem poglavju si bomo ogledali prednosti in slabosti izvornih aplikacij ter njihov potek razvoja za nekaj vodilnih platform na trgu. Predstavljene so tudi nekatere značilnosti in funkcionalnosti, ki jih obravnavana platforma ponuja, ter orodja, ki jih uporabljamo pri razvoju aplikacij.

Izvorne aplikacije (*angl. Native App*) so tiste, katerih razvoj poteka ločeno za vsako platformo posebej. Razvoj lahko poteka z uporabo različnih jezikov, kot je to prikazano v tabeli 2.2.

Izvorni pristop običajno izberemo, ko potrebujemo delujočo aplikacijo brez internetne povezave, bogato grafično podporo in dostop do funkcionalnosti naprave. Oglejmo si prednosti in slabosti razvoja izvornih mobilnih aplikacij.

četrtletje	Android	iOS	Symbian	RIM	Bada	Windows Phone	drugi
1. četrtletje 2012	56,1%	22,9%	8,6%	6,9%	2,7%	1,9%	0,9%
4. četrtletje 2011	50,9%	23,9%	11,7%	8,8%	2,1%	1,9%	0,8%
3. četrtletje 2011	52,5%	15,0%	16,9%	11,0%	2,2%	1,5%	0,9%
2. četrtletje 2011	43,4%	18,2%	22,1%	11,7%	1,9%	1,6%	1,0%
1. četrtletje 2011	36,0%	16,8%	27,4%	12,9%	1,7%	3,6%	1,6%
4. četrtletje 2010	31,1%	16,1%	32,9%	13,1%	1,3%	3,4%	2,2%
3. četrtletje 2010	25,3%	16,6%	36,3%	15,4%	1,1%	2,8%	2,5%
2. četrtletje 2010	17,2%	14,2%	41,2%	18,2%	0,9%	5,0%	3,3%
1. četrtletje 2010	9,6%	15,3%	44,2%	19,7%		6,8%	4,4%
4. četrtletje 2009	7,6%	16,2%	44,7%	19,7%		7,9%	4,0%
3. četrtletje 2009	3,4%	17,0%	44,2%	20,5%		7,9%	7,0%
2. četrtletje 2009	1,8%	13,0%	51,0%	19,0%		9,3%	5,9%
1. četrtletje 2009	1,6%	10,5%	48,8%	20,6%		10,2%	8,2%
4. četrtletje 2008	1,1%	10,6%	46,5%	19,3%		12,2%	9,1%
3. četrtletje 2008	0,6%	13,1%	50,3%	16,1%		11,2%	9,8%
2. četrtletje 2008		2,8%	57,5%	17,5%		12,1%	10,8%
1. četrtletje 2008		4,6%	49,5%	11,6%		10,4%	11,6%
4. četrtletje 2007		5,2%	62,3%	10,9%		11,9%	9,6%
3. četrtletje 2007		3,4%	63,1%	9,7%		12,8%	11,5%
2. četrtletje 2007		1,0%	65,6%	8,9%		11,5%	13,0%
1. četrtletje 2007			61,2%	8,7%		13,4%	16,8%

Tabela 2.1: Deleži prodanih pametnih telefonov po operacijskih sistemih za mobilne naprave.

Operacijski sistem	Programski jezik
iOS	C, C++, Objective-C
Android	C, C++, Java
Windows Mobile	C++, .NET
Windows Phone	.NET
BlackBerry	Java
Symbian	C++
Maemo	C, C++
MeeGo	C++
Bada	C++

Tabela 2.2: Operacijski sistemi in programski jeziki.

Prednosti izvornih aplikacij so naslednje:

- Izvoren pristop je najprimernejši, če aplikacija zahteva napredno grafično podporo.
- Izvorne aplikacije omogočajo polni dostop do vseh funkcij, ki so v napravi: npr. pospeškometer, GPS navigacijske storitve, kamera, kompas, video in zvočni snemalnik, izvorna obvestila, sporočila in drugo.
- Aplikacija je hitro odzivna.
- Izvorne aplikacije omogočajo nemoteno delo brez internetne povezave (*govora je predvsem o izgledu aplikacije; če aplikacija še vedno potrebuje podatke s skupnega strežnika, obvesti uporabnika, da so podatki stari*). Vsa poslovna logika je zapisana v aplikaciji, ob ponovni vzpostavitvi z internetom pa aplikacija le osveži ali podatke sinhronizira, če je to potrebno.
- Aplikacijo lahko prenesemo na spletno trgovino z aplikacijami (*App Store, Google Play, BlackBerry App World, Ovi Store, Windows Phone Marketplace*), vendar le na spletno trgovino platforme, za katero smo aplikacijo razvili.

Slabosti:

- Izdelava aplikacij na izvoren način občutno poveča stroške izdelave.
- Običajno potrebujemo večje število razvijalcev, če želimo aplikacijo delujočo na več platformah.
- Aplikacijo je potrebno ponovno izdelati za vse platforme tudi pri manjših spremembah in jo kot nadgradnjo znova ponuditi v spletni trgovini. Uporabniki morajo dovoliti nadgradnjo aplikacije.
- Razvijalec mora biti pozoren na verzije mobilnih operacijskih sistemov na trgu in ali jih aplikacija podpira. Uporabniki lahko nadgradijo svoj operacijski sistem v prenosni napravi, tako pa je lahko pravilno delovanje že nameščene aplikacije onemogočeno.

2.2.1.1 Android

Android OS [7], [63], [64], [66] je odprtokodni projekt (*z Apache licenco*) in je zgrajen na Linux verziji 2.6. Ponuja številne funkcionalnosti (*npr. podpora medijskim vsebinam: zvok, video vsebine, razni formati slik, NFC, barometer, GPS navigacijski sistem, ipd.*). 2D in 3D grafika je podprta z OpenGL ES 2.0 API. Visokokakovostne so tudi animacije, ki so praviloma del aplikacij. Android mobilni OS podpira tudi večtočkovno zaznavanje dotikov na zaslonu naprave. Spletni brskalnik je grajen na močnem WebKit poganjalniku in vsebuje Chromov V8 JavaScript poganjalnik.

Večopravilnost je podprta in realizirana kot strukturiranje aktivnosti (*angl. Activity [7]*) aplikacij. Aktivnost ima pomembno vizualno predstavitev in mora biti namenjena enemu opravilu (*npr. zajemanje slike s fotoaparatom ali brskanju po imeniku*). Vsaka aplikacija vsebuje vsaj eno aktivnost, zahtevnejše aplikacije pa jih lahko vsebujejo tudi več.

Zahteve za razvoj aplikacij

Za razvijanje Android aplikacij lahko uporabimo okolje Windows, Linux ali Mac OS. Aplikacije so praviloma napisane z Java programskim jezikom, vendar Java virtualni pogon na napravah ni nameščen. Koda je prevedena v Dalvik bitno kodo in teče na Dalvik virtualnem pogonu. Android ne podpira J2ME platforme. Z objavo Android NDK (*angl. Native Development Kit*) razvijalci lahko pišejo svoje izvorne knjižnice [3] v jeziku C in C++ in na ta način zmanjšajo obseg obstoječe kode ali pa pridobijo na učinkovitosti.

[3]Knjižnica (*ang. library*) ali tudi programska knjižnica je v računalništvu zbirka podprogramov oz. funkcij, ki so v pomoč pri razvoju programske opreme. Vsebujejo razrede, objekte in podatke, ki jih lahko uporabimo v neodvisnih programih. Nekatere knjižnice so lahko tudi izvršljiv program. Ob klicu funkcije iz programa, povezovalnik (*angl. Linker*) samodejno pogleda, če funkcija obstaja v knjižnicah in izvrši kodo.

Najbolj priporočljiv IDE je Eclipse [18] z Android SDK. Preko okolja Eclipse si je potrebno namestiti še AVD Manager, ki omogoča kreiranje različnih emulatorjev in Android SDK Manager, s katerim lahko namestimo želeni SDK.

Razvoj aplikacije

Možnost kreiranja novega projekta izberemo, ko imamo pripravljeno okolje za delo v Eclipse IDE. Vpišemo ime projekta, minimalno verzijo Android OS, na katerem aplikacija lahko teče, in ime paketa za identifikacijo projekta. Tako narejen projekt oz. aplikacijo lahko poženemo na emulatorju ali mobilni napravi. Aplikacija sicer prikazuje samo začetno sporočilo na prvem oknu aplikacije, vendar na ta način preverimo njeno delovanje. Z nekaj manjšimi spremembami v kodi posameznega posamezne aktivnosti je lahko uporabniški vmesnik aplikacije prikazan preko izvornega brskalnika v WebView gradniku. Na ta način je mogoča uporaba HTML jezika za prikaz vsebine, kar lahko bistveno olajša delo. Tak način izdelave aplikacij s pridom izkoriščajo medplatformska ogrodja, kar bo prikazano v poglavju 3.

Google Play

Podjetje Google za distribucijo aplikacij nudi spletno trgovino Google Play (*včasih Android Market*). Preden aplikacijo naložimo v spletno trgovino jo je potrebno digitalno podpisati z zasebnim ključem. Ključ lahko generiramo sami z uporabo Keytool orodja. Eclipsov ADT dodatek ta postopek zelo olajša. Čarovnik vodi do podpisa aplikacije ter celotnega postopka za njeno izdajo.

Aplikacijo lahko brez podpisa ponudimo uporabnikom tako, da ponudimo **.apk** datoteko, ki si jo namestijo na pametne mobilne naprave.

Plačljivega računa za razvoj na Android platformi ni, plačljiv (*25$*) je samo prenos aplikacije na Google Play.

2.2.1.2 iOS

iOS [11], [21], [63], [64], [68] je mobilni operacijski sistem podjetja Apple. Izdan je bil leta 2007 in ga ni mogoče licenčno namestiti na mobilnih napravah, ki niso izdelek tega podjetja Apple. Marca 2012 je spletna trgovina Apple App Store ponujala že več kot 550.000 iOS aplikacij, kar je največje število aplikacij na trgu v primerjavi z ostalimi ponudniki. Uporabniški vmesnik aplikacije uporablja večtočkovno zaznavanje dotikov na ekranu naprave in uporablja manjše število fizičnih gumbov kot druge mobilne naprave. iOS podpira in vsebuje napredne knjižnice za delo z 2D in 3D grafiko. Predvsem za izdelavo iger uporablja grafično knjižnico OpenGL ES, ki ima enostaven API. Za risanje 2D elementov imajo razvijalci na voljo Quartz 2D grafični vmesnik, ki omogoča tudi prikazovanje pdf dokumentov.

Zahteve za razvoj aplikacij

Za razvijanje iPhone, iPod Touch, iPad aplikacij potrebujemo računalnik z Intel procesorjem in nameščenim OS X 10.5.7 (*ali novejšo*) verzijo operacijskega sistema. Potrebna je še zadnja verzija iPhone SDK, ki vsebuje Xcode IDE, simulator in veliko različnih orodij za razvoj aplikacij. Ta orodja so v pomoč pri razvoju aplikacije in omogočajo poganjanje le-teh na simulatorju. Pri razvoju aplikacij je potrebno upoštevati tudi nekaj standardov. MVC je način ločevanja kode od poslovne logike in uporabniškega vmesnika aplikacije. Uporabniški vmesnik oz. izgled aplikacije je običajno izdelan z Interface Builder-jem. Za razvoj uporabljamo programski jezik Objective-C, vendar Xcode podpira tudi druge jezike (*kot so C, C++, Fortran, Java, Python, Ruby, ipd.*).

Razvoj aplikacije

V Xcode izberemo možnost za izdelavo novega projekta. Vpišemo ime aplikacije, ime identifikacijskega projekta in izberemo vrsto iOS naprave, na kateri bo aplikacija tekla. Izberemo eno od že pripravljenih predlog za izgled aplikacije, npr. View-based Application. Naš projekt je pripravljen za testni pogon na simulatorju. Če želimo poganjati aplikacije na fizični napravi, potrebujemo razvijalski račun, ki ni brezplačen (*99$*). Tako narejena aplikacija ob zagonu prikaže prazen zaslon, kar je kljub vsemu dovolj, da preverimo, ali so vse zahteve za izdelovanje aplikacij izpolnjene.

App Store

Podjetje Apple ponuja spletno trgovino za aplikacije App Store, če želimo aplikacijo ponuditi tudi drugim uporabnikom. Za prenos aplikacije na App Store moramo imeti odprt razvijalski račun, ki je plačljiv.

2.2.1.3 BlackBerry

BlackBerry OS [16], [63] je izdelek podjetja RIM. Prvi BlackBerry telefon je z optimiziranim načinom za prejemanje elektronske pošte in tipkovnico QWERTY postal uporaben predvsem v poslovnem svetu. Spletni brskalnik, ki je v napravah BlackBerry, se sooča z določenimi omejitvami (*npr. prikazovanje naprednejših HTML5 lastnosti in SVG slik*), vendar podjetje RIM pričakuje, da bodo odpravili večino omejitev z naslednjo izdajo OS in z uporabo brskalnika na osnovi WebKit. 3D grafična podpora je realizirana z uporabo knjižnice OpenGL ES, ki je vključena v BlackBerry SDK, in omogoča izgradnjo bogatih 3D vsebin.

Posebnost pri BlackBerry platformi je BES (*angl. BlackBerry Enterprise Server*), ki zagotavlja napredne funkcionalnosti. BES omogoča obnavljanje in posodabljanje aplikacij, nastavljanje pravic, pomembne funkcionalnosti pa

so tudi sinhronizacija elektronske pošte, koledarja, stikov in drugih opravil s funkcijo potisnih sporočil ob brezžični WiFi povezavi. BES je tudi najpomembnejši razlog, da je BlackBerry močno uveljavljen mobilni OS v poslovnem svetu.

Zahteve za razvoj aplikacij

BlackBerry podpira različne načine izdelave izvornih aplikacij. Spletni razvoj je novejši način, ki ga z uporabo Widget SDK ponuja podjetje RIM. BlackBerry Widgets so manjše samostojne spletne aplikacije, ki uporabljajo jezike HTML, CSS in JavaScript. Razvoj z jezikom Java je standarden način izdelovanja aplikacij za BlackBerry platformo. Java uporablja MIDP 2.0, CLDC 1.1 in RIM-ov API. Kljub temu, da so orodja razvita v jeziku Java, lahko razvijamo aplikacije samo na Windows 32 bitnem OS. Razvoj aplikacij na Mac OS računalnikih je sicer mogoč z namestitvijo Windows OS znotraj virtualnega pogona.

Za razvoj potrebujemo Sun JDK (*angl. Java Development Kit*), ki vključuje JRE (*angl. Java Runtime Enviroment*). Razvojno orodje je Eclipse, ki je popularno orodje za razvijanje Java aplikacij, potreben pa je tudi BlackBerry dodatek za Eclipse in BlackBerry JDE. JDE mora biti skladen s ciljno verzijo platforme, za katero razvijalec razvija aplikacijo.

Orodja za razvoj so brezplačna.

Razvoj aplikacij

V Eclipse IDE izberemo nov BlackBerry projekt, vpišemo ime aplikacije, izberemo minimalno verzijo JDE, na kateri želimo, da aplikacija teče. Za uporabniški vmesnik je potrebno izdelati še poseben razred. Objekt razreda kličemo, ko želimo prikazati uporabniški vmesnik. Koda je prikazana v na-

daljevanju.

```
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;

public class HelloWorld extends UiApplication {
   public static void main(String []args){
      HelloWorld theApp = new HelloWorld();
      theApp.enterEventDispatcher();
   }
   public HelloWorld (){
      pushScreen (new HelloWorldScreen());
   }
}

class HelloWorldScreen extends MainScreen{
   public HelloWorldScreen(){
      super();
      LabelField title = new LabelField("XPlatform Dev");
      setTitle(title);
      add(new RichTextField("Hello World!"));
   }
   public boolean onClose(){
      System.exit(0);
      return true;
   }
}
```

Tako zapisana aplikacija je pripravljena za zagon na simulatorju ali mobilni napravi. Aplikacija ob zagonu prikaže samo pozdravno sporočilo, kar kljub vsemu omogoča preverjanje, ali so vse zahteve za izdelovanje aplikacij na BlackBerry platformi izpolnjene.

BlackBerry App World

Podjetje RIM nudi spletno trgovino za aplikacije BlackBerry App World.

Za prenos aplikacije na App World se moramo registrirati na Vendor portal in pridobiti certifikat, s katerim podpišemo aplikacijo. Za prenos aplikacije na BlackBerry App World potrebujemo račun, ki je plačljiv (*200$ za 10 aplikacij*).

2.2.1.4 Windows Mobile

Windows Mobile operacijski sistem [59], [63] je izdelek podjetja Microsoft in zagotavlja uporabniško izkušnjo podobno tisti na osebnih računalnikih. Grajen je na osnovi Windows CE jedra. Zadnja verzija je Windows Phone 7, ki pa ne nadgrajuje verzije 6.5 in drugih predhodnih verzij. 3D podpora je realizirana z uporabo XNA ogrodja 4.0.

