UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Hafner

Prototip sistema za evidentiranje živine

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2015

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Hafner

Prototip sistema za evidentiranje živine

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Rok Rupnik

Ljubljana, 2015

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Evidentiranje živine je postopek, ki ga ni možno izvajati le iz pisarne v kmetiji in potrebuje tudi informacijsko podporo v stanju mobilnosti. Zasnujte sistem, ki bo preko spletne in mobilne aplikacije kmetijam omogočal izvajanje postopka evidentiranja živine. Mobilna aplikacija naj bo razvita za platformo Android, tehnologije za spletno aplikacijo pa izbirajte med naslednjimi tehnologijami: Javascript, PHP, HTML, CSS, MySQL, SQLite in JSON. Pri mobilni aplikaciji posebno pozornost namenite učinkovitemu uporabniškemu vmesniku.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Andrej Hafner sem avtor diplomskega dela z naslovom:

Prototip sistema za evidentiranje živine

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 14. september 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Roku Rupniku za pomoč in nasvete pri izdelavi diplomskega dela. Obenem gre zahvala tudi družini, ki mi je stala ob strani tekom študija in izdelavi diplomskega dela.

Kazalo

Kazalo slik

Kazalo tabel

Seznam uporabljenih kratic

kratica	angleško	slovensko
QR	Quick response	Hiter odziv
HTML	HyperText Markup Language	Jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	Kaskadne stilske podloge
AJAX	Asynchronous JavaScript and XML	Asinhroni Javascript in XML
API	Application Programming Interface	Vmesnik za programiranje aplikacij
XML	Extensible Markup Language	Razširljiv označevalni jezik
SQL	Script Query Language	Strukturirani povpraševalni jezik za podatkovne baze
HTTP	Hypertext Transfer Protocol	Protokol za prenos nadbesedil
JSON	JavaScript Object Notation	Notacija Javascript objektov
URL	Uniform resource locator	Enolični krajevnik vira
GUI	Graphical user interface	Uporabniški grafični vmesnik
SDK	Software Development Kit	Programski razvojni paket

Povzetek

Dostop do spleta je dandanes mogoč skorajda že od povsod, zato računalniška tehnologija vztrajno prodira tako rekoč na vsa področja. Izvzeta ni niti kmetijska panoga, ki v zadnjih letih vse bolj stremi k tehnološkem napredku in optimizaciji procesov. V diplomski nalogi je predstavljen razvoj prototipa Android in spletne aplikacije za pomoč evidentiranja živali na kmetijah. Sistem je namenjen predvsem manjšim kmetijam, ki bi aplikacijo uporabljale za vodenje lastne evidence. Prototip je ločen na Android in spletno aplikacijo ter aplikacijski strežnik, prek katerega sistem pridobi, obdela in shrani podatke, ki jih uporabljamo pri delu na terenu ali v pisarni. V prvem delu diplomske naloge so opisane uporabljene tehnologije, med njimi Java, Javascript, PHP, HTML, CSS, MySQL, SQLite in JSON, sledijo pa razvojna orodja, med katerimi najdemo Android Studio in DBDesigner. V glavnem delu sledi predstavitev poteka izdelave obeh aplikacij, in sicer od uporabe tehnologij pri izdelavi, uporabniških vmesnikov do arhitekture sistema, ki je bila zasnovana na podlagi uporabniških zahtev. V zadnjem delu so podana sklepna opažanja in možnosti nadgradnje ter izboljšav za narejeni prototip.

Ključne besede: Android, mobilne aplikacije, spletne aplikacije, evidentiranje živine

Abstract

Access to the Internet is now possible almost from everywhere; therefore the computer technology is constantly advancing virtually to all areas. Even the agricultural sector is not exempted from this, which in recent years has been increasingly striving for technological progress and process optimization. The thesis describes the development of an Android prototype and a web application for the assistance of identifying animals on farms. The system is designed primarily for small farms, which could use the application for the management of their own records. The prototype is divided into the Android and Web application and the application server, through which the system acquires, processes and stores data, which is used at the site or work in the field or in the office. In the first part of the thesis the technology used are described, including Java, Javascript, PHP, HTML, CSS, MySQL, SQLite, and JSON, followed by development tools such as Android Studio and DBDesigner. The main part consists of the presentation of the two applications, from the technologies used in the production, user interfaces to the system architecture, which was designed on the basis of user requirements. The final section provides concluding observations and possibilities for upgrades and improvements of the prototype.

Keywords: Android, Web application, mobile application, livestock records

Poglavje 1 Uvod

Pričetki informatizacije v kmetijstvu segajo v 90. leta prejšnjega stoletja, večji mejnik pa predstavlja leto 1994, ko so v Združenih državah Amerike pričeli z uporabo GPS tehnologije za potrebe zalivanja, škropljenja in gnojenja [1]. S prihodom mlajših generacij in tehnološkim napredkom se je pričela informatizacija tudi na manjših kmetijah, ki bi z vodenjem evidenc rade prihranile čas, obenem pa bi imele zanesljiv pregled nad podatki.

Naš prototip je zgrajen iz strežniškega dela, kjer imamo spletno aplikacijo in spletno storitev, ki skrbi za komunikacijo med napravami ter Android mobilno aplikacijo, ki jo kmetje uporabljajo na terenu. Android aplikacija služi za enostavno beleženje podatkov o molžah, zdravju živali in izdaji dokumentov, medtem ko dobimo na spletni aplikaciji jasnejši in podrobnejši pregled nad statistikami. Poleg vseh funkcionalnosti, ki jih vsebuje mobilna aplikacija, ima spletna aplikacija možnost administrativnega dela, s pomočjo katerega izdelamo in tiskamo QR-kode, dodajamo uporabnike mobilne aplikacije in nastavimo parametre o obvestilih, ki jih sistem pošilja na elektronsko pošto.

V prvem delu so opisane uporabljene tehnologije in razvojna orodja. V glavnem delu predstavimo potek izdelave obeh aplikacij, od arhitekture sistema, uporabniških vmesnikov do uporabe tehnologij pri izdelavi. V zadnjem delu so podana sklepna opažanja in možnosti nadgradnje ter izboljšav za narejeni prototip.

1.1 Namen sistema

Namen sistema je pohitritev ter poenostavitev procesa beleženja statistike kmetije ter njene živine. Mobilna aplikacija, ki mora imeti zaradi narave dela čim enostavnejši grafični vmesnik, je namenjena uporabnikom na terenu, spletna aplikacija pa nudi podrobnejši, naprednejši in človeku prijaznejši pregled statistik in podatkov.

Dosedanji način obdelave podatkov je bil zamuden, saj je baziral na papirju, tako da je sčasoma prišlo do napak, izgub, analize podatkov pa so bile težko izvedljive, če ne celo neizvedljive. S tem namenom je bil izdelan prototip aplikacije, ki skrbi za enostaven, varen in učinkovit zajem in obdelavo podatkov.

V aplikaciji se namreč vsi podatki shranjujejo na enem mestu, v centralni podatkovni bazi, za varnost pa skrbijo dnevne rezervne kopije, za katere poskrbi ponudnik spletnega gostovanja.

Za vstop v aplikacijo se mora uporabnik najprej prijaviti, tako da strežnik uporabniku vrne živino in podatke, ki mu pripadajo. Glavne funkcije mobilne aplikacije so enostavno dodajanje novih molž, za kar skrbi čitalec QR-kod, s katerim uporabniku prihranimo tipkanje in morebitne napake, pregled statistik, zdravstvenih kartonov za posamezno žival in izdajo PDF-potrdil o odvozu mleka na terenu.

Spletna aplikacija vsebuje vse funkcije mobilne aplikacije, le da so predstavitve nekoliko podrobnejše, obenem pa ima administrator tudi možnost dodajanja novih uporabnikov mobilne aplikacije, tiskanja QR-kod za delo z mobilno aplikacijo in nastavitve o obvestilih.

Poglavje 2 Razvojna orodja in tehnologije

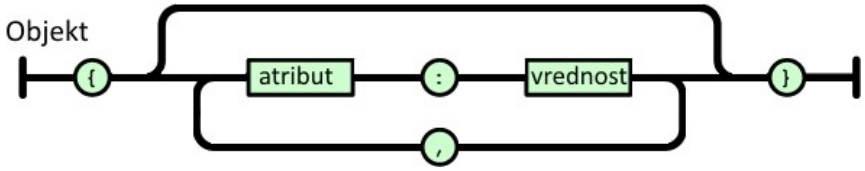
Za razvoj mobilne in spletne aplikacije je bilo uporabljenih več različnih orodij ter tehnologij. V nadaljevanju dela bomo podrobneje predstavili nekatere tehnologije, med drugimi JSON, PHP, JAVA in orodja, kot so Android Studio in DBDesigner. Za uspešen razvoj Android mobilne aplikacije je skrbel Android Studio, ki je v zadnjem letu povsem nadomestil Eclipse. Ta je bil pred tem glavno razvojno orodje za operacijski sistem Android. Za kreiranje podatkovnega modela smo uporabili DBDesigner, na katerem smo ustvarili konceptualni model. Slednjega smo izvozili v SQL-skripto za kreiranje podatkovne baze.

2.1 Tehnologije

2.1.1 JSON

Objektna notacija JSON je preprost format za prenos podatkovnih objektov, ki so sestavljeni po principu množic atribut – vrednost (Slika 1), natančneje pa je opisan v standardih RFC 7159 in ECMA-404 [2]. V našem primeru tehnologijo uporabljamo za prenos podatkov med strežnikom in aplikacijo. Notacijo JSON smo uporabili pred konkurenčnim XML-jem predvsem zaradi njegove lažje uporabe pri delu z Androidom in njegove hitrosti, saj so v večini primerov JSON datoteke zaradi svoje strukture manjše od XML-jevih.

Oče JSON-a je ameriški programer Douglas Crockford, ki je v začetku 20. stoletja kot alternativo XML-ju razvil format, ki je človeku bolj prijazen za branje, hitrejši in enostavnejši [3].



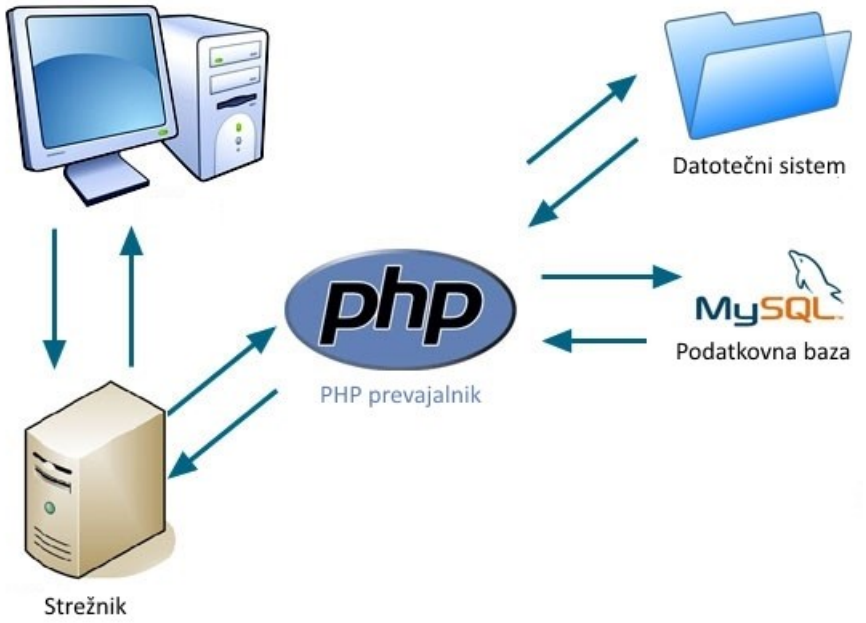
Slika 1 Primer enostavne strukture JSON elementa

2.1.2 PHP

PHP Hypertext Preprocessor je odprtokodni objektno orientiran programski jezik, ki se ga uporablja pri izdelavi dinamičnih spletnih strani v povezavi s HTML elementi. [4] Napisan je v programskem jeziku C, po raziskavah pa PHP uporablja kar 66 odstotkov spletnih strani. PHP se izvaja na strežniški strani, odjemalec preko HTTP protokola pošlje zahtevo na

strežnik, tam pa PHP-jev prevajalnik PHP kodo interpretira in jo izvede (Slika 2). Po končanem izvajanju strežnik največkrat vrne odjemalcu odgovor v HTML obliki.

Za uporabo PHP-ja smo se odločili, ker programski jezik podpira povezavo do večine najbolj uporabljenih podatkovnih baz, med drugim tudi do MySQL-a, ki ga zagotavlja naš ponudnik gostovanja.



