UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Davor Vertelj

Mobilna in spletna aplikacija za vnos meritev s terena

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Davor Vertelj

Mobilna in spletna aplikacija za vnos meritev s terena

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Rok Rupnik

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Podjetja, ki izvajajo meritve na terenu, potrebujejo možnost vnosa rezultatov meritev prek mobilne aplikacije. Poleg mobilne aplikacije pa potrebujejo tudi spletno aplikacijo za spremljanje rezultatov meritev.

Zasnujte in razvite sistem za podporo izvajanju meritev na terenu, v okviru katerega skupno podatkovno bazo uporabljata spletna in mobilna aplikacija. Pri razvoju uporabite Google App Engine, mobilno aplikacijo pa razvijte za platformo Android.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Davor Vertelj, z vpisno številko **63050126**, sem avtor diplomskega dela z naslovom:

Mobilna in spletna aplikacija za vnos meritev s terena

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika
- 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 na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 1. septembra 2014 Podpis avtorja:

Iskreno se zahvaljujem mentorju, doc. dr. Roku Rupniku za dobro voljo ter vso pomoč in nasvete pri izdelavi diplomske naloge. Zahvaljujem se tudi svoji družini za vse lepe besede in pozitiven odnos. Posebej pa bi se rad zahvalil tudi svoji boljši polovici Jerci, ki mi je nesebično stala ob strani celo leto in me v ključnih trenutkih spodbujala ter gnala naprej.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
GAR	Google App Engine	Google App Engine
SDK	Software Development Kit	Orodja za razvoj programske opreme
GQL	Google Query Language	Google Query Language
SQL	Structured Query Language	Structured Query Language
OS	Operating system	Operacijski sistem
IDE	Integrated Development Environment	Integrirano razvojno okolje
JSON	JavaScript Object Notation	JavaScrip Object Notation
XML	eXtensible Markup Language	eXtensible Markup Language
HTTP	HyperText Transfer Protocol	HyperText Transfer Protocol
REST	REpresentational State Transfer	REpresentational State Transfer
URL	Uniform Resource Locator	Enolični krajevnik vira
API	Application Program Inteface	Aplikacijski programski vmesnik

Povzetek

Osnova za diplomsko nalogo je izdelava aplikacije, ki s pomočjo mobilnega telefona omogoča neposreden vnos meritev iz terena v podatkovno bazo, brez potrebe po zapisovanju meritev na list papirja. Namen aplikacije je olajšati delo terenskim delavcem. Aplikacija je sestavljena iz dveh delov, prvi je spletna aplikacija, zgrajena z uporabo platforme Google App Engine, programskega jezika Python in Jinja2 templating language-a. Spletna aplikacija je namenjena administraciji, vsebuje namreč dodajanje, urejanje in brisanje uporabnikov, tipov meritev in meritev samih iz aplikacije. Drugi del aplikacije pa je mobilna aplikacija, ki je narejena za operacijski sistem Android in razvita v programskem jeziku Java. Mobilna aplikacija ima nalogo pošiljanja meritev spletni aplikaciji. V diplomi so predstavljene vse tehnologije in orodja, ki sem jih uporabil da sem lahko razvil aplikacijo in dosegel zastavljeni cilj, olajšanje dela terenskim delavcem.

Ključne besede: Google App Engine, Android, mobilna aplikacija, spletna aplikacija, meritve

Abstract

This thesis is based on developing an application that would enable entering measurements to the database without the need for writing the measurements down to a piece of paper but rather utilising a mobile app for this task. The purpose of this application is to ease the workload of field workers. The application consists of a web application built by using the Google App Engine platform, Python programming language and Jinja2 templating language. The web application's purpose is more that of administrative nature. It includes adding, editing and deleting users, measurement types and the measurements themselves. The second part of the application is the mobile application that's built for Android and developed in Java programming language. The purpose of the mobile application is sending measurements to the web application. All the tools and technologies used in developing this application and thus achieving my goal of making the field workers' job easier are described in this thesis.

Keywords: Google App Engine, Android, mobile application, web application, measurements

Poglavje 1 Uvod

Na področju računalništva je razvoj neverjetno hiter. Po Moorovem zakonu naj bi se število tranzistorjev podvojilo vsake dve leti, kar posledično pomeni hitrejše in zmogljivejše računalnike. Če smo se pred nekaj leti čudili hitrosti naših osebnih računalnikov, je danes razvoj prišel tako daleč, da so mobilni telefoni enako, če ne še bolj zmogljivi.

Tudi razvoj mobilnih tehnologij je vsako leto hitrejši, prav tako pa se iz leta v leto povečuje število njenih uporabnikov. Pametne telefone, ki so bili pred leti dostopni le premožnejšim kupcem, danes najdemo na vsakem koraku. Minili so tudi časi, ko je bilo treba za prenos podatkov plačevati visoke zneske, kajti operaterji danes ponujajo pakete z nekaj GB prenosa zastonj, kar je bilo včasih nepredstavljivo.

Zaradi hitrega razvoja tako računalnikov kot mobilnih tehnologij pa je postalo tudi prepletanje teh dveh področij neizbežno. Prav takšno povezavo med telefonom in računalnikom sem realiziral tudi sam v svoji diplomski nalogi. Glavna ideja za moje delo pa je bila poenostavitev vpisovanje različnih meritev s terena.

Predstavljajte si kontrolorja ali popisovalca v podjetju, ki mora preveriti stanje števcev na različnih lokacijah. Najprej se odpelje na teren do posameznih števcev, si zapiše vrednosti stanja števcev na list, se odpelje nazaj na delovno mesto in meritve vpiše v bazo. Cel proces se lahko močno poenostavi, če se popisovalcu omogoči pošiljanje meritev prek telefona. Na tak način so meritve poslane takoj, opraviti je mogoče več meritev v enakem času kot prej, saj jih ni treba naknadno prepisovati v elektronsko obliko, odpravi pa se tudi možnost napak, kot je na primer izguba lista s popisom s terena.

Diplomsko delo je sestavljena iz teoretičnega in praktičnega dela. V teoretičnem delu predstavim tehnologijo, ki sem jo uporabil za izvedbo svoje ideje. Opisal sem Google App Engine in operacijski sistem Android, na kratko predstavil glavne lastnosti ter prednosti in slabosti. V praktičnem delu pa predstavim svojo rešitev problema, tehnologije, ki sem jih pri tem uporabil, probleme, s katerimi sem se srečeval pri razvoju, nekaj slik same aplikacije ter na koncu še nekaj idej, kako bi aplikacijo lahko še dodatno izboljšal.

Cilj diplomske naloge je olajšati delo popisovalcem na terenu in modernizirati postopek vpisa različnih meritev. Rešitev, ki sem jo naredil, je generična in primerna za kakršenkoli tip meritev, tako da bi bila primerna za uporabo na najrazličnejših področjih, od uporabe v podjetjih ki dobavljajo električno energijo, vodo ali plin, pa vse do prodajalcev sadja.

Poglavje 2 Google App Engine

Google App Engine, v nadaljevanju GAE, je razvojna platforma v oblaku za razvoj in gostovanje spletnih aplikacij v Googlovih podatkovnih središčih. [1] Aplikacija ne teče na posameznem strežniku, temveč je njeno delovanje razdeljeno na več strežnikov, kar omogoča avtomatično skaliranje spletnih aplikacij. To pomeni, da aplikacija, ki mora obdelati veliko število zahtev dnevno, deluje enako hitro kot aplikacija, ki mora dnevno obdelati relativno majhno število zahtev. GAE ob povečani obremenitvi aplikaciji samodejno dodeli več sredstev za njeno delovanje.

Prvič se je GAE pojavil kot predogledna verzija aprila 2008, nato pa je septembra 2011 prišel v produkcijo. Trenutno stabilna verzija je 1.9.0, ki je izšla februarja 2014. [1]

Trenutno podpira razvoj s štirimi programskimi jeziki: Java, Python, PHP in Go. Od teh sta PHP in Go trenutno še v poskusnem stanju, zato so GAE aplikacije najpogosteje razvite v Javi ali Pythonu. Možna je tudi uporaba različnih ogrodij (frameworks) za programske jezike, kot so naprimer Django, Flask in Webapp2 za Python ter Spring za Javo. [2]

Zelo pomembna lastnost GAE je možnost hitrega prototipiranja, ki je omogočeno zaradi kratkega seznama zahtev za začetek razvoja. Na začetku se moramo le odločiti, v katerem programskem jeziku bomo programirali, nato si namestimo ustrezen SDK (Software Development Kit), pripravimo urejevalnik besedil (npr. Sublime), nastavimo nekaj osnovnih nastavitev ter končno poženemo lokalen strežnik kar iz GAE zaganjalnika (launcher), s čimer je proces končan in naša koda oživi.

