Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Primož Bajželj

Uvajanje mobilnih informacijskih poslovnih rešitev v gozdarstvu

diplomsko delo
na univerzitetnem študiju

Mentor: prof. dr. Denis Trček

Ljubljana, 2013

Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 01935 / 2013
Datum: 3.9.2013

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

Kandidat: **PRIMOŽ BAJŽELJ**

Naslov: **UVAJANJE MOBILNIH INFORMACIJSKIH POSLOVNIH REŠITEV V GOZDARSTVU**
IMPLEMENTING MOBILE INFORMATION BUSSINESS SOLUTIONS IN FORESTRY

Vrsta naloge: DIPLOMSKO DELO UNIVERZITETNEGA ŠTUDIJA

Tematika naloge:

V gospodarstvu je ključnega pomena nenehno izpopolnjevanje poslovnih procesov, kjer je tehnologija tista, ki nam to omogoča v veliki meri. Predvsem z razvojem mobilnih naprav in tehnologije zanje jo lahko uporabljamo tudi na mestih, kjer to prej ni bilo mogoče. V diplomski nalogi predstavljamo izdelavo mobilne aplikacije od idejne zasnove do prototipa. Mobilna aplikacija je namenjena uporabi v gozdarstvu in naj bi omogočala optimizacijo ter izboljšanje poslovnih procesov. Najprej bomo predstavili zahteve, ki jim sledi opis razpoložljivih razvojnih okolij, orodij in ostalih tehnologij, uporabljenih pri izdelavi prototipa. V nadaljevanju pa bomo opisali sam razvoj, prikazali prototip ter opisali tehnično ozadje delovanja.

Mentor:

prof. dr. Denis Trček

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Primož Bajželj, z vpisno številko **63080047**, sem avtor diplomskega dela z naslovom:

Uvajanje mobilnih informacijskih poslovnih rešitev v gozdarstvu

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Denisa Trčka,
- 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 11. decembra 2013 Podpis avtorja:

Iskreno se zahvaljujem mentorju prof. dr. Denisu Trčku, družini in prijateljem, ki so mi s podporo, nasveti in spodbujanjem ves čas stali ob strani med celotnim študijem in izdelavo diplomskega dela.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Zahteve **5**

3 Uporabljene tehnologije **7**

3.1 Mobilni računalnik 7
3.2 NetBeans 8
3.3 SDK 9
3.4 JDK 9
3.5 Android SDK 10
3.6 Ant 11
3.7 Google USB driver 11
3.8 NBAndroid 11
3.9 PhoneGap 12
3.10 Weinre 14
3.11 Aardwolf 14
3.12 Ripple 15
3.13 HTML 15
3.14 CSS 16
3.15 JavaScript 17

3.16 PouchDb 23

4 Mobilna aplikacija **27**

4.1 Izbira tehnologij in razvoj 27

4.2 Izdelava vmesnika 28

4.3 Organizacija podatkov 31

4.4 Ključni vnosi 32

4.5 Informacijski sistem in ključne povezave 40

5 Zaključek **45**

Povzetek

V gospodarstvu je ključnega pomena nenehno izpopolnjevanje poslovnih procesov, kjer je tehnologija tista, ki nam to omogoča v veliki meri. Predvsem z razvojem mobilnih naprav in tehnologije zanje jo lahko uporabljamo tudi na mestih, kjer to prej ni bilo mogoče. V diplomski nalogi predstavljamo izdelavo mobilne aplikacije od idejne zasnove do prototipa. Mobilna aplikacija je namenjena uporabi v gozdarstvu in naj bi omogočala optimizacijo ter izboljšanje poslovnih procesov. Najprej bomo predstavili zahteve, ki jim sledi opis razpoložljivih razvojnih okolij, orodij in ostalih tehnologij, uporabljenih pri izdelavi prototipa. V nadaljevanju pa bomo opisali sam razvoj, prikazali prototip ter opisali tehnično ozadje delovanja.

Ključne besede

mobilne aplikacije, tehnologija v gozdarstvu, PhoneGap

Abstract

Information technology plays an important role in constant improvement of business processes. Especially with development of mobile devices and corresponding technology it is possible to use technology in contexts where this was impossible before. This diploma work describes a development of a mobile application from its idea to a prototype. It is intended to be used in forestry where it should enable optimization and improvement of business processes. First, requirements, development environments, tools and other technologies that were used for prototype development are described. These are followed by description of development, prototype and description of technical background of operations.

Keywords

mobile applications, forestry technology, PhoneGap

Poglavje 1

Uvod

Trenutno je informacijska tehnologija v razcvetu in zdi se, da lahko z njeno pomočjo naredimo prav vse. Prebija se v naš vsakdanjik, predvsem s porastom pametnih mobilnih naprav, ki so znatno vplivale na povezanost ljudi s tehnologijo. Tehnologija nam omogoča tudi napredek in prednosti na različnih poslovnih področjih, kjer je zelo pomembno, da sledimo trendom in razvoju, saj lahko tako ohranjamo ali celo pridobivamo konkurenčno prednost.

Eno izmed področij, kjer je veliko možnosti za izboljšave, je tudi gozdarstvo. Največ možnosti za izboljšave in napredek je z olajšanjem dela na terenu. Trenutna praksa na terenu je, da izmerimo vsak hlod posebej ter ga vpišemo v za to pripravljen list. Po izmeri vseh hlodov sledi računanje volumna, kjer si lahko pomagamo s posebej pripravljeno tabelo, katere primer lahko vidimo na sliki 1.1. Na koncu moramo vse skupaj še sešteti, da dobimo končne količine, ki jih potrebujemo za dobavnico. To delo je lahko zelo zamudno in predstavlja osrednji problem diplomskega dela.

V letu izdelave diplomske naloge je država izdala poseben obrazec, ki ga lahko vidimo na sliki 1.2. Za izpolnitev obrazca moramo ponovno izračunati skupne količine, saj obrazec za razliko od dobavnice predvideva združevanje po drugih kriterijih, s čimer je delo na terenu dodatno obremenjeno.

Predlagamo rešitev za izboljšanje omenjenega procesa z uporabo mobilne

cm	1m	2m	3m	4m	5m	6m	7m	8m	9m	10m
20	3	6	9	13	16	19	22	25	28	31
21	3	7	10	14	17	21	24	28	31	35
22	4	8	11	15	19	23	27	30	34	38
23	4	8	12	17	21	25	29	33	37	42
24	5	9	14	18	23	27	32	36	41	45
25	5	10	15	20	25	29	34	39	44	49
26	5	11	16	21	27	32	37	42	48	53
27	6	11	17	23	29	34	40	46	52	57
28	6	12	18	25	31	37	43	49	55	62
29	7	13	20	26	33	40	46	53	59	66
30	7	14	21	28	35	42	49	57	64	71
31	8	15	23	30	38	45	53	60	68	75
32	8	16	24	32	40	48	56	64	72	80
33	9	17	26	34	43	51	60	68	77	85
34	9	18	27	36	45	54	64	73	82	91
35	10	19	29	38	48	58	67	77	87	96
36	10	20	31	41	51	61	71	81	92	102
37	11	21	32	43	54	64	75	86	97	107
38	11	23	34	45	57	68	79	91	102	113
39	12	24	36	48	60	72	84	96	107	119
40	13	25	38	50	63	75	88	100	113	126
41	13	26	40	53	66	79	92	106	119	132
42	14	28	42	55	69	83	97	111	125	138
43	15	29	44	58	73	87	102	116	131	145
44	15	30	46	61	76	91	106	122	137	152
45	16	32	48	64	79	95	111	127	143	159
46	17	33	50	66	83	100	116	133	149	166
47	17	35	52	69	87	104	121	139	156	173
48	18	36	54	72	90	109	127	145	163	181
49	19	38	57	75	94	113	132	151	170	188
50	20	39	59	79	98	118	137	157	177	196
51	20	41	61	82	102	123	143	163	184	204
52	21	42	64	85	106	127	149	170	191	212
53	22	44	66	88	110	132	154	176	198	221
54	23	46	69	92	114	137	160	183	206	229
55	24	47	71	95	119	142	166	190	214	237
56	25	49	74	98	123	148	172	197	222	246

Slika 1.1: Prikaz tabele za ročno izračunavanje volumnov.

PRILOGA | **Črtna koda**

PREVOZNICA:_____**št: __ /__**

1. OSNOVNI PODATKI (*izpolni Zavod za gozdove Slovenije)*

1.1 Izdajatelj odločbe/dovoljenja: Zavod za gozdove Slovenije, krajevna enota_____,(naslov)_____
1.2 Št. odločbe/dovoljenja:_____ , Datum izdaje odločbe/dovoljenja:_____ 1.3 Odsek:_____, Št. manuala:_____ 1.4 Največji možni posek iz odločbe/dovoljenja (m^3)_____ 1.5 Podatki o parceli: k.o._____, št._____

2. PODATKI O PREVOZU *(izpolni prevoznik)*

2.1 Dobavitelj (osebno ime in naslov stalnega prebivališča oz. firma, sedež in matična številka):	
2.2 Kupec (osebno ime in naslov stalnega prebivališča oz. firma, sedež in matična številka): *(Ni treba izpolniti, če gre za prevoz gozdno lesnih sortimentov, namenjenih začasnemu skladiščenju pred prvim dajanjem na trg ali za lastno uporabo. Glej točko 2.12 Namen prevoza)*	
2.3 Prevoznik (osebno ime in naslov stalnega prebivališča oz. firma, sedež in matična številka):	
2.4 Osebno ime in naslov stalnega prebivališča voznika:	2.5 Vrsta vozila:
2.6 Reg. št. vozila: _____ 2.7 Reg. št. priklopnega vozila_____	2.8 Datum nakladanja: _____ 2.9 Čas nakladanja: _____ 2.10 Lokacija nakladanja: _____ 2.11 Lokacija razkladanja: _____
2.12 Namen prevoza gozdno lesnih sortimentov (označi s križcem v kvadratu samo eno polje): ☐ se prvič dajejo na trg ☐ so namenjeni začasnemu skladiščenju pred prvim dajanjem na trg ☐ so namenjeni za lastno uporabo	

3. DEJANSKA KOLIČINA NALOŽENIH GOZDNO LESNIH SORTIMENTOV *(izpolni prevoznik)*

Drevesna vrsta	Enota mere (kos)	Debelinski razred - cm					Skupaj (m^3)
		10-20	20-30	30-40	40-50	nad 50	
Skupna količina (m^3)							
Opombe (neobvezno):							

Slika 1.2: Prevoznica, zahtevana s strani države.

naprave, ki bi omogočala hitro in učinkovito vnašanje podatkov, sprotno pridobivanje informacij o vnesenih podatkih ter avtomatsko računanje končnih količin. Poleg tega pa naj bi bila mogoča tudi sinhronizacija z informacijskim sistemom, s čimer lahko informacije potujejo mnogo hitreje, kar pomeni pohitritev celotnega procesa.