Slika 2 Vloga PHP-ja pri delovanju dinamičnih spletnih strani

2.1.3 JavaScript in jQuery

JavaScript je Netscapeov objektno-skriptni jezik, ki je sicer neodvisno razvit od Jave, vendar si z njo deli številne lastnosti in strukture, med katerimi najbolj izstopa možnost uporabe objektov. Pri razvoju prototipa spletne aplikacije smo uporabili JavaScript v sodelovanju s HTML-kodo, saj tako poživimo aplikacijo z dinamičnim izvajanjem (shranjevanje, spreminjanje stilov, odzivi …), jezik pa se z razliko PHP-ja izvaja pri odjemalcu [5].

JQuery velja za najbolj uporabljeno Javascript knjižnico, ki je po podatkih nameščena na kar 65 odstotkov od 10 milijonov najbolj obiskanih spletnih strani na svetu. Prav tako je knjižnica, kot sam programski jezik odprtokodna in je pod okriljem MIT Licence [6].

2.1.4 Java

Java je objektno usmerjen programski jezik, ki je na trg prišel leta 1996, razvil pa ga je James Gosling v podjetju Sun Microsystem. Sintaksa programskega jezika izhaja iz starejših bratov

C in C++, jezik pa je med programerji priljubljen predvsem zaradi svoje objektne usmerjenosti [7]. Java je primarni razvojni jezik Android aplikacij, prav z razvojem mobilnih aplikacij pa se je Java v juniju 2015 povzpela na prvo mesto najbolj uporabljenih programskih jezikov pred programskimi jeziki, kot so C, C++, C# in Python [8].

2.1.5 CSS

CSS so kaskadne stilske predloge, ki omogočajo urejanje stilske podobe HTML-elementov. Namen je, da vsebinske podatke dokumenta loči od podatkov, ki določajo oblikovne lastnosti dokumenta. S CSS-jem se tudi izognemo ponavljanju kode, saj z značkami (class ter id) zagotovimo množici elementov uporabo iste predloge [9]. V našem primeru smo za podobo spletne strani uporabili Bootstrapovo ogrodje, ki poleg CSS datoteke vsebuje še HTML in Javascript datoteke.

2.1.6 Bootstrap

Ogrodje Bootstrap je odprtokodna rešitev HTML in CSS-predloge z Javascript ter jQuery knjižnicami, ki se jo uporablja za izdelavo dinamičnih spletnih aplikacij. Dizajn je prilagojen tudi telefonskim in tabličnim napravam, poleg velikega števila enostavnih elementov so izdelane še številne sestavljene komponente, kot so navigacijski meniji, zaslonski namigi, spustni meniji, različne tabele, filtri, itd… [10].

Bootstrap je bil sprva namenjen interni uporabi v podjetju Twitter. Zaradi nekonsistentnosti pri internem razvoju sta ga razvila Mark Otto in Jacob Thornton. Avgusta 2011 je rešitev postala odprtokodna. Za uporabo Bootstrapa smo se odločili tudi zaradi dejstva, da je podoba aplikacije enaka na vseh zadnjih različicah spletnih brskalnikov, kot so Chrome, Internet Explorer, Mozilla Firefox, Opera in Safari [11]. Dizajn je od različice Bootstrap 2 dalje tudi odziven (ang. responsive), torej nam podobo aplikacije prilagodi velikosti zaslona naprave (Slika 3).

Slika 3 Odziven dizajn spletne aplikacije

2.1.7 MySQL in SQL

MySQL je najpopularnejša odprtokodna rešitev relacijske podatkovne baze, za poizvedbe pa uporablja SQL, ki je najbolj razširjen in standardiziran povpraševalni jezik za delo s podatkovnimi bazami. MySQL je trenutno v lasti Oracla, izdan pa je bil leta 1995. S pomočjo SQL-zapisov smo na strežniku kreirali MySQL podatkovno bazo. Z SQL-poizvedbami smo iz podatkovne baze pridobili ustrezne podatke, ki smo jih nato v JSON obliki vračali mobilni aplikaciji [12].

2.1.8 SQLite

SQLite je relacijska podatkovna baza, realizirana v programskem jeziku C. Podatkovna baza je priljubljena predvsem za lokalno shranjevanje podatkov na mobilnih napravah, uporabljajo jo namreč vsi najpopularnejši mobilni operacijski sistemi, med njimi [13]:

- Blackberry
- Windows Phone
- Apple OS
- Android
- Symbian OS

SQLite podatkovno bazo smo uporabili za shranjevanje podatkov v brezpovezavnem načinu, podatke pa hranimo do prve sinhronizacije s podatkovno bazo na strežniku.

2.2 Razvojna orodja

2.2.1 Android Studio

Android Studio je Googlovo razvojno orodje za Android mobilne aplikacije. Program je v letu 2015 povsem nadomestil Eclipse, katerega se je pred tem z dodatnimi vtičniki uporabljalo za razvoj Android aplikacij. Program je v celoti napisan v Javi, prav tako je izdan pod licenco Apache 2.0, kar pomeni, da je uporaba brezplačna tako v nekomercialne kot komercialne namene [14].

2.2.2 Operacijski sistem Android

Operacijski sistem Android je v letu 2015 najbolj uporabljen operacijski sistem za pametne telefone. Uporablja ga namreč kar 78 odstotkov uporabnikov pametnih telefonov, na drugem mestu sledi Applov iOS z 18 odstotki, pred Windows Phonom in BlackBerryjem [15].

Android je odprtokodni operacijski sistem, ki je zasnovan na operacijskem sistemu Linux. K takšnemu razvoju je najbolj pripomogel spletni velikan Google, ki je leta 2005 kupil razvojni oddelek podjetja Android Inc. Prav s prihodom Googla v Android se je razvoj mobilnih aplikacij vrtoglavo povzpel, saj je razvoj brezplačen, obenem pa ima na voljo enotno razvijalno okolje.

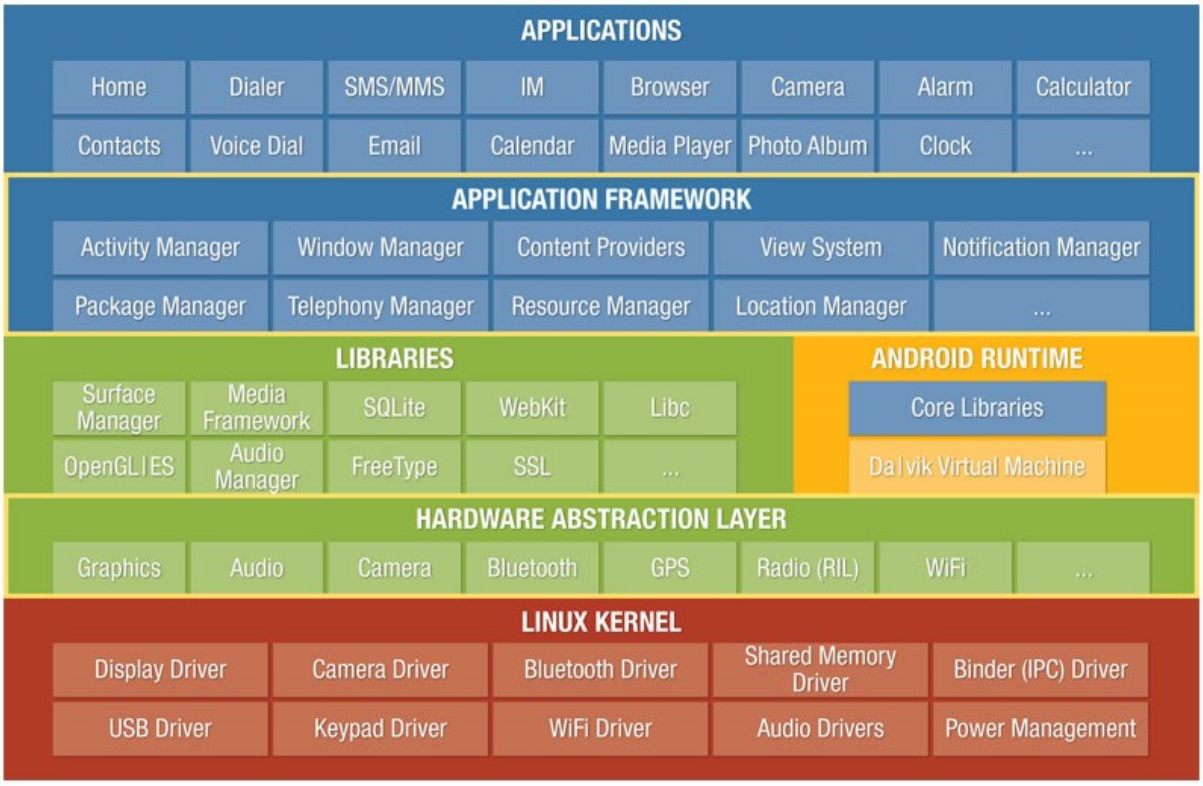
Razvoj aplikacij poteka v visokonivojskem jeziku Java, mogoča pa je tudi kombinacija s C oz. C++ [16].

6. marca 2012 je Google Svetu ponudil aplikacijo v oblaku, imenovano Google Play, za prodajanje in distribucijo aplikacij, spisanih v Androidu. Na spletišče je naloženih že prek 1.5 milijona aplikacij, razlog za takšen uspeh pa je nedvomno nizka cena za razvijalce, saj morajo ti za trajno registracijo odšteti vsega 25 dolarjev, medtem ko iOS -razvijalci na leto odštejejo kar 99 dolarjev [17] [18].

2.2.2.1 Arhitektura sistema

Arhitektura sistema je sestavljena iz šestih komponent, ki skrbijo za tekoče in nemoteno delovanje sistema. To so aplikacije, aplikacijsko ogrodje, zagonsko okolje, knjižnice ter Linux jedro (Slika 4) [19].

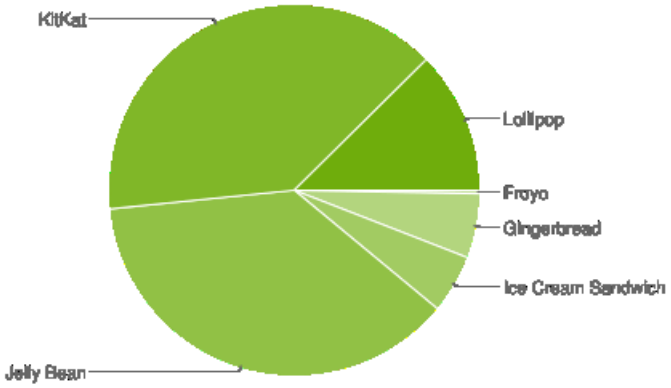
Najpomembnejši del Androida nedvomno predstavlja Linuxovo jedro, ki zagotavlja varnost, upravljanje s pomnilnikom in s procesi. Jedro predstavlja neke vrste abstrakcijo med strojno in programsko opremo, njegove odlike pa so zanesljivost, robustnost in hitrost [20].



Slika 4 Arhitektura sistema

2.2.2.2 Različice sistema

Pri Googlu seveda ne želijo zaspati na lovorikah, zato dnenehno izdajajo nove različice sistema, ki mu dodajo vse več novih funkcionalnosti in možnosti za čim lažji ter učinkovit razvoj. Prva različica sistema se je imenovala 1.5 Cupcake, trenutno pa je najnovejša različica 5.0 Lollipop, ki jo trenutno uporablja nekaj več kot 18 odstotkov Android uporabnikov (Slika 5).



Slika 5 Trenutna uporaba različic sistema Android

Verzije v uporabi, junij 2015:

- Froyo (2.2) 0.3%
- Gingerbread (2.3) 5.6%
- Ice Cream Sandwich (4.0) 5.1%
- Jelly Bean (4.1, 4.2, 4.3) 37.4%
- KitKat (4.4) 39.2%
- Lollipop (5.0, 5.1) 12.5%

Pri razvoju moramo tako spremljati tudi trenutne razmere na tržišču. Nekatere funkcije namreč delujejo le na novejših različicah, torej bi lahko z napačno izbiro API-ja izgubili velik del potencialnih uporabnikov. Priporočljivo je za razvoj uporabljati minimalni API 16, kar pomeni, da je aplikacija na voljo od Android različice 4.0, s tem pa pokrijemo 94 odstotkov Androidovih uporabnikov [21].

2.2.3 DBDesigner

Podatkovni model, ki s shranjevanjem in obdelavo podatkov skrbi za delovanje mobilne ter spletne aplikacije, je izdelan s pomočjo odprtokodne rešitve DBDesigner. Z orodjem najprej ustvarimo konceptualni model z ustreznimi razmerji med entitetami. Orodje omogoča, da model izvozimo v SQL-skripto, to pa uvozimo na podatkovni strežnik. DBDesigner ponuja enostaven in pregleden način izdelave modela, saj vidimo vizualno prestavitev entitet in njihovih razmerij, ki poskrbijo za primarne in sekundarne ključe [22].