Razvoj in uporaba GAE aplikacij je do določene mere brezplačna, kar je nedvomno velika prednost pred ostalimi ponudniki takšnih rešitev. Aplikacijo teko lahko hitro splovimo na internet, vidimo kakšen je odziv in če je ta prevelik za okvirje brezplačne aplikacije, šele takrat plačamo za gostovanje na Googlovih strežnikih. [3]

Slika 1: Logotip Google App Enginea

2.1 Prednosti

Nekaj prednosti razvoja GAE aplikacij je bilo omenjenih že v prejšnem poglavju, v nadaljevanju pa bodo podrobneje opisane. [4]

2.1.1 Brezplačna uporaba do določene meje

Aplikacije tečejo na Googlovih strežnikih, zato porabljajo njihov procesorski čas. Vsaka aplikacija je brezplačna za uporabo, dokler ne preseže kvote, ki določa, kdaj se bo za uporabo strežnikov začelo zaračunavati. Kvote se ponastavljajo na 24 ur. V primeru da aplikacija preseže kvoto, se delovanje aplikacije ustavi, nakar imamo dve možnosti za ponovno vzpostavitev delovanja; bodisi, da počakamo do preteka 24 ur, ko se kvote ponastavijo in se sredstva za delovanje aplikacije spet sprostijo ali pa vključimo zaračunavanje. Pri tem določimo dnevni limit, ki mora biti dovolj velik, da aplikacija brez težav prenese tudi povečane obremenitve zaradi prometa.

Brezplačnost storitve do določene meje, pa ni zgolj vaba za privabljanje čim večjega števila uporabnikov, ki bi bili nato po produkciji primorani v plačevanje, saj so kvote nastavljene precej visoko in jih je z normalno uporabo in obiskom težko preseči. Če pa do tega pride, stroški niso tako visoki. [3]

2.1.2 Ni stroškov z administracijo

GAE aplikacije se nahajajo v oblaku, bolj natančno na Googlovih strežnikih, kar pomeni, da stroškov z nakupom strežnikov ni, saj je za vse podkrbljeno s strani Googla.

Podobno tudi stroškov z administracijo strežnikov ni. Sistemskega inženirja, ki bi ga plačevali samo zato, da skrbi za strežnike, ne potrebujemo, ampak lahko sredstva porabimo najem razvijalca, ki bo delal na sami aplikaciji.

2.1.3 Zanesljivost

Kljub že omenjenim prednostim je strah, da nimamo neposrednega nadzora nad strežniki in njihovo zanesljivostjo odveč

Že Service-level Agreement dokument [5] nam zagotavlja 99.95% nemoteno uporabo GAE, saj je zasnovan tako, da lahko kljub izpadu več podatkovnih centrov še vedno nemoteno deluje naprej.

V primeru, da se odločimo za Premium račun, nam je zagotovljena tudi podpora Googlovih inženirjev[https://developers.google.com/appengine/docs/premier/], sicer pa nam je še vedno na voljo Google Groups in StackOverflow.

2.1.4 Skalabilnost

Ena izmed največjih prednosti GAE pa nedvomno sposobnost skaliranja aplikacij glede na potrebe. Tudi če smo za svojo aplikacijo v začetku predvidevali manjši obisk, bo aplikacija kljub nenadnemu povečanju števila uporabnikov delovala nemoteno, kljub večjim obremenitvam. Pri GAE je poskrbljeno za avtomatsko dodeljevanje dodatnih virov aplikaciji, če jih ta zaradi povečanega obiska ali števila uporabnikov potrebuje. Obstaja nekaj smernic za razvoj skalabilnih GAE aplikacij, vendar gre v večini priemrov zgolj za pravila, kako skalabilnost še izboljšati in ne kako omogočiti.

2.2 Slabosti

Seveda pa ima GAE tudi nekaj slabosti.

2.2.1 Read-only dostop na strežniku

Zaradi načina delovanja GAE, kjer je lahko aplikacija hkrati na več strežnikih po svetu, ne moremo pisati na strežnikov file-system. Na voljo nam je samo branje, za shranjevanje slik ali besedila pa lahko uporabimo Googlovo podatkovno bazo, ki je opisana v nadaljevanju. [6]

2.2.2 Omejitve podatkovne baze

GAE za shranjevanje podatkov uporablja BigTable, ki je kompresiran in visoko zmogljiv Googlov sistem za shranjevanje podatkov. Za iskanje entite v bazi uporablja jezik GQL (Google Query Language), ki je podoben SQL-u (Structured Query Language), vendar ima nekaj omejitev, kot je recimo ta, da v eni poizvedbi ne smemo imeti dveh filtrov neenakosti. To lahko pripelje do neučinkovitosti kode. [7]

Na podlagi zgoraj opisanih dejstev lahko trdimo, da je GAE odlična izbira za startup podjetja in manjša podjetja, ki bi rada na trg čim prej splovila aplikacijo in si ne morejo privoščiti

izgube časa in denarja s postavljanjem vse infrastrukture, ki je sicer potrebna za zagon spletnih aplikacij.

Poglavje 3 Android

Android je mobilni operacijski sistem, ki ga je zasnoval Google. Primarno je namenjen za naprave, ki podpirajo touchscreen, kot so mobilni telefoni in tablice, uporablja pa se tudi kot operacijski sistem pri televizorjih (Android TV), v avtomobilih (Android Auto), v urah (Android Wear), igralnih konzolah, digitalnih fotoaparatih in ostalih elektrotehničnih napravah. OS uporablja vnose z dotikom, kot so poteg, kratek dotik, »ščipanje« in navidezno tipkovnico. [8]

Po zadnjih podatkih je Android v drugem četrtletju 2014 dosegel rekorden delež na trgu, in sicer kar 85%.

Operacijski sistem	Q2 2013	Q2 2014
Android	80.2%	84.6%
Apple iOS	13.4%	11.9%
Microsoft	3.8%	2.7%
BlackBerry	2.4%	0.6%
Ostali	0.2%	0.2%

Tabela 1: Globalni delež operacijskih sistemov na pametnih telefonih [9]

Začetki operacijskega sistema segajo v leto 2003, ko je bil ustanovljen Android Inc. Nato jih je leta 2005 kupil Google in 23. septembra 2008 je uradno izšel Android 1.0.. Trenutna verzija ima zaporedno številko 4.4.4 in nosi ime KitKat. [10]

Programiranje za operacijski sistem Android poteka v Javi, pri čemer se je treba priučiti nekaterih posebnosti razvoja za Android, kot so recimo koncept Activity-jev, Intent-ov in ostalih gradnikov, ki sestavljajo celotno aplikacijo.

3.1 Zgradba in razvoj

Operacijski sistem Android razvija Google, razvit pa je v jezikih C, C++ in Java. Razvoj poteka privatno in sicer dokler niso najnovejši popravki in spremembe pripravljene za splovitev. Takrat postane izvorna koda javno dostopna. Izvorna koda brez modifikacij teče zgolj na določenih napravah (po navadi gre za telefone Google Nexus), različni proizvajalci pa kodo prilagodijo za svoje naprave.

Nadgradnje operacijskega sistema so običajno na voljo vsakih 6-9 mesecev, vendar je čas, v katerem te nadgradnje pridejo na vse naprave, daljši. Glavni razlog za to je dejstvo, da morajo proizvajalci najnovejšo nadgradnjo ponovno prilagoditi za svoje telefone, kar pa pri velikem številu naprav pomeni veliko časa in denarja. Velikokrat se tudi zgodi, da proizvajalci zanemarijo starejše naprave na račun novih, bolje prodajanih.

3.2 Prednosti

Operacijski sistem Android velja za zelo odprtega za novosti, zato ni presenetljivo, da je na področju razvoja aplikacij zanj zelo živahno. Glavne prednosti razvoja za Android v primerjavi z drugim največjim operacijskim sistemom (iOS) so naslednje [11]:

3.2.1 Razvoj ni odvisen od strojne opreme

Za razliko od iOS aplikacij lahko Android aplikacije razvijamo na vsakem računalniku in nismo vezani na točno določeno strojno opremo. Potrebujemo le IDE (Integrated development environment), ki ga pripravimo za delo z Androidom in že lahko začnemo s kodiranjem naše aplikacije.

3.2.2 Hiter razvoj in popravki

Google za objavo aplikacije na svojem marketu (Google Play) ne zahteva predhodne odobritve z njihove strani, kar pomeni, da lahko naložimo še ne popolnoma končano aplikacijo ter jo z nekaj popravki spravimo do 100% verzije, medtem pa že spremljamo odzive uporabnikov.

3.2.3 Velika baza uporabnikov

Ker je Android najbolj razširjen mobilni operacijski sistem, imamo na voljo ogromno potencialnih uporabnikov naše aplikacije. To nam pride prav tako za pridobivanje povratnih informacij, česa bi si v aplikaciji še želeli in ne nazadnje seveda tudi za zaslužek.

3.2.4 Odprtost za nove ideje

Ponovno, ker za objavo aplikacije ni potrebno pridobiti odobritve s strani Googla in ker na splošno omejitev pri tem, kaj bomo razvijali, ni, je odprtost za nove ideje zelo velika, saj lahko razvijemo praktično karkoli, le dobro idejo potrebujemo.

3.3 Slabosti

Odprtos platforme je večinoma sicer pozitivna lastnost, vendar pa vodi tudi do nekaj slabosti.

3.3.1 Ogromno število naprav in verzij operacijskih sistemov

Za razliko od iOS-a, kjer je naprava samo ena (iPhone), je pri Androidu veliko več tipov naprav, ki imajo različne verzije operacijskega sistema, kar otežuje optimizacijo aplikacije. Odločiti se moramo, ali bomo podpirali le najnovejše naprave in s tem uporabljali tudi najnovejše tehnologije ali pa bomo našli nek kompromis pri uporabi tehnologij in s tem zajeli večji spekter naprav.

3.3.2 Možnost hroščev v končni aplikaciji

Ker Google na zahteva preverjanja aplikacije pred naložitvijo na Google Play, je vse testiranje odvisno od razvijalca. Seveda so možnosti za spregledane napake tudi pri Applu, vendar je pri preverjanju pred objavo aplikacije, ki ga izvajajo, vseeno možno, da očitne napake opazijo.