V poglavju 2 sprva opišemo zahteve, na katerih sloni diplomsko delo. V nadaljevanju smo v poglavju 3 predstavili opise vseh razvojnih okolij, jezikov, emulatorjev in naprav, ki smo jih uporabili pri izdelavi prototipa. V poglavju 4 je kratek oris razvoja, prikaz prototipa ter pripadajoče tehnične podrobnosti. Na koncu pa smo v poglavju 5 predstavili še ugotovitve in možne izboljšave prototipa.

Poglavje 2

Zahteve

Za izdelavo uspešne aplikacije smo morali najprej pridobiti čim več informacij in podrobnosti. Prav tako je pomembno, da čim prej odkrijemo težave, ki se pojavijo pri razvoju, ter v kolikor je to mogoče, predvidimo spremembe. Naš namen je bil razviti prototip za delo na terenu z mobilno napravo, ki bi:

- Omogočala delo z dobavitelji, kar zajema dodajanje novih dobaviteljev, pregled vseh dobaviteljev ter možnost izbora dobaviteljev, ki omogoča hitrejše izbiranje posameznih dobaviteljev.
- Omogočala delo s skupinami artiklov, kar zajema dodajanje novih skupin artiklov, pregled vseh skupin artiklov ter možnost izbora skupin artiklov.
- Omogočala delo z artikli, kar zajema vnos artiklov s pripadajočo skupino artiklov, pregled vseh artiklov in možnost izbora artiklov.
- Omogočala vnos hlodovine, ki ji določiš skupino artiklov ali artikel ter dolžino in premer ali volumen.
- Omogočala ustvarjanje odkupa s pripadajočimi podatki o dobavitelju in vneseni hlodovini. Iz ustvarjenega odkupa mora obstajati možnost izpisa povzetka za prevoznico in izpisa vse vnesene hlodovine za dobavnico.

- Omogočala sinhronizacijo z informacijskim sistemom. Pri tem mora obstajati možnost za pošiljanje strank, artiklov, skupin artiklov in nabav v informacijski sistem. Obstajati pa mora tudi možnost za prejem strank, artiklov, skupin artiklov in številk nabave iz informacijskega sistema.

Poglavje 3

Uporabljene tehnologije

V poglavju predstavljamo vse tehnologije in orodja, ki smo jih uporabili za razvoj prototipa.

3.1 Mobilni računalnik

Za uspešen razvoj mobilne aplikacije moramo prej ali slej vse skupaj testirati tudi na pravi napravi. Praviloma moramo to delati čim bolj pogosto, saj se s tem izognemo zapletom, ki se lahko pojavijo na koncu zaradi nekompatibilnosti naprave s katerimkoli delom aplikacije. Deloma to omogoča tudi hitrejši razvoj, saj načeloma velja, da je naprava veliko hitrejša kot nativni emulatorji. V primeru razvoja za PhoneGap to sicer ne drži popolnoma, saj obstaja še hitrejši način, kar pa ne izključuje dejstva, da je treba aplikacijo občasno testirati tudi na pravi napravi.

Pri samem razvoju smo uporabljali napravi Asus Transformer TF300TG ter Samsung Galaxy S4 mini – obe napravi delujeta na operacijskem sistemu Android. Ker je ciljni operacijski sistem Android, sta bili omenjeni napravi primerni za testiranje.

S privzetimi nastavitvami sistem Android ne omogoča razvoja, ampak ga moramo znotraj naprave omogočiti. Preko različnih verzij sistema se dostop razvijalca lahko spreminja, za kar lahko več podrobnosti najdemo v viru [5].

Poleg omenjenega moramo po vklopu razvijalca vklopiti tudi možnost za razhroščevanje preko USB [6], kar omogoča direktno komunikacijo z mobilno napravo preko priključka USB.

3.2 NetBeans

Integrirano razvojno okolje (angl. Integrated development environment – IDE) [8] je programska aplikacija, namenjena razvijalcem programov. Vsebuje veliko zbirko pripadajočih orodij, ki vključujejo:

- urejevalnik programske kode,
- orodja za avtomatiziranje različnih procesov, kot so na primer prevajanje v strojno kodo, ustvarjanje paketov, testiranj in drugo,
- razhroščevalnik.

Moderni IDE vključujejo tudi tehnologijo za avtomatsko dopolnjevanje kode ali vgrajen sistem za revizije. Bolj poznani programi so:

- NetBeans,
- Eclipse,
- Microsoft Visual Studio,
- InteliiJ IDEA.

NetBeans [9, 7] je zastonjsko in odprto-kodno integrirano razvojno okolje, ki primarno omogoča predvsem razvoj v:

- Javi,
- PHP,
- HTML5,

- JavaScript,
- C in C++.

Deluje v operacijskih sistemih s podporo JVM, kot so na primer Windows, OS X, Linux ali Solaris. Napisan je v Javi, sam razvoj pa se je začel leta 1996. Kasneje je bil tudi komercializiran, dokler ga ni kupil in spremenil v odprtokodnega Sun Microsystems leta 1999. Sun in z njim NetBeans je leta 2010 kupil Oracle. Narejen je iz komponent, imenovanih moduli, ki jih je mogoče tudi razširjati. Celotna razvijalska ekipa tesno sodeluje z uporabniki tako pri novih funkcionalnostih kot testiranju, zaradi česar je eno izmed najbolj uporabljenih razvojnih okolij.

3.3 SDK

Programski razvojni paket (angl. Software Development Kit – SDK) [10] je skupek orodij za kreiranje različnih aplikacij za določen operacijski sistem, konzolo ali kakšno drugo platformo. Možne so zelo preproste ali zelo napredne, sofisticirane rešitve, večinoma pa predstavlja skupek orodij, ki razvijalcem v glavnem omogočajo prevajanje in razhroščevanje.

3.4 JDK

Javanski razvojni paket (angl. Java Development Kit – JDK) [11] je razvijalski paket za katerokoli od platform v naslednjih različicah:

- Java SE (angl. Java Standard Edition),
- Java EE (angl. Java Enterprise Edition),
- Java ME (angl. Java Micro Edition).

Razvija ga Oracle, ki ga je pridobil s prevzetjem Sun. Večina implementacije je pod licenco GPL in je namenjena razvijalcem, ki delajo v okoljih

Solaris, Linux, Mac OS X ali Windows. Med drugim naj bi z razvojem platforme Java veljal tudi kot najbolj uporabljen SDK.

Pri razvoju mobilnih aplikacij za sistem Android je prvobitno uporabljena platforma Java. Pri tem pa nam JDK predstavlja ključni gradnik, saj lahko z njim razvito aplikacijo prevedemo v sistemu Android razumljiv jezik.

3.5 Android SDK

Android SDK [12] ponuja množico orodij, ki nam omogočajo kakovostni razvoj za sistem Android. Med drugim lahko v orodjih najdemo tudi orodje za razhroščevanje naprav ter emulatorje, ki so lahko zelo priročni za prvotno vzpostavitev okolja. Deluje na operacijskih sistemih Linux, Mac OS X in Windows. Uradno podprt IDE je Eclipse, ki deluje s pomočjo vtičnika ADT (angl. Android Development Tools). Neuradno ga lahko razvijamo tudi v drugih okoljih IDE, pri tem pa lahko uporabimo zgolj navaden urejevalnik ter ukazno vrstico s potrebnimi orodji (JDK in Ant) za kreiranje, prevajanje in razhroščevanje aplikacij Android.

3.5.1 Android Debug Bridge

Znotraj Android SDK je orodje ADB (angl. Android Debug Bridge), ki vključuje strežniške in odjemalske programe za komunikacijo z napravo. Omogoča, da s pomočjo računalnika lahko razhroščujemo napravo, ki je povezana preko priključka USB.

3.5.2 AVD

Za delo z virtualnimi napravami je na voljo AVD (angl. Android Virtual Device), ki nam omogoča, da ustvarimo poljubno napravo ter ji priredimo želene strojne in programske nastavitve, kot so na primer:

- nastavitve tipkovnice,

- velikost zaslona,
- velikost pomnilnika,
- kartice SD.

3.6 Ant

Ant [13, 14] je programsko orodje za avtomatiziranje gradnje programa. Prepoznamo lahko veliko podobnosti z orodjem Make [15, 16], vendar je Ant za razliko od Make, implementiran v Javi. Razliko med njima lahko sprva opazimo v opisu avtomatiziranja, za katerega Ant uporablja format XML. Glavni namen Ant je avtomatiziranje projektov v platformi Java, možno pa ga je uporabljati tudi za druge aplikacije ali katerikoli proces, ki ga je možno opisati kot cilje.

IDE NetBeans potrebuje Ant, ki ga interno uporablja za svoje procese. Poleg tega je s tem omogočeno, da lahko skripte Ant, ki jih izvozi NetBeans, poganjamo tudi izven okolja NetBeans.

3.7 Google USB driver

Za razliko od Max OS X in Linux je pri razvoju in testiranju iz okolja Windows za povezavo z napravo Android potreben poseben gonilnik USB [17]. Gonilnik je odvisen od naprave, primeren gonilnik pa lahko najdemo na proizvajalčevi strani naprave [17].