2.2.4 PhpMyAdmin

PhpMyAdmin je brezplačna in odprtokodna rešitev, napisana v PHP-ju, ki skrbi za upravljane MySQL podatkovne baze prek spletnega brskalnika. Omogoča vrsto opravil, med njimi so ustvarjanje, urejanje, brisanje in uvoz ali izvoz podatkovnih baz, tabel, atributov ter vrednosti. Zgodovina sega v leto 1998, do danes pa je phpMyAdmin predvsem zaradi svoje enostavne uporabe prevzel vodilno mesto po uporabi pri ponudnikih spletnih gostovanj [23].

Nekaj ključnih funkcij odprtokodne rešitve:

- Spletni vmesnik,
- MySQL urejevalnik,
- možnost uvoza CSV ali SQL datotek,
- izvoz podatkov v različnih formatih, med njimi CSV, SQL, XML, PDF, Word, Excel, Latex,

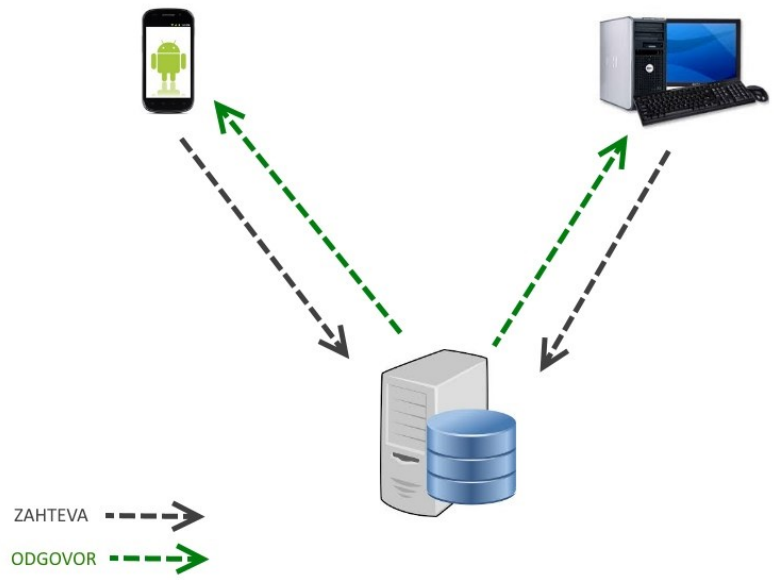
- možnost izvedbe kompleksnih poizvedb s pomočjo QBE,
- deluje na različnih operacijskih sistemih,
- prikaz grafov v realnem času za nadzor aktivnosti MySQL strežnika, kot so aktivne povezave, procesi, poraba pomnilnik in procesorja.

Poglavje 3 Razvoj sistema

V poglavju je podrobneje prestavljen rezultat in potek izdelave diplomskega dela.

3.1 Arhitektura sistema

Sistem je sestavljen iz treh večjih delov. To so Android mobilna aplikacija, spletna aplikacija in aplikacijski strežnik (Slika 6). Vlogi mobilne in spletne aplikacije sta podobni, saj obe na strežnik pošiljata zahtevke po določenih podatkih, ki jih strežnik prejme, kot odgovor pa posreduje podatke, ki so v našem primeru zapisani v JSON-formatu. S takšno arhitekturo dosežemo, da je logika ločena od aplikacij, kar nam zagotavlja večjo neodvisnost in možnosti za lažji nadaljnji razvoj.



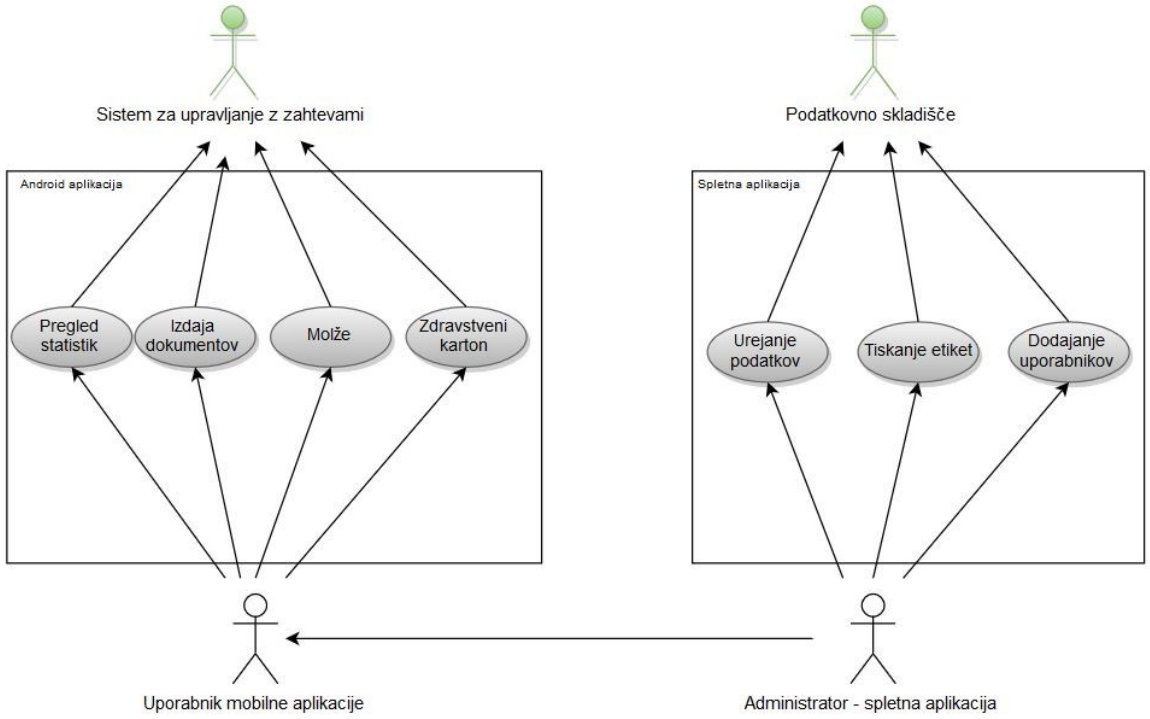
Slika 6 Arhitektura aplikacije

3.1.1 Primeri uporabe

Glavna akterja sistema sta uporabnik in administrator, ki ima poleg dodatnih tudi vse pravice navadnega uporabnika. Uporabnik lahko do sistema dostopa prek Android mobilne aplikacije, medtem ko administrator do spletne aplikacije dostopa prek spletnega brskalnika.

Uporabnik lahko dodaja nove molže, zdravstvene kartone, izdaja dobavnice in pregleduje statistike. Medtem lahko administrator poleg zgoraj naštetih akcij dodaja nove uporabnike

mobilne aplikacije, ureja podatke o kmetiji in živalih, generira QR kode in ureja administrativne nastavitve. Oba sistema sta povezana s podatkovnim skladiščem, ki je v našem primeru MySQL podatkovna baza (Slika 7).

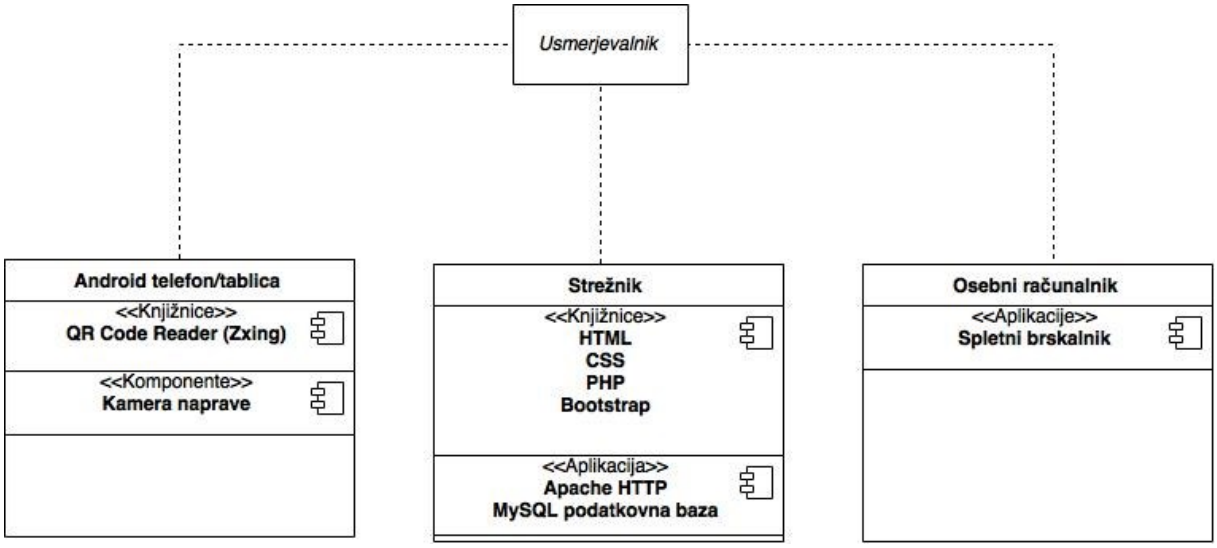


Slika 7 Primeri uporabe

3.1.2 Implementacijski pogled

Na sliki (Slika 8) je viden komponenti diagram, ki vsebuje dele, ki jih sistem uporablja za svoje delovanje. Android aplikacija za potrebe dodajanja nove molže potrebuje dostop do napravine kamere, ki s pomočjo odprtokodne knjižnice ZXing prebere podatke o živali, ki jo dodamo molži. Android naprava za komunikacijo s strežnikom uporablja HTTP protokol, tako, da strežnik s strani klienta prejme zahteve, te obdela ter jih nato posreduje nazaj h klientu.

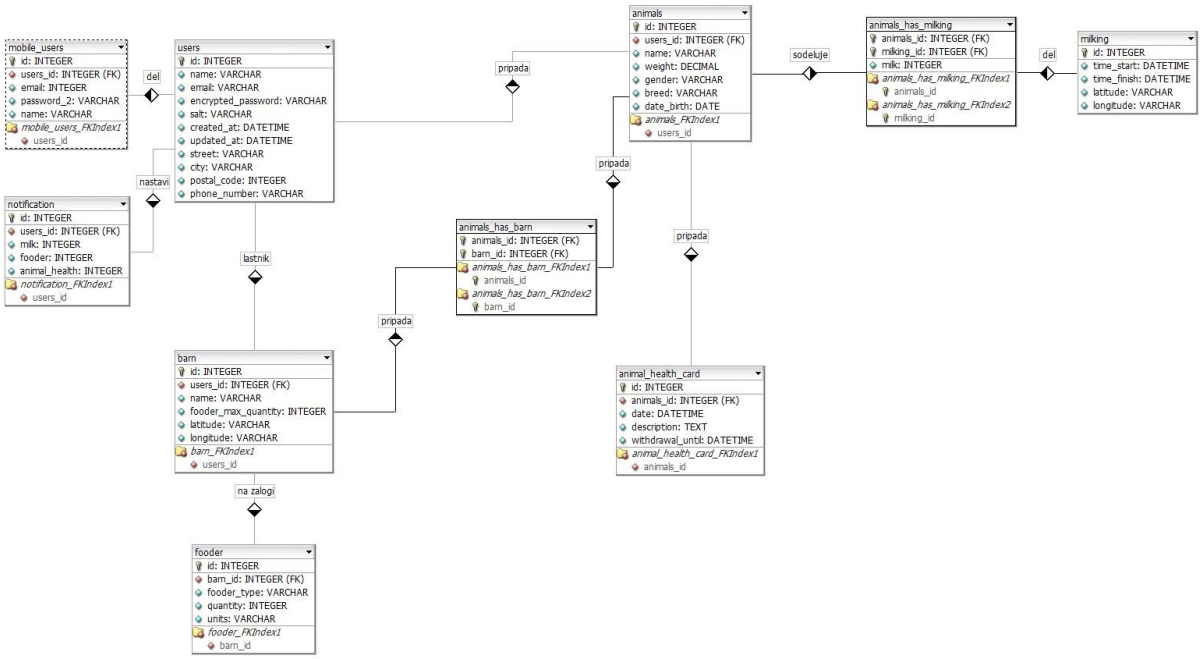
Za dostop do spletne aplikacije uporabljamo osebni računalnik ali pameten telefon, kajti za uporabo potrebujemo le spletni brskalnik.