Poglavje 4 Uporabljene tehnologije in orodja

Za rešitev zastavljenega problema sem uporabil več različnih tehnologij in orodij, ki jih bom v tem poglavju tudi podrobneje opisal. Ker sem delal v dveh različnih tehnologijah, bom orodja ločil glede na tip tehnologije, pri katerem sem to orodje uporabil.

4.1 Orodja, uporabljena pri Google App Engine aplikaciji

Za dokončanje GAE aplikacije sem uporabil nekaj različnih orodij, vsako od njih je prispevalo svoj delež h končni rešitvi.

4.1.1 Sublime text 2

Za kodiranje dela aplikacije, ki teče na GAE sem uporabil kar Sublime text 2, ker je odličen tako za kodiranje v Python-u, kot v veliko ostalih jezikih. Res je, da nima vseh naprednih funkcij, ki jih najdemo v bolj naprednih IDE-jih kot so Eclipse, Netbeans ali IntelliJ IDEA, vendar je s svojo široko paleto zelo uporabnih možnosti urejanja besedil/kode zelo dobra izbira. Na voljo je za Windows, Linux in OS X. [12]

Njegove glavne lastnosti so: [12]

- **Zelo zmogljiv »iskalnik dokumentov«.** S kombinacijo tipk ctrl+P (na Windowsih) odpremo pojavno okno, v katerega začnemo vpisovati ime datoteke, ki bi jo radi odprli. Rezultati se sproti prikazujejo in osvežujejo dokler ne najdemo iskanega dokumenta. Omogoča tudi skok na točno določeno funkcijo ali vrstico v dokumentu

- **Hkratno izbiranje več delov teksta**. Ta lastnost nam omogoča, da hkrati spreminjamo kodo na več mestih. Če moramo recimo popraviti kakšno for zanko, ki se ponovi večkrat v kodi, potem to lahko enostavno storimo tako, da označimo vse for

zanke, ki jih želimo spremeniti in vnesemo spremembe. Nič več spreminjanja vsake besede posebej

- **Način brez motenj (Distraction free mode).** Gre za način, pri katerem celo površino zaslona zasede Sublime, tako da nas razna obvestila, npr. o prejeti pošti, ne motijo.

- **Ogromno možnosti za prirejanje urejevalnika po svojih željah**. Skoraj vse je s pomočjo JSON datotek mogoče priredi željam posameznika, od bližnjic na tipkovnici do Makrojev in dokončevanja (completion) kode. Spremembe so lahko globalne in veljajo za celotno aplikacijo, lahko pa jih omejimo tudi samo na določen tip datoteke ali na en projekt.

4.1.2 Google App Engine launcher

Po postavitvi lokalnega okolja in vseh potrebnih nastavitvah, moramo pognati lokalni strežnik. Za to imamo na voljo dve možnosti. Lahko ga poženemo preko konzole, lahko pa za to uporabimo GAE launcher. Sam sem uporabljal slednjo rešitev, zato jo bom tudi podrobneje opisal.

Launcher omogoča enostavno izvajanje vseh ukazov, ki jih potrebujemo za delo z GAE aplikacijami. Omogoča nam zagon in zaustavitev strežnika, na katerem teče aplikacija. Ko strežnik zaženemo, lahko na prvo stran aplikacije pridemo kar z gumbom v launcherju. Launcher je uporaben tudi za iskanje napak in razhroščevanje pri razvoju, saj nam omogoča da odpremo arhiv (loge) izpisov aplikacije in velikokrat nam ravno to pove, kjer se nahaja napaka, zaradi katere nam aplikacija ne dela tako, kot si želimo. Na voljo je tudi gumb za odprtje SDK konzole, ki je v bistvu administrativna stran naše aplikacije, kjer so vidime vse informacije o njej. Najpomembnejša funkcija launcherja poleg zagona strežnika pa je splovitev aplikacije na internet. Ko smo zadovoljni s svojim delom, lahko aplikacijo splovimo, da je dostopna celemu svetu. To lahko storimo z enostavnim pristiskom na gumb, vpisom uporabniškega imena in gesla in že smo na spletu, pripravljeni na delo.

Med samo uporabo sem launcher največkrat uporabljal za zagon strežnika in pa za razhroščevanje s pomočjo arhiva izpisov aplikacije.

4.1.3 Programski jezik Python

Ves razvoj GAE aplikacije je potekal v programskem jeziku Python. Kot sem omenil v uvodu, sem imel na izbiro za programiranje še Java, PHP in Go, vendar sem se zaradi predznanja in zabavnega programiranja odločil za Python.

Python je visokonivojski programski jezik, pri katerem je poudarjena razumljivost kode, njegova sintaksa pa programerjem omogoča, da le z manj vrsticami kode opravijo delo, za katerega bi v katerem drugem programskem jeziku, recimo Javi, potrebovali neprimerno več vrstic. Za primer si lahko pogledamo enostaven »Hello world« programček v Javi in Pythonu:

Java

```
public class Hello {
        public static void main(String[] args) {
                System.out.println(»Hello world!«);
        }
}
```

Python

```
print(»Hello world«)
```

Od večine popularnejših programskih jezikov se Python razlikuje tudi po tem, da za ločevanje blokov kode Python ne uporablja ločil, temveč zamike vrstic, kar nam lahko ob prehodu iz kakšnega drugega jezika na začetku povzročati nekaj težav.

Podpira več različnih programerskih paradigm:

- objektno usmerjeno programiranje,
- imperativno programiranje,
- funkcijsko programiranje in
- proceduralno programiranje.

Python ima ogromno število standardnih knjižnic, zaradi česar je že takoj po namestitvi na računalnik primeren za veliko večino opravil. Pravimo da Python sledi sloganu »Baterije priložene«, kar pomeni da naj ne bi imeli dodatnega dela z nastavljanjem in iskanje potrebnih knjižnic za delo, ampak imamo že vse vključeno v osnovni paket. To zagotovo ena izmed njegovih najmočnejših prednosti. [13]

4.1.4 Jinja2 templating language

Jinja2 je moderen in uporabniku prijazen templating jezik za Python. Jinja2 je osnovana na Django templating jeziku, zato sta sintaksa in uporaba podobni. Glavne prednosti uporabe templating jezika kot je Jinja2 so [14]:

- **Osnovna HTML datoteka, od katere nato vse ostale podedujejo elemente, ki smo jih nastavili.**
- **Enostavno spreminjanje izgleda naše strani.** Ker vse podstrani dedujejo od osnovne, spremenimo samo osnovno in sprememba se bo poznala na vseh straneh, torej ni potrebe po spreminjanju vsake strani posebej.
- **Ločevanje kode od predstavitve.**
- **Izboljšana varnost spletne strani.** Zaradi autoescape-anja se izognemo možnostim za XSS napade.
- **Enostavna povezava med kodo in predstavitvijo**. V kodi deklariramo spremenljivke, ki jih bomo nato na strani uporabili za prikaz.

Primer enostavne for zanke, ki nam sestavi seznam elementov [14]:

```
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

4.1.5 Webapp2 framework

Webapp2 je framework, ki na nek način nadgrajuje Google App Engineov lastni webapp. Deluje zelo podobno kot webapp s to razliko, da ima bolje narejeno lovljenje izjem, URI routing, poln response objekt in bolj fleksibilen mehanizem za pošiljanje.

Zakaj sploh uporabljati framework? Enostaven odgovor je zato, da si močno poenostavimo delo. V primeru da bi se odločili, da bomo vse sprogramirali sami, bi imeli že z navadnimi operacijami, kot je recimo branje URL naslova, veliko odvečnega dela. [15]

4.2 Orodja, uporabljena pri Android aplikaciji

Pri programiranju aplikacije za Android sem se prvič srečal s principi programiranja mobilnih aplikacij. Uporabil sem nekaj različnih orodij in tehnologij.

4.2.1 Eclipse

Za programiranje mobilne aplikacije sem uporabil urejevalnik Eclipse. Trenutna verzija je 4.4.0 in nosi ime Luna. Primarno je Eclipse namenjen programiranju v Javi vendar ima veliko možnosti nadgradenj, s čimer ga lahko pripravimo za programiranje v skoraj vseh jezikih. Omogoča nam ustvarjanje novih projektov, uvoz starih, prevajanje kode in še veliko več, vendar so to že najpomembnejše funkcije, ki jih pričakujemo od urejevalnika.

Za zagotavljanje vseh funkcionalnosti se Eclipse zanaša na nadgradnje v obliki pluginov. Na voljo nam jih je ogromno, med katerimi je tudi plugin za razvoj Android aplikacij, ki sem ga moral namestiti, preden sem lahko začel z razvojem. Imamo tudi možnost, da kar sami napišemo plugin, če tak, ki bi rešil naš problem še ne obstaja.

Torej, če na hitro naštejemo nekaj lastnosti Eclipse-a [16]:

- urejevalnik, ki podpira mnogo programskih jezikov,
- hitro programiranje s pomočjo samodejnega dokončevanja kode (code completion) ter
- veliko možnosti za nadgradnje in modifikacijo po željah uporabnika.

4.2.2 REST

REST (Representational state transfer) je abstraktcija arhitekture svetovnega spleta. Bolj natančno povedano, REST je arhitekturni stil, sestavljen iz medsebojno povezanih pravil, ki veljajo za komponente, povezovalne elemente in podatkovne elemente znotraj distribuiranega hipermedijskega sistema. REST ignorira same podrobnosti implementacije komponent in sintakste protokola z namenom, da se osredotoči na vloge komponent, na pravila njihovih interakcij z ostalimi komponentami in na njihove interpretacije pomembnih podatkovnih elementov [17].

Da zgoraj napisanemu dodam še praktično razlago, bom opisal, kako sem sam uporabil REST pri svoji nalogi. Naloga je bila v Android aplikacijo spraviti seznam vseh uporabnikov, pri čemer mi je prav prišel REST. V GAE backendu sem napisal funkcijo za sestavljanje tega seznama, potem pa sem s telefonom klical to funkcijo tako, da sem obiskal URL naslov »/rest/getUsers«. Nato mi je prej napisana funkcija izpisala seznam uporabnikov, ki sem ga vzel, zapakiral v tabelo JSON objektov in potem uporabil za nadaljne delo. Primer seznama uporabnikov:

```
{
```

```
1. result: 1,
2. clients:

    [

     o  "Angelca",
     o  "Črna Žaba",
     o  "Davor",
     o  "Narator",
     o  "Jerca"

     ],

3. timestamp: 1409258006

}
```

4.2.3 JSON

JSON (JavasScript object notation) je format odprtega standarda, ki za prenos podatkovnih objektov uporablja človeku enostavno berljiv tekst. Podatkovni objekti so sestavljeni iz parov atribtut – vrednost. Največ se uporablja za prenos podatkov med strežnikom in spletno aplikacijo kot alternativa XML-ju (eXtensible Markup Language). [18]

Pri svoji nalogi sem JSON uporabljal za prenos in urejanje podatkov. Zgoraj omenjeni seznam uporabnikov sem najprej shranil v tabelo JSON objektov in nato v aplikaciji prikazal seznam uporabnikov. Kasneje sem JSON uporabljal še za pošiljanje meritev na strežnik.

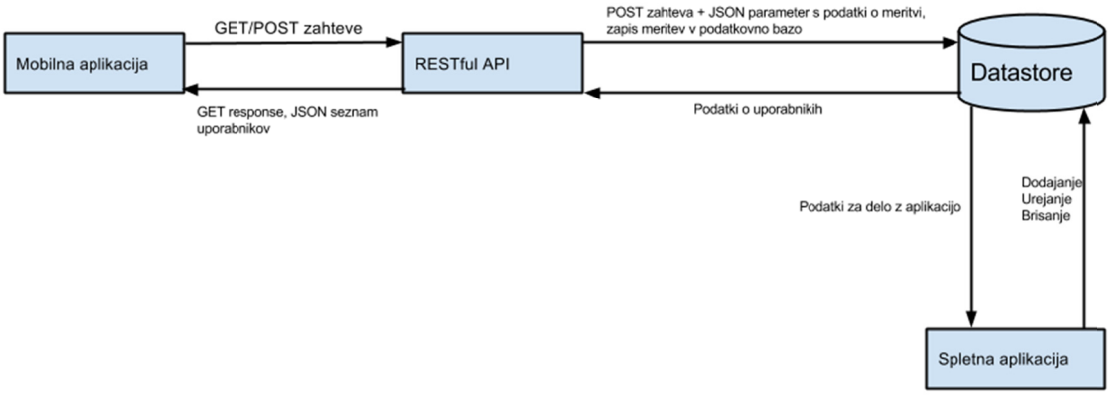
Poglavje 5 Izvedba aplikacije

Preden sem se lotil izdelave aplikacije, sem si zastavil problem, ki bi ga rad rešil. Namen je bil rešiti problematiko vnosa meritev s terena s pomočjo mobilne aplikacije, ki skrbi za vnos in prenos podatkov o meritvah ter s pomočjo strežnika, kjer se te meritve zapisujejo v bazo in so nam kadarkoli na voljo za pregled ali urejevanje. Rešitev je bila torej kombinacija spletne in mobilne aplikacije.

Takšna aplikacija bi močno olajšala delo popisovalcem, zaposlenim v podjetju, ki je naročilo aplikacijo, prav tako pa bi prišla prav končnim odjemalcem, ki že sedaj porabo sproti popisujejo, s to razliko da ne bi rabili prek telefona klicati na podjetje, čakati na prostega operateja in potem sporočiti meritev. Trenutna verzija aplikacije je sicer namenjena popisovalcem iz podjetja, vendar pa bi vsi principi in pravila veljali tudi pri aplikaciji za končne odjemalce.

5.1 Arhitektura aplikacije

Aplikacija ima torej dva glavna dela, to je spletna aplikacija in pa mobilna aplikacija. Vsaka posebej nam ne koristi ravno veliko, zato je tukaj še RESTful API, ki skrbi za povezavo med spletno in mobilno aplikacijo. Mobilna aplikacija prek protokola HTTP izvede GET zahtevo na RESTful spletno storitev, od tukaj dobi potrebne podatke, da lahko zgradi seznam uporabnikov. Ko vpišemo meritve in pritisnemo na gumb »Send«, mobilna aplikacija ponovno prek protokola HTTP izvede zahtevno, s to razliko da je tokrat ta zahteva tipa POST. RESTful storitev na strežniku za tem prebere parametre, ki jih je mobila aplikacija poslala v zahtevi, pobere ven vse podatke in meritev zapiše v podatkovno bazo.



Slika 2 – shema arhitekture aplikacije

5.2 Delo s podatki

Celotna aplikacija je za delovanje potrebovala več različnih podatkov. Tukaj bom opisal sestavne dele, ki skupaj tvorijo celoto za delo s podatki.

5.2.1 Podatkovna baza

[19] GAE za shranjevanje podatkov uporablja svojo bazo, ki nosi ime Datastore. Datastore ni klasična relacijska podatkovna baza kot je to v navadi pri MySQL-u in podobnih tehnologijah, temveč je zgrajena na osnovi BigTable, kar pomeni, da bomo morali o snovanju baze razmisliti nekoliko drugače. BigTable je namreč distribuiran sistem za shranjevanje podatkov, kar pomeni, da se podatki ne nahajajo samo na enem strežniku, temveč so distribuirani med več njih. Dobra stran je, da v so primeru izpada enega strežnika podatki še vedno na voljo.

Podatki se v bazi shranjujejo kot entitete, ki jih moramo pred shranjevanjem definirati z modeli. Za iskanje po bazi se uporablja GQL, ki ima podobno sintakso kot SQL, vendar z nekaterimi omejitvami, kot je recimo samo en neenakostni filter v eni poizvedbi ali odsotnost ukaza JOIN za branje podatkov iz več tabel hkrati.

Za izvedbo aplikacije sem najprej razmislil, katere modele bom potreboval za dosego cilja. Na koncu sem prišel do treh modelov:

1. **Client –** Entitete tipa Client v aplikaciji predstavljajo uporabnike storitve, za katero se vnašajo meritve. Praktičen primer je recimo uporabnik storitev elektro podjetja, za katerega se vnašajo podatki o porabi električne energije.

   Vsak Client ima dva parametra, ki ga določata. Prvi je **userid**, ki predstavlja unikaten ID uporabnika, po katerem lahko dobimo točno določenega uporabnika. Userid je sestavljen iz črk in številk, to pa zato, da dobimo večji razpon možnih ID-jev kot pa če bi bili omejeni recimo samo na 5 številk.

   Drugi parameter pa je **measurementTypes**, ki predstavlja seznam vseh tipov meritev, ki jih uporabnik ima. Posledično lahko za določenega uporabnika vpisujemo samo meritve za tiste tipe meritev, ki so jih uporabnik trenutno ima.

2. **MeasurementType –** Entitete tipa MeasurementType predstavljajo tipe meritev, ki so na voljo v celotni aplikaciji. Recimo da ima prej omenjeno elektro podjetje tudi podružnico, ki se ukvarja z dobavo plina, nimajo pa vsi uporabniki elektro storitev hkrati tudi njihovi storitev. V tem primeru morajo imeti nekaj tipov za elektro del podjetja, recimo višjo in nižjo tarifo, potrebujejo pa tudi kakšen tip za plinarsko podružnico, recimo mesečna poraba. Na koncu ima podjetje tri tipe meritev, vendar se vsi trije tipi ne uporabljajo na vseh uporabnikih.

   MeasurementType ima tri parametre, ki določajo unikaten tip meritve. Prvi parameter je **name**, ki predstavlja unikatno ime tipa meritve. Name se uporabniku nikoli ne prikaže, uporablja se samo za lažje delo z iskanjem in urejanjem tipov meritev in delo v celotni aplikaciji. Name je lahko sestavljen iz črk in številk, je URL-friendly prezentacija tipa meritve, zato ne sme imeti presledkov. Primer je »visja_tarifa«.

   Drugi parameter je **label**. Label se uporablja za uporabniku prijazen prikaz imena tipa meritve. Če je name tipa meritve »visja_tarifa«, potem je label recimo »Višja tarifa«. Label se uporablja povsod, kjer poteka interakcija z uporabnikom, saj ne želimo da se zaradi nenavadnega imena tipa meritve uporabnik zmoti in vnese napačno meritev.