3.8 NBAndroid

NetBeans v osnovi ne podpira razvijanja za Android, ki je sicer odlično podprt v okolju Eclipse. Za podporo znotraj okolja NetBeans imamo na voljo odprtokodno razširjavo NBAndroid [18].

Inštalacijo najlažje izvedemo preko za to pripravljene opcije za razširjavo v NetBeans, s čimer je omogočeno tudi avtomatsko posodabljanje. Z inštalacijo [19] le-tega dobimo povezljivost med Android SDK in NetBeans, kar nam omogoči preprosto uporabo emulatorja ali naprave Android.

3.9 PhoneGap

PhoneGap [2, 4, 20, 21] je mobilno razvijalsko okolje podjetja Adobe Systems in temelji na Apache Cordova [22]. Razvijalcem za mobilne naprave omogoča, da namesto z nativnimi jeziki naprave pišejo s spletnimi jeziki JavaScript, HTML5 in CSS3. To med drugim omogoča tudi, da določeno kodo napišemo enkrat in jo lahko uporabimo na celi množici naprav z različnimi sistemi. Aplikacije, napisane v tem okolju, so deloma internetne, saj je celoten prikaz narejen kot internetna stran, ter deloma nativne, ker so zapakirane kot paket oziroma aplikacija in imajo dostop do funkcionalnosti naprav.

Večinoma aplikacije PhoneGap uporabljajo spletne jezike HTML5 in CSS3 za prikaz ter jezik JavaScript za logiko. HTML5 ima podporo do strojne opreme naprav, a ker standard HTML5 ni popolnoma podprt v vseh napravah, PhoneGap uokviri HTML5 kodo znotraj nativnega internetnega prikazovalnika v napravi in pri tem uporablja mehanizem FFI za dostop do nativnih funkcij naprave.

PhoneGap nudi tudi razširjanje nativnih vtičnikov, kar razvijalcem omogoča dodajanje funkcionalnosti, ki jih lahko pokličemo s pomočjo jezika JavaScript, kar omogoči direktno komunikacijo med HTML5 stranjo in napravo. V osnovi je sicer za večino naprav podprt:

- merilnik pospeška,
- kamera,
- mikrofon,
- kompas,

- datotečni sistem.

Zaradi uporabe internetnih tehnologij in potrebnih okvirov, ki sicer omogočajo enotno kodo in učinkovit dostop do funkcij naprave, lahko aplikacije delujejo počasneje kot bi podobne nativne aplikacije. Pri tem obstaja celo možnost, da je aplikacija na primer pri razvoju za sistem Apple zavrnjena zaradi prepočasnega delovanja ali nekonsistenčnega izgleda v primerjavi z drugimi vsebinami v sistemu.

Ena izmed največjih prednosti razvoja s PhoneGap je pisanje iste razvojne kode za več sistemov. Ob pisanju diplomskega dela so bili podprti sistemi:

- iOS,
- Android,
- webOS,
- Windows Phone,
- Symbian,
- BlackBerry,
- Tizen.

S tem se PhoneGap uvršča med okolja s podporo za vse večje sisteme mobilnih naprav.

3.9.1 Foreign Function Interface

FFI (angl. Foreign Function Interface) [23] je mehanizem, s pomočjo katerega lahko iz programa, ki je napisan v enem programskem jeziku, kličemo funkcionalnosti drugega jezika. V večini primerov je to uporabljeno tako, da se v višjem jeziku kličejo in uporabljajo funkcionalnosti iz jezika, ki je nižje. To se na primer uporablja zaradi dostopa do različnih servisov, ki nam jih ponuja API operacijskega sistema ali zaradi optimiziranja hitrosti.

3.10 Weinre

Weinre (angl. WEb INspector REmote) [24, 25] je razhroščevalnik, namenjen uporabi pri razvoju internetnih strani. Za razliko od podobnih orodij lahko deluje na daljavo, kar omogoča razhroščevanje mobilnih naprav. Orodje uporabljata JavaScript in nenativne kode brskalnikov, zaradi česar ima omejene zmožnosti ter ne podpira razhroščevanja programske kode v JavaScript. Pri tem je lahko bolj uspešno orodje Aardwolf. Weinre za strežnik uporablja paket NodeJs, ki ga lahko dobimo znotraj upravljalnika paketov Npm [26].

3.10.1 Node.js

NodeJs (Node.js) [27] je programsko okolje, namenjeno razvoju razširljivih omrežnih aplikacij. Napisano je v jeziku JavaScript in omogoča poganjanje omrežnega strežnika samo po sebi. Zasnovano je bilo za uporabo pri izdelavi spletnih strani s »push« možnostjo.

3.11 Aardwolf

Aardwolf [28] je razhroščevalnik za jezik JavaScript, napisan v jeziku JavaScript, ki deluje na daljavo in je primarno namenjen za razhroščevanje naprav:

- iPhone,
- Android,
- WindowsPhone,
- BlackBerry.

Omogoča postavljanje prekinitvenih točk, pregled kode na prekinitvenih točkah, premikanje po izvorni kodi, oddaljeno upravljanje z JavaScript in tudi opozarjanje na napake.

3.12 Ripple

Ripple [2, 29, 30] je na voljo kot razširitev za Google Chrome in ga lahko najdemo v spletni trgovini Chrome. Namenjen je poenostavljenemu razvoju in testiranju aplikacij HTML5 za mobilne naprave. Omogoča razvoj za več mobilnih platform ter odpravlja marsikatero težavo, ki se lahko pojavi pri samem razvoju mobilnih aplikacij.

Gre za navidezni emulator znotraj brskalnika Chrome, kar nam omogoča popoln pogled v ozadje izvajanja aplikacije. Možno je preverjati:

- strukturo DOM,
- razhroščevanje JavaScript, ki je lahko sicer zelo oteženo,
- avtomatsko testiranje, ki je sicer praktično nemogoče.

Med drugim je Ripple namenjen prav razvoju aplikacij za PhoneGap. Z uporabo le-tega lahko čas razvoja in testiranja aplikacije znatno pohitrimo, saj je hitrejši od emulatorjev, ki nam jih ponujajo proizvajalci sistema Android, kot tudi od fizičnih naprav.

Sama uporaba poteka tako, da celotno mobilno aplikacijo poženemo s strežniškimi programi, kot je na primer Apache. S prej nameščeno razširjavo za brskalnik Chrome je potem možno obiskati stran mobilne aplikacije, ki v veliki meri emulira pravo napravo.

3.13 HTML

HTML (angl. HyperText Markup Language) [31, 32] je glavni označevalni jezik za izdelavo spletnih strani.

Napisan je s pomočjo elementov HTML, katerih opisi so znotraj znaka manjše in večje. Za večino elementov tudi velja, da nastopajo v parih, kjer prvi par predstavlja začetek in drugi konec. Na ta način dobimo hierarhično

strukturo elementov. S pomočjo elementov lahko določimo strukturo, preusmeritev, obliko in druge podrobnosti strani. Med drugim poznamo elemente, kot so:

- naslovi,
- odstavki,
- seznami,
- naslovi,
- slike,
- izbiranje datuma.

Struktura HTML oziroma koda služi kot navodilo spletnemu brskalniku za prikazovanje strani. Poleg same kode HTML lahko brskalnik za obliko uporabi tudi pripadajoč CSS.

Dokumenti HTML so lahko kar običajne datoteke, v večini primerov pa je dokument HTML prenesen s spletnega strežnika preko protokola http ali z e-pošto. V primeru PhoneGap so na napravi večinoma vsebine HTML že prisotne in jih je treba zgolj še interpretirati ter prikazati.

Najnovejša različica jezika HTML je HTML5. Namen zadnje različice je predvsem odprava pomanjkljivosti prejšnjih verzij, kot so na primer podpora multimediji, nepovezavni način, vleci-spusti način ali shranjevanje, s katerim je med drugim tudi omogočen nepovezavni način.

3.14 CSS

CSS (angl. Cascading Style Sheets) [33] je jezik za oblikovanje sloga, s katerim se določa oblika in videz označevalnega jezika. Največkrat je uporabljen v jeziku HTML in je primarno namenjen ločitvi vsebine dokumenta ter prikaza. Z uporabo le-tega lahko izboljšamo veliko stvari, kot so:

- izboljšana fleksibilnost in izboljšan nadzor oblike, saj uporabljamo namenski jezik,
- možnost souporabe oblike med več stranmi, kar zmanjša kompleksnost in ponavljanje,
- omogočen različen prikaz za različne načine prikaza (kot so zaslon ali tiskanje) ter različne naprave in velikosti zaslona oziroma naprave,
- z ločitvijo dokumenta lahko uporabnik tudi enostavno zamenja oblikovni dokument oziroma uporabi enega izmed več predpripravljenih, ki so mu ponujeni.

Elementi so izbrani s posebnimi izbiralniki, kjer velikokrat pomagajo identiteta, razred ali tip elementa. Poleg tega so specificirani tudi posebni razredi za pomoč, kot je na primer lebdenje miške nad elementom.

Vsi moderni brskalniki znajo uporabljati CSS, vendar obstaja tudi težava, saj ga vsi ne interpretirajo pravilno. Zato je občasno treba uporabljati tudi različne nestandardne obhode, s katerimi lahko specificiramo različno obliko za različne brskalnike. Deloma je tudi zaradi tega testiranje na več brskalnikih nujen in pomemben korak pri samem razvoju.

3.15 JavaScript

JS (JavaScript) [34] je interpreterski računalniški jezik, ki je narejen po vzoru jezika C. Doda funkcionalnosti, kot so na primer interakcije z uporabniki, nadzor brskalnika, asinhrona komunikacija ali spreminjanje obstoječe vsebine. Z razvojem se veča uporaba jezika JavaScript tudi v strežniškem programiranju, programiranju iger, namiznih aplikacijah in podobno.