Slika 8 Implementacijski pogled

3.2 Podatkovni model

Pred pričetkom izdelave aplikacije je bilo treba ustrezno nastaviti podatkovni model. Z ustreznim in fleksibilnim podatkovnim modelom je mogoč tudi nadaljnji razvoj, kar pomeni, da je začetna zasnova modela še kako pomembna. V našem primeru začetna shema vsebuje osem entitet, v katerih hranimo podatke o uporabnikih, živini, molžah, hlevih, zalogah in zdravstvenih kartonih (Slika 9).



Slika 9 Podatkovni model aplikacije

Entiteta »users« hrani podatke o uporabnikih oz. administratorjih sistema. To so ime, elektronski naslov, telefonska številka, kraj, poštna številka, geslo za dostop do aplikacije in sol, ki skrbi za varnost pred vdori v uporabniški račun. Tabela vsebuje primarni ključ, ki se z novimi vnosi samodejno povečuje. Entiteta je povezana še s štirimi preostalimi entitetami, to so »barn«, »mobile_users«,«notification« in »animals«.

Entiteta »mobile_users« hrani podatke o uporabnikih mobilne aplikacije. Te lahko uporabnik oz. administrator dodaja prek spletne aplikacije. Entiteta poleg svojih atributov vsebuje tudi tuji ključ, ki ga pridobi iz tabele »users«.

Entiteta »animals« hrani podatke o živini, ki pripada določenemu lastniku. Prav zato vsebuje tabela tudi tuji ključ uporabnika, saj gre v tem primeru med entitetama za razmerje 1,n. V tabeli hranimo še vrednosti o imenu, teži, spolu, pasmi in datum rojstva.

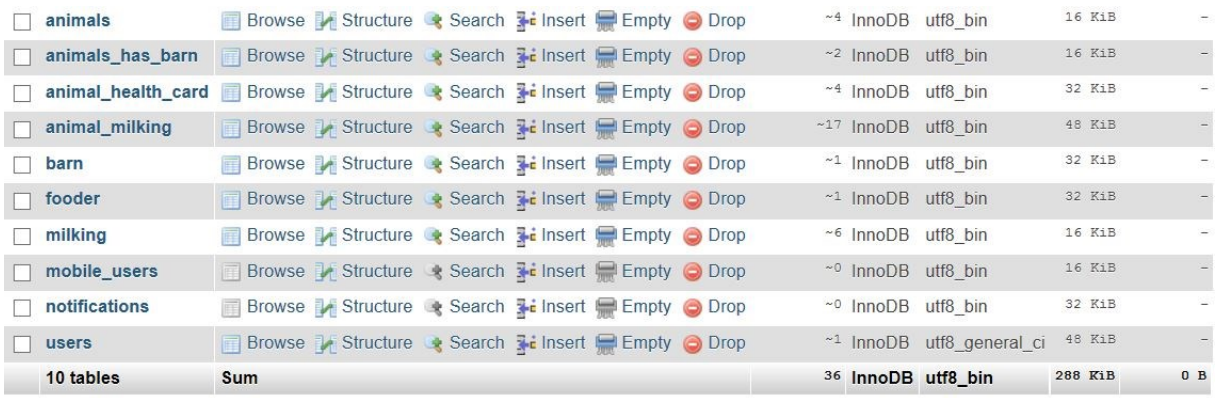
Entiteta »animal_health_card« vsebuje podatke o kurativah, ki smo jih vnesli za posamezno žival. Aplikacija omogoča tudi možnost hkratnega vnosa, ki določeno bolezen ali bolezenske znake pripiše večjemu številu živali hkrati. Tabela poleg primarnega ključa, datuma vnosa, časa karence in opisa bolezni vsebuje tudi tuji ključ živali.

Entiteta »milking« vsebuje čas pričetka in konca molže ter njeno lokacijo. Slednja entiteta je povezana z entiteto »animals«, zaradi razmerja m,n se kreira vmesna tabela »animals_has_milking«. Ta vsebuje dva tuja ključa in podatek o številu litrov, ki ga določena žival da na posamezni molži.

Entiteta »barn« vsebuje podatke o hlevu oz. planini, ki si jo lasti uporabnik. Tukaj hranimo podatke o imenu, lokaciji in kapacitetah skladišča za krmo. Entiteta vsebuje tudi tuji ključ uporabnika oz. administratorja. Poleg povezave z entiteto »users« ima razmerje še z entitetama »fooder« in »animals«. Zaradi m,n razmerja z entiteto »animals« se ponovno kreira vmesna entiteta »animals_has_barn«

Sledita še entiteti »fooder« kjer hranimo podatke o krmi, ki pripada hlevu oz. planini in »notification«, kjer nastavimo vrednosti za sistemsko obveščanje.

Po izdelanem ER modelu se je na strežnik izvozilo SQL datoteko, ki je kreirala tabele (Primer 1) in relacije med njimi. Podatkovno bazo tako skupaj z vmesnimi tabelami sestavlja deset tabel (Slika 10).



animals	Browse Structure Search Insert Empty Drop	~4	InnoDB	utf8_bin	16 KiB	-
animals_has_barn	Browse Structure Search Insert Empty Drop	~2	InnoDB	utf8_bin	16 KiB	-
animal_health_card	Browse Structure Search Insert Empty Drop	~4	InnoDB	utf8_bin	32 KiB	-
animal_milking	Browse Structure Search Insert Empty Drop	~17	InnoDB	utf8_bin	48 KiB	-
barn	Browse Structure Search Insert Empty Drop	~1	InnoDB	utf8_bin	32 KiB	-
fooder	Browse Structure Search Insert Empty Drop	~1	InnoDB	utf8_bin	32 KiB	-
milking	Browse Structure Search Insert Empty Drop	~6	InnoDB	utf8_bin	16 KiB	-
mobile_users	Browse Structure Search Insert Empty Drop	~0	InnoDB	utf8_bin	16 KiB	-
notifications	Browse Structure Search Insert Empty Drop	~0	InnoDB	utf8_bin	32 KiB	-
users	Browse Structure Search Insert Empty Drop	~1	InnoDB	utf8_general_ci	48 KiB	-
10 tables	**Sum**	**36**	**InnoDB**	**utf8_bin**	**288 KiB**	**0 B**

Slika 10 Tabele podatkovnega modela

```
CREATE TABLE users (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  name VARCHAR NULL,
  email VARCHAR NULL,
  encrypted_password VARCHAR NULL,
  salt VARCHAR NULL,
  created_at DATETIME NULL,
  updated_at DATETIME NULL,
  street VARCHAR NULL,
  city VARCHAR NULL,
  postal_code INTEGER UNSIGNED NULL,
  phone_number VARCHAR NULL,
  PRIMARY KEY(id)
);
```

Primer 1 SQL koda za kreiranje tabele »users«

3.3 Android mobilna aplikacija

3.3.1 Datoteki Android Manifest in Gradle

Datoteka AndroidManifest.xml je datoteka, ki skrbi za ključne nastavitve v naši aplikaciji. V njej namreč določimo minimalno in ciljno SDK-verzijo, ime aplikacije, podobo ikone za zagon aplikacije, dovoljenja aplikacije ter navedemo vse aktivnosti, ki so del aplikacije. V našem primeru aplikacija za delovanje potrebuje dvojno dovoljenje. Za komunikacijo s spletnim strežnikom in zaznavanju GPS-lokacije potrebuje aplikacija dostop do spleta in podatke o lokaciji, kar omogočimo s spodnjimi zapisi (Primer 2).

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Primer 2 Dovoljenja v datoteki AndroidManifest.xml

Gradle je še ena izmed najpomembnejših Androidovih datotek, v kateri med drugim vključujemo tudi zunanje knjižnice, ki jih uporabljamo v aplikaciji. V našem primeru smo tako vključili knjižnico ZXing čitalca črtnih kod (Primer 3).

```
compile('eu.livotov.labs:zxscanlib:2.0.1@aar') { transitive = true }
```

Primer 3 Vključevanje zunanjih knjižnic preko Gradla

3.3.2 Uporabniški vmesnik

V našem primeru je bilo potrebno izdelati dva tehnično in vizualno povsem različna uporabniška vmesnika – enega čim enostavnejšega za delo na mobilni napravi ter drugega nekoliko bolj kompleksnega in podrobnejšega, saj delo z njim v večini primerov poteka prek osebnega računalnika.

Grafični uporabniški vmesnik, ali po angleško GUI, prikazuje elemente, kot so ikone, okna, temelje uporabniških vmesnikov, kot jih poznamo sedaj, pa je postavil raziskovalni oddelek Xerox PARC že daljnega leta 1973 [24].

Pri razvoju uporabniškega vmesnika za Android smo uporabljali XML-označevalni jezik, s katerim smo vsaki aktivnosti ali fragmentu namenili ustrezno .xml datoteko, ki je ponazarjala podobo zaslona. V primeru, da je šlo za aktivnost ali fragment s spremljajočo se podobo (dodajanje gumbov, novih polj …) pa lahko dodajamo nove elemente neposredno prek javanskih ukazov (Primer 4).

```
Button milkinButton = new Button(this);
milkinButton.setText(getResources.getText(R.string.milking));
milkinButton.setLayoutParams(new LayoutParams(AbsListView.LayoutParams.FILL_PARENT,
                                    AbsListView.LayoutParams.WRAP_CONTENT));
linerLayout.addView(milkinButton);
```

Primer 4 Programsko dodajanje elementov na platno

V Android aplikaciji smo največkrat uporabili gradnike kot so EditText, ki skrbi za vnos besedila, Spinner za izvlečni seznam, TextView za prikaz besedila (Primer 5), ListView za generično dodajanje besedila in Button, ki predstavlja gumb.

```
<TextView
    android:id="@+id/waitLabel"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/white"
    android:gravity="center"
    android:text="@string/please_wait"/>
```

Primer 5 Primer polja za prikaz besedila

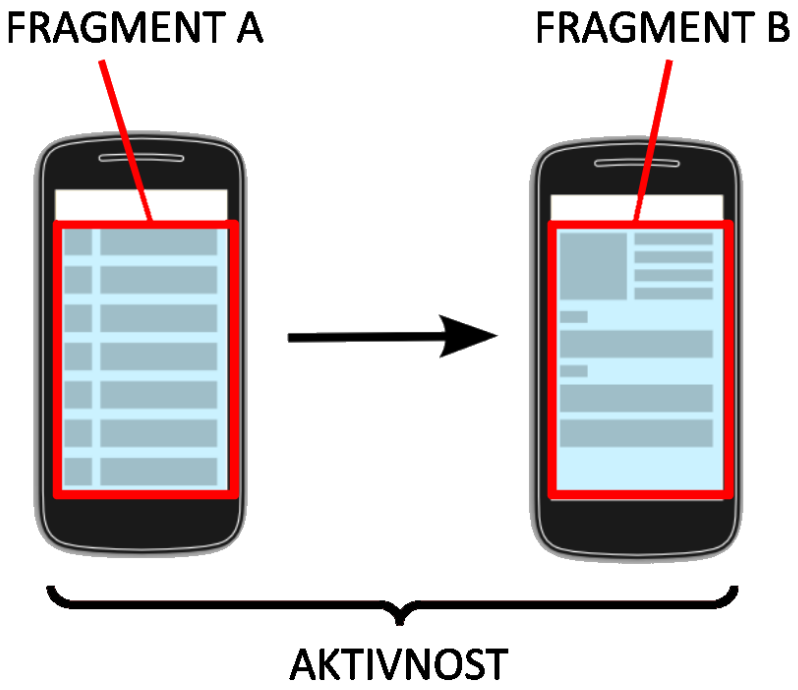
3.3.3 Aktivnosti in fragmenti

Primarno stvar oziroma srce mobilne aplikacije predstavljajo Androidove aktivnosti, ki jim določimo XML-datoteko za podobo in javansko datoteko, ki skrbi za delovanje ter obnašanje aktivnosti. Aktivnost je tako sestavljena iz javanske in XML datoteke, za uporabo pa jo je potrebno navesti v AndroidManifest datoteki (Primer 6).

```
<activity
    android:name=".pdf.PdfActivity"
    android:label="@string/title activity pdf" >
</activity>
```

Primer 6 Deklariranje aktivnosti v AndroidManifest.xml

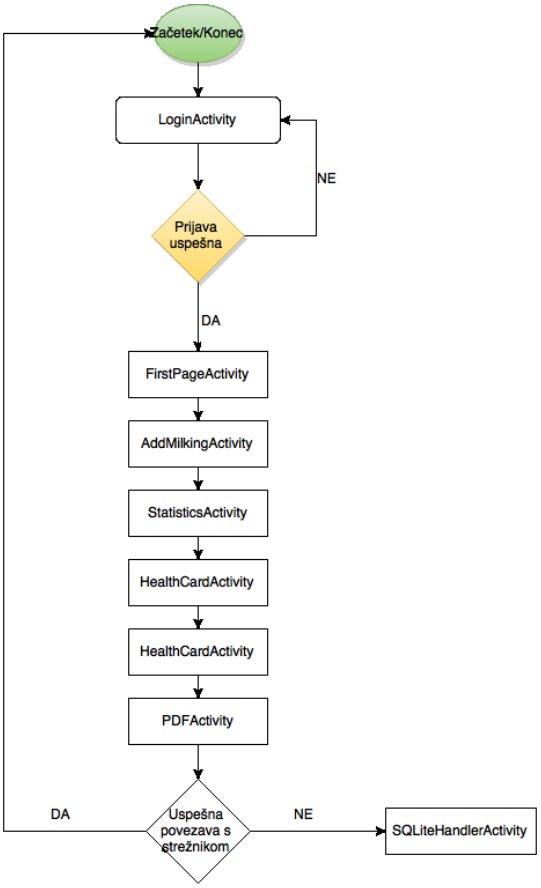
Poleg aktivnosti v naši aplikaciji uporabljamo tudi fragmente, ki so del aktivnosti. Slednje prav tako sestavljata javanska in XML-datoteka. Fragment je del aktivnosti, kar pomeni, da znotraj aktivnosti lahko gostuje več fragmentov, ki med seboj komunicirajo [25](Slika 11).