   Tretji parameter je **unit**. Unit predstavlja enoto tipa meritve, primer je recimo »kW/h« pri porabi električne energije. Unit se podobno kot label uporablja pri delih aplikacije, kjer se dogaja interakcija z uporabnikom. Uporablja se zato, da uporabnik ve, v kakšnih enotah mora vpisati meritve. Ne želimo recimo, da bi zaradi napake pri vnosu kakšnemu uporabniku zaračunali preveč stroškov, zato mora uporabnik pri vnosu pogledati enoto, v kateri se meritev pošilja in po potrebi pretvoriti podatke, da bodo ustrezni.

3. **Measurement** – Entitete tipa Measurement predstavljajo posamezne meritve, ki jih uporabniki pošiljajo na strežnik. Teh entitet je zaradi namena aplikacije največ.

   Measurement ima štiri parametre, s katerimi določimo vse potrebne informacije o meritvi. Prvi parameter je **client**. Gre za referenčni parameter, pove pa nam, kateremu Clientu pripada meritev, saj ne moremo imeti meritev, ki ne pripadajo nobeni stranki in kar ležijo naokrog ter zasedajo prostor.

   Drugi parameter je **value**. Value je, logično, vrednost meritve. Podprte so vrednosti z decimalno piko in negativne vrednosti, saj moramo biti pripravljeni na vse tipe meritev in se ne smemo omejiti recimo le na pozitivna cela števila, saj smo potem lahko v težavah če se stranka odloči da bi rada spremljala recimo tudi temperaturo hladilnih prostorov kjer so temperature večinoma pod ničlo.

   Tretji parameter je **created**. Ta parameter se zapiše samodejno ob vnosu meritve, gre pa za datum poslane meritve. To je edini parameter, ki se ga ne more spreminjati v primeru napačnega vnosa, to pa zato, ker bi v nasprotnem primeru lahko prihajalo do zlorab in vpisovanja starejših datumov, ki so že bili zaračunani in posledično izogiba plačilu.

   Zadnji parameter je **mType**. Podobno kot prvi parameter client je tudi mType referenčni parameter, ki nam pove, kateremu tipu meritve pripada vnešena meritev. Brez tega nam meritev ne bi kaj dosti koristila, saj ne bi imeli podatka za kateri tip meritve gre, niti kakšna je enota.

5.2.2 RESTful service

Ko so bili modeli za predstavitev vseh potrebnih podatkovnih tipov sestavljeni, sem moral ugotoviti, kako bi najlažje poskrbel za komunikacijo med strežnikom in mobilno aplikacijo. Zaradi enostavne izdelave in komunikacije sem se odločil za uporabao RESTful service-a, saj nam omogoča komunikacijo prek protokola HTTP ter prikaz podatkov v obliki JSON-a, od koder pa ni več težko vzeti stvari, ki so pomembne za naše delo.

Za potrebe aplikacije sem potreboval dva service-a, eden je služil za prikaz seznama uporabnikov s pomočjo get requesta, drugi pa je služil za pošiljanje meritev nazaj na strežnik in njihov posledičen zapis v bazo.

- **Prikaz seznama uporabnikov** – za potrebe mobilne aplikacije sem moral napisati service za prikaz seznama uporabnikov. Service teče na strani strežnika in je napisan v sklopu GAE aplikacije, deluje pa tako, da s pomočjo GQL poizvedbe pripravi seznam

vseh entitet tipa Client v aplikaciji, potem vse skupaj zapakira v JSON objekt in na koncu na točno določenem URL naslovu prikaže vsebino tega JSON objekta. Mobilna aplikacija nato izvrši get request na ta URL naslov, v response-u od strežnika pa dobi prikazan JSON objekt. Iz tega JSON objekta potem za prikaz zgradim seznam uporabnikov aplikacije.

- **Pošiljanje vnešenih meritev na strežnik** – Ker je ena glavnih nalog mobilne aplikacije pošiljanje meritev nazaj na strežnik, sem moral napisati tudi service za zapisovanje vrednosti meritev iz mobilne aplikacije na strežnik. Tega sem se lotil tako, da sem najprej v mobilni aplikaciji v en JSON objekt zbral vse vnešene meritve in uporabnika, za katerega te meritve veljajo, nato pa prek post requesta dostopal do točno določenega URL naslova, kot dodaten parameter v URL naslov pa sem dodal še JSON objekt. Na zadnje sem na strežniku prebral ta argument, iz njega pobral vse vrednosti, ki me zanimajo in jih zapisal v bazo.

5.2.3 Urejanje podatkov

Ker nismo nezmotljivi in obstaja verjetnost, da pri vnosu uporabnikov, tipov meritev ali meritev lahko pride do neželjene napake, sem v aplikacijo dodal možnost urejanja in brisanja podatkov.

- **Urejanje podatkov** – Vnešene podatke lahko v primeru, da se na primer, spremenijo podatki o uporabniku, tipi meritev, ki jih ima uporabnik ali vrednosti kakšne meritve tudi urejamo.

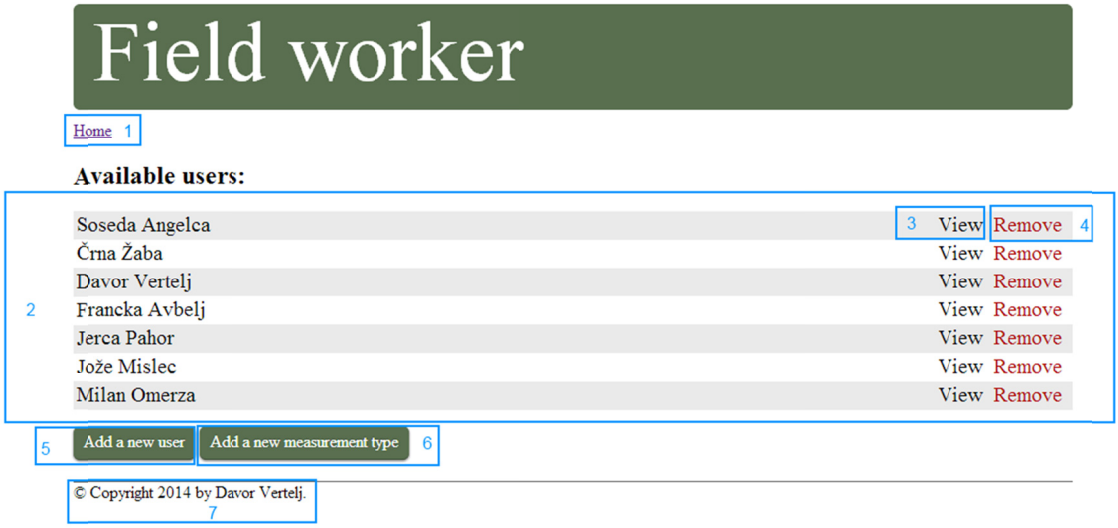
5.3 Uporaba spletne aplikacije

Ozadje delovanja aplikacije sem že predstavil, v nadaljevanju pa bom podrobneje opisal tudi samo delovanje aplikacije za uporabnika.

Aplikacija zahvaljujoč uporabi Jinja2 templating languagea deluje po principu dedovanja predlog, kar pomeni, da bodo imele vse strani aplikacije določene elemente, kot je recimo header, postavitev vsebine in copyright obvestilo na dnu strani postavljene enako.

5.3.1 Začetna stran

Na začetni strani se znajdemo, ko obiščemo URL na katerem je aplikacija dostopna.



Slika 3 – začetna stran spletne aplikacije

1. **Home** – gumb, oziroma povezava, s katero se lahko vedno vrnemo na začetno stran. Ker je povezava vidna na vseh straneh, je ne bom več opisoval pri drugih slikah.

2. **Seznam uporabnikov** – to je seznam, na katerem so izpisani vsi trenutno aktivni uporabniki v aplikaciji. Vsak na novo dodan uporabnik se seveda avtomatsko izpiše na tem seznamu. Zaradi bolj preglednega seznama so sode in lihe vrstice pobarvane z drugačno barvo, ob postavitvi miškinega kurzorja na uporabnika pa se cela vrstica obarva zeleno. Seznami po vsej aplikaciji uporabljajo enak stil, saj sem želel doseči enoten izgled skozi celotno aplikacijo.

3. **View** – vsak uporabnik ima na voljo podrobnejši pregled, kjer lahko ročno vnašamo meritve, urejamo podatke o uporabniku in vidimo sledeče podatke o uporabniku:

    - tipi meritev na uporabniku,
    - predstavitev zgodovine meritev z grafi,
    - predstavitev zgodovine meritev v tabeli.