Tipi v jeziku JavaScript so vezani na vrednosti in ne na spremenljivke, podobno kot pri drugih skriptnih jezikih. JavaScript večinoma temelji na objektih, ki so asociativni seznami. S tem je omogočena notacija pike (obj. x) ali pa uporaba objekta kot seznama (obj['x']) ter s tem tudi možnost

sprehajanja čez lastnosti objekta z zanko. Dva izmed redkih objektov, ki so že vgrajeni v sam JavaScript, sta Function in Date. JavaScript med drugim pozna tudi funkcijo eval, s pomočjo katere lahko poženemo niz znakov, kot je JavaScript, a je potrebna previdnost, saj lahko ob nepravilni uporabi predstavlja varnostno luknjo.

Funkcije v jeziku JavaScript so kar objekti sami. Kot taki imajo tudi nekaj standardnih metod, kot sta na primer call() in bind(). Znotraj ene funkcije je lahko ugnezdena tudi druga funkcija, ki prevzame lastnosti funkcije, znotraj katere se nahaja.

Objekti v JavaScript ne uporabljajo razredov, ampak temeljijo na uporabi prototipov, kjer se dedovanje uporablja s pomočjo kloniranja obstoječih objektov.

Najpogostejši način uporabe jezika JavaScript je pisanje funkcij, ki so vključene v strani HTML in upravljajo z modelom strani DOM. Nekateri primeri uporabe JavaScript so:

- povezava s strežnikom preko AJAX brez ponovnega nalaganja strani,
- nalaganje novih vsebin strani,
- animiranje elementov,
- predvajanje zvoka ali videa,
- preverjanje vnosnih polj.

JavaScript poganjamo lokalno, zato se lahko hitreje odzove na uporabniške želje in s tem aplikacijo naredimo bolj odzivno. S pomočjo JavaScript lahko tudi prepoznamo druge aktivnosti uporabnika, kot je na primer pritisk tipke, kar nam sam HTML ne omogoča.

JavaScript je edini jezik, ki je podprt v vseh modernejših brskalnikih in je postal ciljni jezik za veliko drugih platform kljub dejstvu, da sprva temu ni bil namenjen. Obnašanje jezika JavaScript se lahko v različnih okoljih razlikuje, zato je treba vsa okolja posebej pretestirati. Včasih je treba zaznati različno

okolje in zanj ustrezno prirediti kodo. Stari brskalniki, mobilni telefoni z nepodprtim ali neomogočenim JavaScript so lahko primeri, ko se aplikacija morda ne bo odzivala pravilno, zato je treba pri nekaterih aplikacijah to upoštevati.

JavaScript z veliko zmožnostmi ponuja tudi veliko ranljivosti. Razvijalci brskalnikov zato uporabljajo dve pomembnejši omejitvi, in sicer da lahko JavaScript izvaja zgolj akcije, povezane z brskalnikom ter z zaprtim okoljem, zaradi katerega naj ne bi bil možen dostop do informacij druge strani. Varnostni problem predstavlja dejstvo, če uspe napadalcu internetno stran prisiliti v izvajanje skripte, ki se nahaja na drugi strani. Brskalniki lahko to deloma preprečijo, a ne popolnoma, saj lahko v nekaterih primerih napadalec kodo skrije v podatkovno bazo, ki jo potem brskalnik žrtve požene. Za preprečitev omenjenega napada je torej potreben primeren dizajn baze in strani. Prav tako lahko napadalec izvede napad, tako da na svojo stran postavi povezavo na napadeno stran, ki izrabi žrtvine podatke, kot so na primer piškotki. Pri izdelavi strani moramo biti previdni in predvidevati, da JavaScript, ki je na strani klienta, ni tisti, za katerega domnevamo, da je. Na strežniški strani moramo to upoštevati in preveriti, ali so prejeti podatki res pravilni.

Pri razvoju velikih programov postane razhroščevanje znatnega pomena. Zaradi razlik med različnimi brskalniki pa je včasih pomembno imeti dostop do razhroščevalnikov na vseh ciljnih brskalnikih. Podporo za razhroščevanje imajo brskalniki:

- Internet Explorer,
- Firefox,
- Safari,
- Chrome,
- Opera.

Poleg izvajanja JavaScript znotraj brskalnikov obstaja veliko okolij, ki

prav tako omogočajo izvajanje JavaScript. Večina jezika pri tem ostaja ista, na primer:

- razširitve brskalnika Chrome,
- Opera,
- Safari,
- Adobe Reader,
- Photoshop,
- OpenOffice,
- Unity.

3.15.1 Razredi

JavaScript nativno ne podpira razredov, zato jih je treba za uporabo ustrezno implementirati. Eden izmed načinov je enostavno dedovanje Johna Resiga [3, 35]. Pri implementaciji je vedno treba razširiti enega izmed obstoječih razredov. Najbolj osnovni razred mora torej razširiti vsaj osnovni razred Class, ki med drugim velja kot skupni prednik vsem razredom. S pomočjo te implementacije je možna uporaba enostavnega konstruktorja s pomočjo metode init in možnost dostopa do prepisane metode s pomočjo _super.

3.15.2 jQuery in jQueryMobile

jQuery [1, 36, 37] je knjižnica JavaScript za poenostavljeno pisanje kode. jQuery je zastonjsko odprto-kodno programje pod MIT licenco. Med drugim omogoča:

- enostavno navigiranje po dokumentu,
- izbiranje elementov DOM,

- ustvarjanje animacij,
- upravljanje z dogodki,
- AJAX klice.

jQuery Mobile [38, 39] je knjižnica, izpeljana iz jQuery, in je optimizirana za mobilne naprave. Poleg jQuery omogoča še funkcionalnost jQueryUI [40, 41], pri katerem je poskrbljeno, da deluje na čim več pametnih telefonih in tabličnih računalnikih. Iz jedra jQuery ga je razvila ista ekipa kot jQuery, zato sta si v marsičem podobna. Prav tako pa je omogočena podpora z drugimi mobilnimi okolji, kot je na primer Worklight ali PhoneGap.

3.15.3 JSON

JSON (angl. JavaScript Object Notation) [42] je posebna oblika formata, ki je enostavno berljiva in omogoča shranjevanje ter pošiljanje podatkovnih objektov. Gre za odprt standard, razvit kot alternativa formatu XML ter je primarno namenjen izmenjavi podatkov med strežnikom in internetno aplikacijo. Kljub primarnemu razvoju v jeziku JavaScript pa gre za jezikovno neodvisen format, tako da lahko funkcije za delo s formatom najdemo v veliko jezikih.

Znotraj formata so podprti vsi osnovni tipi, kot so števila, niz znakov, logična vrednost »true« oziroma »false«, urejen seznam, objekt ali null. Za ostale tipe pa je potrebna posebna pretvorba, s katero sta seznanjena oba konca povezave. Taki tipi so na primer datum, napaka, regularni izraz ali funkcije.

Posebna uporaba formata JSON je v kombinaciji z AJAX, kjer je večinoma zamenjava za format XML.

3.15.4 AJAX

AJAX (angl. Asynchronous JavaScript and XML) [1, 43] je množica povezanih tehnik pri spletnem razvoju, namenjenih ustvarjanju asinhronih spletnih

aplikacij. Pri razvoju spletnih aplikacij je pomembno, da je obnašanje čim bolj tekoče in nemoteče, zato je, v kolikor je to mogoče, pomembna uporaba asinhronih povezav, ki nam jih omogoča AJAX. AJAX omogoča tudi sinhrono delovanje, ki ga je treba posebej navesti, a je uporabo priporočljivo zaradi odzivnosti minimizirati.

AJAX je tako postal sinonim za skupino tehnologij, ki v ozadju omogočajo komunikacijo spletne strani s strežnikom, ne da bi se stanje strani kakorkoli spremenilo. To nam na primer dovoljuje, da smo ves čas na isti strani in dobimo aktualne podatke iz strežnika. Za izmenjavo se večinoma uporablja format JSON, možni pa so seveda tudi drugi formati, kot je XML ali neformatirani podatki.

3.15.5 Single-page application

SPA (angl. single-page application) [44, 45] ali SPI (angl. single-page interface) je spletna aplikacija oziroma stran, ki je v celoti na eni strani in je narejena z namenom ponuditi boljšo uporabniško izkušnjo, ki bi bila podobna namiznim aplikacijam. Na strani SPA so HTML, JavaScript in koda CSS že naloženi ob prvem dostopu oziroma se ves čas sproti posodabljajo, po navadi kot odziv na uporabniške akcije. Ob nobeni točki se stran ne zamenja ali naloži ponovno, čeprav moderne tehnologije, kot je HTML5, omogočajo percepcijo, kot da se je to zgodilo.

Pri izdelavi teh strani je med drugim treba paziti na sledeče:

- Pred HTML5 ni bilo možnosti direktne manipulacije z zgodovino brskalnika, kar je pomenilo, da operacije v brskalniku, kot so na primer prejšnja ali naslednja stran, ne bi pravilno delovale. S tem je bilo prav tako onemogočeno pomnjenje lokacije na strani in vrnitev oz. deljenje le-te. HTML5 se s tem uspešno spopade, saj omogoča delo z zgodovino brskalnika, kar odpravlja omenjene težave.

- Dinamične strani lahko otežijo delo uporabnika s prikazom dinamičnih vsebin ob neprimernem času. To se lahko pojavi ob različnih situacijah,

predvsem če je na voljo slaba povezava.

- Če je treba indeksirati stran, moramo zaradi tega ponuditi alternativne možnosti, saj večina »pajkov« ne zna uporabljati jezika JavaScript.
- Brskalniki ali mobilne naprave morda ne podpirajo (oziroma imajo zaradi varnostnih razlogov onemogočeno) tehnologije AJAX oziroma JavaScript.
- AJAX ima morda onemogočeno delovanje v drugih domenah.