Slika 11 Aktivnost in fragment

Ker naša aplikacija vsebuje več aktivnosti, moramo v datoteki AndroidManifest.xml določiti aktivnost, ki se ob zagonu aplikacije izvede prva. V našem primeru je to »LoginActivity«, ki takoj ob zagonu preveri, ali ima uporabnik veljavno sejo. V primeru pozitivnega odgovora se

uporabnika preusmeri na aktivnost »FirstPageActivity«, kjer preko menija izbiramo nadaljnje akcije. (Slika 12).



Slika 12 Potek aktivnosti v mobilni aplikaciji

3.3.3.1 Aktivnost LoginActivity

Aktivnost LoginActivity se uporablja kot uvod v aplikacijo. Ob zagonu le-te namreč sistem preveri, ali obstaja veljavna seja (Primer 7), V primeru seje se akcije aktivnosti preskočijo in aplikacija zažene aktivnost FirstPageActivity, kjer nato prek drsečega menija izbiramo nadaljnje akcije, prek katerih v podatkovno bazo shranjujemo in obdelujemo podatke (Slika 13).

```
if (session.isLoggedIn()) {
    Intent intent = new Intent(LoginActivity.this, FirstPageActivity.class);
    startActivity(intent);
    finish();
}
```

Primer 7 Preverjanje seje

Slika 13 Drsni meni na aktivnosti

V primeru neveljavne seje moramo izpolniti polja »Uporabniško ime« in »Geslo«. S pritiskom na gumb »Prijava« se na strežnik pošljejo vneseni podatki, ki jih z SQL-poizvedbo preverimo in sporočimo, ali so vneseni podatki pravilni.

Kot strežniški odgovor nato prejmemo podatke v obliki JSON (Primer 8), za lažjo nadaljnjo uporabo podatkov uporabimo javansko knjižnico GSON, ki podatke samodejno pretvori v javanski objekt.

```
{"tag":"login","error":false,"uid":"5589bdb802dc96.27465285","user":{"id":2,"name":"Andrej","e
mail":"test@gmail.com","created_at":"2015-06-23 22:12:40","updated_at":null}}
```

Primer 8 JSON odgovor s strani strežnika

Ključna stvar, ki jo moramo kot programerji poznati, je struktura odgovora. Vedeti moramo, kakšen odgovor pričakujemo. Tako javanskemu razredu nastavimo atribute z enakim imenom in tipom, kot so atributi v JSON-odgovoru. V našem primeru najprej preverimo, če je atribut »error« nastavljen na »false«, če to drži, pretvorimo vsebino objekta, ki se nahaja v atributu »user« v javanski objekt (Primer 9).

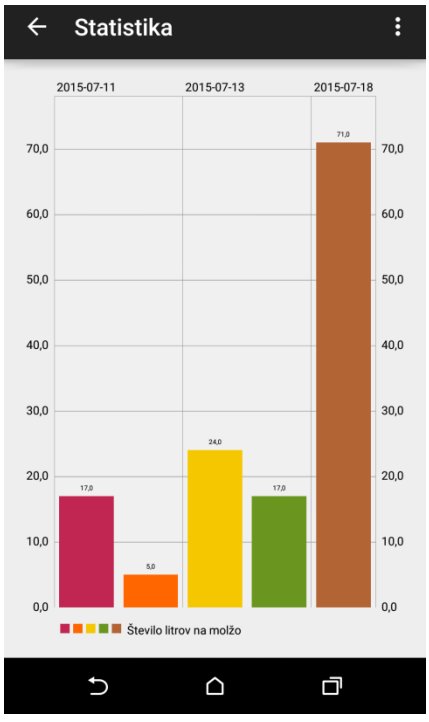
```
public class User {

    private int id;
```

```
    private String name;
    private String email;
    private String created_at;
    private String updated_at;

}
```

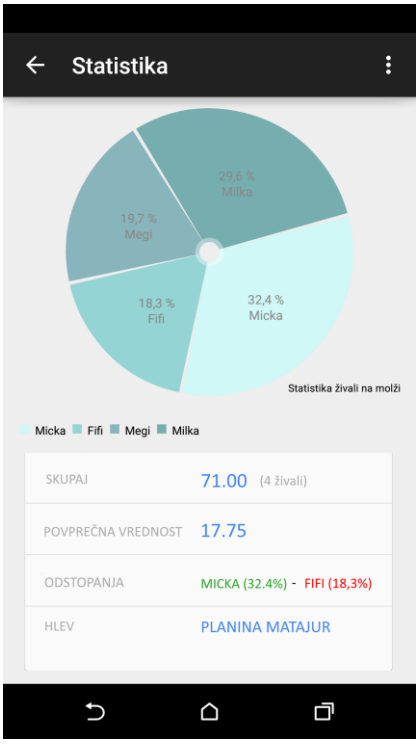
Primer 9 Javanski razred User

3.3.3.2 Aktivnost FirstPageActivity

Aktivnost FirstPageActivity vsebuje drsni meni za izbiro vseh možnih aktivnosti in velja za glavni zaslon, kamor se vračamo iz preostalih aktivnosti. Aktivnost gostuje tudi dva fragmenta. Fragment »MilkingGraphFragment« na zaslon izriše graf zadnjih petih molž (Slika 14), ki jih je opravil uporabnik. Fragment ponuja tudi možnost interakcije z grafom. Ob kliku na stolpec podatke ustrezno razberemo, te pa v fragmentu »MilkingDetailFragment« izpišemo kot rezultat podrobnega pregleda izbrane molže (Slika 15).



Slika 14 Pregled zadnjih molž



Slika 15 Fragment podrobnega pregleda molže

3.3.3.3 Aktivnost ScanActivity

Aktivnost ScanActivity predstavlja primarno dejavnost mobilne aplikacije. V njej namreč generiramo novo molžo, s pomočjo QR-bralca kod pa na molžo dodajamo živino. Vsaka žival ima namreč pri sebi QR-kodo, ki jo v administrativni spletni aplikaciji generira program, tako da je izdelava teh povsem enostavna. QR-koda namreč vsebuje podatke o imenu živali in njenem ID-ju (enolični identifikator oziroma primarni ključ v tabeli), program pa na podlagi prebranega zapisa generira javanske objekte (Slika 16).

Kaj pa, če je žival trenutno v karanteni? V primeru podatkovnega omrežja sistem namreč ob vsakem zajetem objektu prek čitalca kod na strežniku preveri, ali je žival ustrezna za opravljanje molže. V tabeli »animal_health_card« namreč hranimo datum, do katerega je žival neprimerna za opravljanje molže. V primeru zaznave prekoračenja karence je o tem obveščen uporabnik, živali pa tudi ni mogoče uvrstiti na molzni seznam. Za ustrezno preverjanje datumov skrbi knjižnica Joda Time.

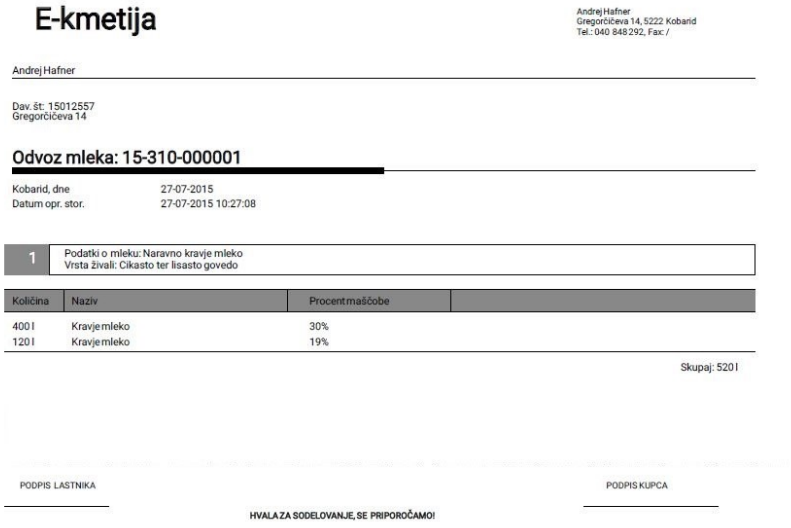
Veljavne objekte shranjujemo v povezanem seznamu (ang. ArrayList), ob končani molži pa s pomočjo knjižnice GSON na strežnik posredujemo podatke. Te obdelamo in jih shranimo na podatkovnemu strežniku. V primeru, da Android naprava nima dostopa do spleta, se ti podatki shranijo na lokalni podatkovni bazi SQLite do prve možne sinhronizacije.

Za shranjevanje podatkov na napravi smo kreirali razred »SQLiteHandler«, ki razširja Androidov razred »SQLiteOpenHelper«. Android »SQLiteOpenHelper« je namreč abstraktni razred v paketu »android.database.sqlite«, s pomočjo katerega kreiramo, vstavljamo, urejamo in beremo podatke iz mobilne naprave. Razred vsebuje kar nekaj metod, med njimi »onCreate«, ki se kliče ob primeru, da je baza narejena prvič, »onOpen«, ki skrbi za odpiranje podatkovne baze in »getWritableDatabase«, ki kreira in/ali odpre podatkovno bazo, ki jo bomo uporabljali za pisanje ali branje [26].

Slika 16 Dodajanje živali na molžo

3.3.3.4 Aktivnost PdfActivity

V aktivnosti uporabniku mobilne aplikacije omogočimo izdajo PDF potrdil o odvozu mleka (Slika 17). Generiranje potrdil nam omogoča Androidova privzeta knjižnica »android.graphics.pdf«. Kreiranje dokumenta poteka programsko, saj na platno (ang. Canvas) izrisujemo like in besedilo. [27]



E-kmetija

Andrej Hafner
Gregorčičeva 14, 5222 Kobarid
Tel.: 040 848 292, Fax: /

Andrej Hafner

Dav. št: 15012557
Gregorčičeva 14

Odvoz mleka: 15-310-000001

Kobarid, dne 27-07-2015
Datum opr. stor. 27-07-2015 10:27:08

1 Podatki o mleku: Naravno kravje mleko
Vrsta živali: Cikasto ter lisasto govedo

Količina	Naziv	Procent maščobe	
400 l	Kravje mleko	30%	
120 l	Kravje mleko	19%	

Skupaj: 520 l

PODPIS LASTNIKA

PODPIS KUPCA

HVALA ZA SODELOVANJE, SE PRIPOROČAMO!

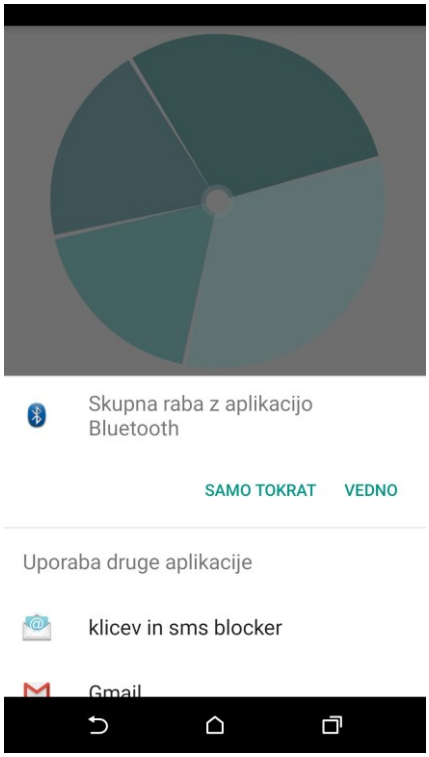
Slika 17 Generiran PDF dokument

Programsko moramo poskrbeti tudi za dodajanje novih vrstic in številčenje strani. Tako po vsakem dodanem zapisu preverimo, ali je stran pri koncu, v tem primeru dodamo novo stran, kjer nadaljujemo s pisanjem vsebine PDF-dokumenta (Primer 10).

```
if(position > 800){
    document.finishPage(page);
    pageInfo = new PdfDocument.PageInfo.Builder(595, 842, 1).create();
    page = document.startPage(pageInfo);
    position = titleBaseLine;
}
```

Primer 10 Preverjanje dolžine dokumenta in kreiranje nove strani

Po uspešno generiranem PDF dokumentu lahko tega direktno natisnemo, saj nam sistem ponudi opcijo pošiljanja preko povezave modrega zoba (ang. Bluetooth) direktno na tiskalnik (Slika 18).