4. **Remove** – ker lahko pride do prekinitve poslovnega razmerja in se uporabnik odloči, da bo sodeloval z drugim ponudnikom, je na voljo tudi možnost izbrisa uporabnika iz aplikacije. V primeru, da se odločimo za izbris uporabnika se seveda izbrišejo tudi vse meritve, ki so bile zanj opravljene, saj nočemo, da nam v bazi zasedajo prostor.

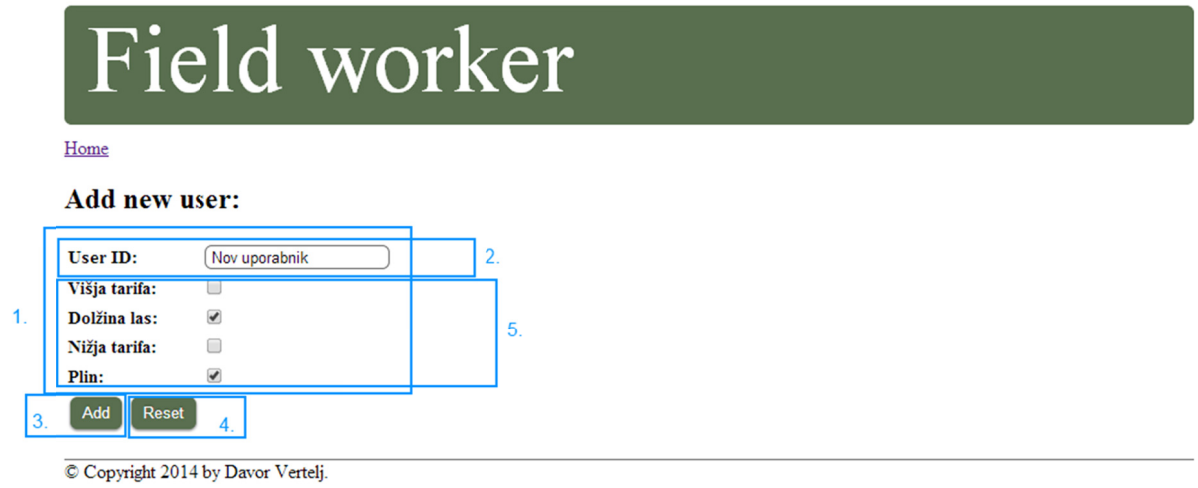
5. **Gumb »Add a new user«** – ko pridobimo novega uporabnika, ga moramo dodati v aplikacijo. To storimo s klikom na gumb »Add a new user«, vpisom potrebnih podatkov in potrditvijo. Po uspešnem dodajanju uporabnika v aplikacijo se ta uporabnik prikaže na seznamu vseh uporabnikov in z njim lahko operiramo tako kot z ostalimi.

6. **Gumb »Add a new measurement type«** - kaj se zgodi, ko podjetje, ki se ukvarja z električno energijo kupi podjetje, ki skrbi za dobavo plina v gospodinjstva? V takem primeru je treba vnesti nov tip meritve, ki bo primeren za meritve porabe plina oziroma česarkoli.

7. **Copyright obvestilo** – ker je aplikacija intelektualna lastnina, ima tudi copyright obvestilo. Obvestilo je na vseh straneh enako, zato se v opisih ne bo več pojavljalo.

5.3.2 Dodajanje novega uporabnika

Podjetje je dobilo novega naročnika in sedaj bi rado spremljalo in vnašalo meritve za tega naročnika. Preden to lahko stori, mora biti novi uporabnik dodan v aplikacijo in nastaviti se mu morajo tipi meritev, ki jih ima.



Slika 4 – stran za dodajanje novega uporabnika

1. **Seznam parametrov** – prva točka predstavlja seznam vseh parametrov, ki jih lahko določimo za novega uporabnika. V trenutni verziji ima uporabnik lahko dva parametra, prvi je »User ID«, ki predstavlja uporabnikov ID ter tipi meritev, ki predstavljajo tipe meritev, ki so trenutno asociirani s tem uporabnikom.

2. **User ID** – v polje »User ID« se vpiše uporabnikov ID, lahko kar ime in priimek ali pa kakršna koli kombinacija črk in številk.

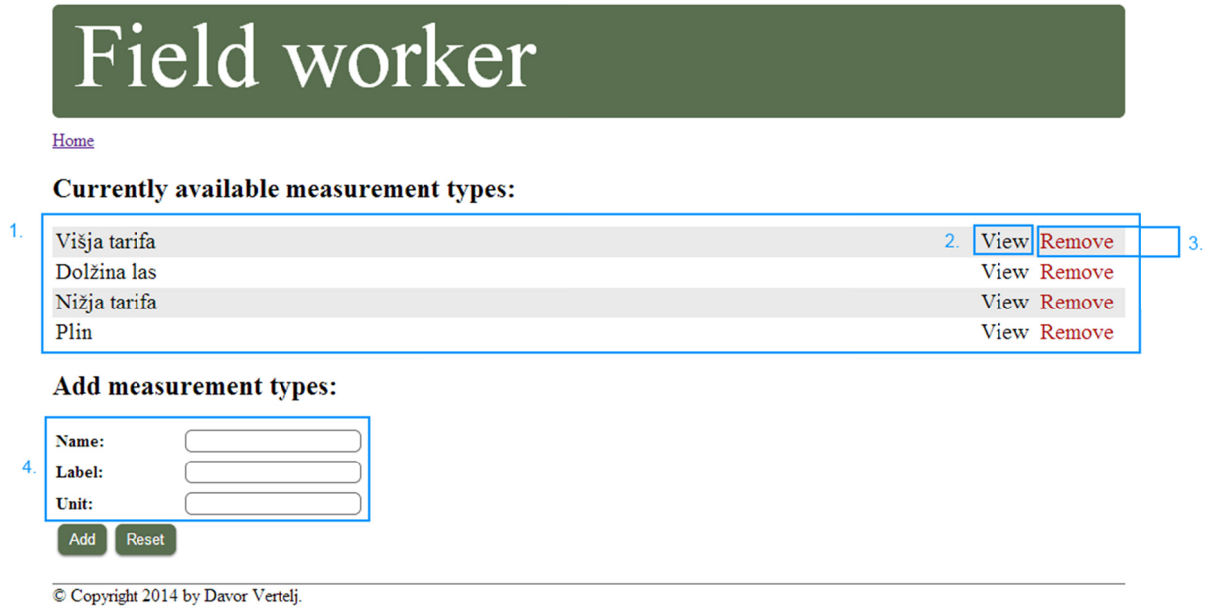
3. **Gumb »Add«** - s klikom na gumb »Add« potrdimo da so vsi podatki pravilni in dodamo uporabnika v aplikacijo. Gumb »Add« se pojavi na več mestih v aplikaciji in povsod ima enako funkcijo, zato ga pri nadaljnih slikah ne bom posebej opisoval.

4. **Gumb »Reset«** - s klikom na gumb »Reset« zbrišemo vse do sedaj vnešene nastavitve za novega uporabnika. Uporaben je, če smo se zmotili pri veliko parametrih in bi radi takoj začeli s ponovnim vpisovanjem. Tako kot gumb »Add« se tudi gumb »Reset« pojavi na več mestih v aplikaciji, zato bom ponovno opisovanje v nadaljevanju izpustil.

5. **Seznam tipov meritev** – prikazan je seznam vseh možnih tipov meritev v aplikaciji, s kljukico označimo tiste tipe, ki bi jih naj uporabnik imel.

5.3.3 Dodajanje novega tipa meritve

Podobno kot z uporabniki, lahko tudi pri tipih meritve dodajamo nove tipe v aplikacijo. To moramo storiti, če recimo želi naročnik poleg obstoječih tipov od sedaj naprej slediti tudi kakšnemu novemu tipu meritve



Slika 5 – stran za pregled in dodajanje novega tipa meritve

1. **Seznam obstoječih tipov meritev** – to je seznam, v katerem so izpisani vsi trenutno obstoječi tipi meritev v aplikaciji. Ko se doda nov tip meritve, se avtomatsko doda na ta seznam.

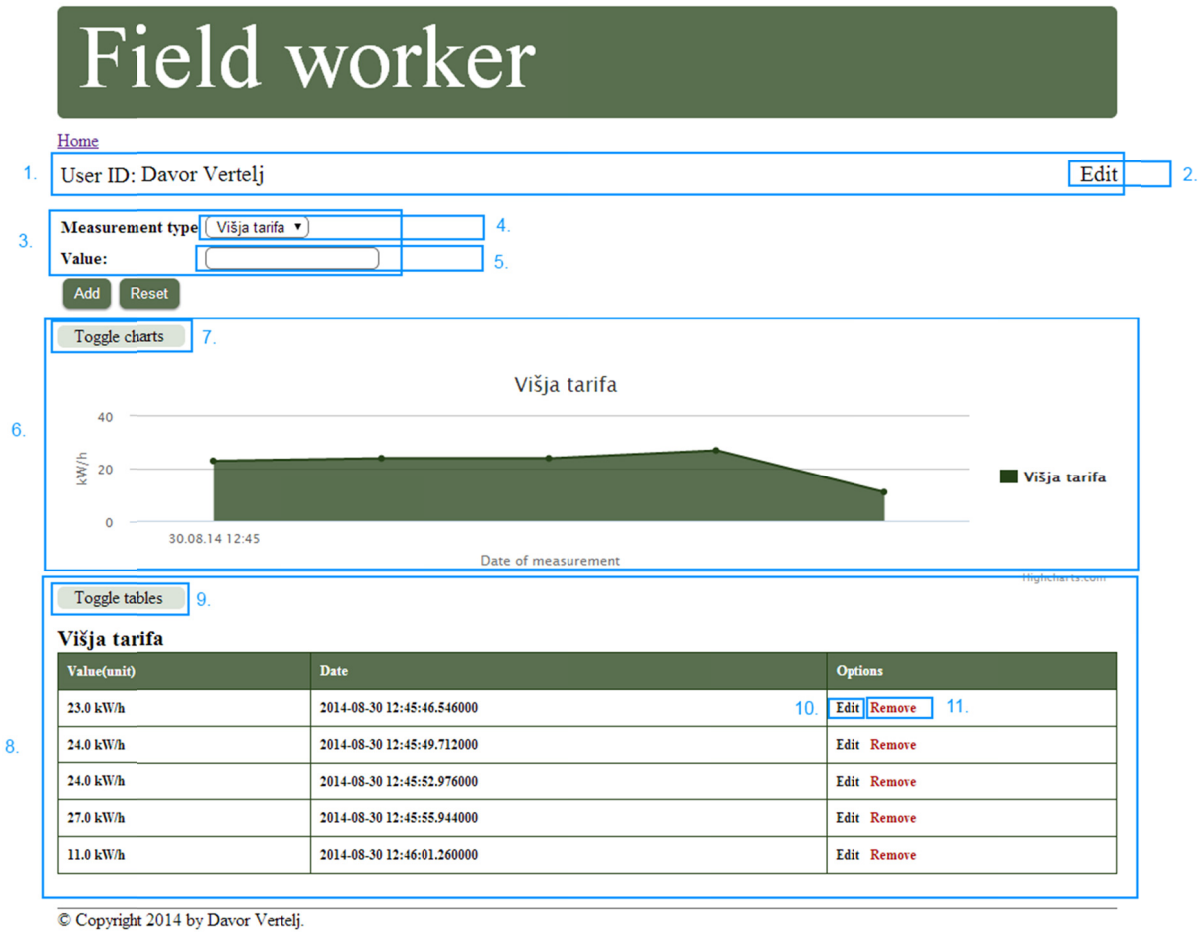
2. **View** – s klikom na »View« pridemo na stran z nekaj več podrobnostmi o izbranem tipu meritve. Če pride do potrebe po spremembi podatkov lahko na tej strani tip meritve tudi urejamo in spreminjamo.

3. **Remove** – s klikom na »Remove« odstranimo tip meritve iz aplikacije. Zaradi konsistence seveda odstranimo tudi vse meritve, ki so imele ta tip meritve nastavljen za svoj tip meritve, ker bi v nasprotnem primeru lahko prihajalo do napak in nepotrebnega zasedanja prostora.

4. **Obrazec za dodajanje tipa meritve** – tukaj lahko dodajamo nove tipe meritev, vpisati moramo vse tri parametre in klikniti na gumb »Add«. Parameter »Name« je unikatno, URL-friendly ime, ki se uporabniku nikjer ne prikaže. Parameter »Label« je ime meritve, ki se prikazuje uporabnikom, zato naj bo zapisano v bralno prijazni obliki. Parameter »Unit« pa predstavlja enoto tipa meritve, pove nam, koliko nečesa potem predstavljajo meritve, ki imajo nastavljen ta tip meritve.

5.3.4 Podroben pregled uporabnika

Največkrat bomo želeli pogledati kakšno je stanje za določenega uporabnika, oziroma mu bomo želeli ročno vpisati meritve. Zanimajo nas meritve zadnjega meseca, katerih tipov so te meritve bile, v nekaterih primerih pa bi želeli tudi spremeniti podatke o uporabniku.



Slika 6 – podrobnejši pregled podatkov o uporabniku

1. **Podatki o trenutno odprtem uporabniku** – tukaj lahko vidimo, podrobnosti katerega uporabnika si trenutno ogledujemo.

2. **Edit** – s klikom na »Edit« pridemo na stran za urejanje podatkov o uporabniku. Na tej strani mu lahko spreminjamo obstoječe parametre. Trenutno imajo vsi uporabniki samo dva parametra, ID in tipe meritev, tako da posledično na strani za urejanje lahko spremenimo ID ali pa uporabniku dodamo/odstranimo tipe meritev.

3. **Dodajanje meritev** – tukaj lahko odprtemu uporabniku dodajamo meritve. Meritve bi na ta način lahko dodajali v primeru, ko bi imeli težave s prenosom preko mobilne aplikacije in bi si odčitene meritve zapisali v beležko za kasnejši vnos.

4. **Measurement type** – v dropdown meniju izberemo tip meritve, ki jo vpisujemo. V meniju so na voljo le tipi meritev, ki jih uporabnik trenutno ima.

5. **Value** – v vnosno polje vpišemo vrednost, ki smo jo odčitali. Za ostalo nam ni treba skrbeti, kliknemo le še na gumb »Add« in meritev je dodana ter prikazana v grafu in tabeli.

6. **Grafični prikaz meritev** – v grafu se prikazujejo vse meritve v zadnjem mesecu. Posamezne meritve so prikazane s točko na grafu na katero lahko pokažemo z miško in prikaže se nam okenček z dodatnimi informacijami. Če ima uporabnik več različnih tipov meritev, se prikaže več različnih grafov za vsak tip meritve posebej. V primeru, da uporabnik ima nek tip meritve, ni pa za ta tip pri tem uporabniku še nobene meritve se ne prikaže nič.