SPA zagotavlja množico prednosti, kot so:

- Lahko se zmanjša velikost prenosa podatkov, saj se, ko je ogrodje naloženo, spreminjajo le še podatki.
- Ob primerni zasnovi je nepovezan način na straneh SPA lahko precej boljši.
- Hitrejša navigacija in odzivnost, saj je večina dela opravljena na odjemalčevi strani. Na strežniku ostanejo le nujne stvari, kot so validacija, overjanje in avtorizacija. Možni so sicer tudi pristopi, kjer večina dela ostane strežniku.
- Boljša interakcija, saj uporabniki ostajajo na isti strani.

3.16 PouchDb

PouchDB [46, 47] je odprtokodna JavaScript NoSQL baza, ki temelji na shranjevanju elementov in izhaja iz CouchDB ter je namenjena uporabi znotraj brskalnika.

Baza je bila razvita z namenom, da bi razvijalcem omogočili razvoj aplikacij, ki bi delale enako v nepovezavnem kot povezavnem načinu. Omogoča, da aplikacije v nepovezavnem načinu podatke shranjujejo lokalno ter jih sinhronizirajo s CouchDB, ko vstopijo v povezavni način. Podatki so shranjeni v formatu JSON ter se pretvarjajo s pomočjo jezika JavaScript.

Deluje v vseh modernejših brskalnikih, med drugim tudi v Apache Cordova, kar omogoča razvoj za PhoneGap aplikacije. Podpira enostavno sinhronizacijo z drugo bazo PouchDB ali CouchDB, medtem ko je za sinhronizacijo s preostalimi bazami treba poskrbeti ročno.

Med razvojem je bilo veliko pozornosti posvečene sinhronizaciji, ki lahko postane težavna ob uvedbi nepovezavnih aplikacij. Na voljo je sicer zgolj ob povezavi z drugo bazo PouchDB oziroma CouchDB. Sinhronizacija je zasnovana tako, da nobena baza ne deluje kot centralna, ampak so vse enotne. Za sinhronizacijo samo je skoraj popolnoma poskrbljeno v ozadju, kar omogoča, da tako v nepovezavnem kot povezavnem načinu operiramo na isti način ter še vseeno dobimo sinhronizirane baze. To istočasno tudi omogoča, da v povezavnem načinu lahko brez sprememb še vedno izvajamo vse operacije, četudi tehnologija za lokalno shranjevanje, ki jo PouchDB uporablja v ozadju, ni na voljo.

PouchDB v ozadju uporablja različne tehnologije za različne brskalnike:

- V Firefox je uporabljen IndexedDB, kjer je velikost sicer neomejena, povečanje pa je treba potrditi na vsakih 50 MB.

- Brskalnik Chrome prav tako uporablja IndexedDB, velikost pa izračuna iz velikosti prostega prostora na disku. Pri tem ima vsaka aplikacija na voljo 20% skupnega prostora, ki je polovica prostega prostora na disku.

- Mobilni Safari uporablja WebSQL in ima 50 MB limita, medtem ko njegova namizna različica s potrditvijo lahko shranjuje od 5 MB do 500 MB.

- V Operi sta uporabljena LevelDB in Node.js, velikost pa je neomejena.

- Podpora je bila vsaj v preteklosti najslabša za Internet Explorer, ki je omejen z 250 MB prostora.

3.16.1 NoSQL

NoSQL, včasih poimenovana tudi »Not Only SQL«, je baza za shranjevanje in iskanje podatkov, ki je nastala kot alternativa relacijskim bazam. Je manj konsistentna od relacijskih baz, omogoča pa lažji razvoj, horizontalno skaliranje in boljšo kontrolo razpoložljivosti. S to bazo sta poenostavljeni operaciji za pridobivanje in shranjevanje podatkov ter izboljšana hitrost. Slednja je predvsem pomembna pri rastočih potrebah v realno-časovnih aplikacijah in aplikacijah z ogromno podatkov. Najbolj znane baze, ki jih lahko najdemo, so:

- Cassandra,
- CouchDB,
- Hbase,
- MongoDB,
- Redis,
- Google Big Table.

Obstaja več načinov shranjevanja v NoSQL bazah:

- Key-value oziroma ključ-vrednost, ki med drugim omogoča, da dodajamo nove nedefinirane podatke ali uporabimo zgolj podatke, ki jih rabimo, kjer bi v relacijski bazi morali pustiti prazno polje.
- Document je podoben načinu ključ-vrednost, le da ima lahko kot vrednost kar cele dokumente, ki so med seboj različni in so lahko predstavljeni na primer kot XML, JSON ali PDF.
- Column je podoben načinu document z izjemo, da ima znotraj posamezne vrstice stolpce, ki nastopajo v parih ključ-vrednost ter med drugim omogočajo tudi indeksiranje oziroma zajem potrebnih stolpcev.

- Graph je namenjen za shranjevanje podatkov, katerih relacije so lahko dobro predstavljene v grafu s poljubnim številom povezav.

NoSQL ima tako zagovornike kot nasprotnike. Nekaj slabosti, ki jih lahko najdemo pri bazi NoSQL, so naslednje:

- je še v razvoju in obstajajo neizpopolnjeni deli,
- slabša podpora indeksiranju,
- brez ACID (angl. Atomicity, Consistency, Isolation, Durability), ki zagotavljajo kvaliteto podatkov,
- brez standardov, s čimer je tudi premik na drug NoSQL lahko zelo drag.

Glavne prednosti NoSQL baze pa so:

- horizontalno skaliranje, ker ni kompleksnih poizvedovanj za rezultati, kar omogoča, da lahko podatke enostavno razbijemo in procesiramo paralelno. To je med drugim tudi zelo primerno za baze, kjer je količina podatkov ogromna,
- ni definiranega podatkovnega modela, kar omogoča hiter razvoj, dodajanje novih podatkov in spremembo strukture, ko je to potrebno,
- enostaven za uporabo, saj je po principu precej bližje objektnemu pristopu in s tem tudi programerjem,
- možnost shranjevanja kompleksnih podatkov na eni lokaciji,
- lažje vzdrževanje v primerjavi z relacijskimi bazami.

Poglavje 4

Mobilna aplikacija

4.1 Izbira tehnologij in razvoj

Zaradi razvoja za mobilne platforme smo izbrali vmesnik PhoneGap, ki nam omogoča tudi prenos na alternativno platformo. Zaradi predvidenega večjega števila sprememb in poenostavljenega razvoja smo uporabili NoSQL bazo, in sicer PouchDB, ki je minimizirana različica baze CouchDB. Kot glavni format prenosa med mobilno napravo in informacijskim sistemom smo izbrali format JSON, ki se prenaša s pomočjo AJAX. Zaradi enostavnejšega dela s strukturo DOM smo v jeziku JavaScript uporabili tudi Mobile jQuery. Za potrebe razhroščevanja smo uporabili različna orodja, in sicer Weinre in Aardolf za razhroščevanje mobilne naprave ter Ripple kot emulator naprave, ki jo lahko razhroščujemo z brskalnikom Chrome.

Prvi korak v razvoju predstavlja vzpostavitev okolja s pomočjo okolja NetBeans in emulatorjev v Android SDK. Po uspešnem testiranju na emulatorju smo odkrili pričakovano težavo z njegovo počasnostjo, kar je zahtevalo drug pristop. Po namestitvi ustreznega gonilnika za napravo in omogočenju razvijalskega načina smo razvoj lahko nadaljevali v napravi, kar je omogočalo znatno hitrejše delo. Kljub hitrejšemu razvoju smo relativno kmalu odkrili problem z razhroščevanjem jezika JavaScript. V prvi fazi je sicer zadostoval Weinre, a smo zaradi načina razvoja v JavaScript in nezmožnosti raz-

hroščevanja le-tega v Weinre potrebovali drugo rešitev, ki bi omogočila tudi razhroščevanje kode JavaScript. Omenjeni težavi je kljuboval Ripple, ki je služil kot odličen emulator za aplikacije PhoneGap. Emulator deluje znotraj brskalnika, s čimer so omogočene vse funkcionalnosti, ki nam jih ponuja brskalnik. Na ta način smo lahko razhroščevali tudi JavaScript, kar je bilo pri razvoju projekta zelo pomembno. Poleg omenjenega je delo v emulatorju še pohitrilo razvoj, saj ni bilo potrebe po prevajanju in prenašanju programa v napravo. Kasneje se je pojavil tudi program Aardwolf, ki pa pri razvoju zaradi hitrega emulatorja Ripple ni prišel v poštev, ampak zgolj pri prenosu in testiranju v napravi.

4.2 Izdelava vmesnika

Z izdelavo vmesnika smo želeli odpraviti direktno delo z elementi HTML ter omogočiti bolj objektni pristop, kar bi pomenilo tudi lažje spreminjanje. Vmesnik nam ponuja tudi enostavnejše prehajanje med različnimi obrazci ali zgolj klicanje prejšnjega, kar pomeni poenostavljen pristop za SPA aplikacije. Poleg omenjenega nam vmesnik omogoča tudi delo s podatki, ki na enoten način nudi njihovo upravljanje in združevanje.

4.2.1 Upravljalec obrazcev

Upravljalec obrazcev smo razvili z namenom prikazovanja enega ali več obrazcev. Poleg omenjenega nam omogoča tudi skrivavanje oziroma brisanje prikazanih obrazcev ter poenostavlja prehajanje v prej prikazan obrazec.

4.2.2 Obrazec

V osnovni je obrazec namenjen prikazu in skrivanju pripadajočih elementov ter poenostavljenemu gnezdenju, ki nam je v pomoč pri prehajanju med obrazci.

4.2.3 Prikazni element

Prikazni element večinoma predstavlja most z elementom HTML in ga uporabljamo za pravilen prikaz in skrivanje elementov.