Zahteve za razvoj aplikacij

Za razvoj aplikacij potrebujemo Microsoft Windows operacijski sistem. Razvoj Windows Phone 7 aplikacij je podprt samo na operacijskem sistemu Windows Vista in Windows 7. Razvojno orodje je Microsoft Visual Studio Profesional IDE. Express verzija Visual Studia ni dobra, saj ni kompatibilna z Microsoft Mobile Development Tool Kit. Pozorni moramo biti tudi pri verziji Visual Studia, saj novejša verzija z leta 2010 ne podpira razvoja aplikacij za starejše verzije (*Windows Mobile 6.5 in starejše*). Torej, če želimo razvijati za cel spekter Windows Mobile naprav, moramo imeti nameščeni verziji z leta 2008 in 2010. Razvijalci uporabljajo programski jezik C++ in C# z .NET ogrodjem. Potrebujemo tudi verzijo Microsoft Windows Mobile Professional SDK, na kateri želimo, da aplikacija teče, in Microsoft Mobile Active Sync ali Windows Vista Mobile Device Center, da lahko aplikacijo prenesemo na mobilno napravo.

Razvoj aplikacij

Za izdelavo aplikacije zaženemo Visual Studio IDE ter izberemo nov Smart

Device projekt. Projektu moramo dodati ime ter izbrati minimalno verzijo SDK naprave, na kateri bo aplikacija lahko tekla. Izbrati moramo tudi verzijo .NET CF ogrodja ter vrsto aplikacije. Z Visual Studio-m je izdelava aplikacij relativno preprosta, saj z načinom "povleci in spusti" (*angl. drag & drop*) dodajamo različne objekte v okno aplikacije.

Windows Marketplace

Microsoft nudi spletno trgovino za aplikacije Windows Marketplace. V preteklosti je bilo mogoče prenesti na Marketplace vsako aplikacijo, z razvojem verzije Windows Phone 7 pa aplikacije predhodno potrjuje Microsoft. Za prenos aplikacije na Windows Marketplace potrebujemo razvijalski račun, ki je plačljiv (*99$*).

Poglavje 3

Medplatformski razvoj mobilnih aplikacij

3.1 Zakaj medplatformski razvoj mobilnih aplikacij?

Besedna zveza mobilna aplikacija je že tako pomembna, da je dobila svoje mesto v slovarju, pomeni pa majhen specializiran program, ki si ga uporabniki sami naložijo z interneta in namestijo v mobilno napravo. Mobilna aplikacija je izdelek programske rešitve razvijalca oz. razvojne ekipe za nek določen namen [63], [59].

Na trgu obstaja veliko pametnih mobilnih naprav in statistični podatki (*za leto 2012*) kažejo, da njihova uporaba še narašča [34]. Trenutno je med vsemi mobilnimi telefoni kar četrtina pametnih mobilnih telefonov. Ob tem se pojavlja tudi vedno večja potreba po razvoju številnih aplikacij za pametne mobilne naprave. Na trgu obstajajo tudi številna podjetja, ki izdelujejo pametne mobilne naprave z nameščenimi lastnimi operacijskimi sistemi. Zastavlja se vprašanje, kako izdelati aplikacijo, ki lahko teče na čim večjem številu mobilnih naprav in s tem zadovoljiti veliko večino njihovih uporabnikov. Razvoj mora biti ob tem hiter in čim cenejši.

Razvoj aplikacije je lahko za vsako platformo drugačen, saj si operacijski sistemi niso enaki. Dodatno delo otežuje tudi uporaba različnih programskih jezikov. V tem primeru se morajo razvijalci praviloma naučiti večjega števila programskih jezikov in postopkov izdelave aplikacije na določeni platformi, uporabe njihovih SDK-jev in podobno. Razvijalec ali razvojna ekipa se lahko odloči in aplikacijo razvije za vsak operacijski sistem ločeno. To je včasih težko, saj ni vedno na razpolago razvijalca, ki bi znal razvijati aplikacije za vsak izbran operacijski sistem, hkrati pa je morebitno povečanje števila razvijalcev finančno prevelik zalogaj. Težave lahko nastanejo tudi zaradi sprememb deležev operacijskih sistemov na trgu, ki se skozi leta hitro spreminjajo. Trenutno (*glej Tabela 2.1: Deleži prodanih pametnih telefonov po operacijskih sistemih za mobilne naprave*) je na vodilnem mestu operacijski sistem Android podjetja Google z 46.9% deležem vseh pametnih mobilnih naprav na tržišču. Na drugem mestu je iPhone operacijski sistem (*iOS*) podjetja Apple z 28.7% deležem. Sledijo mu RIM z 16.6% deležem, Microsoft z 5.2% deležem in Symbian z 1.5% deležem vseh pametnih mobilnih naprav na tržišču. Za primer si lahko ogledamo podatke za leto 2008, ko je operacijski sistem Android šele prihajal na trg. Medtem ko so napovedovali vzpon operacijskima sistemoma Windows in iOS, se je izkazalo, da se je tržni delež zviševal slednjemu. Ne smemo tudi pozabiti, kaj se je dogajalo s Symbian OS, katerega tržni delež se je zaradi počasnega prilagajanja razmeram na tržišču skokovito zmanjšal. Izbira operacijskega sistema za razvoj aplikacije, ki bi hkrati zadostil tudi potrebam po čim večjem deležu uporabnikov pametnih mobilnih telefonov, je zato postala tvegana odločitev.

Potrebe in prihod vedno več operacijskih sistemov na trg so narekovale razvoj medplatformskih ogrodij za razvoj mobilnih aplikacij. Ogrodja bi omogočala razvoj aplikacij za vse platforme skupaj. Cilj medplatformskih orodij je enkraten razvoj aplikacije, ki jo je mogoče prevesti in namestiti na različne pametne mobilne telefone z različnimi operacijskimi sistemi. Medplatformaska

ogrodja razvijajo številni proizvajalci, zato število medplatformskih orodij danes hitro narašča. Orodja za medplatformski razvoj so si glede značilnosti precej različna. V tabeli 3.1 so po različnih karakteristikah predstavljena številna medplatformska ogrodja, med drugim tudi nekatera, ki po definiciji niso medplatformska, saj podpirajo razvoj samo za eno mobilno platformo.

Primerjava medplatformskih ogrodji je razkrila, da se ogrodja medsebojno razlikujejo po številnih kriterijih. Aplikacije narejene z medplatformskimi ogrodji Rhodes, PhoneGap, MoSync, Mono in Appcelerator Titanium so po kakovosti zelo podobne izvornim aplikacijam, večina teh ogrodij pa je brezplačnih. Dokumentacija izbranih medplatformskih ogrodij je dobra in jasno prikazuje, kakšne aplikacije je mogoče izdelati. Podjetja ponujajo dobro podporo razvijalcem (*običajno za plačilo*), vendar so vprašanja in odgovori objavljeni na javno dostopnih spletnih forumih. Prav zaradi teh lastnosti smo jih izbrali in jih podrobneje opisali v poglavju 3.2.

3.2 Podrobnejši opis izbranih medplatformskih ogrodij

3.2.1 Rhodes

Rhodes, podjetja RhoMobile, je medplatformsko ogrodje, ki se je pojavilo na trgu leta 2008 [4], [5], [47], [48], [63]. Je eno starejših medplatformskih ogrodij za razvoj mobilnih aplikacij in hkrati, po mnenju mnogih razvijalcev, tudi eno najboljših. Kot bo v nadaljevanju predstavljeno, mora biti izbira ogrodja Rhodes premišljena.

Rhodes podpira razvoj aplikacij za več mobilnih platform (*iOS, Android, Windows Mobile, BlackBerry in Symbian*). Razvijalci za razvoj aplikacije uporabljajo programske jezike HTML, CSS, JavaScript in Ruby. Programski jezik Ruby je lahko razlog, da se za izbiro Rhodes ogrodja ne odločimo, saj

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIKI	VRSTE APLIKACIJ	NE PODPIRA FUNKCIONAL-NOSTI NAPRAVE	CENA	KRATEK OPIS
Adobe Flex [6]	Android, iOS, BlackBarry	ogrodje	ActionScript, C++, HTML, CSS, JavaScript	izvorna aplikacija, aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	NFC, uporaba imenika, uporaba kamere, prikaz reklam, prikaz obvestil	plačljivo	Brezplačni SDK. Doplačilo knjižnice za grafične prikaze 800€. Če aplikacijo želimo prodajati, je plačljiva. IDE Adobe Flash Builder (cena 250$), ampak je mogoče razvijati v drugem IDE. Aplikacijo naredimo na podoben način kot izvorno. V IDE je vgrajenih več simulatorjev za različne naprave.
Appcelerator Titanium [4], [5], [10], [68], [71]	Android, iOS, BlackBerry (plan)	ogrodje	JavaScript, HTML, CSS	izvorna aplikacija, aplikacija, ki teče v izvornem WebView, hibridna aplikacija		brezplačno	IDE Titanium Studio. Titanium SDK povezan z Android in iOS SDK. Razvoj izvornih aplikacij z JavaScript in uporabo Titanium SDK. Plačljivi so dodatni moduli, ki lahko olajšajo delo (npr. modul za delo z grafi).
App Inventor [8], [73], [74]	Android	ogrodje	brez kodiranja	aplikacija, ki teče v izvornem WebView	NFC, uporabe kamere, imenika	brezplačno	Potrebujemo Google račun. App Inventor Designer IDE. Programiramo s »povleci in spusti« načinom, enako se izdela logika aplikacije. Podobno kot Visual Basic za Android. Podpira le platforme z Java podporo.
app.cat [9]	iOS, Android	ogrodje	brez kodiranja	aplikacija, ki teče v izvornem WebView	ne podpira nobene funkcionalnosti naprave	plačljivo	IDE App Studio. Plačljivo 70$ na aplikacijo in razvijalca. Povleci in spusti način kodiranja. Aplikacija je na koncu kot vizitka. Ena predloge za aplikacije.

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONAL-NOSTI NAPRAVE	CENA	KRATEK OPIS
Application Craft [13]	Vse platforme, razen Maemo. Možna tudi izdelava namiznih aplikacij.	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	kompas in opozarjanje s tresenjem	brezplačno	Doplačilo za številne funkcionalnosti ogrodja (izbris pozdravne reklame, podpora skupnosti, povezava s PhoneGap, uporaba naprednejših pripomočkov, poljubno ime domene, ...) = 35$ na mesec. Application Craft IDE. Potrebujemo povezavo z PhoneGap za delovanje izvornih funkcionalnosti. Narejeno aplikacijo storitev v oblaku prevede v aplikacije za različne mobilne naprave.
Application Markup Language [12]	Android, iOS (plan), WP7 (plan)	knjižnica	izvorni jezik AML	izvorna aplikacija, aplikacija, ki teče v izvornem WebView, hibridna aplikacija	podpira malo izvornih funkcij	brezplačno	Podobno kot pri PhoneGap, naredimo aplikacijo v izvornem jeziku in po želji uporabljamo knjižnico. Uporaba knjižnice je brezplačna, vendar ni uporabna za izris grafičnih prikazov (uporabiti je potrebno drugačen pristop). Uporabimo jezik AML, ki je podoben XML. Še vedno je potrebno znanje programiranja v izvornem jeziku.

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONAL-NOSTI NAPRAVE	CENA	KRATEK OPIS
appMobi [14]	iOS, Android	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave, hibridna aplikacija	odvisno od vrste aplikacije	brezplačno	Brezplačen razvoj. Plača se prostor v oblaku 5$ in več. Moženrazvoj iger z HTML5, CSS3, canvas. Uporaba GameDev XDK je plačljiva 99$. Aplikacijo izdelujemo z brskalnikom, XDK je v oblaku. Aplikacije so na izgled povsod enake. Ni izvornega zgleda. Aplikacije na uradnih spletnih trgovinah potrebujejo veliko dovoljenj, kar uporabnike odvrača.
appsbuilder [15]	iOS, Android, WP7	spletno ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView		brezplačno	Brezplačna verzija vsebuje reklame. Plača se 19$ na aplikacijo na mesec ali več, odvisno od podpore, ki jo potrebujemo. Brez kodiranja, samo dodajamo komponente. Povleci in spusti način izdelave. Lahko dodamo svojo HTML5 in JS kodo za popolnoma svojo spletno stran. Spletna izdelava aplikacije kot vizitke.
Corona [17]	iOS, Android, Nook, Kindle	ogrodje	Lua [30]	izvoma aplikacija	NFC, opozarjanje s tresenjem in modri zob.	plačljivo	Plačljivo ogrodje = 350$ na leto (če potrebujemo samo iOS ali Android razvoj, je cena 199$ na leto). Corona SDK povezan z OpenGL. Omogoča izdelavo iger. API za izvorne funkcionalnosti naprave (obvestila, sporočila, GPS, senzorji naprave).

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONALNOSTI NAPRAVE	CENA	KRATEK OPIS
FeedHenry [19]	iOS, BlackBerry, Android, WP	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView	ne podpira nobene funkcionalnosti naprave	brezplačno	V oblaku. Ni potrebno nalagati SDK-jev. Izdelava aplikacij preko App Studio IDE, ki deluje preko spleta.
Gideros Mobile [22]	iOS, Android	ogrodje	Java, Lua, Action Script	izvoma aplikacija		brezplačno	Brezplačna verzija vsebuje reklame ob zagonu aplikacije, sicer se plača 150$ na leto. Gideros ponuja IDE. API tudi za grafiko. Box2D, OpenGL, Open AL. Deluje na AMRv6 in AMPv7 platformah.
iWebKit [24], [58], [63]	iOS, Android (plan), BlackBerry (plan)	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	ni podatka	brezplačno	Sprva je bilo ogrodje namenjeno izdelavi spletnih strani optimiziranih za mobilne naprave, sedaj z uporabo »Web UI view« tudi za aplikacije. IWebKit podpira CSS3 v Safari brskalniku. Mogoč razvoj izvornih iOS aplikacij z vstavljanjem iWebKit v projekt (WebApp). IwebKit deluje tudi z PhoneGap. Če aplikacijo želimo prodajati, plačamo 20$.
JO [26]	iOS, Android, BlackBerry, Chrome OS	knjižnica	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView	ne podpira nobene funkcionalnosti naprave, le z uporabo PhoneGap.	brezplačno	Podoben kot PhoneGap. Uporabniški vmesnik se izdela z uporabo CSS. JO ogrodje dela tudi s PhoneGap.
jQTouch [27], [63]	iOS, Android	knjižnica	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	ne podpira veliko izvornih funkcionalnosti naprave	brezplačno	Knjižnico uporabimo v izvornem projektu izdelave aplikacije vendar z WebView ali pa za spletno stran. Zazna vrsto mobilne naprave in temu primerno izriše uporabniški vmesnik.

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONALNOSTI NAPRAVE	CENA	KRATEK OPIS
jQuery Mobile [28]	vse platforme, razen Maemo	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	ne podpira nobene funkcionalnosti naprave	brezplačno	Uporaba AJAX-a.
JULY Systems [29]	Vse	ogrodje	HTML, CSS, JavaScript	spletna stran za mobilne naprave	ne podpira nobene funkcionalnosti naprave	plačljivo	Povleci in spusti vmesnik za dodajanje komponent. Na voljo več že izdelanih izgledov aplikacije. Možnost pošiljanja sms sporočil. MiStudio za izdelavo spletnih strani
Mamalade [31]	iOS, Android	ogrodje	C / C++, HTML, JavaScript	izvorna aplikacija, hibridna aplikacija		plačljivo	Plačljivo ogrodje = 149$. Podpora 3D grafiki. Uporabno za izdelovanje iger.
MobiOne [35]	iOS, Android (plan)	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	ne podpira nobene funkcionalnosti naprave	plačljivo	Plačljivo ogrodje (99$). IDE MobiOne Studio.
Monocross [38]	vse platforme	ogrodje	C#	izvorna aplikacija, aplikacija, ki teče v izvornem WebView, hibridna aplikacija	pospeškometer, uporaba imenika, kompas in opozarjanje s tresenjem	brezplačno	MVC ogrodje za izdelavo aplikacij z uporabo MonoTouch (iOS) , MonoDroid (Android) ter ASP.NET (HTML5, JavaScript, CSS).