Slika 18 Pošiljanje datoteke preko modrega zoba

Izdaja PDF dokumentov je mogoča le ob predpogoju, da uporabnik uporablja napravo z Android verzijo vsaj 4.4 z imenom KitKat (API level 19). V predhodnih različicah sistem Android namreč še ni ponujal možnosti kreiranja takšnih dokumentov [28].

3.3.3.5 Aktivnost MapsActivity

Kot je razvidno iz podatkovnega modela, ima lahko uporabnik več hlevov oz. planin z živino, v katerih opravlja molžo. Da bi uporabniku mobilne aplikacije čim bolj poenostavili delo, aplikacija prek mobilne naprave razbere GPS-koordinate (Primer 11).

```
private void findLocation(){

    LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    Criteria criteria = new Criteria();
    String provider = lm.getBestProvider(criteria, true);
    Location location = lm.getLastKnownLocation(provider);
    double latitude = location.getLatitude();
    double longitude = location.getLongitude();

}
```

Primer 11 Zaznava GPS lokacije

Sistem prek Androidove knjižnice Criteria izbere v trenutku najbolj natančno opcijo za izračun, to pa sta GPS na podlagi satelitskih signalov ali pa izračun lokacije s pomočjo WI-FI-ja in mobilnega omrežja. V podatkovni bazi ima hlev namreč zapisane koordinate. Čeprav te koordinate niso identične s pridobljenimi, pa s pomočjo metode (Primer 12) pridobimo razdalje do vseh hlevov. Med razdaljami izberemo najkrajšo, koordinate tega objekta pa pripišemo molži. Tako brez dodatnega dela pridobimo podatke o lokaciji molže, ki je bila opravljena v hlevu ali planini.

```
public static double getDistance(LatLng barn, LatLng position){

    double d2r = Math.PI / 180;
    double dLong = (position.longitude - closestPoint.longitude) * d2r;
    double dLat = (position.latitude - closestPoint.latitude) * d2r;
    double a = Math.pow(Math.sin(dLat / 2.0), 2) + Math.cos(closestPoint.latitude * d2r)
            * Math.cos(position.latitude * d2r) * Math.pow(Math.sin(dLong / 2.0), 2);
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    double d = 6367000 * c;
    return Math.round(d);

}
```

Primer 12 Razdalja med dvema koordinatama

Aktivnost vsebuje tudi možnost prikaza hlevov in osnovnih podatkov na Google zemljevidih (Slika 19). Za uporabo zemljevida potrebujemo naložen Google Play Services SDK in veljavni Google Maps API key [29]. Slednjega kreiramo v Google API konzoli ali pa na Googlovi strani (https://console.developers.google.com), namenjeni tako spletnim kot Android razvijalcem. Generirani ključ nato vključimo v datoteko »google_maps_api.xml« (Primer 13).

```
<string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">
    AI*****y0ON9-pHN******h7SgjFpBTe6****c
</string>
```

Primer 13 Vnos API ključa za prikaz Google zemljevida

Slika 19 Prikaz hlevov na Google zemljevidih

Android Studio razvijalcem ponudi možnost za avtomatsko generiranje aktivnosti, ki vključujejo Google zemljevid. Tako se generira aktivnost, ki gosti fragment z zemljevidom. Medtem ko fragment skrbi za interakcijo zemljevida, v aktivnosti navedemo atribute in obnašanje zemljevida. Tako na zemljevidu dodajamo oznake (ang. markers) s pripisi, nastavimo pogled in rišemo objekte (Primer 14).

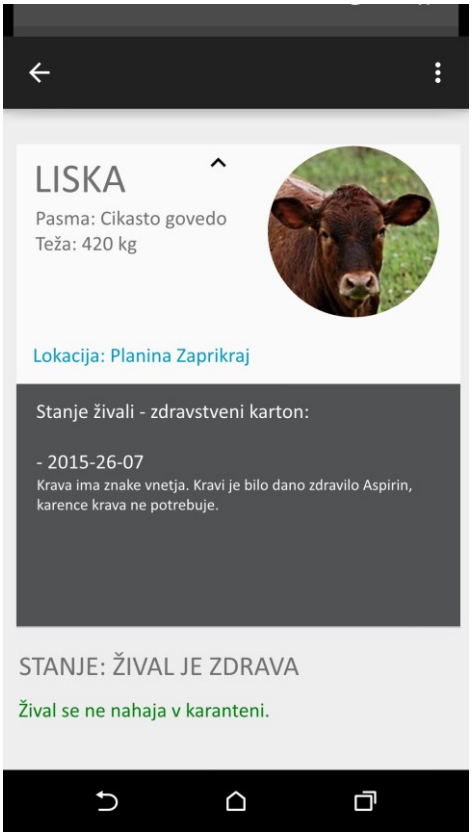
```
private void setUpMap() {
    MarkerOptions marker = new MarkerOptions();
    marker.position(new LatLng(barn.getLatitude(), barn.getLongitude)).title(barn.getName());

    mMap.addMarker(marker);
}
```

Primer 14 Dodajanje oznake na zemljevid

3.3.3.6 Aktivnost HealthCardActivity

Poleg dodajanja nove molže je ena najuporabnejših aktivnosti tudi pregled in dodajanje zdravstvenih kartonov. Uporabnik lahko prek izbirnega seznama ali prek odčitane QR-kode izbere žival, o kateri pridobi ustrezne informacije, ali pa vnese podatke o zdravstvenem stanju. V primeru prihoda veterinarja lahko te podatke izvozi v PDF-datoteko, uporabnik pa

pred pričetkom z delom preveri, ali je žival zdravstveno sposobna in kakšne so morebitne zahteve za njeno nego (Slika 20).



Slika 20 Pregled zdravstvenega kartona

3.3.4 Sinhronizacija podatkovne baze

Kot smo že omenili, je bil eden izmed pogojev mobilne aplikacije tudi možnost delne uporabe aplikacije v brezpovezavnem načinu. Za ta namen smo uporabili SQLite relacijsko podatkovno bazo, kamor smo podatke shranili v primeru, da naprava ni imela kakršnekoli povezave s podatkovnim omrežjem. Tako smo pred vsako zahtevo na strežniku preverili, ali ima naprava dostop do spleta, to pa smo storili s pomočjo spodnje kode (Primer 15).

```
private boolean networkAvailable() {
    ConnectivityManager cm
            = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = cm.getActiveNetworkInfo();
    if (activeNetworkInfo != null && activeNetworkInfo.isConnected()){
        return true;
    }else{
        return false;
    }
}
```

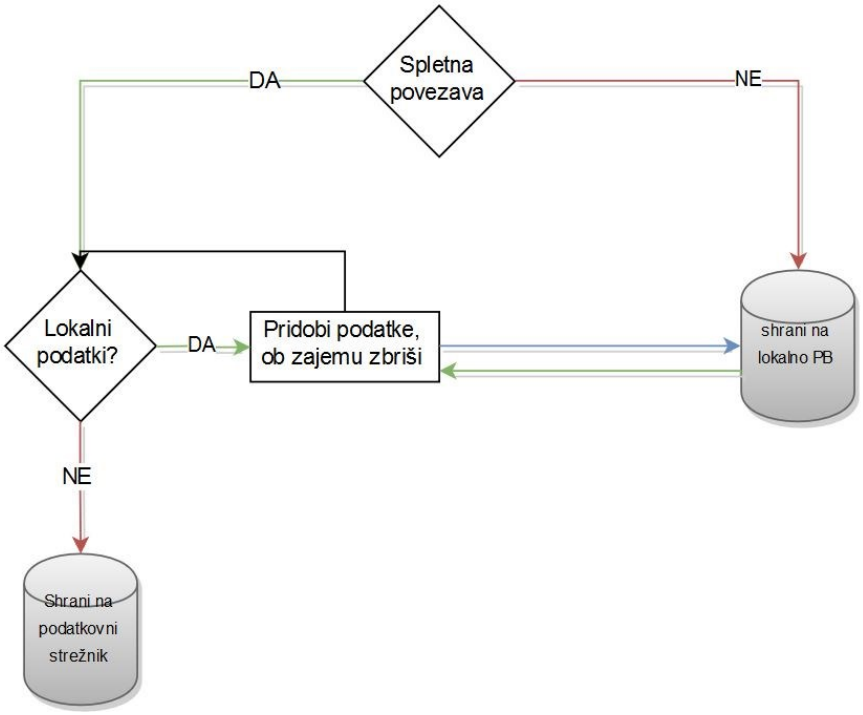
Primer 15 Preverjanje podatkovne povezave

Pozitiven oz. trdilni (ang. true) odgovor metode networkAvailable pomeni, da ima naprava podatkovno povezavo. Pred pošiljanjem zahtev na strežnik preverimo stanje na lokalni podatkovni bazi. Preverjanje tabel je tako odvisno od vrste akcije, ki jo trenutno opravljamo. V primeru, da tabele vsebujejo podatke, te priključimo v povezan seznam. Ob uspešnem zajemu podatkov lokalne tabele pobrišemo, pridobljene podatke pretvorimo v format JSON, ki jih preko metode POST pošljemo na podatkovni strežnik (Slika 21).

V nasprotnem primeru oz. negativnem odgovoru (ang. false) smo na telefonu, če ta že ni imel kreirane podatkovne baze in tabel, te ustvarili in vanje vpisali objekte (Primer 16).

```
public void addHealthCard(HealthCard healthCard){
    Log.d("addNewLocalHealthCard", healthCard.toString());
    // 1. Prejmemo referenco do vseh lokalnih podatkovnih baz
    SQLiteDatabase db = this.getWritableDatabase();

    // 2. Pripravimo podatke atribut/vrednost
    ContentValues values = new ContentValues();
    values.put(KEY_ANIMAL_ID, healthCard.getID());
    values.put(KEY_DATE, healthCard.getDate());
    values.put(KEY_DATE_ WITHDRAWAL, healthCard.getWithdrawalDate(););
    values.put(KEY_DESCRIPTION, healthCard.getDescription());

    // 3. Vstavljanje podatkov v tabelo
    db.insert(TABLE_HEALTHCARD, null, values);
    // Zapiranje povezave s podatkovno bazo
    db.close();
}
```

Primer 16 Vstavljanje v tabelo lokalne podatkovne baze



Slika 21 Logika nepovezanega načina

3.3.5 Uporabljene Android knjižnice

Pri sami izdelavi Android mobilne aplikacije smo uporabili več odprtokodnih knjižnic. Slednje bomo v nadaljevanju tudi opisali.

3.3.5.1 ZXing čitalec QR kod

ZXing je ena izmed najbolj razširjenih in optimiziranih odprtokodnih knjižnic za branje 1-D ali 2-D črtnih kod. Pri projektu je do sedaj sodelovalo že prek 130 programerjev, med njimi je veliko takšnih, ki so zaposleni prav pri Androidovem Googlu. Čitalec omogoča branje črtnih kod, ki jih prek spletne kamere zajamemo, knjižnica pa nato zajeto sliko analizira in razbere zapis na kodi (Primer 17) [30].

```
public boolean onCodeScanned(final String data)
{
        if (data != null) {
        if (data.equalsIgnoreCase(lastEmbeddedScannerScannedData))
        {
            return false;
        }
        else
        {
            displayScannedResult(data);
            lastEmbeddedScannerScannedData = data;
            lastEmbeddedScannerScannedDataTimestamp = System.currentTimeMillis();
            return true;
```

```
        }
    }
    return false;
}
```

Primer 17 Branje QR kode

3.3.5.2 Gson

V aplikaciji je knjižnica Gson največkrat uporabljena, saj se jo uporablja ob vsakem branju ali pošiljanju podatkov na strežnik. Gson skrbi za pretvorbo iz objektov v format JSON in obratno (Primer 18 in Primer 19). Knjižnica je bila sprva napisana za interno uporabo pri spletnem velikanu Google, od leta 2008 pa je brezplačno na voljo vsem Android razvijalcem [31].

```
Gson gson = new Gson();
Cow cow = new Cow("Micka", 200, "Cikasta");
Gson.toJson(cow);
```

Primer 18 Uporaba Gson razreda

```
{
    "name" : "Micka"
    "weight": 200
    "breed": "Cikasta"
}
```

Primer 19 Rezultat pretvorbe javanskega objekta v JSON notacijo

3.3.5.3 MPAndroidChart

Knjižnica omogoča izris skoraj vseh vrst grafov, od stolpčnih, črtnih, tortnih, mehurčnih pa do polarnih grafikonov. Za svoje delovanje potrebuje knjižnica API-level 8, poleg izrisa grafa pa omogoča tudi skaliranje po X in Y-osi, hkratno kombiniranje več grafov na eni maski, označevanje, shranjevanje v .txt ali .jpg datotekah, itd. [32]

3.3.5.4 Joda Time

Knjižnica predstavlja kvalitetnejšo zamenjavo s primarnimi javanskimi časovnimi in datumskimi razredi. Knjižnica ponuja ogromno metod, ki razvijalcu precej olajšajo delo (seštevanje, odštevanje, iskanje datumov,…) [33].