4.2.4 Skladišča

Skladišča nam omogočajo lažje delo s podatki, ki so v ozadju lahko shranjeni v pomnilniku ali bazi. Vmesnik je sestavljen iz več skladišč:

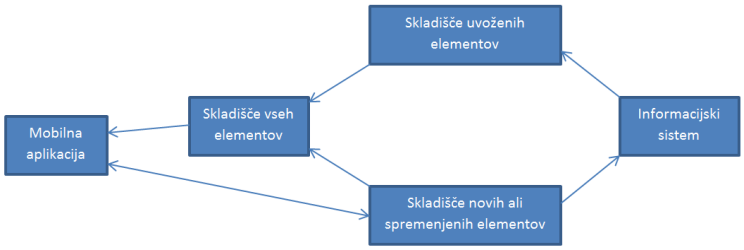
- osnovno skladišče, ki ga uporabljamo za shranjevanje in branje iz pomnilnika,
- skladišče za podatkovne baze, s pomočjo katerega beremo in shranjujemo podatke iz baze PouchDB,
- virtualno skladišče, ki ga priključimo na osnovno skladišče ali skladišče za podatkovne baze in nam omogoča lažje prehajanje med posameznimi elementi,
- združevalno skladišče, ki nam omogoča združevanje več virtualnih skladišč, ki so potem prikazana kot eno.

4.2.5 Predpripravljeni elementi in obrazci

Vmesnik nam za lažje delo in splošnejšo uporabo ponuja več predpripravljenih elementov in obrazcev:

- Gumb, ki pozna dogodek pritiska in ga lahko splošneje uporabimo tudi kot gumb za nazaj, ki ga najdemo na napravi.
- Niz gumbov, ki nam omogoča nizanje več gumbov, od katerih se vsak odzove na svoj dogodek.
- Enostavni vnašalec za številke, ki nam v kombinaciji z nizom gumbov omogoča enostavno vnašanje v vnosno polje tako za cela števila kot za decimalna števila.

- Zlom, ki ga lahko uporabimo kot element za prelom.
- Potrditveno polje, ki nam omogoča element okenca potrditve.
- Element čiščenja, ki nam omogoča urejanje dokumenta, če je to potrebno.
- Izbiralnik datuma, s pomočjo katerega je lažje izbrati datum v vnosnem polju.
- Izbiralec predstavlja obrazec, s pomočjo katerega izberemo poljuben element iz seznama. Pridružimo mu lahko dve skladišči, in sicer skladišče za izbrane elemente in neobvezno skladišče, ki pričakuje skladišče z vsemi elementi. V ozadju je skladišče za izbrane elemente uporabljeno z združevalnim skladiščem, katerega drugi del je spreminjajoče se skladišče, ki se posodablja v primeru, ko izberemo neznan element iz neobveznega skladišča z vsemi elementi. Na ta način je poskrbljeno, da imamo zelo majhno število izbir, ki se hitro posodobijo, če je to potrebno.
- Enostavni izbiralec je element, ki nam omogoča hitro premikanje med elementi, ki so izbrani. V primeru, ko elementa ne najdemo, pa ta gradnik omogoča hiter pregled s pomočjo izbiralca, ki nudi možnost izbire kateregakoli elementa.
- Grupni element, ki ga lahko uporabimo za pomoč grupiranja.
- Gradnik slike, ki nam koristi za prikaz slike.
- Gradnik vnosnega polja, ki prepozna dogodek spreminjanja.
- Gradnik, ki ga lahko uporabimo za prikaz besedila.
- Gradnik seznam, ki mu lahko podamo seznam elementov, za katere je omogočen dogodek pritiska.



Slika 4.1: Prikaz organizacije skladišč o dobaviteljih, skupinah artiklov in artiklih.

- Obrazec za spreminjanje izbranih elementov, ki nam omogoča posplošen način spreminjanja skladišč izbranih elementov.

4.3 Organizacija podatkov

Iz zahtev lahko razberemo potrebo po več podatkih:

- dobavitelji, za katere hranimo podatke o nazivu, naslovu, kraju in transakcijskem računu,
- za skupine artiklov s pripadajočim nazivom,
- za artikle s pripadajočim nazivom in skupino artiklov,
- za vneseno hlodovino, kjer hranimo podatke o artiklih ali skupinah artiklov in pripadajočih volumnov oziroma premerov ter dolžin,
- za odkupe, v katerih hranimo podatke o dobaviteljih, datumu opravljene storitve in pripadajoče vnesene hlodovine.

Podatke o dobaviteljih, skupinah artiklov in artiklih hranimo v dveh ločenih skladiščih, kot je to prikazano na sliki 4.1. Eno skladišče predstavlja nove (oziroma spremenjene) podatke, medtem ko drugo predstavlja uvožene. Na ta način lahko podatek prestavljamo iz enega v drugo ter poenostavimo

Slika 4.2: Glavni meni.

ločevanje med sinhroniziranimi in nesinhroniziranimi podatki. Obe skladišči nato združimo s pomočjo združevalnega skladišča, kar lahko uporabimo za prikaz vseh podatkov.

Poleg omenjenih pa potrebujemo še podatke o nastavitvah, znotraj katerih so informacije za povezavo z informacijskim sistemom in številke za nabavne dokumente.

4.4 Ključni vnosi

4.4.1 Glavni meni

Po pozdravnem obrazcu, ki se pojavi v času nalaganja, se prikaže glavni meni, ki je iztočna točka za vse ostale funkcionalnosti. Vidimo ga lahko na sliki 4.2 in podpira naslednje funkcionalnosti:

- nov odkup,
- iskanje odkupov,
- nov artikel,
- iskanje artiklov,

Slika 4.3: Prikaz iskanja skupin artiklov.

- izbor artiklov,
- vnos nove skupine artiklov,
- iskanje skupine artiklov,
- izbor skupin artiklov,
- vnos novega dobavitelja,
- iskanje dobaviteljev,
- izbor dobaviteljev,
- vnos količin,
- pregled nedokončanih,
- sinhronizacija,
- nastavitve.

Slika 4.4: Prikaz za izbiranje skupin artiklov.

4.4.2 Iskanja

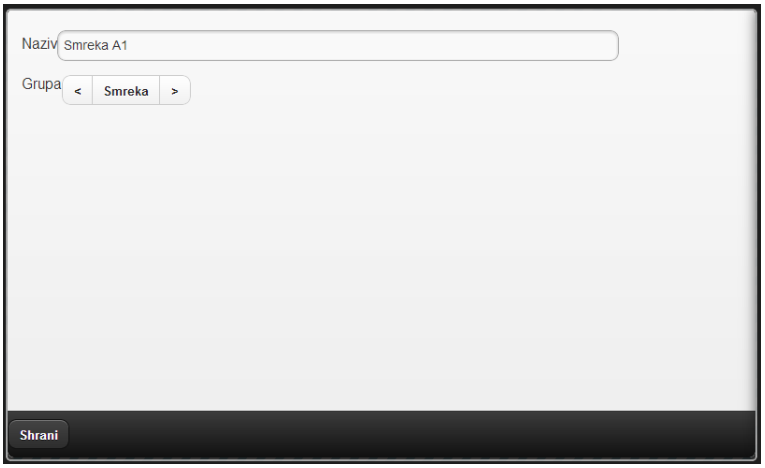
Iskanja smo posplošili tako, da se lahko uporabljajo s pomočjo predpripravljenega izbiralnega obrazca. Iskanje odkupov, artiklov, skupin artiklov in dobaviteljev smo tako naredili na isti način. Obrazec za iskanje skupin artiklov prikazuje slika 4.3. Iz predpripravljenega izbiralnega obrazca pa smo na podoben način naredili tudi pregled nedokončanih dokumentov.

4.4.3 Obrazci za zožanje izborov

S pomočjo predpripravljenega obrazca za spreminjanje izbranih elementov smo posplošili izbiranje artiklov, skupin artiklov in dobaviteljev, primer je prikazan na sliki 4.4. Izbrane elemente lahko brišemo ali pa dodajamo, za kar se nam odpre novo okno s celotnim izborom, ki smo ga naredili na podoben način kot iskanja.

4.4.4 Nov artikel

Z obrazcem za nov artikel lahko dodajamo nove še neobstoječe artikle. V osnovi potrebujemo zgolj podatke o nazivu in pripadajoči skupini, kasneje pa



Slika 4.5: Obrazec za ustvarjanje in spreminjanje artiklov.

se lahko podatki dopolnijo v informacijskem sistemu. Obrazec za dodajanje artikla lahko vidimo na sliki 4.5.

4.4.5 Nova skupina artiklov

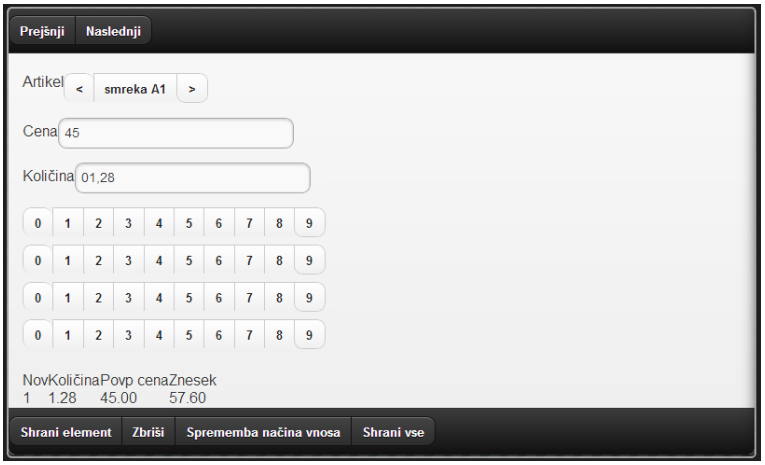
Skupina artiklov je skupna več artiklom, katere glavni pomen je naziv, ki ga dodamo z obrazcem, prikazanim na sliki 4.6. Skupina artiklov ima velik pomen tudi pri izpiskih, pri katerih lahko določeno skupino artiklov povežemo in za katero nato prikažemo grupirane podatke.