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONALNOSTI NAPRAVE	CENA	KRATEK OPIS
MonoTouch [36], [64], [65], [68], [69], [70]	iOS	ogrodje	C#	izvorna aplikacija		plačljivo	Skupaj z MONO MonoTouch omogoča razvoj iOS aplikacij z uporabo .NET ogrodja. IDE MonoDevelop. Plačljivo ogrodje = 250$ in več.
MoSync [39], [40]	Android, iOS, Windows Mobile, Symbian, Java, ME Moblin	ogrodje	C/C++, HTML, CSS, JavaScript	izvorna aplikacija, aplikacija, ki teče v izvornem WebView, hibridna aplikacija		brezplačno	Omogoča izdelavo izvornih aplikacij z uporabo OpenGL ES grafičnih knjižnic z jezikom C/C++. Za izdelavo hibridnih aplikacij je razvijalcu na voljo mosync_bridge.js knjižnica, ki poskrbi za komunikacijo med C/C++ in HTML deli kode.
Netbiscuits [41]	vse platforme	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView	ne podpira nobene funkcionalnosti naprave	brezplačno	Razvoj v oblaku. Brezplačno, če aplikacije ne prodajamo (sicer je cena 69$ na mesec).
NimbleKit [42]	iOS	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, hibridna aplikacija	odvisno od vrste aplikacije	plačljivo	Plačljivo ogrodje = 100$. V Xcode naredimo projekt podobno kot PhoneGap.
NS Basic/App Studio [43]	iOS, Android, BlackBerry, MeeGo	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	NFC, modni zob	plačljivo	Plačljivo ogrodje = 99$. Aplikacije ne naložimo na App store in Google play, ampak jo razpošljemo preko spleta in nadziramo dostop. Uporablja se WebKit.

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONAL-NOSTI NAPRAVE	CENA	KRATEK OPIS
Resco.net [47]	WM, WinCE, WP7, Win32, iOS, Android	ogrodje	C#	izvoma aplikacija		plačljivo	Ogrodje je plačljivo = 599$ za vsako platformo. Plačljiv je tudi razvojni IDE (Resco MobileApp Studio).
RhoMobile [4], [5], [47],[48], [63]	iOS, Android, Windows Mobile, BlackBerry, Symbian	ogrodje	HTML, CSS, JavaScript, Ruby	aplikacija, ki teče v izvornem WebView		brezplačno	MVC način izdelave aplikacij. Kontroler razvit z jezikom Ruby. Izgled razvit z HTML, CSS. IDE RhoStudio zelo podoben Eclipce IDE. Mogoča uporaba IDE v oblaku RhoHub, ki je plačljiv (49$ do 499$ na mesec). Ne potrebujemo nameščenih SDK-jev vseh platform.
PhoneGap [4], [5], [44],[45], [63], [67]	iOS, Android, BlackBerry, Palm, webOS, Windows Mobile, Bada, Symbian	knjižnica	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView		brezplačno	Za vsako izbrano platformo je potrebno narediti projekt na izvoren način, nato se v projekt doda PhoneGap ogrodje. Videz aplikacije je enak na vseh platformah, z uporabo drugih knjižnic ali samostojnega razvoja različnih CSS skript lahko naredimo drugačen izgled aplikacije.
Sencha Touch [49], [63]	iOS, Android, BlackBerry	knjižnica	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	ne podpira veliko izvornih funkcionalnosti naprave	plačljivo	Knjižnica za izvorni izgled aplikacije. Plačljiva podpora razvijalcem = 299$. Grafi so plačljivi dodatki (99$), ampak niso za dovoljivi. Deluje skupaj z knjižnico PhoneGap, ki dostopa do izvirnih funkcionalnosti naprave.

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONALNOSTI NAPRAVE	CENA	KRATEK OPIS
Smartface Designer [51]	iOS, Android	ogrodje	C++	izvoma aplikacija, hibridna aplikacija	kompas, obvestila, branje iz datotek, modri zob, NFC	brezplačno	Brezplačna verzija ne dovoljuje objave aplikacije na uradnih spletnih trgovinah, vsebuje reklame. Plača se 1490$ na aplikacijo. Povleci in spusti način pri izdelavi aplikacije. Izgled je narejen s slikami. Tudi gumbi. Akcije se gumbom dodajo s povleci in spusti načinom.
TapLynx [54]	iOS	ogrodje	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView	ne podpira nobene funkcionalnosti naprave	plačljivo	Plačljivo ogrodje: iPhone SDK = 299$ in iPad SDK = 499$. Aplikacijo izdelamo kot spletno stran.
Unity [55]	iOS, Android	ogrodje	C#, JavaScript, Phyton - Boo	izvoma aplikacija		plačljivo	Plačljiv pogon (400$ za vsako platformo), ki omogoča razvoj mobilnih iger v 3D prostoru.
Verivo [56]	iOS, Android, BlackBerry	ogrodje	Ni kodiranja	aplikacija, ki teče v izvornem WebView	pomanjkljiva dokumentacija	brezplačno	IDE Application Studio, podoben kot Visual Studio. Ni izvornega izgleda. Potrebujemo veliko strojne in programske opreme, ker aplikacija deluje na strežniku (IIS strežnik, .NET Framework, SQL strežnik 2005 ali 2008)
WebApp-Net [57], [70]	iOS, Android, WebOS	knjižnica	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView	ne podpira nobene funkcionalnosti naprave	brezplačno	Brezplačna mikro knjižnica, ki poskrbi za izgled aplikacije.
Wink [60]	iOS, Android, BlackBerry, Bada, WebOS	knjižnica	HTML, CSS, JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	ne podpira veliko funkcionalnosti naprave	brezplačno	Izgled narejen z CSS. Deluje s PhoneGap.

IME	PODPIRA PLATFORME	VRSTA	RAZVOJNI JEZIK	VRSTA APLIKACIJ	NE PODPIRA FUNKCIONAL-NOSTI NAPRAVE	CENA	KRATEK OPIS
XUI [61]	Vse platforme	knjižnica	HTML CSS JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	ne podpira nobene funkcionalnosti naprave	brezplačno	Knjižnico uporabimo pri izdelavi izvorne aplikacije, vendar aplikacija teče v WebView. Izdelamo lahko tudi spletno stran prilagojeno mobilnim napravam z uporabo knjižnice.
Zepeto.js [62]	iOS, Android, BlackBerry, Symbian, WebOS, Bada	knjižnica	JavaScript	aplikacija, ki teče v izvornem WebView, spletna stran za mobilne naprave	PhoneGap omogoča podporo funkcionalnosti naprav	brezplačno	Samo za moderne brskalnike. Sodeluje s PhoneGap. Podobno kot Jquery, samo manjša datoteka in hitrejše metode.

Tabela 3.1: Primerjalni prikaz medplatformskih ogrodij in knjižnic.

lahko učenje dodatnega programskega jezika vzame preveč dragocenega časa za razvoj aplikacije. Še pomembnejši razlogi za odločitev o izbiri orodja so omejitve ogrodja. Aplikacija izdelana v medplatformskem ogrodju Rhodes je pravzaprav spletna stran, ki teče na izvornem brskalniku mobilne naprave in je v njej tudi shranjena. To lahko predstavlja nekaj ključnih omejitev pri funkionalnosti končne aplikacije. Izvorni brskalniki pametnih naprav se med seboj razlikujejo glede na sposobnosti prikazovanja zahtevnejših spletnih strani, ki uporabljajo naprednejše tehnologije za prikaz podatkov. Tako npr. na Android, BlackBerry in Symbian platformi ne moremo uporabiti prikaz podatkov s pomočjo SVG tehnike. Ogrodje Rhodes pravtako ni primerno za izdelavo hitrih in grafično zahtevnih iger, kot tudi drugih aplikacij, ki zahtevajo bogato grafično podporo.

Rhodes za poslovno logiko uporablja jezik Ruby, ta pa uporablja najboljše prakse MVC načina razvoja aplikacij. MVC loči poslovno logiko z izgledom aplikacije. Izgled je narejen z uporabo jezikov HTML, CSS, JavaScript, nadzorniki (*angl. Controllers*) pa z jezikom Ruby. Koda je tako bolj pregledna in jo je zato lažje vzdrževati.

Rhodes ponuja razvojno orodje RhoStudio, ki je zelo podobno orodju Eclipse. Razvoj Rhodes aplikacij je mogoč tudi z drugimi razvijalskimi orodji, ki podpirajo Ruby in HTML, kot so Eclipse, IntelliJ, NetBeans, Visual Studio, Textmate, VIM, Emacs in drugi.

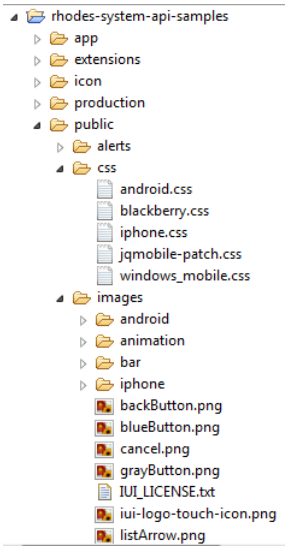
Na voljo je tudi IDE v oblaku RhoHub. RhoHub vsebuje vse kar potrebujemo za razvoj aplikacije, zato posebne namestitve na računalniku niso potrebne. Povezan je z GitHub (*spletni repozitorij*), na ta način pa omogoča pregled nad kodo. RhoHub je plačljiv glede na to, koliko aplikacij želimo razvijati, koliko prostora potrebujemo za svoje datoteke in ali je koda javno dostopna ali ne.

Ogrodje Rhodes z Rhom API omogoči povezavo s podatkovno bazo SQLite in HSQLDB. Način shranjevanja podatkov je sicer nekoliko drugačen (*v primerjavi s shranjevanjem podatkov v podatkovnih bazah*), toda olajša delo razvijalcem. Uporablja se ključ, tj. vrednost način shranjevanja podatkov.

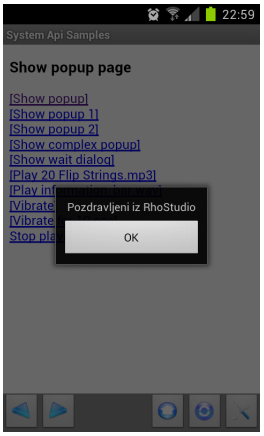
Zanimivo je tudi zagotavljanje izvornega izgleda aplikacije. Uporabniki praviloma pričakujejo drugačen izgled aplikacij na različnih platformah. To je realizirano z uporabo že napisanih CSS slogovnih pol, ki jih ogrodje ponuja. K temu spadajo tudi pripadajoče slike različnih gumbov in drugih slik za prikaze na različnih platformah. Slogovne pole lahko poljubno spreminjamo ter dodajamo slike. Rhodes si pri izgledu pomaga tudi z drugimi JavaScript knjižnicami (*kot so: JQTouch, iWebKit, xui in Sencha Touch*), ki še dodatno izboljšajo izvorni izgled aplikacije. Aplikacija teče na izvornem brskalniku mobilne naprave. Če katera lastnost naprave ni podprta, kot npr. NFC, je mogoče napisati svojo kodo z uporabo izvornih razširitev.

Na sliki 3.1 so prikazane različne slogovne pole projekta za posamezne platforme in slike za različne gumbe, vnosna polja in podobno. Slika 3.2 prikazuje dostop do izvornih sporočil, v tem primeru opozorila (*angl. Alert*). Nekoliko nižje je prikazana še koda, ki jo uporabimo na vseh platformah. Opozorilo je na vsaki platformi vidno drugače in je realiziran s klicem izvornih API (*in ne z uporabo različnih slogovnih pol*). Predstavljena koda je uporabljena tudi za prikaz strani, ki je v ozadju sporočila. Podjetje RhoMobile sicer meni [48], da je izgled aplikacije nerejen na ta način (*z uporabo slogovnih pol*) uporabniku prijaznejši, vendar je potrebno o tem premisliti.

```
require 'rho/rhocontroller'
class AlertController < Rho::RhoController
   @layout = :simplelayout
   def index
      @flash = "Alerts"
      render :back => '/app'
   end
```



Slika 3.1: Prikaz različnih slogovnih pol za posamezne platforme in slike različnih gumbov, vnosnih polj v projektu.



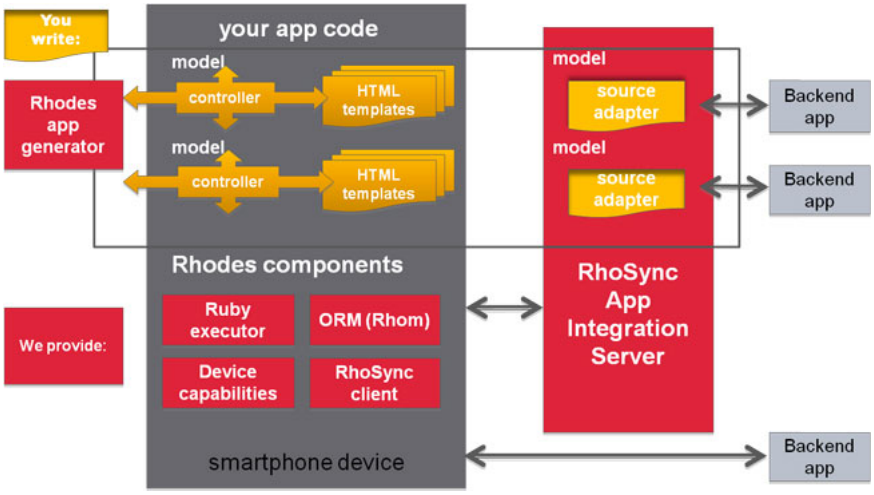
Slika 3.2: Prikaz dostopa do izvornih sporočil (*Android*).

```
def show_popup
   @flash = "Show popup page"
   Alert.show_popup "Pozdravljeni iz RhoStudio"
   render :action => :index, :back => '/app'
end
def show_popup1
   @flash = "Show popup page"
   Alert.show_popup(
      :message=>"The new password can't be empty.\n",
      :title=>"",
      :buttons => ["Ok"]
   )
   render :action => :index, :back => '/app'
end
def show_popup2
   @flash = "Show popup page"
   Alert.show_popup(
      :message=>"The new password can't be empty.\n",
      :title=>"MyTest",
      :buttons => ["Ok", "Cancel"],
      :callback => url_for(:action => :popup_callback)
   )
   render :action => :index, :back => '/app'
end
def show_popup_complex
   @flash = "Show popup page"
   Alert.show_popup :title => "This is popup", :message =>
   "Some message!", :icon => :info,
      :buttons => ["Yes", "No", {:id => 'cancel', :title => "Cancel"}],
      :callback => url_for(:action => :popup_callback)
      render :action => :index, :back => '/app'
end
def show_popup3
   @flash = "Show popup page"
   Alert.show_popup :title => "Wait...", :message => "Wait ..."
   Rho::Timer.start 5000, url_for(:action => :wait_callback), ""
   render :action => :index, :back => '/app'
end
```

```
  def wait_callback
    Alert.hide_popup
    WebView.navigate url_for(:action => :index)
  end
  def popup_callback
    puts "popup_callback: #{@params}"
    WebView.navigate url_for(:action => :index)
  end
  def vibrate
    @flash = "Vibrate page"
    Alert.vibrate
    render :action => :index, :back => '/app'
  end
  def vibrate_for_10sec
    @flash = "Vibrate for 10 sec page"
    Alert.vibrate 10000
    render :action => :index, :back => '/app'
  end
  def play_file
    @flash = "Play file page"
    Alert.play_file @params['file_name'], @params['media_type']
    render :action => :index, :back => '/app'
  end
  def play_file_1
    @flash = "Play file page"
    Alert.play_file @params['file_name']
    render :action => :index, :back => '/app'
  end
  def stop_playing
    Rho::RingtoneManager.stop
    render :action => :index, :back => '/app'
  end
end
```

Aplikacija ob zagonu najprej preveri na kateri platformi je zagnana. Na osnovi te informacije se izbere ustrezen CSS in se s tem določi izgled aplikacije.

Za razvijanje Rhodes aplikacij je potrebno namestiti Ruby in Rhodes ogrodje, ki zago-



Slika 3.3: RhoMobile razvijalsko okolje.

tovi vsa potrebna orodja za razvoj aplikacij. Potrebe s seznanjenostjo z izvornim SDK-jem vsake platforme ni, kar je sicer namen orodij za medplatformski razvoj, vendar pa moramo imeti SDK platforme (*za katere razvijamo aplikacijo*) nameščene na računalniku. Upoštevati je potrebno tudi, da je razvoj za Apple iOS mogoč le na Mac OS X računalnikih, za BlackBerry pa le na računalnikih z nameščenim Windows operacijskim sistemom.

Rhodes ponuja tudi RhoSync strežnik. Ta je nekaj posebnega, saj omogoča sinhronizacijo podatkov v ozadju, še preden jih aplikacija dejansko potrebuje. Aplikacija je tako lahko aktivna tudi brez uporabe interneta, saj so podatki sinhronizirani. To je sicer mogoče tudi na aplikacijah napisanih z drugimi razvijalskimi ogrodji, vendar mora za to poskrbeti razvijalec sam. Prednost RhoSync je torej njegova enostavna uporaba.

Slika 3.3 prikazuje RhoMobile razvijalsko okolje.