7. **Toggle charts** – gumb, s katerim lahko zaradi boljše preglednosti skrijemo grafični prikaz. Ko so grafi skriti, se gumb obarva temno zeleno in ob ponovnem kliku nanj se grafi spet prikažejo.

8. **Tabelarični prikaz meritev** – podobno kot v grafu, se tudi v tabeli prikazujejo vse meritve zadnjega meseca. V tabeli lahko vidimo vrednost in enoto meritve, kdaj je bila meritev zabeležena in pa še možnost urejanja ali izbrisa meritve. Tako kot pri grafih se za vsak tip meritve izpiše ena tabela.

9. **Toggle table** – gumb opravlja podobno funkcijo kot gumb »Toggle charts«, to je skrivanje tabel z meritvami zaradi boljše preglednosti. Kadar so tabele skrite se gumb obarva temno zeleno, ob ponovnem kliku nanj se tabele spet prikažejo.

10. **Edit** – s klikom na »Edit« se nam odpre pojavno okno, s pomočjo katerega lahko uredimo vrednost meritve. Zaradi možnosti goljufanja pri datumih zapisov meritev je urejanje datuma zapisa meritve onemogočeno.

11. **Remove** – s klikom na »Remove« odstranimo izbrano meritev. Posledično se meritev odstrani tudi iz grafičnega prikaza, če pa je bila to edina meritev določenega tipa, potem se, logično, odstanita tako graf kot tabela za prikaz meritev tega tipa.

5.3.5 Urejanje podatkov o uporabniku

Pri urejanju podatkov o uporabniku se znajdemo na podobni strani kot pri dodajanju novega uporabnika, s to razliko, da so pri urejanju uporabnika parametri, ki jih ima uporabnik, že

izpolnjeni.

Slika 7 – urejanje podatkov o uporabniku

Če želimo uporabniku spremeniti ID zgolj spremenimo obstoječega, če pa mu želimo dodati ali odvzeti kak tip meritve, pa dodamo oziroma odstranimo kljukico pri tipu meritve, ki bi ga radi dodali ali odstranili.

5.3.6 Urejanje tipa meritve

Tako kot pri urejanju podatkov o uporabniku se tudi pri urejanju tipa meritve znajdemo na podobni strani, kot za dodajanje tipa meritve, s tem da so tukaj vnosna polja že izpolnjena z obstoječimi podatki o tipu meritve.

Slika 8 – urejanje podatkov o tipu meritve

Za spremembo podatkov o tipu meritve spremenimo trenutno vpisane vrednosti v vnosnih poljih in pritisnemo gumb »Edit«. Tip meritve se avtomatsko popravi na vseh mestih v aplikaciji.

5.4 Uporaba mobilne aplikacije

Za rešitev problema je bila ključna tudi izdelava mobilne aplikacije za operacijski sistem Android, saj bi brez tega vse meritve še vedno vpisovali po starem, najprej bi si jih zapisali na list papirja, potem bi potrebovali dostop do računalnika, od koder bi končno lahko vpisali meritve, ki smo jih za tisti dan odčitali, v elektronsko obliko.

S pomočjo mobilne aplikacije lahko odčitane meritve vpišemo kar na licu mesta, brez skrbi kam si jih bomo zapisali in kasneje vpisovali v računalnik ter brez nevarnosti, da bi jih mogoče celo pozabili vpisati.

Vnos meritev prek mobilne aplikacije je zelo enostaven, najprej iz seznama uporabnikov izberemo uporabnika, za katerega želimo vpisati meritve. Ob kliku na uporabnika se nam odpre nova stran, na kateri so vsi tipi meritev, ki jih uporabnik ima. Zdaj nam preostane samo še da vpišemo vrednosti in pritisnemo na gumb »Send« ter s tem pošljemo meritve na strežnik, ki jih zapiše v podatkovno bazo.

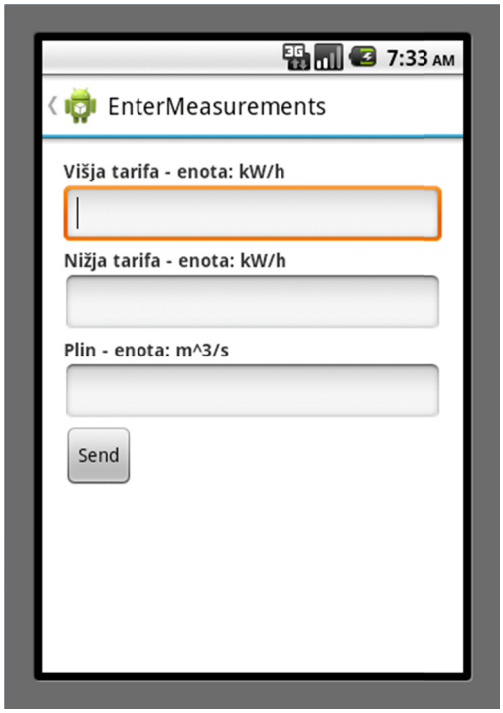
5.4.1 Seznam uporabnikov in izbira uporabnika

Mobilna aplikacija nas takoj pripelje na seznam uporabnikov, na katerem so prikazani vsi aktivni uporabniki v aplikaciji. Za izbiro uporabnika, ki bi mu radi vpisali meritve, moramo zgolj klikniti nanj in aplikacija nas bo odpeljala na stran za vpis meritev.

Slika 9 – seznam uporabnikov

5.4.2 Vnos in pošiljanje meritev

Na strani za vnos meritev so našteti vsi tipi meritev, ki trenutno obstajajo na uporabniku. Odčitane meritve vpišemo in jih pošljemo s pritiskom na gumb »Send«.



Slika 10 – stran za vnos meritev

Trenutno izbran uporabnik ima (Slika 9), tri tipe meritev: Višja tarifa, Nižja tarifa in Plin. V primeru da bi radi vpisali zgolj eno meritev, vpišemo samo tisto vrednosti, ki smo jo odčitali, aplikacija pa sama poskrbi, da se vpiše izključno ta vrednost in ne prihaja do napak. Po pritisku gumba »Send« se vse vrednosti izbrišejo z ekrana, tako da lahko ponovno vpišemo novo meritev, če želimo to storiti.

5.5 Dostop do aplikacije

Ker je aplikacija v osnovi namenjena podjetjem, ki opravljajo svoje meritve za svoje uporabnike, seveda ne želimo, da bi bil dostop do spletne ali mobilne aplikacije omogočen vsakomur.

Za rešitev tega problema sem poskrbel na dva načina:

1. Dostop do spletne aplikacije je omogočen le točno določenim poštnim naslovom, katere se določi na administrativni strani aplikacije. Ko uporabnik pride na URL naslov aplikacije, se mu odpre pojavno okno za vpis uporabniškega imena (email) in gesla. V primeru da uporabnik nima pravic za dostop do aplikacije, se bo njegova pot tukaj končala, če pa je uporabnik dodan med dovoljene uporabnike za aplikacijo, se bo po vpisu uporabniškega imena in gesla znašel na prvi strani aplikacije. S pomočjo

administrativne strani aplikacije in nekaj malih sprememb v kodi lahko zaklenemo tudi samo posamezne dele aplikacije, to bi nam prišlo prav recimo v primeru, ko bi želeli končnim odjemalcem omogočiti dostop do podatkov o njihovi porabi, seveda pa ne bi želeli, da si spreminjajo ali vpisujejo meritve.

2. Distribucija mobilne aplikacije poteka samo med naročnikom aplikacije in izvajalcem rešitve, zato se za distribucijo ne uporablja servisa GooglePlay. Ker je nezaželeno, da bi nek naključni, nepooblaščeni uporabnik vpisoval meritve za končnega odjemalca, imajo aplikacijo nameščeno le operaterji, ki bodo hodili po terenu in vpisovali meritve, ali pa stranke podjetja, zaradi česar je možnost za napačne ali prirejene vnose meritev veliko manjša.

Poglavje 6 Sklepne ugotovitve in zaključek