4.4.6 Nov dobavitelj

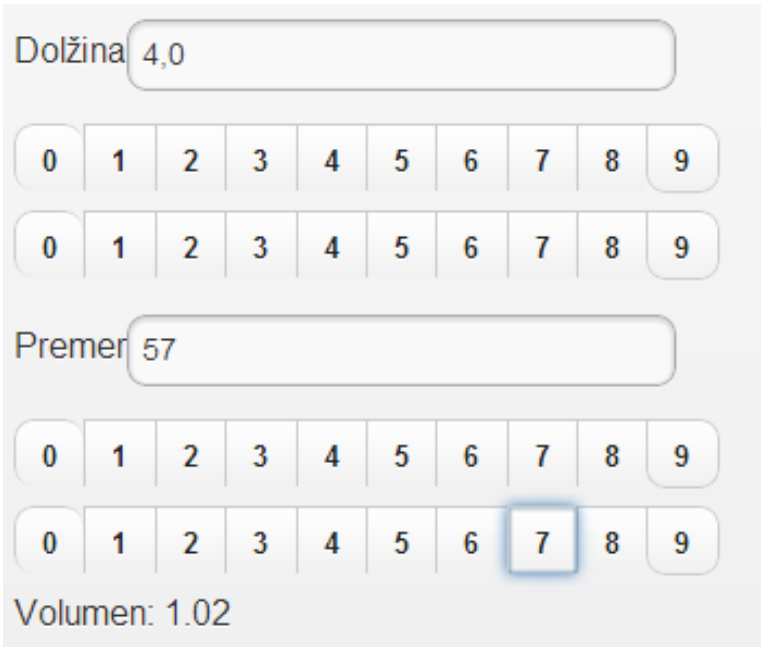
En izmed glavnih podatkov za odkup je tudi dobavitelj, primer vnosa pa vidimo na sliki 4.7. Potrebni podatki za vnos so naziv, naslov, kraj in transakcijski račun, ki pa jih ob prenosu v informacijski sistem lahko dopolnimo, če je to potrebno.

Slika 4.6: Obrazec za ustvarjanje in spreminjanje grup artiklov.

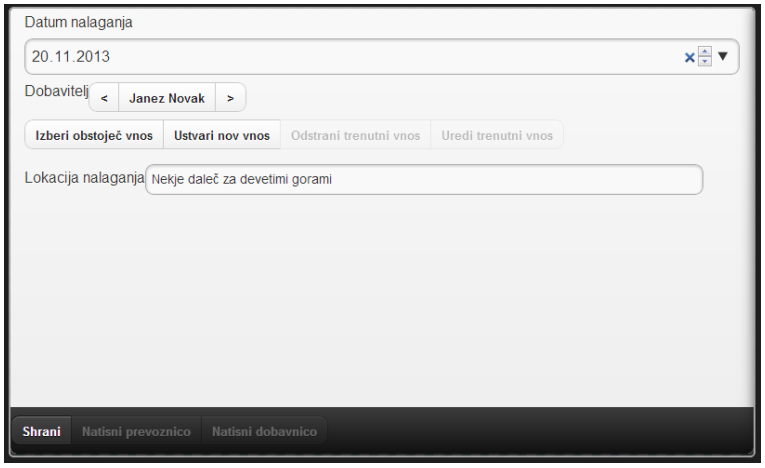
Slika 4.7: Obrazec za ustvarjanje in spreminjanje dobaviteljev.



Slika 4.8: Prikaz vnosa količin z vnašanjem volumna.



Slika 4.9: Primer vnosa hloda z dolžino in premerom.



Slika 4.10: Prikaz priprave odkupa.

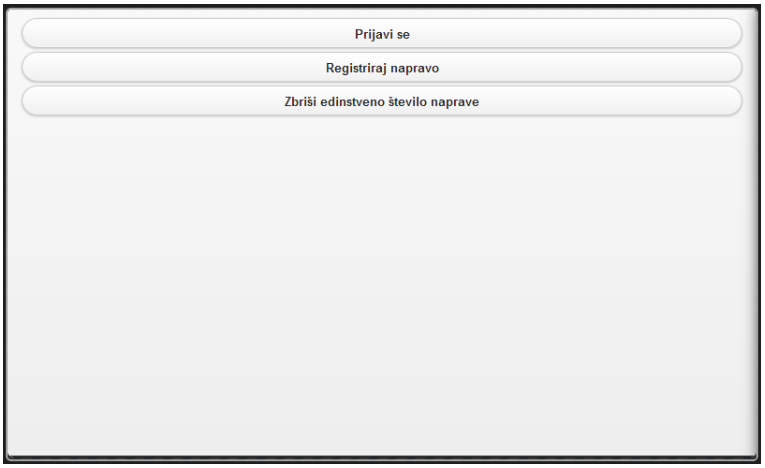
4.4.7 Vnos količin

Vnos količin je ena izmed osrednjih funkcionalnosti, pri kateri je ključnega pomena to, da nam omogoča čim bolj enostavno in hitro uporabo. Obrazec za vnos lahko vidimo na sliki 4.8. S pomočjo gumbov v glavi lahko menjamo že vnesene artikle, ki jih s pomočjo gumbov v nogi shranimo ali izbrišemo. Notranja vsebina je sestavljena iz izbire artikla, njegove cene in vnosa konkretnih podatkov o artiklu. Prikazan vnos na sliki 4.8 je namenjen direktnemu vnosu volumna in ga lahko spremenimo s pomočjo primernega gumba, ki ga najdemo v nogi in je prikazan na sliki 4.9. V nogi pa lahko najdemo tudi gumb, ki omogoča shranitev celega vnosa.

4.4.8 Nov odkup

Nov odkup nam je v pomoč pri pripravi dobavnice, iz katere lahko prikažemo podatke, ki so potrebni za prevoznico, natisnemo vse podatki o dobavnici oziroma prenesemo dobavnico v informacijski sistem, kjer jo dodatno obdelamo. Podatki, potrebni za dobavnico, so prikazani na sliki 4.10:

- datum nalaganja, ki ga lahko izberemo s pomočjo ustreznega koledarja,



Slika 4.11: Obrazec za sinhronizacijo v nepovezavnem načinu.

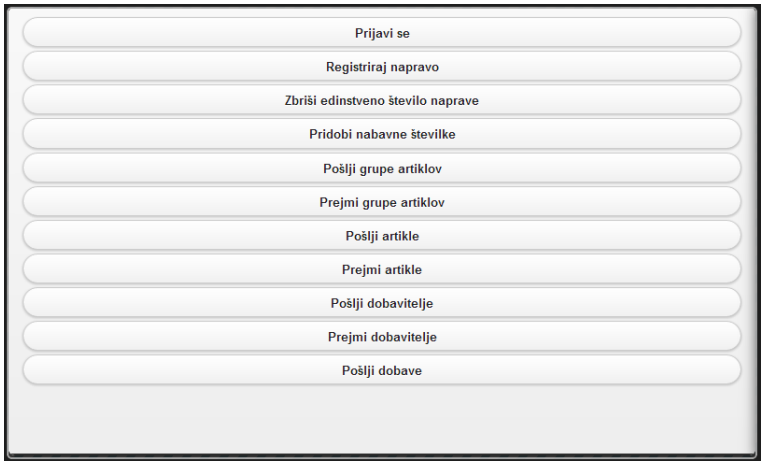
- dobavitelj, ki ga lahko izberemo s pomočjo seznama predpripravljenih dobaviteljev,
- lokacije nalaganja,
- vnos hlodovine, čemur lahko služi že predpripravljen vnos ali vnos, ki ga vnesemo na novo.

Dobavnica predstavlja osrednji dokument, ki je most med dobaviteljem in odkupovalcem ter tako tudi celotnim sistemom.

4.4.9 Sinhronizacija

Sinhronizacija je ključnega pomena za pohitritev celotnega procesa. Pred uporabo moramo najprej znotraj nastavitev nastaviti potrebne podatke in registrirati napravo s pomočjo gumba, ki ga lahko najdemo na obrazcu, prikazanem na sliki 4.11. Vse možnosti za sinhronizacijo, ki jih lahko vidimo na sliki 4.12, so na voljo šele po prijavi in zajemajo:

- pošiljanje in prejemanje skupin artiklov,
- pošiljanje in prejemanje artiklov,



Slika 4.12: Obrazec za sinhronizacijo v povezavnem načinu.

- prejemanje nabavnih številk,
- pošiljanje in prejemanje dobaviteljev,
- pošiljanje dobavnic.

4.4.10 Nastavitve

V nastavitvah, ki jih lahko vidimo na sliki 4.13, lahko nastavljamo prijavne podatke, ki so namenjeni sinhronizaciji, in generiramo številke odkupa, za katere gumb se prikaže šele ob pomanjkanju le-teh. Obrazec za nastavitve prijave lahko najdemo na sliki 4.14, kjer so obvezni podatki o imenu, geslu in naslovu, kjer se nahaja informacijski sistem.

4.5 Informacijski sistem in ključne povezave

Informacijski sistem je ključen za sinhronizacijo celotnega sistema. Služi kot zbirni center, s čimer se pohitri pretok informacij od začetne do končne točke.

Slika 4.13: Obrazec za nastavitve.

Slika 4.14: Obrazec za spreminjanje sinhronizacijskih nastavitev.

4.5.1 Prijava

Pred izmenjavo katerekoli informacije se moramo prijaviti, kar opravimo s pomočjo posebnega gumba za prijavo znotraj sinhronizacije. Pred poskusom prijave pa moramo znotraj nastavitev vnesti potrebne podatke za prijavo:

- naslov informacijskega sistema,
- uporabniško ime,
- uporabniško geslo.

Ob vsaki uspešno izvedeni prijavi se na strežniški strani ustvari poseben ključ dostopa in pošlje mobilni napravi, ki ga nato uporabljamo pri ostalih dejanjih za sinhronizacijo.

4.5.2 Registracija

Napravo moramo pred kakršnokoli praktično uporabo prej tudi registrirati znotraj informacijskega sistema. Registracija poteka v več korakih:

1. Moramo biti prijavljeni, s čimer naprava pridobi ključ dostopa.
2. Znotraj določenega časovnega intervala po prijavi moramo napravo potrditi v informacijskem sistemu.
3. S pritiskom ustreznega gumba za registracijo pod sinhronizacijo v nastavitvah lahko za napravo pridobimo edinstveno številko, s katero je registrirana znotraj informacijskega sistema.