Na platformah, kjer je izvorni razvijalski jezik Java (*npr. na BlackBerry platformi*), se Ruby aplikacije prevedejo v Java kodo in se izvajajo izvorno. Na ostalih platformah se aplikacija prevede v Ruby bitno kodo. Izvajanje aplikacije na Android platformi je včasih še hitrejše, saj je Rhodes ogrodje napisano v jeziku C++ z Android NDK. Ker pa veliko aplikacij uporablja RhoSync za sinhronizacijo podatkov, so podatki tako vedno dostopni lokalno, kar dodatno pospeši izvajanje aplikacije.

Rhodes ponuja za testiranje aplikacije storitev RhoGallery, ki je dodatna storitev v Rho-

Hub. Dostop do teh aplikacij imajo samo uporabniki RhoHub.

Rhodes je odprtokoden in se ga lahko spreminja, posodablja, prilagaja po lastnih potrebah, kar je njegova prednost. Kljub vsemu pa razvoj z Rhodes ni enostaven. Neizkušen razvijalec lahko porabi veliko časa za preučevanje vsaj osnovnih vsebin, ki jih ponuja Rhodes ogrodje. Težava pa ni v veliki številčnosti vsebin, ampak v pomankljivi dokumentaciji in primernih aplikacijah. Hkrati so vsi osnovni primeri aplikacij izdelani preveč kompleksno, vsebujejo preveč datotek, zato je iz primerov težko izdelati aplikacijo z npr. samo enim oknom, ki vsebuje neko poljubno sporočilo. Prevelik je tudi poudarek na RhoSync-u in RhoHub-u, zato lahko pride do zmede pri vprašanju, kako aplikaciji zagotoviti vse potrebne podatke. Razlaga orodja je nejasna, zato lahko kljub obširnemu preučevanju nevede in nehote izpustimo najpomembnejše vsebine medplatformskega ogrodja Rhodes. Rhodes je plačljiv, če hočemo aplikacijo prodajati na spletnih trgovinah z aplikacijami.

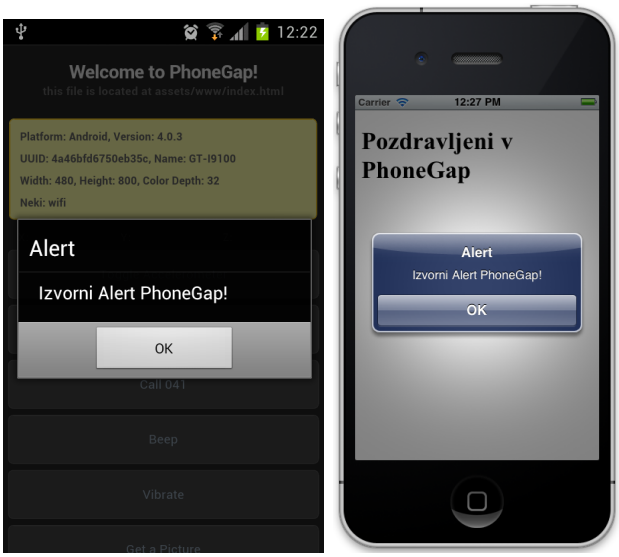
3.2.2 PhoneGap

Razvojno ogrodje PhoneGap [4], [5], [44], [45], [63], [67] je na vrhu priljubljenosti izbranih ogrodij za razvoj medplatformskih aplikacij. Podrobnejši pregled razkrije nekaj ključnih elementov, ki so pomembni za odločitev o izbiri ogrodja.

PhoneGap podpira kar sedem mobilnih platform (*iOS, Android, BlackBerry, Palm webOS, Windows Mobile, Bada in Symbian*). Z njim izdelujemo izvorne mobilne aplikacije z uporabo jezikov HTML, CSS in JavaScript.

Kljub vsemu, podrobnejši pregled in uporaba ogrodja razkrije, da aplikacije niso ravno izvorne, kot se promovira. Aplikacije so zelo podobne spletnim aplikacijam, saj tečejo v izvornem oknu brskalnika vsake platforme (*ob tem je razumljivo, zakaj je razvoj realiziran z jeziki, ki jih uporabljajo spletni razvijalci*). Različna okna aplikacije so pravzaprav različne spletne strani. Da dosežemo različen izvoren izgled aplikacije, je potrebno za vsako platformo napisati drugačen CSS. To pa pomeni, da razvoj aplikacije praviloma ne poteka tako hitro.

Izvornost aplikacij, razvitih z ogrodjem PhoneGap se kaže v omogočanju uporabe različnih funkcionalnosti posameznega pametnega telefona in naprave ter v začetni izdelavi aplikacije na izvoren način. Lahko dostopamo do različnih senzorjev naprave (*pospeškometer, žiroskop, barvni senzor, senzor bližine, kompas, barometer*), kot tudi do telefonskega imenika, kamere, spomina, avdio ter video vsebin in tudi do izvornih dogodkovnih opozoril in sporočil. Za primer si oglejmo opozorilo, ki je generirano z enako kodo, vendar se na



Slika 3.4: Prikaz opozirila na Android in iOS platformi.

vsaki platformi pokaže drugače. Opozorilo je za Android in iOS platformi prikazano na sliki 3.4, koda pa je predstavljena v nadaljevanju.

```
function alert1()
{
   navigator.notification.alert("Izvorni Alert PhoneGap!")
}
<a href="#" class="btn large" onclick="alert1();">
Prikaži izvorno sporočilo</a>
```

Zanimiva je tudi izdelava aplikacije za izbrano platformo. Za vsako platformo moramo upoštevati vsa pravila za izdelavo aplikacije na tej platformi in izdelati osnovni začetni projekt. Za lažjo predstavo bo v nadaljevanju podrobneje prikazan primer za Android platformo, kasneje pa še za ostale platforme.

3.2.2.1 Začetek izdelave aplikacije za Android

Na razvijalski računalnik je potrebno na začetku namestiti PhoneGap. Nov projekt v Eclipse IDE naredimo po enakih postopkih, kot bi izdelovali izvorno aplikacijo [1]. Ko je projekt pripravljen, je običajno že na začetku mogoče zagnati aplikacijo. Aplikacija pokaže

[1] Opisano v poglavju 2.2.1 Izvorni razvoj aplikacij.

začetni zaslon kot pripomoček, da smo na pravi poti. Nato nadaljujemo z vstavljanjem PhoneGap knjižnice v naš projekt. Izdelati moramo dve novi mapi */libs* in */assets/www*. Kopiramo datoteko "phonegap.js", ki jo smo dobili s prenosom PhoneGap-a na računalnik v obe prej navedeni mapi. Kopiramo še xml mapo, ki smo jo prav tako dobili s prenosom Phonegap-a v mapo */res*. Narediti moramo še nekaj prilagoditev v začetni aktivnosti, kot je v kodi prikazano spodaj.

```
import android.os.Bundle;
import com.phonegap.*;

public class TestApp extends DroidGap {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      //setContentView(R.layout.main); //pred PhoneGap
      super.loadUrl("file:///android_asset/www/index.html");
   }
}
```

Razred ne razširja več razreda `Activity` ampak `DroidGap`. Klic funkcije, ki odpre prvo stran aplikacije:

`setContentView(R.layout.main)` zamenja klic:

`super.loadUrl("file:///android_asset/www/index.html")`. Ta prikaže prvo stran aplikacije. Pokliče se datoteka `index.html`, ki se nahaja v mapi `www` (*tukaj najdemo tudi PhoneGap knjižnico*). Tako lahko s pomočjo HTML-ja preko `phonegap.js` knjižnice dostopamo do izvornih funkcij naprave. Za izgled aplikacije moramo poskrbeti z CSS datotekami ali na kakšen drug podoben način. Izdelati je potrebno še `index.html` datoteko, ki jo lahko tudi drugače poimenujemo, vendar je v tem primeru potrebno prilagoditi klice te datoteke v aplikaciji. Nastavimo še dodatne pravice v datoteki `AndroidManifest.xml` in projekt je pripravljen. Po želji lahko dodajamo nove html datoteke, pri čemer je razvoj potem enak izdelavi spletnih strani.

3.2.2.2 Razvoj na ostalih platformah

Podobno kot za Android platformo, moramo postopek ponoviti še za vsako platformo posebej, tj. namestiti PhoneGap na razvijalčev računalnik in izdelati nov projekt na izvoren način. Za različne platforme je značilnih nekaj posebnosti. Tako se na iOS platformi v Xcode, za kreiranje novega projekta, pokaže nova možnost "Cordova-based Application",

ki jo moramo izbrati. Na koncu imamo toliko projektov, kolikor platform izberemo. Skupno vsem projektom so enake html datoteke. Pri razvoju aplikacije za več platform tako prenesemo html datoteke na druge projekte. S tem ogrodje PhoneGap opraviči medplatformski razvoj. V nadaljevanju je potrebno določiti še izgled aplikacije. Če ne želimo enakega izgleda aplikacije na vseh platformah, moramo uporabljati različne slogovne pole. PhoneGap namreč ne definira izvornega izgleda (*gumbov, vnosnih polj, ipd.*) z uporabo izvornega SDK-ja vsake platforme.

Aplikacijo lahko razvijemo tudi s kombinacijo HTML datotek ter z izvornim načinom, vendar tovrstnega izvornega načina ne moremo prenašati na ostale platforme. Takim aplikacijam rečemo hibridne aplikacije.

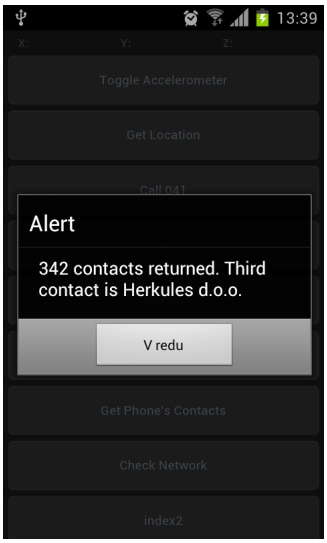
Aplikacijo prenesemo na spletne trgovine z aplikacijami na enak način, kot to velja za izvorno aplikacijo, saj je aplikacija dejansko narejena na izvoren način. Izpostaviti je potrebno še omejitve glede vsebine prikaza izvornih brskalnikov na pametnih napravah, tako na Android OS, Symbian, BlackBerry, Palm webOS, Windows Mobile, Bada ne moremo prikazati grafike v SVG tehniki in uporabljati nekaterih naprednejših funkcij HTML5.

Zanimiva značilnost PhoneGap ogrodja je, da lahko uporabimo storitev PhoneGap Build. Njena prednost je, da ni potrebno imeti nameščnega SDK-ja za vsako platformo in uporabljano razvojno okolje, ampak lahko izdelamo aplikacijo samo z uporabo HTML, CSS in JavaScript. Izdelek v celoti pošljemo storitvi v oblaku, ki poskrbi za ustrezen prevod. Slabost storitve PhoneGap Build-a je plačljivost, ki je odvisna od števila izdelanih zasebnih aplikacij, tj. da ne želimo kode javno izpostavljati (*cena se giblje od 12 do 90$ na mesec*).

PhoneGap je zelo primeren za razvoj aplikacij, za katere želimo, da delujejo na številnih platformah. Izgled aplikacije je enak na vseh teh platformah, če ne uporabimo različnih slogovnih pol. Pri izbiri PhoneGap-a je potrebno vzeti v ozir še kompleksnost aplikacije, saj je potrebno upoštevati tudi podporo izvornega brskalnika naprave.

V spodnji kodi vidimo primer branja podatkov iz imenika.

```
function contacts_success(contacts) {
   alert(contacts.length
   + ' contacts returned.'
   + (contacts[2] && contacts[2].name ? (' Third contact is '
   + contacts[2].name.formatted) : ''));
}
```



Slika 3.5: Izpis stika z opozirilom na Android platformi.

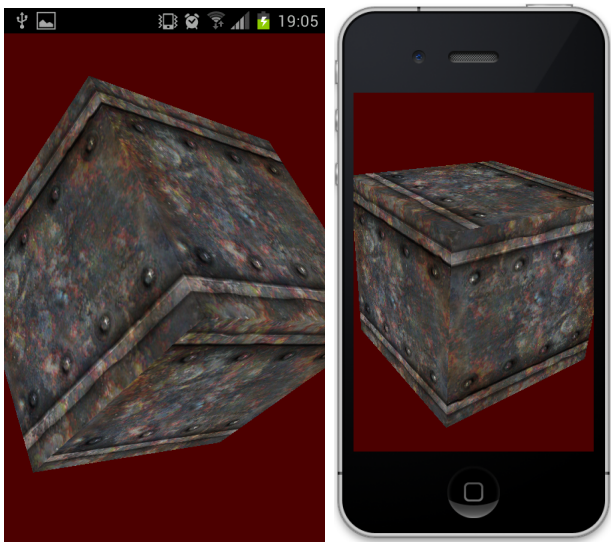
```
function get_contacts() {
   var obj = new ContactFindOptions();
   obj.filter = "";
   obj.multiple = true;
   navigator.contacts.find(
      [ "displayName", "name" ], contacts_success,
      fail, obj);
}
```

Slika 3.5 prikazuje, kako lahko z uporabo PhoneGap ogrodja dostopamo do podatkov s telefonskega imenika s pomočjo zgoraj predstavljene kode.

3.2.3 MoSync

MoSync [39], [40] je ogrodje, ki je namenjeno predvsem razvijalcem z znanjem jezika C in C++. Trenutno podpira Android, iOS, Windows Mobile, Symbian, Java ME in Moblin platforme.

Z namestitvijo MoSync dobimo Eclipsu podobno razvojno ogrodje ter MoSync SDK in "Whormhole" JavaScript knjižnico, ki omogoča komunikacijo med C/C++ in HTML deli aplikacije.



Slika 3.6: Obračanje 3D kocke na platformah (*Android levo in iOS desno*).

Z MoSync je omogočen razvoj aplikacij, ki tečejo v izvornem brskalniku mobilne naprave in so podobne spletnim aplikacijam. Za razvoj se uporabljajo že znani jeziki (*HTML5, CSS in Java Script*). Kljub temu, da MoSync podpira razvoj za veliko različnih mobilnih platform, pa lahko aplikacije, ki tečejo v izvornem oknu brskalnika naprave izdelujemo le za Android, iOS in Windows Phone. Izgled aplikacije je realiziran z uporabo CSS slogovnih pol, izvorne funkcije naprave pa so dosegljive preko "Wormhole" JavaScript knjižnice. Tako lahko dostopamo do imenika, različnih senzorjev naprave, GPS navigacijske storitve, kot tudi do pošiljanja SMS sporočil in do izvornih sporočil (*sporočila, obvestila, opozorila*) operacijskega sistema oz. platforme. Na voljo imamo veliko že predhodno izdelanih aplikacij narejenih s HTML, CSS in JavaScript, ki so nam lahko v pomoč pri razvoju lastne aplikacije.

Aplikacije, izdelane s programskima jezikoma C/C++, se zelo približajo izvorni izdelavi, ker se tudi izgled aplikacije tvori z izvornimi gradniki z uporabo izvornih SDK-jev (*gumbi, različni pogledi, polja za vnos besedila, ipd.*). Aplikacija teče v izvornem oknu aplikacije in ne v brskalniku mobilne naprave. Na ta način je zagotovljena boljša odzivnost aplikacije in je zato mogoča izdelava bolj kompleksnih aplikacij. Tak pristop omogoča uporabo podpore OpenGL ES knjižnice, s pomočjo katere lahko izdelujemo tudi grafično zahtevnejše igre. Slika 3.6 prikazuje obračanje 3D kocke na platformah Android in iOS. Možen je tudi hibridni pristop, tj. del aplikacije se izdela z uporabo HTML, CSS in JavaScript in del

s C/C++. MoSync omogoča komunikacijo med jezikoma C++ in JavaScript z uporabo "mosync_bridge.js" knjižnice.

Na uradni strani MoSync najdemo veliko testnih aplikacij, ki so nam lahko v pomoč pri učenju in dobro urejeno dokumentacijo. MoSync je tudi združljiv z PhoneGap, kar prinaša dodatne možnosti pri izdelavi aplikacije.

3.2.4 Mono

Izdelava mobilnih aplikacij z Mono medplatformskim orodjem je posebna. Mono namreč temelji na Microsoft .NET tehnologiji [1], [2], [3], [37], [59], [68], [69]. Microsoft, Apple in drugi proizvajalci operacijskih sistemov so tekmeci, zato je potek razvoja aplikacij z Microsoft orodji za Apple iOS operacijski sistem toliko bolj zanimiv.