Nekaj glavnih razredov knjižnice:

- LocalDate,
- LocalTime,

- DateTime,
- Interval.

S pomočjo knjižnice na prototipu preverimo, čez koliko dni bo žival prešla iz karence (Primer 20).

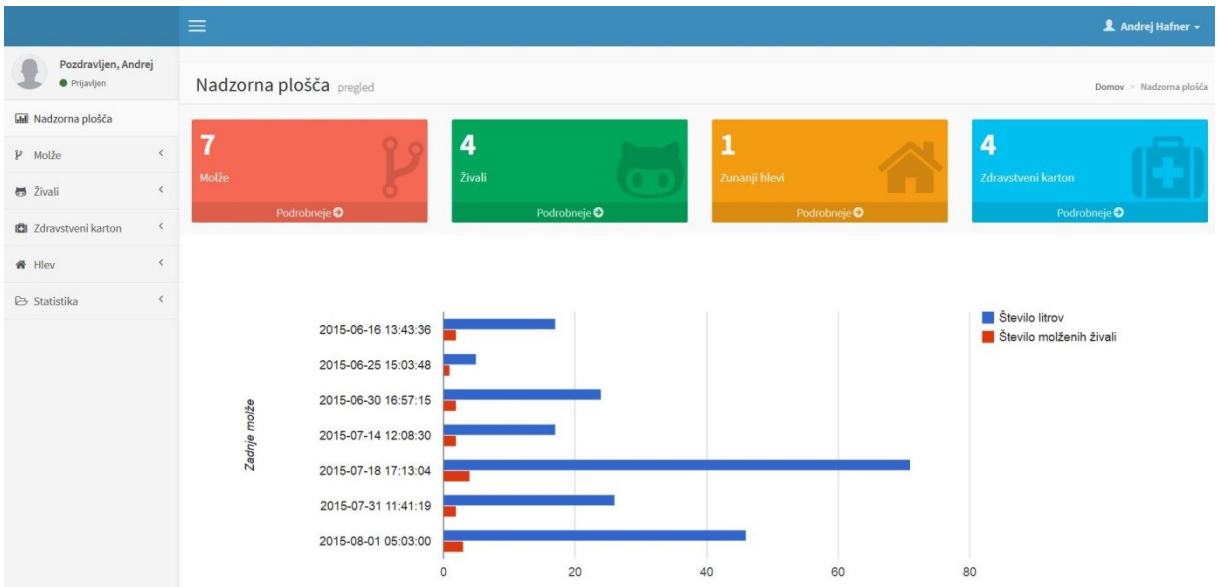
```
public Days daysToWithdrawal(LocalDate fromDate, LocalDate nowDate) {
    return Days.daysBetween(untilDate, nowDate);
}
```

Primer 20 Računanje števila dni med podanima datumoma

3.4 Spletna aplikacija

Po končani izdelavi Android mobilne aplikacije je na vrsto prišla spletna aplikacija (Slika 22). Razlog za razvoj dveh aplikacij je boljša preglednost in lažja uporaba, saj želimo, da je aplikacija na terenu čim enostavnejša in hitrejša za uporabo. Do sistem dostopamo prek spletnega brskalnika, ob uspešni avtentikaciji nam sistem poleg osnovnih akcij nudi dodajanje novih uporabnikov mobilne aplikacije, urejanje administrativnih nastavitev in tiskanje črtnih kod. Spletna aplikacija temelji na Bootstrapovem ogrodju, za komunikacijo s podatkovno bazo MySQL pa smo uporabljali programski jezik PHP.

Slednjega se je za asinhrono delovanje v ozadju kombiniralo z Javascriptom, kar nam omogoča uporabo Ajaxa. Z uporabo Ajaxa si lahko spletne aplikacije izmenjujejo podatke brez osveževanja strani, kar je v današnjem času predpogoj za dobro aplikacijo.



Slika 22 Začetni zaslon spletne aplikacije

3.4.1 Uporabniški vmesnik

Uporabniški vmesnik oziroma ogrodje spletne strani (angl. Framework) bazira na Bootstrapovi rešitvi. V paketu namreč dobimo HTML-ogrodje ter CSS in Javascript datoteke, ki skrbijo za podobo in interaktivnost elementov.

```
<div class="col-md-4 column">
          <div class="box box-solid">
                    <div class="box-header">
                              <i class="fa fa-fw fa-bar-chart-o"></i>
                              <h3 class="box-title">Ostali podatki</h3>
                    </div>
                    <div class="box-body">
                              <dl>
                                        <dt>Kraj</dt>
                                        <dd><?php echo $row["city"]; ?></dd>
                                        <dt>Naslov</dt>
                                        <dd><?php echo $row["street"]; ?></dd>
                                        <dt>Poštna številka</dt>
                                        <dd><?php echo $row["postal code"]; ?></dd>
                              </dl>
                    </div>
          </div>
</div>
```

Primer 21 HTML ogrodje

Za podobo objektov skrbijo CSS-pravila. Za prilagodljivo podobo (responsive) v CSS- datotekah preverimo, prek kakšne velikosti zaslona uporabnik dostopa do aplikacije. V spodnjem primeru za zaslone z minimalno širino 768px in maksimalno 991px prilagodimo prikaz nekaterim atributom (Primer 22).

```
@media (min-width: 768px) and (max-width: 991px) {
  .visible-sm {
    display: block !important;
  }
  table.visible-sm {
    display: table !important;
  }
  tr.visible-sm {
    display: table-row !important;
  }
  th.visible-sm,
  td.visible-sm {
    display: table-cell !important;
  }
}
```

Primer 22 Preverjanje velikosti zaslona

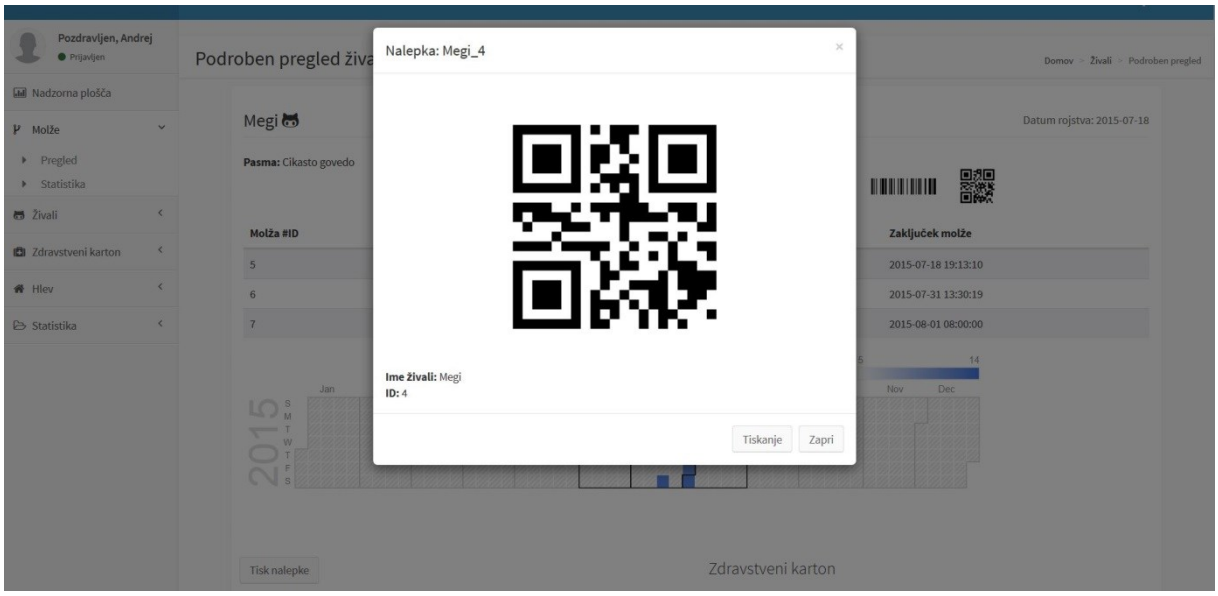
3.4.2 Tiskanje nalepk

Ker Android mobilna aplikacija za potrebe skeniranja potrebuje ustrezno generirane QR-kode, nam administrativni del to ponuja. Ob dodani novi živali ali pri pregledu obstoječe lahko le z enim klikom generiramo kodo, to pa pošljemo naprej na tiskalnik. Generiranje QR-kode poteka s pomočjo Googlove rešitve Google Chart API, ki prek parametrov v url-ju generira

PNG sliko (Slika 23) [34]. Android aplikacija kot vhod pričakuje tekst v obliki »ime_id«, za to pa poskrbi spodnja koda (Primer 23).

```
<img src="https://chart.googleapis.com/chart?chs=350x350&cht=qr&chl=<?php echo $name." ".$id;
?>%2F&choe=UTF-8" title="<?php echo $name; ?>" />
```

Primer 23 Generiranje QR kode



Slika 23 Generirana ustrezna QR koda

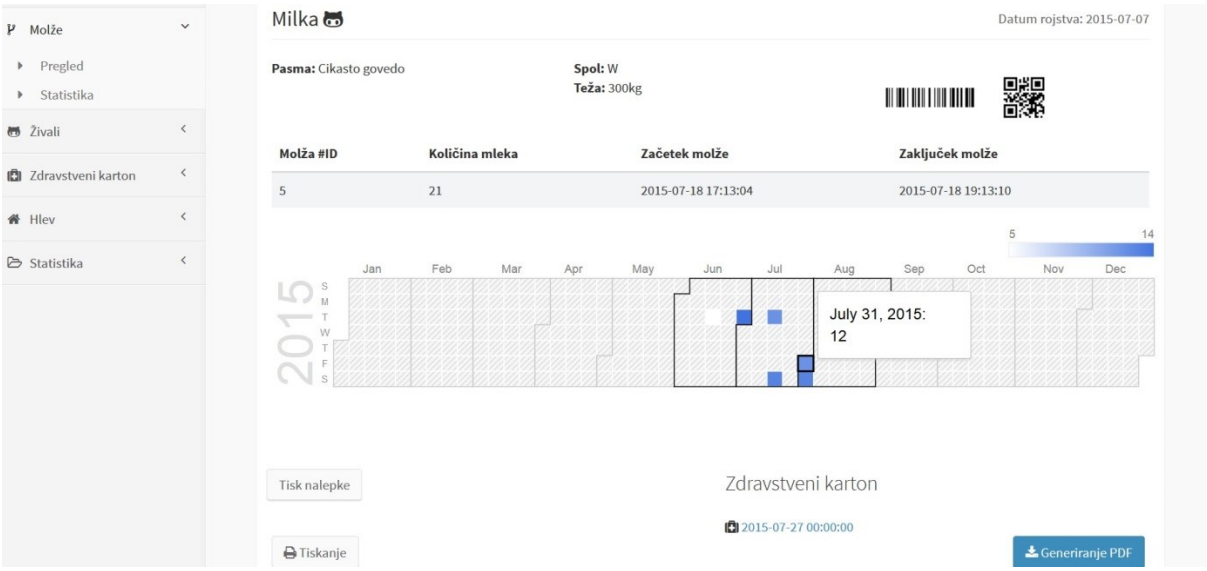
3.4.3 Prikaz podatkov v grafih

Za ustrezen prikaz podatkov v grafih uporabljamo Google Chart, ki za razliko od predhodnika (Google Chart API, ki ga uporabljamo za prikaz QR kode) grafa ne vrne v slikovni različici, ampak ga s pomočjo Javascripta izriše, to pa nam omogoča tudi interakcijo grafa (Slika 24). Za delovanje moramo na spletno aplikacijo vključiti AJAXOV API, nato pa prek Javascript spremenljivk določimo tip grafa in mu po želji dodajamo atribute in podatke za izris (Primer 24).

```
<script type="text/javascript">
          google.load('visualization', '1', {packages: ['corechart', 'bar']});
          google.setOnLoadCallback(drawBasic);

          function drawBasic() {

                    var data = google.visualization.arrayToDataTable([
                              ['Molža', 'Število litrov','Število molženih živali'],
                    ]);

                    var options = {
                              title: '',
                              height: 500,
                              chartArea: {width: '50%'},
                              hAxis: {
```

```
                    title: 'Skupaj litrov',
                    minValue: 0
                },
                vAxis: {
                    title: 'Zadnje molže'
                }
            };

            var chart = new
google.visualization.BarChart(document.getElementById('chart div'));

            chart.draw(data, options);
        }
</script>
```