Nekatera opravila se danes kljub ogromni naprednosti mobilnih in spletnih tehnologij in možnostiuporabe ali razvoja naprednejših rešitev še vedno opravljajo ročno, kar ni le potrata časa in denarja, temveč predstavlja tudi večje tveganje za napake, ki bi ga lahko s pomočjo prej omenjenih tehnologij zmanjšali.

Z namenom olajšati eno izmed takšnih opravil, to je popis meritev na terenu, sem razvil aplikacijo, predstavljeno v tej diplomski nalogi. Aplikacija je enostavna za uporabo, spletni del aplikacije je namenjen bolj pisarniškim administrativnim nalogam kot je dodajanje uporabnikov in tipov meritev, mobilni del pa ima zgolj eno funkcijo: vnos meritev, kar ponavadi opravljajo popisovalci na terenu.

Moja rešitev se mi zdi dobra predvsem zato, ker poenostavi naloge delavca na terenu. Vrednosti mu ni več potrebno zapisati najprej v blokec, se odpeljati nazaj na delovno mesto in tam prebrati vrednosti ter jih vpisati v arhive. S pomočjo predstavljene aplikacije terenski delavec pride do števca, prebere vrednost na števcu, vnese vrednost v mobilno aplikacijo, s čimer je naloga opravljena, delavec pa lahko brez skrbi začne z opravljanjem druge naloge. Aplikacija reši tudi vprašanje hranjenja podatkov, saj se vsi podatki shranjujejo v bazo in so vedno dostopni.

Vsaka aplikacija ima vedno prostor za izboljšave, zato tudi ta ni izjema. Naštel in opisal bom nekaj izboljšav, za katere sem prepričan, da bi bile koristne za večjo razširjenost in uporabo moje aplikacije.

Nadgradnja mobilne aplikacije na funkcionalnost spletne - trenutno je mobilna aplikacija namenjena samo pošiljanju meritev. Lahko bi implementirali dodatno funkcionalnost za dodajanje/urejanje/brisanje različnih podatkovnih struktur. S tem bi se mobilna aplikacija po funkcionalnosti močno približala spletni in mogoče sčasoma tudi izničila potrebo po njej in bi GAE deloval samo za namene strežnika na katerem se shranjujejo meritve, ne bi pa prikazoval ničesar.

Ločena aplikacija za končne uporabnike - trenutna verzija mobilne aplikacije je namenjena recimo operaterjem v elektro podjetju, se pravi terenskim delavcem in ne končnim odjemalcem storitve. Operater se zapelje do števca, odmeri porabo in meritve zapiše s pomočjo aplikacije. Ideja je, da bi izdelal še eno ločeno aplikacijo, ki bi bila namenjena končnim odjemalcem in bi omogočala sporočanje stanja števcev. Aplikacija bi bila javno dostopna na GooglePlay, avtentikacija pa bi bila možna s prijavnim oknom, kamor bi vpisali svojo kodo, ki jo imamo pri določenem podjetju.

Izdaja računov na podlagi meritev - dodala bi se možnost, da operater na podlagi mesečnih meritev uporabniku izda račun. Za implementacijo te storitve bi seveda potrebovali tudi podatek o ceni enote.

Več podatkov o uporabnikih - trenutna verzija aplikacije ima za uporabnike samo dva parametra, ID in tipe meritev na uporabniku. Če bi želeli implementirati zgoraj omenjeno izdajanje računov, bi seveda potrebovali več podatkov o uporabnikih, razmišljal sem o dodajanju naslednjih parametrov:

- ime
- priimek
- naslov
- telefon
- email
- davčna številka

Arhiv - dobra ideja bi bila tudi možnost vpogleda v arhiv meritev. Ta možnost zdaj že obstaja, vendar samo s pomočjo administrativne strani aplikacije, ki je naročniki ne bi videli. V arhivu bi hranili vse meritve v zadnjih 365 dneh, funckionalnost pa bi bila ista kot pri sedajšnjem pregledu meritev za uporabnika, meritev bi lahko uredili ali pa jo izbrisali.

Aktivacija in deaktivacija uporabnikov - ta možnost bi prišla prav v primeru, ko bi se nek končni uporabnik odločil, da ne bo več koristil storitev ponudnika, ki ima aplikacijo za spremljanje meritev. Zdaj je aplikacija narejena tako, da moramo v takem primeru uporabnika zbrisati iz aplikacije, s tem pa zbrišemo tudi vse njegove meritve. Boljša opcija bi bila, da se v takem primeru uporabnika deaktivira, s tem se uporabnika ne bi videlo na seznamu aktivnih uporabnikov, ohranijo pa se vse njegove meritve, če se mogoče vrne.

Literatura

[1] Python Runtime Environment, dostopno na: https://developers.google.com/appengine/docs/python/?csw=1

[2] Google App Engine: Platform as a Service, dostopno na: https://developers.google.com/appengine/

[3] App Engine pricing, dostopno na: https://developers.google.com/appengine/pricing

[4] Razprava »Pros & Cons of Google App Engine«, dostopno na: http://stackoverflow.com/questions/1306279/pros-cons-of-google-app-engine

[5] App Engine Servie Level Agreement, dostopno na: http://developers.google.com/appengine/sla?csw=1

[6] Odgovor na »Why can't I write to this file?«, dostopno na: https://developers.google.com/appengine/kb/java?csw=1#writefile

[7] Wikipedia o BigTable, dostopno na: http://en.wikipedia.org/wiki/BigTable

[8] Wikipedia o Android (Operating system), dostopno na: http://en.wikipedia.org/wiki/Android_(operating_system)

[9] Report: Android reached record 85% smartphone market share in Q2 2014, Xiaomi now fifth-largest vendor, dostopno na: http://thenextweb.com/google/2014/07/31/android-reached-record-85-smartphone-market-share-q2-2014-report/

[10] A short history of Android – infographic, dostopno na: http://businessideaslab.com/android-history-infographic/

[11] Pros & Cons of iOS and Android Apps, dostopno na:

http://www.teazel.com/articles/which-platform-is-better-ios-vs-android/

[12] Sublime text domača stran, dostopno na:
http://www.sublimetext.com/

[13] Domača stran programskega jezika Python, dostopno na:
https://www.python.org/

[14] Jinja2 domača stran in dokumentacija, dostopno na:
http://jinja.pocoo.org/docs/dev/

[15] Domača stran Webapp2 frameworka, dostopno na:
https://webapp-improved.appspot.com/

[16] Wikipedia o Eclipse (software), dostopno na:
http://en.wikipedia.org/wiki/Eclipse_(software)

[17] Wikipedia o Representational state transfer, dostopno na:
http://en.wikipedia.org/wiki/Representational_state_transfer

[18] Wikipedia o JSON, dostopno na:
http://en.wikipedia.org/wiki/JSON

[19] The Python DB Datastore API, dostopno na:
https://developers.google.com/appengine/docs/python/datastore/