3.2.4.1 .NET

V letu 1990 je Microsoft pričel z razvojem .NET ogrodja. Prva verzija pa je bila uporabnikom na voljo leta 2002. Trenutno je na trgu četrta verzija ogrodja, ki obstaja tako za 32- kot tudi za 64-bitne sisteme. Najdemo ga tudi v drugih napravah (*npr. XBOX*) in v mobilnih telefonih z verzijo CF (*angl. Compact Framework*). Razvijalsko orodje je Visual Studio .NET in je integrirano razvijalsko okolje za .NET ogrodje. .NET ogrodje je skupek vmesnikov in knjižnic, ki omogočajo lažje razvijanje aplikacij. Ogrodje nudi knjižnice za gradnjo uporabniških vmesnikov, spletnih komunikacij, dostop do različnih podatkovnih baz in drugo. Dobra stran ogrodja .NET je, da uporablja isto ogrodje pri več različnih programskih jezikih. Aplikacije napisane v ogrodju .NET lahko namreč uporabimo na sistemih Windows, Mac in Linux, čeprav se slednja soočata z omejitvami.

Mono vsebuje vso potrebno programsko opremo za razvoj in poganjanje .NET aplikacij na operacijskih sistemih Windows, Linux, Solaris, Mac OS X, Nintendo's Wii, Sony PlayStation 3 in drugih. Cilj Mono projekta je celovit prenos Microsoft .NET razvijalskega okolja na UNIX okolje in omogočen razvoj .NET aplikacij za več platform. Mono je odprtokodni projekt, ki zagotavlja C# prevajalnik in skupni jezik na več operacijskih sistemih.

3.2.4.2 MonoDevelop

IDE, v katerem razvijamo aplikacije, se imenuje MonoDevelop. Sprva je bil razvoj mogoč le na operacijskem sistemu Linux, toda z verzijo 2.2 je razvoj mogoč tudi na Mac OS X, Windows in mnogo drugih ne-Linux platformah. Čeprav je .NET ogrodje zelo razširjeno,

se pri razvoju za iOS pojavljata dve vprašanji. Prvo, že omenjeno, je konkurenčno okolje Appla in Microsofta in vprašljivost skupnega delovanja in drugo, .NET aplikacije so dinamično prevedene v času izvajanja le-teh. To pa na Apple iOS operacijskem sistemu ni dovoljeno.

Koda, ki jo poganjamo na Microsoft .NET ogrodju, je med izvajanjem prevedena v strojno kodo. Pričakovati je, da se bo koda napisana z Mono ogrodjem enako prevajala med izvajanjem, vendar temu ni tako. Ker iOS tega ne dopušča, ima Mono razvito ADO tehnologijo, ki kodo prevede pred izvajanjem. Za razvijanje aplikacij za mobilne naprave pa sam Mono še ni dovolj.

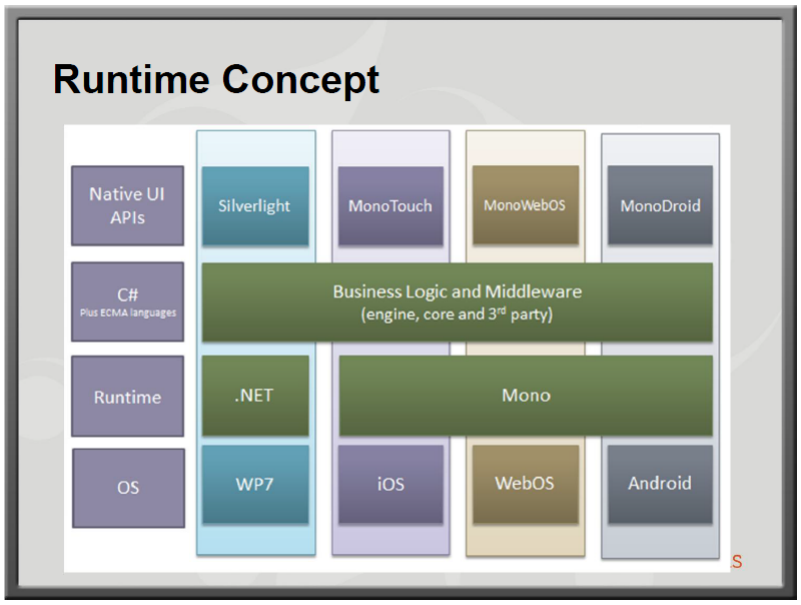
3.2.4.3 MonoTouch

Za razvijanje aplikacij za iOS platformo potrebujemo tudi MonoTouch [36], [64], [65], [68], [69], [70]. Izšel je leta 2009 in omogoča .NET razvijalcem razvijanje izvirnih aplikacij za iOS v jeziku C#. Za razvoj aplikacij za iOS potrebujemo Mac z OS 10.5.7 ali novejši, Mono, MonoTouch, MonoDevelop in iPhone SDK.

MonoTouch je paket orodij, ki razvijalcem iPhone in iPod Touch aplikacij na Mac OS omogoča uporabo znanj .NET ogrodja. MonoTouch API omogoča kombinacijo med .NET 3.5 ogrodjem in izvornim iPhone SDK. Da je to mogoče, je .NET ogrodje prevedeno do ARM kode in zato lahko teče na iPhone napravi. Osnovni iPhone API je tako prepleten z C# API. MonoTouch zagotavlja implementacijo Apple Cocoa Touch knjižnice na .NET način, zato pri razvoju ni omejitev. Razvijalcem je na ta način omogočen razvoj aplikacij v .NET ogrodju z jezikom C# za Apple iPhone, iPad, iPod Touch naprave z uporabo iPhone API. Mono Touch združuje Objective C in C API iz Cocoa Touch v C# CLI (*angl. Common Intermediate Language*) API. Nad MonoTouch je interoperabilni pogon, ki zagotavlja povezavo med Cocoa Touch API kot tudi s Core Foundation in UIKit. To vključuje tudi grafične knjižnice, kot so Core Graphics in OpenGl ES. Implementacija MonoMac zagotavlja tudi razvoj aplikacij za Mac OS X.

3.2.4.4 MonoDroid

Za razvoj aplikacij za Android platformo potrebujemo Mono za Android [70]. Osnovni koncept je enak kot pri MonoTouch. Razlika je le v tem, da lahko razvijamo tudi na Windows računalnikih in nam za IDE ni potrebno uporabljati MonoDevelop, ampak se lahko odločimo za Visual Studio 2010.



Slika 3.7: Prikaz za katere platforme je možen razvoj z uporabo .NET ogrodja in kako so med seboj povezani Mono, .NET, MonoTouch, MonoDroid, MonoWebOS ter Silverlight.

3.2.4.5 Silverlight

Windows aplikacije se izvorno razvijajo že z .NET tehnologijo [70]. Uporabimo lahko tudi Silverlight razvojno platformo.

3.2.4.6 Medplatformski razvoj z Mono ogrodjem

MonoTouch je povezan z MonoDroid [23], [32], [70]. Lahko celo rečemo, da sta v sorodu, le da je MonoDroid namenjen razvijanju aplikacij za Android OS. Tako je tudi realiziran medplatformski razvoj, saj lahko skoraj vso napisano kodo v C# uporabimo še za razvoj aplikacij na drugih platformah. Ker je koda napisana v C# in .NET BCL (*Base Class Library*), je ista koda lahko dodeljena tudi za razvoj Windows Mobile 7 aplikacij. To močno zmanjša stroške razvoja aplikacije, saj je lahko na ta način razvoj omogočen na treh vodilnih mobilnih platformah. Slika 3.7 prikazuje, za katere platforme je možen razvoj z uporabo .NET ogrodja in kako so med seboj povezani Mono, .NET, MonoTouch, MonoDroid, MonoWebOS ter Silverlight [31]. Tako MonoTouch kot tudi MonoDroid sta plačljiva (*cena za vsakega posebej je 399$ za osnovni paket in za enega razvijalca*). Brezplačno je mogoče zaganjati aplikacije samo na simulatorju.

3.2.5 Appcelerator Titanium

Appcelerator Titanium je medplatformsko ogrodje za razvoj mobilnih, tabličnih in namiznih aplikacij z uporabo spletnih tehnologij. Razvil ga je Appcelerator Inc. in je za uporabo na voljo od leta 2008 [4], [5], [10], [68], [71]. Podpora za razvoj iPhone in Android je bila omogočena v letu 2009, podpora za razvoj na iPad pa leto kasneje. Appcelerator Titanium omogoča razvoj aplikacij tudi za platformo BlackBerry, vendar je še v fazi testiranja (*beta verzija*).

Appcelerator Titanium je zelo priljubljen med spletnimi razvijalci, saj lahko le - ti uporabijo obstoječe znanje za ustvarjanje izvirnih aplikacij za iOS in Android. Razvijalec se mora tako naučiti le novi Titanium API in po potrebi JavaScript, kar je enostavneje kot učenje novega jezika (*Objective C ali Java*). Čeprav delo s Titaniumom daje hitre rezultate (*zaradi česar je Titanium zelo primeren za izdelavo prototipov*), obstajajo vprašanja glede razlik v obnašanju API-ja na različnih platformah, ter stabilnosti v upravljanju s pomnilnikom. Nekateri razvijalci zato še vedno raje razvijajo aplikacije za vsako platformo posebej.

Appcelerator Titanium z uporabo modula Box2D omogoča tudi razvoj iger, vendar le za iOS platformo. Omogoča tudi izris osnovnih objektov kot sta krog in pravokotnik ter njihovo dinamično premikanje.

3.2.5.1 Opis priprave okolja za delo z Appcelerator Titanium

Prvi korak je registracija uporabnika na portalu Appcelerator. Izbiramo lahko med brezplačnim in plačljivim računom. Brezplačni račun nudi podporo skupnosti, izobraževanja preko video posnetkov ter osnovni mobilni API. Plačljiv račun nudi pomoč razvijalcem s strani strokovne ekipe podjatja Appcelerator, ter naprednejši API. Modul za delo z grafičnimi prikazi je potrebno plačati. Naslednji korak je namestitev Eclipsu podobnega IDE Titanium Studio, ki je dosegljiv na spletni strani Appcelerator Titaniuma [10]. Ob tem se namesti tudi SDK. Titanium Studio napisano kodo prevede v izvorno kodo vsake platforme posebej z uporabo izvornih SDK-jev. Predhodno je zato potrebno imeti izpolnjene vse zahteve za razvoj aplikacij na posamezni platformi[2]. Okolje je primerneje namestiti na računalnik z Mac operacijskim sistemom, ker je aplikacijo za iOS platformo mogoče razvijati samo v tem operacijskem sistemu. Titanium Studiu je v naslednjem koraku potrebno določiti pot do iOS in Android SDK. Titanium Studio je tako pripravljen za delo.

[2]Zahteve za razvoj aplikacij na posamezni platformi so bile predstavljene v poglavju 2.2.1. Izvorni razvoj aplikacij.

Pri izdelavi projekta je potrebno na začetku definirati, na katerih platformah bo aplikacijo možno poganjati (*iOS, Android, WebOS*). Če razvoj aplikacije za iOS platformo ni potreben, potem razvoj na računalniku z Mac OS ni nujen. Projektu nastavimo še ime aplikacije. Tak projekt je že pripravljen za prenos na mobilno napravo ali pa za poganjanje na Android ali iOS simulatorju. Tako narejena začetna aplikacija prikazuje dva okna, kar je dovolj za testiranje delovanja okolja.

Appcelerator Titanium smo izbrali za izdelavo aplikacije "Sončne elektrarne". Zato bo več o njem napisanega tudi v pogavju 4.1 Zakaj Appcelerator Titanium?

3.3 Vrste mobilnih aplikacij izdelanih z medplatformskimi ogrodji

Pred razvojem medplatformskih ogrodij so se razvijalci še posebej posvečali vprašanju o aplikacijah, ki bi, s čim manj spremembami v kodi in razvoju, tekla na čim več mobilnih platformah. Pri tem so uporabljali različne tehnike, kmalu pa ugotovili, da je mogoča uporaba integriranega spletnega brskalnika v mobilni napravi. Na ta način se je pričela izdelava spletnih strani, prilagojenih mobilnim napravam. Uporaba gradnika WebView je omogočala dostop do spletne strani, zato je bila mogoča izdelava aplikacije, ki se že ob zagonu poveže na internet in prikaže vsebino. Uporabniki so si nameščali izvorne aplikacije, ob zagonu le-teh, pa se je prikazala samo prilagojena spletna stran. Spletne strani so lahko shranjene v mobilni napravi in so tako dostopne tudi brez povezave. Na tak način je izdelanih veliko HTML iger, ki ne potrebujejo podatkov s strežnikov. Kmalu za tem so se na trgu začela pojavljati medplatformska orodja, ki so praviloma uporabila prav ta pristop. Tak pristop prinaša tako prednosti kot slabosti, zato si ga bomo ogledali v nadaljevanju [70]. Medplatformska orodja so hkrati narekovala razvoj drugačnih pristopov, ki bodo prav tako predstavljeni v nadaljevanju.

3.3.1 Aplikacije izdelane z .NET ogrodji

Aplikacije narejene z .NET medplatformskim ogrodjem, se kar najbolj približajo izvornim aplikacijam. Aplikacije izgledajo kot izvorne in so tako tudi odzivne. Grafično je nabor bogat in tako bistvenih omejitev pri razvoju ni. Trenutno obstaja zelo majno število tovrstnih medplatformskih ogrodij (*.NET ogrodje je pravzaprav eno samo, medplatformskih*

ogrodij za razvoj aplikacij, pa lahko nastane več), hkrati pa je vsako od njih plačljivo[3]. Prednosti in slabosti razvoja mobilnih .NET aplikacij so predstavljene v nadaljevanju.

Prednosti:

- Izdelava aplikacij z močnim IDE Visual Studio.
- Bogata grafična podpora aplikacijam.
- .NET ogrodje pozna veliko število razvijalcev.

Slabosti:

- Ogrodja za razvoj so plačljiva.
- Še vedno je potrebno znati razvijati aplikacije z izvrnimi jeziki kot je Objektni-C. V nasprotnem primeru ne moremo slediti dokumentaciji. Upravičeno se zastavlja vprašanje, ali ni bolje narediti aplikacijo na izvoren način.

3.3.2 Medplatformske aplikacije

Medplatformske aplikacije izdelane z ogrodji, kot so npr. RhoMobile, Titanium, so po funkcionalnosti zelo podobne izvornim aplikacijam. Razvijalcem ni potrebno poznati SDK-jev in API-jev vsake platforme, ampak le SDK in API medplatformskega ogrodja. Oglejmo si prednosti in slabosti razvoja mobilnih aplikacij s pomočjo medplatformskih aplikacij.

Prednosti:

- Izdelava medplatformskih aplikacij je hitra.
- Podpira večino funkcionalnosti mobilne naprave (uporabo kamere, imenika, shranjevanja podatkov, branja iz datoteke, itd.).
- Ob spremembi aplikacije samo enkrat popravimo kodo in generiramo nove datoteke za prenos na spletne trgovine.

Slabosti:

- Večina medplatfomskih orodij podpira samo iOS, Android in BlackBerry platforme.
- Na razvijalskem računalniku je potrebna namestitev razvojnega okolja za vsako platformo posebej.
- Ne podpirajo vseh funkcionalnosti mobilnih naprav.

[3]Več o .NET aplikacijah najdemo v poglavju 3.2.4.1 .NET.

- Ob nadgradnji mobilnega operacijskega sistema nekaj časa traja, da medplatformsko ogrodje posodobi oz. pripravi svoj SDK in API za nove funkcionalnosti.
- Dovoljenje za prenos aplikacije na spletne trgovine je včasih težje pridobiti.
- Potrebno se je naučiti uporabo SDK-ja in API-ja medplatformskega ogrodja.

3.3.3 Hibridne aplikacije

Hibridne aplikacije (*angl. Hybrid App*) so mešanica izvornih in spletnih mobilnih aplikacij. Ta pristop omogoča uporabo prednosti obeh razvojev. V nadaljevanju so predstavljene prednosti in slabosti razvoja hibridnih aplikacij.

Prednosti:

- Hiter razvoj aplikacij za več platform (*praviloma skoraj za vse platforme*) z uporabo HTML, CSS, JavaScript.
- Ni potrebno programirati v izvornih jezikih (*potrebno je izdelati le osnovni projekt za vsako platformo*).
- Z uporabo izvornih API-jev in kontrol se aplikacija obnaša podobno kot izvorna aplikacija.
- Podprtih veliko funkcionalnosti naprave.
- Če je v aplikaciji na nekem mestu potrebno zagotoviti popolno izvorno podporo, je to še vedno mogoče z uporabo izvornega razvoja tistega dela aplikacije.
- Hibridna aplikacija je pravzaprav zavita v izvorni razvoj, zato jo lahko ponudimo na spletni trgovini mobilnih aplikacij.
- Fleksibilnost pri izbiri načina razvoja aplikacije.
- Stroški razvoja aplikacije so cenejši, še posebej v primerjavi z izvornim razvojem.

Slabosti:

- Potrebujemo razvojno okolje vsake platforme.
- Na razvijalskem računalniku je potrebna namestitev razvojnega okolja za vsako platformo posebej.
- Ne podpirajo vseh funkcionalnosti mobilnih naprav.
- Razvite dele aplikacije na izvoren način je potrebno razviti na vsaki platformi posebej.
- Odzivnost aplikacije je slabša (*z delom HTML kode je do izvornih kontrol potreben prehod skozi API, ki upočasni delovanje aplikacije*).