4.5.3 Pošiljanje

Za sinhronizacijo moramo pošiljati podatke o dobaviteljih, skupinah artiklov, artiklih in dobavah. Vsa pošiljanja smo posplošili in delujejo na podoben način. Funkcija, ki skrbi za pošiljanje, sprejema naslednje parametre:

- zadnja sprememba, s katero določimo zadnjo spremembo, ki je bila storjena na podatkih, ki jih pošiljamo,
- zadnje pošiljanje, ki ga primerjamo z zadnjo spremembo, pošiljanje pa izvedemo le v primeru, ko je bila sprememba narejena po zadnjem pošiljanju,
- ciljni naslov, s katerim določimo naslov, kamor pošiljamo podatke,
- skladišče, za katerega velja, da imamo v njem nove (oziroma spremenjene) podatke,
- vrnitvena funkcija, ki jo pokličemo ob uspešno izvedenem pošiljanju.

Vse poslane podatke moramo s strani informacijskega sistema potrditi in šele nato premakniti v pravilno skladišče, kar pa izvedemo na strani prejemanja.

4.5.4 Prejemanje

Prejemna stran sinhronizacije nam omogoča prejemanje podatkov o dobaviteljih, skupinah artiklov, artiklih in številkah nabav. Prav tako omogoča, da potrjene podatke prenesemo v pravilno skladišče. Posplošena funkcija za vsa prejemanja ima naslednje parametre:

- Naslov za prejemanje, kamor se funkcija poveže za pridobitev podatkov.
- Funkcijo za konvertiranje podatkov, ki jih pošlje informacijski sistem. Funkcija mora med drugim nujno izluščiti edinstven ključ, ki predstavlja ključ za določeno vsebino v informacijskem sistemu in ki je ista v vseh napravah.
- Funkcijo, ki omogoča posodobitev novega podatka iz starega, saj obstajajo podatki, ki so edinstveni za napravo in jih kot take ne smemo povoziti.

- Skladišče novih (oziroma spremenjenih) in uvoženih podatkov. Najprej preverimo, ali je določen podatek prišel iz te naprave, nato pa ga prestavimo v skladišče uvoženih podatkov. V primeru, ko temu ni tako, ustvarimo nov oziroma posodobimo podatek iz skladišča uvoženih podatkov.
- Funkcijo, ki ji podamo dva elementa, za katera mora ugotoviti, ali sta ista, in to vrniti.
- Vrnitveno funkcijo, ki jo pokličemo po uspešnem prejemanju in shranjevanju.

Prejemanje nabavnih številk

Posebno vrednost imajo nabavne številke, ki jih tudi prejemamo na drugačen način. Za te številke velja, da morajo biti nujno enolične na vseh napravah. To smo uredili tako, da ob povezavi z informacijskim sistemom in ob zahtevi za nabavne številke na strežniški strani rezerviramo določeno število nabav za določeno napravo. Vse te številke so pri tem tudi poslane mobilni napravi, saj je lahko pred naslednjim prejemanjem narejenih več nabav, od katerih mora biti vsaka edinstvena v celotnem sistemu.

Poglavje 5

Zaključek

V diplomski nalogi smo želeli rešiti predstavljeni primer, izbrati primerne tehnologije, razviti prototip in pridobiti čim več podatkov o težavah, ki bi se pri tem pojavile, kar bi nam bilo v pomoč kot osnova za razvoj končnega produkta. Na začetku smo opisali naprave, orodja in druge tehnologije, namenjene razvoju prototipa, ki smo jih kasneje tudi bolj podrobno predstavili. V procesu razvoja smo uporabili tudi tehnologije, za katere velja, da se še razvijajo. Taka je na primer baza PouchDB, ki predstavlja bazo NoSQL, kar se je pokazalo kot dobra rešitev, saj nam omogoča hitrejši razvoj in spreminjanje. Kot minimizirana različica za mobilne naprave in tehnologija v razvoju pa se pri bazi NoSQL opazi tudi določene pomanjkljivosti, ki pa niso nerešljive.

Sam prototip je pokazal, da ima potencial, s katerim bi bilo možno pohitriti celoten proces dela. Potrebno bi bilo še veliko dela, ki bi med drugim zajemal predvsem:

- Delo na vmesniku, ki zajema tako estetsko plat kot izboljšanje enostavnosti uporabe.
- Povezavo z mobilnim tiskalnikom, ki bi omogočil zamenjavo trenutnega načina dela.
- Bolj transparentno sinhronizacijo z informacijskim sistemom, ki bi de-

loval samodejno brez uporabnika.

- Razvoj pametnega sistema nadgrajevanja aplikacije, ki bi omogočil hitro prilagajanje na spremembe.

V nadaljevanju se kot eden izmed ključnih problemov lahko pojavi pri uveljavljanju obstoječega dela z novo rešitvijo, ki pa bi s časom lahko znatno doprinesla delu.

Literatura

[1] David Sawyer McFarland, *JavaScript and jQuery: The Missing Manual 2nd Edition*, Sebastopol: O'Reilly Media, Inc, 2011, Združene države Amerike, pogl. 4, 11.

[2] Jeff McWherter, Scott Gowell *Professional Mobile Application Development*, Indiana: John Wiley and Sons, Inc, Zvezne države Amerike, 2012, pogl. 11.

[3] John Resig, *Secrets of the JavaScript Ninja*, Manning Publications Co., December, 2012, pogl. 5.

[4] John M. Wargo, *PhoneGap Essentials: Building Cross-Platform Mobile Apps*, Indiana: RR Donnelley in Crawfordsville, Združene države Amerike, 2012, pogl. 1.

[5] (2013) Using hardware devices. Dostopno na: http://developer.android.com/tools/device.html

[6] (2013) Remote Debugging on Android. Dostopno na: https://developers.google.com/chrome-developer-tools/docs/remote-debugging

[7] (2013) NetBeans. Dostopno na: http://netbeans.org/

[8] (2013) IDE. Dostopno na: http://en.wikipedia.org/wiki/Integrated_development_environment

[9] (2013) NetBeans. Dostopno na:
http://en.wikipedia.org/wiki/Net_Beans

[10] (2013) Software Development Kit. Dostopno na:
http://en.wikipedia.org/wiki/Software_development_kit

[11] (2013) Java Development Kit. Dostopno na:
http://en.wikipedia.org/wiki/Java_Development_Kit

[12] (2013) Android Software Development. Dostopno na:
http://en.wikipedia.org/wiki/Android_software_development

[13] (2013) The Apache Ant Project. Dostopno na:
http://ant.apache.org/

[14] (2013) Apache Ant. Dostopno na:
http://en.wikipedia.org/wiki/Apache_Ant

[15] (2013) GNU Make. Dostopno na:
http://www.gnu.org/software/make/

[16] (2013) Make (software). Dostopno na:
http://en.wikipedia.org/wiki/Make_(software)

[17] (2013) OEM USB Drivers. Dostopno na:
http://developer.android.com/tools/extras/oem-usb.html

[18] (2013) NBAndroid. Dostopno na:
http://nbandroid.org/

[19] (2013) NBAndroid installation. Dostopno na:
http://nbandroid.org/wiki/index.php/Installation

[20] (2013) PhoneGap. Dostopno na:
http://phonegap.com/

[21] (2013) PhoneGap. Dostopno na:
http://en.wikipedia.org/wiki/PhoneGap

[22] (2013) Apache Cordova. Dostopno na:
http://cordova.apache.org/

[23] (2013) Foreign Function Interface. Dostopno na:
http://en.wikipedia.org/wiki/Foreign_Function_Interface

[24] (2013) weinre. Dostopno na:
http://people.apache.org/~pmuellr/weinre/docs/latest/

[25] (2013) Getting started with Weinre mobile debugging on Windows. Dostopno na:
http://dustint.com/post/482/getting-started-with-weinre-mobile-debugging-on-windows

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

[27] (2013) Node.js. Dostopno na:
http://en.wikipedia.org/wiki/Nodejs

[28] (2013) Aardwolf. Dostopno na:
http://lexandera.com/aardwolf/

[29] (2013) Apache Ripple. Dostopno na:
http://ripple.incubator.apache.org/

[30] (2013) Ripple Emulator. Dostopno na:
https://chrome.google.com/webstore/detail/ripple-emulator-beta/geelfhphabnejjhdalkjhgipohgpdnoc

[31] (2013) HTML. Dostopno na:
http://en.wikipedia.org/wiki/HTML

[32] (2013) HTML5. Dostopno na:
http://en.wikipedia.org/wiki/HTML5

[33] (2013) Cascading Style Sheets. Dostopno na:
http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[34] (2013) JavaScript. Dostopno na:
http://en.wikipedia.org/wiki/JavaScript

[35] (2013) Simple JavaScript Inheritance. Dostopno na:
http://ejohn.org/blog/simple-javascript-inheritance/

[36] (2013) jQuery. Dostopno na:
http://jquery.com/

[37] (2013) jQuery. Dostopno na:
http://en.wikipedia.org/wiki/JQuery

[38] (2013) jQuery mobile. Dostopno na:
http://jquerymobile.com/

[39] (2013) jQuery Mobile. Dostopno na:
http://en.wikipedia.org/wiki/JQuery_Mobile

[40] (2013) jQuery user interface. Dostopno na:
http://jqueryui.com/

[41] (2013) jQuery UI. Dostopno na:
http://en.wikipedia.org/wiki/JQuery_UI

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

[43] (2013) Ajax (programming). Dostopno na:
http://en.wikipedia.org/wiki/Ajax_(programming)

[44] (2013) Should You Design your SaaS as a Single Page Application? Dostopno na:
http://sandhill.com/article/should-you-design-your-saas-as-a-single-page-application

[45] (2013) Single-page application. Dostopno na:
http://en.wikipedia.org/wiki/Single-page_application

[46] (2013) PouchDB. Dostopno na:
http://pouchdb.com

[47] (2013) POUCHDB: AN INTRO. Dostopno na:
http://calvinmetcalf.com/post/39926807885/pouchdb-an-intro