3.3.4 Spletne mobilne aplikacije

Mobilne spletne aplikacije (*angl. Web App*) so najpogostejši način izdelave aplikacij za mobilne naprave z medplatformskimi ogrodji. Aplikacije tečejo na strežniku tako kot spletne strani za namizne računalnike in prenosnike ali pa so integrirane v aplikacijo in za delovanje aplikacije ni potrebna povezava z internetom. Za izvorni izgled aplikacij ogrodja uporabljajo CSS ali druge knjižnice. Prednosti in slabosti spletnih mobilnih aplikacij so predstavljene v nadaljevanju.

Prednosti:

- Izdelava aplikacije z že znanimi pristopi za izdelavo spletnih strani (*uporaba HTML, CSS, JavaScrip*).
- Za izdelavo mobilne aplikacije običajno ne potrebujemo novega kadra, niti niso potrebni veliki stroški za izobraževanje.
- Ni potrebno poznavanje izvornih SDK-jev in API-jev vsake platforme, ampak samo SDK in API medplatformskega ogrodja.
- Izdelava aplikacij je hitra, izgled aplikacij je lahko na vsaki platformi drugačen z uporabo različnih CSS slogov.
- Uporabimo lahko številne knjižnice za pomoč pri izdelavi izgleda aplikacije.

Slabosti:

- Aplikacija podpira manjše število funkcionalnosti naprave.
- Aplikacija ne deluje, če je shranjena na strežniku in mobilna naprava ni povezana z internetom.
- Uporabniki se zavedajo zmogljivosti svoje naprave, zato pričakujejo dobro odzivnost aplikacije.
- Na različnih platformah je podpora integriranega brskalnika drugačna, zato je potreben premislek, kako narediti aplikacijo, da bo delovala na vseh izbranih platformah.
- Razvijalci so omejeni pri razvoju, ker brskalniki ne podpirajo kompleksnejših interakcij z uporabnikom.
- 2D in 3D grafika sta podprti v omejenem obsegu (*samo prikaz na izvornih brskalnikih naprave*). S pomočjo HTML5, Canvas ter drugimi JavaScript grafičnimi knjižnicami lahko relativno dobro razvijamo kompleksnejše igre ter grafične prikaze, vendar zadostno podporo ponuja le izvorni brskalnik na iOS platformi.

3.3.5 Spletne strani prilagojene za mobilne naprave

Spletne strani prilagojene za mobilne naprave (*angl. Mobile Friendly Site*) so podobne spletnim stranem za namizne in prenosne računalnike. Pravzaprav to niso aplikacije, vendar pa kljub temu predstavljajo pomembno vlogo pri prikazovanju vsebine na mobilnih napravah. Shranjene so na strežnikih in dostopne preko URL (*angl. Uniform Resource Locator*) naslova. Glavni poudarek je na prikazu pomembne vsebine, torej brez reklam in animacij, ki bi upočasnjevale prikaz spletne strani in zasedale prostor na majhnem zaslonu mobilne naprave. Največkrat take spletne strani nastanejo iz že obstoječih spletnih strani. Ob prihodu uporabnika na spletno stran "user_agent preveri", če uporabnik dostopa do spletne strani preko mobilne naprave ali preko namiznega računalnika. Temu primerno se prikaže prava spletna stran. Mobilne spletne strani sicer ne moremo primerjati z mobilnimi aplikacijami, kljub vsemu pa izpostavljamo nekaj prednosti in slabosti.

Prednosti:

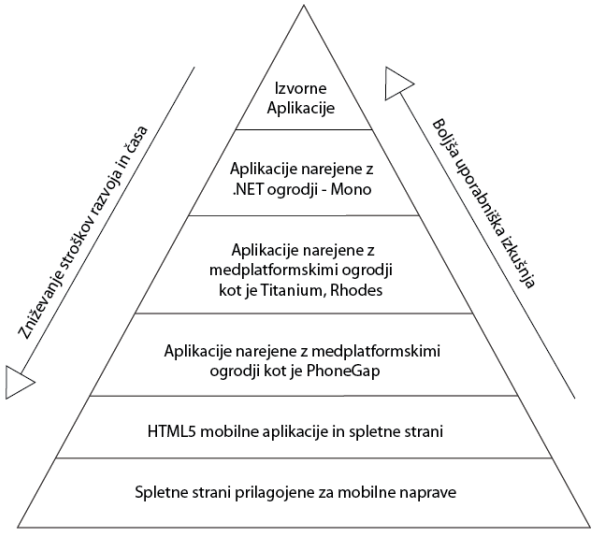
- Enostavna izdelava aplikacije - strani v primerjavi z drugimi pristopi.

Slabosti:

- Ne moremo dostopati do izvornih funkcionalnosti naprave.
- Pri razvoju se ne posveča veliko pozornosti izgledu spletnih strani (*npr. da bi spletna stran izgledala na Android platformi drugače, kot na mobilni napravi z nameščenim iOS operacijskim sistemom*).
- Uporabniki največkrat ne vedo, da bodo ob vpisu naslova spletne strani naleteli na spletno stran prilagojeno mobilni napravi, ampak to opazijo šele takrat, ko se stran naloži.

Izpostaviti še velja, da so tovrstne spletne strani nastale kot dodatna potreba uporabnikov. Vedno več uporabnikov namreč uporablja mobilne naprave (*v primerjavi z namiznimi in prenosnimi računalniki*) in preko le-teh dostopajo tudi do spletnih strani. Ker imajo mobilne naprave drugačne značilnosti kot namizni in prenosni računalniki, so bile potrebne prilagoditve značilnostim mobilnih naprav.

Slika 3.8 prikazuje uporabniške izkušnje, stroške ter čas razvoja aplikacij z uporabo različnih medplatformskih ogrodij.



Slika 3.8: Medplatformska in druga orodja glede na uporabniške izkušnje in glede na stroške ter čas razvoja aplikacij [25].

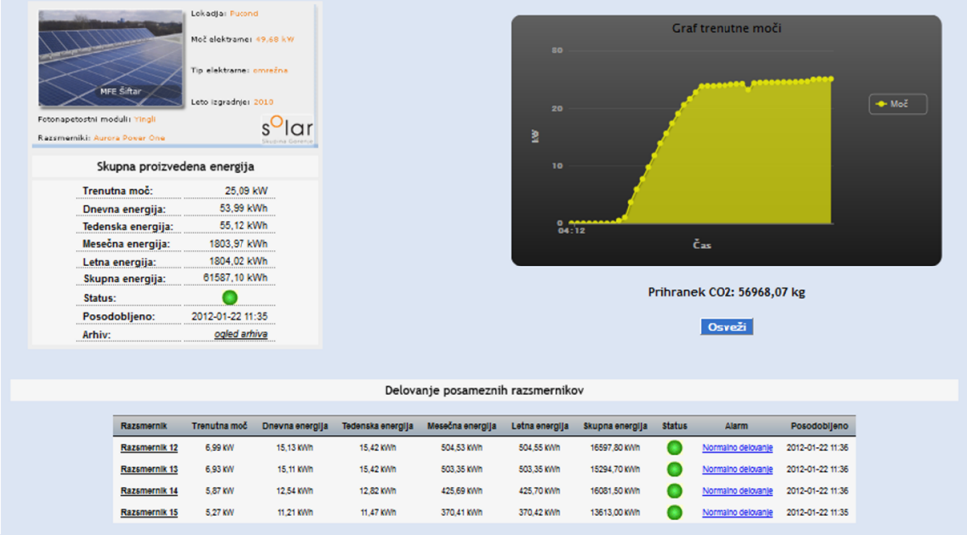
Poglavje 4

Razvoj aplikacije

4.1 Zakaj Appcelerator Titanium?

Za izbiro medplatformskega ogrodja Appcelerator Titanium smo se odločili, ker najbolje ustreza zahtevam za izdelavo želene aplikacije. Sicer podpira razvoj le za Android in iOS platformo (*podpora za BlackBerry je v nastajanju*), vendar je ogrodje brezplačno, Android in iOS platformi pa sta vodilni na trgu (*skupaj obsegata več kot 75% delež trga [34]*). Aplikacije izdelane z Appcelerator Titanium uporabljajo izvorne funkcijonalnosti naprave. Izvorni izgled dosežemo z uporabo izvornega SDK-ja posamezne platforme in ne s CSS-i, kot to velja za večino drugih ogrodij. Aplikacije so zato hitro odzivne in omogočajo boljšo interakcijo z uporabnikom. Ogrodje sicer nekoliko slabše podpira delo z grafiko oz. so moduli za delo z grafiko plačljivi. V nadaljevanju bo razvidno, da je bilo mogoče aplikacijo razviti tudi brez dodatnih modulov. Razvoj z Appcelerator Titanium je preprost tudi, ker za neizkušene razvijalce zelo jasno podaja primere aplikacij v testne namene. Dobro je tudi razloženo, kaj Appcelerator Titanium je in kakšne aplikacije je možno z njim narediti[1]. Podani so primeri obsežnih aplikacij, ki vsebujejo skoraj vse ključne elemente in uporabljajo skoraj vse funkcionalnosti naprave, priložena pa je tudi izvorna koda. Dobro pojasnjena sta tudi SDK in uporaba API-ja. Za dodatne informacije nam je v pomoč skupnost, ki v rubriki vprašanja in odgovori ponuja pomoč razvijalcem, hkrati pa si medsebojno pomoč nudijo tudi razvijalci.

[1] V primerjavi z drugimi ogrodji, kjer je to zelo pomanjkljivo definirano.



Slika 4.1: Spletna stran Fotonapetostni portal, Skupina Gorenje Solar, podjetja MIEL, d.o.o.

4.2 Zahteve mobilne aplikacije

Podjetje MIEL, d.o.o [33] je razvilo spletni portal [20], ki je namenjen spremljanju delovanja fotonapetostnih elektrarn, priključenih na sistem daljinskega nadzora. Spletni portal omogoča spremljanje delovanja posamezne fotonapetostne elektrarne, posameznih fotonapetostnih vej in posameznih razsmernikov. V poljubnih časovnih obdobjih je omogočeno spremljanje merjenja različnih vrednosti (*celotne proizvedene energije fotonapetostne elektrarne, proizvedene energije na letni, mesečni, tedenski ali dnevni ravni ter tudi trenutno moč fotonapetostne elektrarne in drugih različnih podatkov, kot so trenutna napetost, trenutni tok in trenutna frekvenca napetosti*). Vsi podatki se shranjujejo v podatkovno bazo, ki omogoča pregled zgodovine delovanja posameznih fotonapetostnih elektrarn. Prednost spletnega portala je tudi obveščanje o alarmnih stanjih in daljinskem vzdrževanju sistema. Primer prikaza podatkov o delovanju fotonapetostne elektrarne z grafičnim prikazom je predstavljena na sliki 4.1.

Cilj naloge je bila izdelava mobilne aplikacije za platformo Android in iOS, ki bo registriranemu uporabniku, podobno kot na že obstoječi spletni aplikaciji, prikazovala informacije o sončni elektrarni oz. o večjemu številu sončnih elektrarn. Glavni namen mobilne aplikacije je prikazovanje podatkov o proizvedeni energiji, v določnem časovnem obdobju s pomočjo grafičnih prikazov.

Definirane so bile tudi dodatne zahteve:

- aplikacija se mora povezovati na strežnik in od tam pridobivati vse podatke;
- prikazani podatki morajo biti konsistentni s podatki prikazanimi na spletni aplikaciji, ki jo podjetje že uporablja;
- uporabnikovo geslo mora biti v kriptirani obliki in shranjeno na strežniku;
- aplikacija mora biti dvojezična (*v slovenskem in angleškem jeziku*);
- prikazati se mora sporočilo ob povezovanju na strežnik;
- prikazati se mora čas zadnje posodobitve podatkov;
- podatki morajo biti prikazani s pomočjo (*dnevnih, mesečnih, letnih*) grafičnih prikazov;
- aplikacija mora biti oblikovno skladna, dinamična in uporabniku prijazna.

4.2.1 Videz mobilne aplikacije

Na trgu obstaja mobilna aplikacija Sunny portal, SMA Solar Technology AG [50], ki je namenjena prikazu podatkov o proizvedeni energiji sončnih elektrarn. Na sliki 4.2 je prikazana mobilna aplikacija Sunny portal.

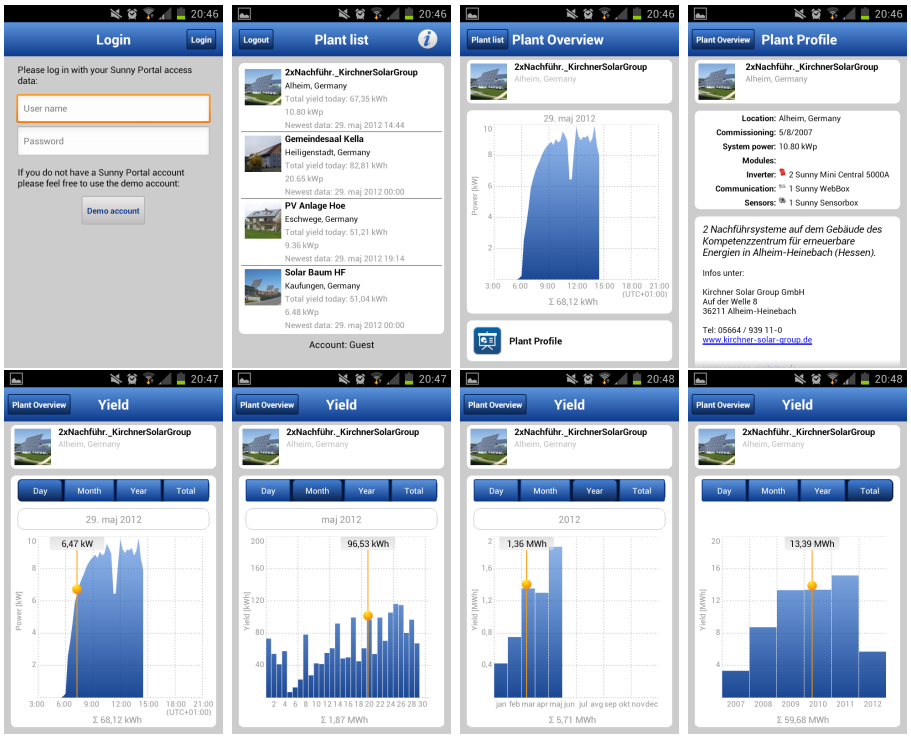
Začetna referenca glede izgleda oz. razvoja lastne rešitve je bila aplikacija zgoraj omenjenega podjetja. Čeprav so bile na ta način vse zahteve glede videza mobilne aplikacije že definirane, so kljub temu predstavljale dodaten izziv. Naslednji korak je predstavljalo kodiranje.

4.3 Razvoj

4.3.1 Komunikacija s strežnikom

Aplikacija za delovanje potrebuje podatke s podatkovne baze, ki je shranjena na strežniku. Za namene spletne aplikacije ima podjetje MIEL, d.o.o. že postavljen Apache spletni strežnik tako, da postavitev ni bila potrebna. Potrebno je bilo narediti nekaj PHP vmesnikov, ki so izvrševali SQL stavke (*glede na prejete parametre iz aplikacije*) nad že obstoječo bazo podatkov. Podatki se aplikaciji vračajo v obliki JSON (*JavaScript Object Notation*). Primer vrnjenega zapisa iz strežnika v JSON obliki:

```
[{"id_elektrarna":"1","naziv":"RPS Gorenje", "naslov":"Velenje",
"right_id":"5","instalirana_moc":"50,0",
"tip_elektrarne":"omre\u017ena","fotonapetostni_moduli":"Yingli",
```



Slika 4.2: Mobilna aplikacija Sunny portal, SMA Solar Technology AG.

```
if (username.value != '' && password.value != '')
{
    loginReq.open("POST",Titanium.App.Properties.getString("naslov_serverja")+"Solar/app.php");
        var params = {
        funkcija: 'login',
        username: username.value,
        password: Titanium.Utils.md5HexDigest(password.value)
    };
    loginReq.send(params);
}
```

Slika 4.3: Primer funkcije za pridobivanje podatkov iz strežnika.

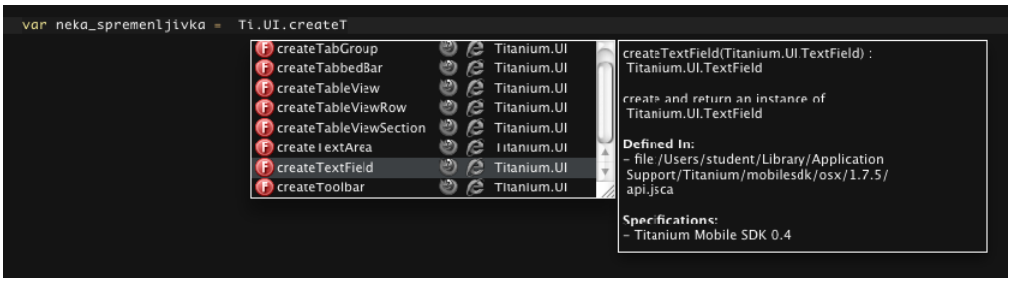

```
"datum_prikljucitve":"2011","tip_razsmernika":"Aurora Power One",
"img_path":"images\/RPS\/rps_mala.gif",
"img_path_big":"images\/RPS\/rps.gif",
"trenutna_moc":"31925.2","updated":"2012-01-22 13:59:14.071",
"dnevna_energija":"148.968"},{"id_elektrarna":"2",
"naziv":"MSPGorenje","naslov":"Sostanj","right_id":"6",
"instalirana_moc":"50,0","tip_elektrarne":"omre\u017ena",
"fotonapetostni_moduli":"Yingli","datum_prikljucitve":"2010",
"tip_razsmernika":"Aurora Power One",
"img_path":"images\/MSP\/msp_mala.gif",
"img_path_big":"images\/MSP\/msp.gif",
"trenutna_moc":"0","updated":"2012-01-22 13:20:38.075",
"dnevna_energija":"0"}]
```


Če SQL stavek ne vrne podatka, se aplikaciji kljub vsemu vrne JSON podatek z določeno vsebino in aplikacija se temu primerno odzove.

V aplikacijo smo dodali funkcijo za pridobivanje podatkov s strežnika. Slika 4.3 prikazuje primer preverjanje uporabniškega imena in gesla uporabnika.

4.3.2 Razvoj aplikacije

Appcelerator Titanium nam, poleg obširne dokumentacije za izdelavo aplikacije, podaja tudi primere, skupaj z izvorno kodo, kjer so v eni aplikaciji zajeti ključni elementi za izdelavo aplikacije. To nam olajša delo, v pomoč pa so nam tudi vprašanja in odgovori na uradni strani appcelerator.com [46], [53]. Titanium Studio IDE nam omogoča pomoč pri kodiranju z namigi (slika 4.4). Prvi korak je predstavljal prikaz osnovnega okna z nekaj teksta, gumbi in vnosnim poljem za uporabniško ime in geslo.



Slika 4.4: Primer prikaza pomoči Titanium Studia pri kodiranju.

Naslednji korak razvoja je bila predstavitev pridobljenih podatkov s strežnika v tabeli. Na ta način je bilo mogoče preveriti pravilnost podatkov [2], s pomočjo katerih se v nadaljevanju izrisujejo grafični prikazi.

Izris grafičnih prikazov podatkov je bil, kot je bilo že omenjeno, nekoliko težaven, saj brezplačna verzija računa za Titanium Studio ne vsebuje modula za risanje po površini. Na voljo sta bili dve možnosti: Izris grafičnega prikaza v oknu brskalnika ali pa izris grafičnih prikazov z dodajanjem posameznih grafičnih elementov (*angl. view*) določene velikosti na osnovno okno. Prva možnost je bila sprva zelo elegantna, saj aplikacija odpre okno brskalnika. Za prikaz grafičnih prikazov na spletnem brskalniku pa obstaja že veliko knjižnic, ki izrisujejo lepe grafične prikaze (*Google charts, JS Charts, Highcharts JS, Raphael JS, Rgraph, PHP Diagram, itd.*). Ker pa je bila ena od zahtev interaktivnost grafičnih prikazov, so se pojavile težave na Android platformi, saj se grafični prikazi niso izrisali, medtem ko so se na iOS platformi prikazovali brez težav. Težave so posledica samega Android OS, ki še ni sposoben prikazovati SVG-ja in naprednejših HTML5 grafičnih prikazov. Možna je samo uporaba ne-interaktivnih grafičnih prikazov, kar pa za našo aplikacijo ni prišlo v poštev. Postopek izrisa grafičnih prikazov je opisan v nadaljevanju.

4.3.3 Izris grafičnih prikazov

V okno aplikacije lahko dodajamo različne grafične elemente (*gumbe, vnosna polja, oznake, poglede, izbirne gumbe, itd.*). Njihov položaj in izgled v oknu določimo s pomočjo atributov (*višine, širine, pozicije, barve, itd.*). Za grafični prikaz podatkov smo uporabili element pogled, ki se v okno dodaja dinamično. Več pogledov skupaj tvori grafični prikaz. Na-

[2]Podatki so bili pravilni, ko so se ujemali s podatki v obstoječi bazi podatkov in so bili zaokroženi na dve decimalni vrednosti.

```
if(id_gumb == 1){
    for (var c=0;c<stevilo_podatkov;c++)
    {
        var datumS = json_3[c].updated;

        var datum_ura = datumS.split(" ");
        var  ura_minuta_sekunda = datum_ura[1].split(":");

        var ura_zaokroena =  Math.round( ura_minuta_sekunda[0] );
        var minuta_zaokroena =  Math.round( ura_minuta_sekunda[1] );
        var minuta_zaokroena_text = minuta_zaokroena;
        if(minuta_zaokroena < 10){
            minuta_zaokroena_text = '0'+minuta_zaokroena;
        }

        var trenutna_moc_xxx = json_3[c].max/1000;
        var trenutna_okrajsana_xxx = trenutna_moc_xxx.toFixed(2);
        var visina = trenutna_okrajsana_xxx * 100 / max_vrednost ;
        var visina_zaokrozena = Math.round(visina  );
        var visina_v_procentu = nastavi_visino(visina_zaokrozena);

        function dobi_casovne_podatke(casovni_polozaj_v_tabeli){
            if(minuta_zaokroena < 20)
            {
                if(tabela_vrednost_po_casu[casovni_polozaj_v_tabeli] == -1){
                    tabela_vrednost_po_casu[casovni_polozaj_v_tabeli] = trenutna_okrajsana_xxx;
                    cas_stolpec[casovni_polozaj_v_tabeli].height = visina_v_procentu;
                    cas_stolpec[casovni_polozaj_v_tabeli].neki = trenutna_okrajsana_xxx;
                    cas_stolpec[casovni_polozaj_v_tabeli].cas = ura_zaokroena+':'+minuta_zaokroena_text;
                    tocke[casovni_polozaj_v_tabeli] = visina_zaokrozena;
                }
             }
             if(minuta_zaokroena > 19 && minuta_zaokroena < 40)
             {
```

Slika 4.5: Del realizacije izračuna višine stolpcev.

slednja faza je razvoj dinamičnega prikazovanja grafičnih elementov, ki bodo sovpadali s podatki in tako tvorili stolpce v grafičnem prikazu. V nadaljevanju bo opisan postopek za prikaz letnega grafičnega prikaza. V prvem koraku je bilo pripravljeno ozadje za grafični prikaz, ki vsebuje sliko ozadja, hkrati pa se tudi omeji maksimalna višina stolpca. Širina stolpcev je izračunana glede na število, ki ga potrebujemo za posamezen grafični prikaz [3]. Naslednji korak je določitev višine stolpcev. S pomočjo maksimalne energije, ki je bila dosežena v nekem mesecu, se določi maksimalna vrednost grafičnega prikaza. To je vrednost na zgornji črti grafičnega prikaza. Glede na maksimalno vrednost grafičnega prikaza se, s pomočjo križnega računa, določi višina vsakega stolpca kar grafični prikaz oziroma prikaz stolpcev osveži. Na sliki 4.5 je prikazan del realizacije izračuna višine stolpcev. Dnevni grafični prikaz, ki prikazuje trenutno moč je zvezen, ima nekaj več dodatkov kot letni grafični prikaz. Na začetku je, s pomočjo stolpcev, izrisan na enak način kot letni

[3]Npr. za prikaz letnega grafičnega prikaza potrebujemo dvanajst stolpcev, za vsak mesec posebej. Širina stolpca je tako 8% (100% - (12 * 8%) = 96%). Ostanejo 4%, kar je razporejeno na začetek in konec grafa. Vsak stolpec se oblikuje ob klicu funkcije za izris letnega grafičnega prikaza s pomočjo stavka »for«. Stolpcu se določi višina 0% in levi odmik. Vsak naslednji stolpec je od levega roba grafa oddaljen za število že oblikovanih stolpcev + 2%.

grafični prikaz. Stolpcev je 48, zato se podatki prikazujejo na dvajset minutnem intervalu. Širina stolpca je 2%. Stolpci so popolnoma prosojni in se jih na grafičnem prikazu ne vidi, čeprav so prisotni. Vsakemu stolpcu se doda tudi vrednost trenutne moči. Če podatka na tem intervalu ni, je na tem mestu vrednost stolpca -1. Za izris zveznega, vidnega grafičnega prikaza, pa je uporabljena višina in levi odmik stolpcev (*ki sicer niso vidni*). Z upoštevanjem teh vrednosti in z upoštevanjem resolucije zaslona, na kateri teče aplikacija, se izračunajo še koordinate najvišje in najbolj leve točke vsakega stolpca posebej. Vse točke se po vrsti zapišejo v tabelo, nato se izvrši funkcija za izris točk. Izrišejo se novi - vidni stolpci širine ene slikovne točke. Ker je širina stolpca samo ena slikovna točka, je graf viden kot zvezen. Ta funkcija vzame po vrsti dve točki iz tabele in nariše povezavo med njima. Točke so po širini vedno enako oddaljene, razlikujejo pa se lahko po višini. Del med dvema točkama se izriše s pomočjo "for" zanke. Ta kreira toliko elementov, koliko jih je potrebno za vsako resolucijo, da pride od ene točke do druge. Za vsako slikovno točko se ustrezno določi višina elementa. Grafični prikaz je izrisan, ko funkcija obhodi vse točke.

4.3.3.1 Prikaz črte, kroglice in energije na grafičnem prikazu

Grafični prikaz vsebuje tudi funkcijo prikaza izbranega podatka. Npr. v mesečnem grafičnem prikazu, kjer je 31 stolpcev [4], nas zanima koliko energije je izbrana elektrarna proizvedla na šesti dan v mesecu. Ena rešitev je izpis vrednosti stolpca s pomočjo klika, vendar je na manjših, pa tudi večjih zaslonih, težko določiti želeni stolpec. Rešitev omogočata interaktivna črta in kroglica, ki se izrišeta in premikata, če se dotaknemo grafičnega prikaza na zaslonu. Pri rešitvi nam pomaga funkcija `touchmove`, ki vrača informacije o koordinatah dela ekrana, kjer je zaznana akcija. Ker je ta funkcija definirana le za območje grafičnih prikazov, se le na tem mestu, s pomočjo x koordinate, nariše vertikalna črta. To je grafični element širine ene slikovne točke in je obarvan z zeleno barvo. Tudi kroglica je grafični element, s sliko ozadja zelene kroglice. Izriše se na podoben način kot črta, le da se kroglica premika tudi glede na višino stolpca. Višina stolpca in kroglica nista neposredno povezana, povezana pa sta s pomočjo levega odmika od roba grafa. Tako lahko s pomočjo x koordinate črte izračunamo položaj kroglice in, če se ujema s položajem stolpca, se ustrezno določimo višino kroglice. Na ta način je določen tudi položaj stolpca. Nad grafičnim prikazom se izpiše še njegova vrednost, ki predstavlja proizvedeno energijo.

Na dnevnem grafičnem prikazu je prikaz kroglice identičen, saj so v ozadju narisani prozorni stolpci, zato se višina kroglice spreminja glede na njihove vrednosti.

Na sliki 4.6 je prikazan del realizacije premikanja črte in kroglice po grafičnem prikazu ter

[4] Za 31 dni v mesecu.

```
view_za_graf_dan_po_casu.addEventListener('touchmove', function(e)
{
    crta_dan_po_casu.left = e.x;
    kroglica_dan_po_casu.left = (e.x - kroglico_na_sredino );

        if(     (  e.x >= (interval -100 )   ) &&     (   e.x < (interval )  )       ){
            crta_dan_po_casu.hide();
            kroglica_dan_po_casu.hide();
            label_za_vrednost_nad_crto_dan_po_casu.hide();
        }

        else if(     (  e.x >= (interval)   ) &&     (   e.x < (interval * 2)  )       ){
            label_za_vrednost_nad_crto_dan_po_casu.left = (e.x - (interval * 5) );
            kroglica_se_premika(0);
        }
        else if(     (  e.x >= (interval * 2 )   ) &&     (   e.x < (interval * 3)  )       ){
            label_za_vrednost_nad_crto_dan_po_casu.left = (e.x - (interval * 5) );
            kroglica_se_premika(1);
        }
        else if(     (  e.x >=(interval * 3 )   ) &&     (   e.x < (interval * 4)  )       ){
```

Slika 4.6: Del realizacije premikanja črte in kroglice po grafičnem prikazu ter klic funkcije za izpis vrednosti nad grafičnim prikazom.

klic funkcije za izpis vrednosti nad grafičnim prikazom. Prikaz črtice in kriglice na oknu aplikacije vidimo na sliki 4.8 (prikaza 4 in 5). Slika 4.7 prikazuje realizacijo premikanja kroglice po grafičnem prikazu.

4.3.4 Razlike v kodiranju za iOS in Android

Vsaka platforma ima svoje zahteve, zato je bilo pri kodiranju včasih potrebno napisati kodo posebej za Android in posebej za iOS. Aplikacija s pomočjo funkcije zazna, na kateri platformi teče. S pomočjo te funkcije dobimo podatek o platformi. Tako se izvrši pravi del kode za določeno platformo.

Največja razlika v kodiranju je pri kreiranju dnevnih grafičnih prikazov. Grafični prikaz je na iOS platformi realiziran s pomočjo stolpcev in ni prikazan zvezno. Kot je bilo že omenjeno, obstajajo razlike v obnašanju API-ja na različnih platformah, od tega pa je odvisno upravljanje s pomnilnikom. Na iOS platformi se pomnilnik pri izrisu grafičnega prikaza ne sprosti vedno, zato se aplikacija ustavi.

Druge manjše razlike so še:

- platformi shranjene podatke s pomočjo datoteke shranjujeta na različnih lokacijah zato ni možna uporabe ene funkcije za dostop do shranjenih podatkov.
- izbirni gumb (*angl. check box*), ki nakazuje, ali naj si aplikacija zapomni uporabnika ali ne, je različen oziroma izviren za vsako platformo posebej. Posebnost pa je ta, da zopet

```
function kroglica_se_premika(indexkrogle){
    var temp = cas_stolpec[indexkrogle].height;
        if(temp == 0){
            kroglica_dan_po_casu.bottom = '0%';
        }
        else{
            if(android){
                kroglica_dan_po_casu.bottom = cas_stolpec[indexkrogle].height;
            }
            else{
                var visina_kroglice = nastavi_visino(cas_stolpec[indexkrogle].height);
                kroglica_dan_po_casu.bottom = visina_kroglice;
            }
        }
        if(cas_stolpec[indexkrogle].neki == -1){
            crta_dan_po_casu.show();
            label_za_vrednost_nad_crto_dan_po_casu.hide();
            kroglica_dan_po_casu.hide();
        }
        else{
            if(Titanium.App.Properties.getBool("SLO") == true){
                label_za_vrednost_nad_crto_dan_po_casu.text =
                'Čas '+cas_stolpec[indexkrogle].cas +'\nMoč '+ cas_stolpec[indexkrogle].neki+' kW';
            }
            else {
                label_za_vrednost_nad_crto_dan_po_casu.text =
                'Time '+cas_stolpec[indexkrogle].cas +'\nPower '+ cas_stolpec[indexkrogle].neki+' kW';
            }
            crta_dan_po_casu.show();
            label_za_vrednost_nad_crto_dan_po_casu.show();
            kroglica_dan_po_casu.show();
        }
}
```

Slika 4.7: Del realizacije premikanja kroglice po grafičnem prikazu.

ne moremo uporabiti ene funkcije. Za to mora razvijalec poskrbeti sam in napisati kodo posebej za Android platformo kot tudi za iOS.

- gumbi so izvirni za vsako platformo posebej, vendar pa gumb na Android platformi potrebuje dodatne parametre. To pomeni, da mora razvijalec ponovno zapisati "if" stavek za Android z dodatnimi parametri.

4.4 Primer uporabe aplikacije

Predpostavimo, da je uporabnik registriran in ima v lasti sončno elektrarno.

- Uporabnik zažene aplikacijo, vpiše uporabniško ime in geslo ter pritisne gumb "vpis". Prikažejo se vse elektrarne do katerih ima dostop ter nekaj podatkov o vsaki elektrarni.
- Uporabnik pritisne na določeno elektrarno, za katero želi podrobnejše podatke.
- Odpre se drugo okno, kjer se prikaže grafični prikaz trenutne moči tega dne, nekaj osnovnih podatkov o elektrarni in gumbi za nadaljnji izbor.
- V kolikor uporabnik želi vpogled v podatke o zgodovini pridobljene energije te elektrarne, pritisne na gumb "zgodovina".

Slika 4.8: Aplikacija na Android in iOS (iPad in iPhone) platformi.

- Prikaže se dnevni grafični prikaz trenutne moči, osnovni podatki o elektrarni in gumbi za izbiro grafičnega prikaza, gumba za premikanje naprej in nazaj po časovnici, na voljo pa ima tudi izbiro datuma, kjer lahko določi datum želenih podatkov za prikaz. Za izbiro grafičnih prikazov ima na voljo dnevni grafični prikaz, mesečni grafični prikaz, letni grafični prikaz in grafični prikaz skupne proizvedene energije skozi vsa leta.

- Uporabnik pritisne na grafični prikaz, da se mu prikaže pokončna črta. Na ta način se lažje premika po grafičnem prikazu in izbere določen podatek.

Slika 4.8 prikazuje končni videz aplikacije na Android platformi (*prikazi od 1 do 5), iOS platformi (prikaza 6 in 7 iPad, prikaz 8 iPhone*). V tem poglavju je bila predstavljena mobilna aplikacija izdelana s pomočjo medplatformskega ogrodja Appcelerator Titanium. Aplikacija deluje na platformi Android in iOS, ter prikazuje podatke o proizvedeni energiji, v določnem časovnem obdobju, s pomočjo grafičnih prikazov. Vse zastavljene zahteve so bile uspešno realizirane. Največjo težavo je predstavljal izris grafičnih prikazov, ki so sicer ključni elementi aplikacije. Rešitev sta predstavljala spretnost in matematično znanje, pridobljeno na fakulteti, s pomočjo katerih je bilo izdelanih kar nekaj funkcij s katerimi

se izrisujejo grafični prikazi glede na pridobljene podatke. Ta rešitev je bila ustrezna za večji del grafičnih prikazov, dodatne težave pa so se pojavile pri prikazovanju dnevnega grafičnega prikaza na iOS platformi. Rešitev so predstavljali stolpci, ki se prikazujejo namesto zveznega grafa. Aplikacije še ne najdemo na App Store in Google Play, kar je sicer eden izmed ciljev za prihodnost.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo prikazali razvoj aplikacij na izvoren način in z uporabo medplatformskih ogrodij. V začetku smo predstavili zgodovinski razvoj pametnih mobilnih naprav, saj na ta način lažje napovemo dogajanje na trgu v prihodnosti in lažje izberemo medplatformsko ogrodje za razvoj mobilne aplikacije. Predstavili smo številna ogrodja in knjižnice, ki so trenutno dostopna na trgu. Izpostavljene so razlike med njimi, posebej pa smo se osredotočili na nekaj izbranih medplaformskih ogrodij.

Izdelava aplikacij za pametne mobilne naprave predstavlja vedno bolj pomemben produkt številnih podjetij doma in po svetu. Ob stalnem variiranju deležev na trgu pametnih mobilnih naprav različnih proizvajalcev in vedno večjemu številu operacijskih sistemov, ki so nameščeni na napravah, se je pojavila težava, kako zagotoviti kar najbolj ugoden razvoj aplikacij za čim več platform. Raziskava medplatformskih ogrodij in knjižnic je tako lahko v veliko pomoč nekomu, ki želi razvijati mobilne aplikacije z uporabo medplatformskega ogrodja.

Ugotovili smo, da pogosta izbira medplatformskega ogrodja ni primerna. Zlasti vprašljiva je izbira medplatformskega ogrodja pri razvoju aplikacije, ki mora biti zelo odzivna. Kot primer lahko vzamemo predstavljeno aplikacijo "Sončne elektrarne", ko se pri prikazovanju grafičnega prikaza dotaknemo zaslona in s premikanjem po zaslonu pridobivamo želene podatke. Izkušnje s prikazom na HTML način in z uporabo JavaScript knjižnic za prikazovanje grafičnih prikazov kažejo, da bi prikazovanje podatkov na tak način potekalo zelo počasi.

Vprašljivo je tudi, kako je smiselno uporabiti medplatformsko ogrodje, ki podpira razvoj za največ dve mobilni platformi. Hkrati je vprašljiva tudi podpora razvijalcem, ki

uporabljajo brezplačna ogrodja, včasih pa ogrodja celo vsebujejo veliko hroščev, kar lahko vpliva na končni izdelek. Nejasna je tudi uporaba API-jev ogrodja, kar je tudi posledica iskanja tržnih deležev ponudnikov medplatformskih ogrodij. Pogosto se zgodi, da je teoretično sicer mogoče določene rešitve realizirati z ogrodjem, vendar v praksi pogosto ugotovimo, da so težave z implementacijo dela programa. Kot konkreten primer lahko navedemo težavo pri branju z datoteke z uporabo medplatformskega ogrodja Titanium Appcelerator. Branje naj bi bilo realizirano z uporabo funkcije: `Ti.Filesystem.getFile (Ti.Filesystem.applicationDataDirectory, 'text.txt')`, kot je to zapisano v dokumentaciji, vendar funkcija ne deluje. Naj še dodamo, da so medplatformska ogrodja v razvoju vedno korak ali dva za operacijskim sistemom pametne mobilne naprave. To pomeni, da ob prihodu nove verzije določenega mobilnega operacijskega sistema medplatformska ogrodja potrebujejo relativno veliko časa, da implementirajo novosti novih mobilnih operacijskih sistemov.

Medplatformska ogrodja se neprestano izboljšujejo, s tem pa raste tudi cena razvojnemu ogrodju (*predhodno brezplačno ogrodje lahko postane plačljivo*). Tako npr. Appcelerator Titanium za enkrat še omogoča brezplačni razvoj, vendar je za pričakovati, da bo tudi to ogrodje enkrat v prihodnosti postalo plačljivo.

Veliko težav pri ponudbi aplikacije na trg pa predstavljajo tudi licence razvojnega ogrodja. Te niso vedno konkretno definirane, zato ne vemo, ali lahko aplikacijo ponudimo na trg brez dodatnega plačila podjetju, ki je razvilo uporabljeno ogrodje za razvoj aplikacije.

Zastaviti si je potrebno tudi vprašanje, kaj se bo zgodilo z našimi aplikacijami, če podjetje medplatformskega ogrodja propade. Tak primer je npr. OpenPlug Studio [44].

Produkt te diplomske naloge je tudi mobilna aplikacija Sončne elektrarne, ki jo uporablja podjetje MIEL, d.o.o.. Razvili smo jo s pomočjo medplatformskega ogrodja Appcelerator Titanium, ki je predstavljal najboljšo izbiro. Na koncu se je pokazalo, da je bila izbira ogrodja dobra. Kljub vsemu pa je pred izbiro ogrodja potrebno preveriti vse ključne elemente aplikacije, če jih ogrodje podpira. Npr. od meseca maja 2012 je uporabnikom Android OS na voljo nadgradnja operacijskega sistema na verzijo 4.0.3. Appcelerator Titanium trenutno podpira razvoj do verzije 3.x.x. Nadgradnja sistema je razkrila, da ima mobilna aplikacija težave pri pridobivanju podatkov s strežnika v primeru 3G omrežja. Pred nadgradnjo Android operacijskega sistema z verzije 2.3.5 na 4.0.3 teh težav ni bilo.

Ena od zahtev aplikacije je bil tudi prikaz podatkov s pomočjo interaktivnih grafičnih prikazov. Pri tem so se pojavile težave na Android platformi, saj se grafični prikazi niso

izrisali, medtem ko so se na iOS platformi prikazovali brez težav. Težave so posledica samega Android OS, ki še ni sposoben prikazovati SVG-ja in naprednejših HTML5 vsebin na gradniku WebView. Ni bilo mogoče uporabiti tudi številnih napisanih knjižnic, ki so prosto dostopne na spletu. Mogoča je bila uporaba ne-interaktivnih grafičnih prikazov, kar pa za našo aplikacijo ni prišlo v poštev. Težavo smo odpravili z izdelavo funkcij, ki dinamično dodajajo "View" elemente na okno aplikacije glede na pridobljene podatke s strežnika in tako nastaja grafični prikaz podatkov. Ta pristop se je pokazal za zelo dobrega. Prikaz ni odvisen od drugih knjižnic, kar tudi pomeni, da lahko grafične prikaze spreminjamo po naših željah.

Kljub nekaterim nevšečnostim pri izdelavi aplikacije, smo vse zahteve realizirali. Aplikacija je v fazi testiranja, pozitivni odzivi pa prihajajo tudi s strani uporabnikov aplikacije.

Raziskavo bi lahko izboljšali s poglobljenim vpogledom v plačljiva medplatformska ogrodja.

Literatura

[1] (2004) E. Dasque, "The Mono Development Platform". Dostopno na: http://www.willydev.net/descargas/WillyDEV_MonoUmass.pdf.

[2] (2007) R. Kumar Swaminathan, "Cross-Platform Application Development using .NET and Mono". Dostopno na: http://meetrajesh.com/publications/work_reports/report2.pdf.

[3] (2010) P. Ferrill, "Developing Cross-Platform C# Applications with Mono". Dostopno na: http://www.codeguru.com/csharp/article.php/c17535/Developing-CrossPlatform-C-Applications-with-Mono.htm.

[4] (2010) J. Rowberg, "Comparison: App Inventor, DroidDraw, Rhomobile, PhoneGap, Appcelerator, WebView, and AML". Dostopno na: http://www.amlcode.com/2010/07/16/comparison-appinventor-rhomobile-phonegap-appcelerator-webview-and-aml.

[5] (2011) G. Hartmann, G. Stead, A. DeGani, "Cross-platform mobile development". Medical Mobile Development Project: D4. Dostopno na: http://www.mole-project.net/images/documents/deliverables/WP4_crossplatform_mobile_development_March2011.pdf.

[6] (2012) AdobeFlex. Dostopno na: http://www.adobe.com/products/flex.html.

[7] (2012) Android. Dostopno na: http://www.android.com/developers.

[8] (2012) App Inventor. Dostopno na: http://www.appinventor.mit.edu.

[9] (2012) AppCat. Dostopno na: http://app.cat.

[10] (2012) Appcelerator Titanium. Dostopno na:
http://www.appcelerator.com.

[11] (2012) Apple. Dostopno na:
https://developer.apple.com.

[12] (2012) Application Markup Language. Dostopno na:
http://www.amlcode.com.

[13] (2012) Applicationcraft. Dostopno na:
http://www.applicationcraft.com.

[14] (2012) AppMobi. Dostopno na:
http://www.appmobi.com.

[15] (2012) AppsBuilder. Dostopno na:
http://www.apps-builder.com.

[16] (2012) Blackberry. Dostopno na:
http://us.blackberry.com.

[17] (2012) Corona. Dostopno na:
http://www.anscamobile.com.

[18] (2012) Eclipse. Dostopno na:
http://www.eclipse.org.

[19] (2012) Feedhenry. Dostopno na:
http://feedhenry.com.

[20] (2012) Fotonapetostni portal, Skupina Gorenje Solar. Dostopno na:
zaradi zaupnosti podatkov je naslov spletne strani dostopen pri predstavnikih podjetja MIEL, d.o.o.

[21] (2012) GameTeam. Dostopno na:
http://gameteam.fri.uni-lj.si.

[22] (2012) Gideros mobile. Dostopno na:
http://www.giderosmobile.com.

[23] (2012) Hello, Android. Dostopno na:
http://docs.xamarin.com/android/getting_started/hello_world.

[24] (2012) iWebKit. Dostopno na:
http://snippetspace.com.

[25] (2012) J. Sunil, "Comparison Chart for Mobile App Development Methods". Dostopno na:
http://www.alliancetek.com/downloads/article/comparison-chart-for-mobile-app.pdf.

[26] (2012) Jo. Dostopno na:
http://joapp.com.

[27] (2012) jQTouch. Dostopno na:
http://www.jqtouch.com.

[28] (2012) jQuery Mobile. Dostopno na:
http://jquerymobile.com.

[29] (2012) JULY Systems. Dostopno na:
http://julysystems.com.

[30] (2012) Lua. Dostopno na:
http://www.lua.org/about.html.

[31] (2012) Marmalade. Dostopno na:
http://www.madewithmarmalade.com.

[32] (2012) W. B. McClure, "Android Development for .NET/C# Developers with Mono for Android". Dostopno na:
http://www.devconnections.com/updates/McClure_updates_Introduction%20to%20MonoDroid.pdf.

[33] (2012) MIEL Elektronika, d.o.o.. Dostopno na:
http://www.miel.si.

[34] (2012) Mobile operating system. Dostopno na:
http://en.wikipedia.org/wiki/Mobile_operating_system.

[35] (2012) MobiOne Studio. Dostopno na:
http://www.genuitec.com/mobile.

[36] (2012) Mono Touch. Dostopno na:
http://xamarin.com/monotouch.

[37] (2012) Mono. Dostopno na:
http://www.mono-project.com/Main_Page.

[38] (2012) Monocross. Dostopno na:
http://itr-mobility.com/products/monocross.

[39] (2012) MoSync. Dostopno na:
http://en.wikipedia.org/wiki/MoSync.

[40] (2012) MoSync. Dostopno na:
http://www.mosync.com.

[41] (2012) Netbiscuits. Dostopno na:
http://www.netbiscuits.com.

[42] (2012) Nimblekit. Dostopno na:
http://nimblekit.com/index.php.

[43] (2012) NS Basic/App Studio. Dostopno na:
http://www.nsbasic.com/app.

[44] (2012) OpenPlug Studio. Dostopno na:
http://www.open-plug.com.

[45] (2012) PhoneGap. Dostopno na:
http://phonegap.com.

[46] (2012) Q&A Titanium Studio. Dostopno na:
http://developer.appcelerator.com/questions.

[47] (2012) Resco. Dostopno na:
http://www.resco.net.

[48] (2012) RhoMobile. Dostopno na:
http://www.rhomobile.com.

[49] (2012) Sencha Touch. Dostopno na:
http://www.sencha.com/products/touch.

[50] (2012) SMA Solar Technology AG – Sunny portal. Dostopno na:
http://www.sunnyportal.com/Templates/Start.aspx.

[51] (2012) Smartface Designer. Dostopno na:
http://www.mobinex.biz/smartface-designer.html.

[52] (2012) Smartphone. Dostopno na:
http://en.wikipedia.org/wiki/Smartphone.

[53] (2012) Stackverflow. Dostopno na:
http://stackoverflow.com.

[54] (2012) TapLynx. Dostopno na:
http://www.taplynx.com.

[55] (2012) Unity. Dostopno na:
http://unity3d.com/unity.

[56] (2012) Verivo. Dostopno na:
http://www.verivo.com.

[57] (2012) WebApp.Net. Dostopno na:
http://webapp-net.com.

[58] (2012) WebKit Open Source Project. Dostopno na:
http://www.webkit.org.

[59] (2012) Windows Mobile. Dostopno na:
http://create.msdn.com/en-US.

[60] (2012) Wink. Dostopno na:
http://xuijs.com.

[61] (2012) XUI. Dostopno na:
http://xuijs.com.

[62] (2012) Zepeto.js. Dostopno na:
http://zeptojs.com.

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

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

[65] M. Bluestein, *"Learning MonoTouch: A Hands-On Guide to Building iOS Applications with C# and .NET"*, New York: Addison-Wesley, 2011.

[66] W. M. Lee, *"Beginning Android Application Development"*, Indianapolis Wiley Publishing, Inc., 2011.

[67] A. Lunny, *"PhoneGap Beginner's Guide"*, Birmingham: Packt Publishing, 2011.

[68] M. Mamone, *"Migrating to iPhone and iPad for .NET Developers"*, New York: Apress, 2011.

[69] W. B. McClure, R. Blyth, C. Dunn, C. Hardy, M. Bowling, *"Professional iPhone Programming with MonoTouch and .NET/C# (Wrox Programmer to Programmer)"*, Indianapolis: Wiley Publishing, Inc., 2010.

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

[71] B. Pollentine, *"Appcelerator Titanium Smartphone App Development Cookbook"*, Birmingham: Packt Publishing, 2011.

[72] S. N. Potežica, *"Razvoj aplikacij za pametne telefone"*, diplomska naloga, Fakulteta za računalništvo in informatiko: Ljubljana, 2009, str. 9-11.

[73] D. Wolber, "App Inventor and Real-World Motivation", v zborniku *"Proceedings of the 42nd ACM technical symposium on computer science education"*, New York, ZDA, 2011, str. 601-606.

[74] D. Wolber, H. Abelson, E. Spertus, L. Looney, *"App Inventor: Create Your Own Android Apps"*, Sebastopol: O'Reilly Media, 2011.

[75] P. Zheng, L. M. Ni, *"Smart Phone and Next Generation Mobile Computing"*, San Francisco: Morgan Kaufmann, 2005.