Primer 24 Javascript klic za kreiranje grafa



Slika 24 Primer grafa molž

3.5 Spletna storitev

Spletna storitev (ang. Webservice) skrbi za komunikacijo med strežniškim delom in Android mobilno aplikacijo. Storitev deluje na istem strežniku kot naša spletna aplikacija, delovanje pa poteka na principu parametrov, ki jih prek mobilne aplikacije z metodo POST pošiljamo na spletni strežnik. Preko metode POST na strežniški strani razberemo zahtevek, ki ga nato v nadaljevanju uporabimo za klice v naprej napisanih funkcijah, ki vsebujejo SQL-poizvedbe (Primer 25).

```
public function getAllAnimals($id user){
        $return arr = array();
        $fetch = mysql_query("SELECT * FROM animals WHERE id_user='$id_user'") or
die(mysql_error());
        // check for result
        $no of rows = mysql num rows($fetch);
        while($row = mysql_fetch_array($fetch)) {

                $row_array['id'] = $row['id'];
```

```
                $row_array['name'] = $row['name'];
                $row_array['gender'] = $row['gender'];
                $row_array['weight'] = $row['weight'];
                $row_array['breed'] = $row['breed'];
                $row_array['date birth'] = $row['date birth'];

                array_push($return_arr, $row_array);
        }
        return $return_arr;
}
```

Primer 25 Vračanje vseh živali, ki pripadajo uporabniku

Strežnik nam odgovor vrne v JSON-formatu. V primeru napake se atribut »error« nastavi na »true«, vsebina odgovora pa vsebuje vrsto napake, prek katere lažje razberemo vzrok za negativen odgovor. V nasprotnem primeru vrnjen odgovor z GSON-om pretvorimo v javanske objekte, ki jih nato uporabljamo za nadaljnjo rabo.

3.6 Samodejno obveščanje – CRON sistemski klic

Vprašali smo se, kako delo kmetu čim bolj olajšati. V primeru večje kmetije je pregled nad zalogo krme, zdravjem živali in količino pridelanega mleka kljub sistemu lahko dolgotrajen, obstaja pa tudi možnost, da bi se kakšen podatek spregledal. Tako sistem vsako uro vse dni v letu preverja stanje vseh atributov in v primeru ujemajočih se pogojev na elektronski naslov uporabnika pošlje poročilo (Slika 25).

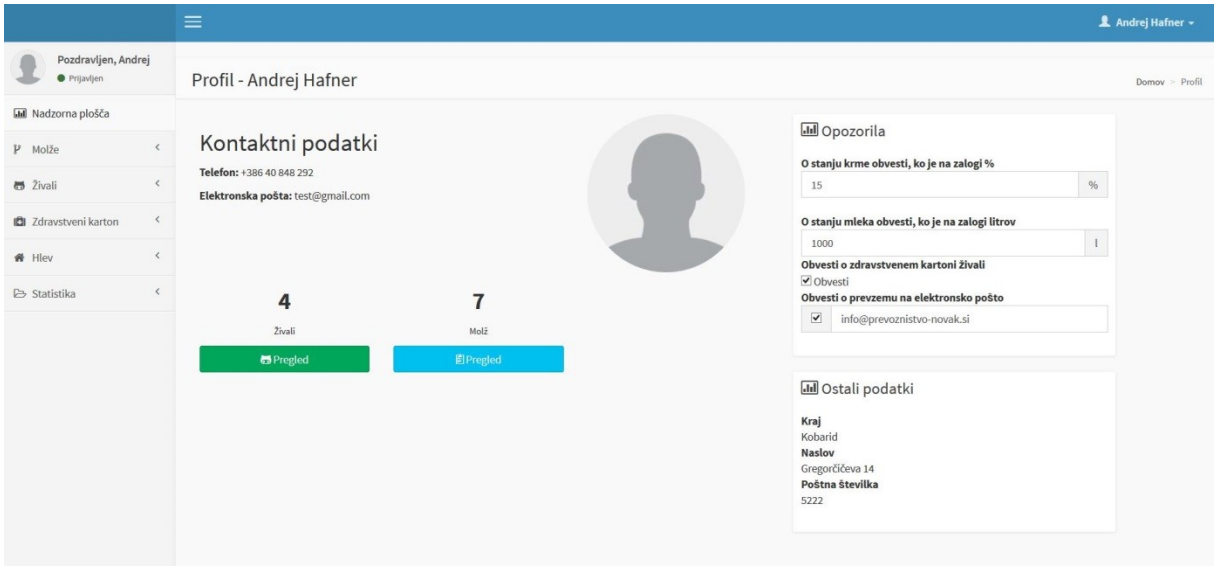

Imate novo sistemsko sporočilo:

Sporocilo: Prišlo je dosega določene kvote mleka (Trenutna zaloga: 950l | Kvota: 900l). Elektronska pošta je bila odposlana tudi prevozniku 'Prevozništvo Novak s.p.'

E-kmetija

Slika 25 Obvestilo preko elektronske pošte

Administrator ima na spletni aplikaciji možnost upravljanja z obvestili, kjer nastavi želene vrednosti, ki služijo kot robni pogoj za obvestilo. Uporabnik lahko določi procentualno vrednost minimalne zaloge krme, maksimalno količino litrov mleka in obvestila o zdravstvenih kartonih živali (Slika 26).



Slika 26 Administrativne nastavitve obvestil

Za konstantno preverjanje podatkov v podatkovni bazi skrbi Cron opravilo (ang. Cron job). To je sistemski proces, ki teče na Unix Sistemih [35]. Z ukazom na primeru (Primer 26), določimo, da se proces izvede vsako uro, vse dni v letu, podamo pa tudi pot do PHP-datoteke, ki opravi preverjanje in pošiljanje elektronske pošte prek SMTP-strežnika. Sistem omogoča pošiljanje obvestil tudi izbranemu prevozniku, ki je prek elektronske pošte obveščen, da je uporabnik dosegel kvoto za prevzem mleka.

```
0 * * * * /usr/bin/wget -O - nasa-domena.si/android__api/reminder.php> /dev/null 2>&1
```

Primer 26 CRON sistemski klic

Časovni ukaz vsebuje pet atributov, podrobneje opišimo strukturo.

* * * * *

* Predstavlja minute, možne vrednosti od 0-59

* Predstavlja ure, možne vrednosti od 0-23

* Predstavlja dan v mesecu, možne vrednosti 1-31

* Predstavlja mesec, možne vrednosti 1-12

* Predstavlja dan v tednu, možne vrednosti 0,6 (številka 0 predstavlja nedeljo, 6 pa dan sobote)

V primeru da vrednost pustimo na * pomeni, da se atribut izvede na vseh možnih vrednostih. [36]

Tabela 1 Nekaj osnovnih primerov CRON časovne strukture

CRON vrednost	**Opis**
0 0 1 1 *	Enkrat letno ob polnoči 1. januarja
0 0 1 * *	Enkrat mesečno prvega dne v mesecu
0 0 * * *	Enkrat dnevno ob polnoči

Poglavje 4 Sklepne ugotovitve

V diplomski nalogi smo razvili prototip spletne in Android aplikacije v povezavi s strežniškim delom, ki skrbi za hranjenje ter komunikacijo med podatkovno zbirko in aplikacijama. Cilj diplomskega dela je bilo razviti prototip aplikacije, ki bi predvsem manjšim kmetijam olajšala delo in prihranila čas pri beleženju statistik ter podatkov. Vsebina diplomske naloge predstavlja potek razvoja, od zasnove, strukture do končne izvedbe prototipa. V delu so predstavljeni tudi krajši izseki programske kode, ki so del prototipa.

Delovanje prototipa v testnem okolju se je izkazalo za zanesljivo, morebitne težave v produkcijskem okolju pa bi lahko nastale pri branju QR-kode, saj je zajem slike v temačnih prostorih nezanesljiv in dostikrat dolgotrajen.

Z iskanjem razvojnih rešitev ni bilo večjih problemov, saj je razvoj spletnih in mobilnih aplikacij v visokem porastu, zato je na spletu ogromno dokumentacije ter zapisov, ki so nam služili kot pomoč pri izdelavi prototipa.

Najzanimivejši del diplomske naloge mi je predstavljala izdelava podatkovnega modela in spletne storitve, saj sem bil pred tem že dobro seznanjen z razvojem spletnih ter Android aplikacij.

4.1 Možnosti za nadaljnji razvoj

Prototip je lahko deležen še mnogih izboljšav, ena najtežjih je povezava aplikacije z mehanskimi stroji, kot je molzni stroj, prek katerega bi prejemali podatke o pridobljenih litrih mleka.

Nekatere možnosti za nadaljnji razvoj:

- Povezava z molznim strojem,
- izboljšati podobo mobilne aplikacije,
- izdaja računov za opravljene storitve,
- sporočanje o stanju preko sms sporočil,
- povezava z računovodskim sistemom.

Literatura

[1] „Great achievements,“ [Elektronski]. Dosegljivo: http://www.greatachievements.org. [Dostop 2015].

[2] „JSON notacija,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/JSON. [Dostop 2015].

[3] „JSON notacija,“ [Elektronski]. Dosegljivo: http://www.json.org/xml.html. [Dostop 2015].

[4] „PHP,“ [Elektronski]. Dosegljivo: https://sl.wikipedia.org/wiki/PHP. [Dostop 2015].

[5] „Javascript,“ [Elektronski]. Dosegljivo: https://sl.wikipedia.org/wiki/JavaScript. [Dostop 2015].

[6] „JQuery,“ [Elektronski]. Dosegljivo: http://api.jquery.com/. [Dostop 2015].

[7] „Java,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/Java_%28programming_language%29. [Dostop 2015].

[8] „Tiobe Programming Community,“ 2015. [Elektronski]. Dosegljivo: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. [Dostop 2015].

[9] „CSS,“ [Elektronski]. Dosegljivo: https://sl.wikipedia.org/wiki/CSS. [Dostop 2015].

[10] „Bootstrap,“ [Elektronski]. Dosegljivo: http://getbootstrap.com/. [Dostop 2015].

[11] „Bootstrap,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/Bootstrap_%28front-end_framework%29. [Dostop 2015].

[12] „MySQL,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/MySQL. [Dostop 2015].

[13] „SQLite,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/SQLite. [Dostop 2015].

[14] „Android Studio,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/Android_Studio. [Dostop 2015].

[15] „Mobile operating system market share,“ [Elektronski]. Dosegljivo: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1. [Dostop 2015].

[16] „Android operating system,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/Android_%28operating_system%29. [Dostop 2015].

[17] „Apple developer,“ [Elektronski]. Dosegljivo: https://developer.apple.com/support/compare-memberships/. [Dostop 2015].

[18] „Google Play,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/Google_Play. [Dostop 2015].

[19] „ Sam svoj programer“. [Elektronski]. Dosegljivo: http://www.monitor.si/clanek/sam-svoj-programer3/125026/. [Dostop 2015].

[20] „Android,“ [Elektronski]. Dosegljivo: http://android.fri.uni-lj.si/index.php/Platforma. [Dostop 2015].

[21] „Android,“ [Elektronski]. Dosegljivo: http://developer.android.com/intl/zh-CN/about/dashboards/index.html. [Dostop 2015].

[22] „DBDesigner,“ [Elektronski]. Dosegljivo:

http://www.fabforce.net/dbdesigner4/. [Dostop 2015].

[23] „PhpMyAdmin,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/PhpMyAdmin. [Dostop 2015].

[24] „Uporabniški vmesnik,“ [Elektronski]. Dosegljivo: https://sl.wikipedia.org/wiki/Grafi%C4%8Dni_uporabni%C5%A1ki_vmesnik. [Dostop 2015].

[25] Android, [Elektronski]. Dosegljivo: http://developer.android.com/guide/components/fragments.html. [Dostop 2015].

[26] „SQLite,“ 2015. [Elektronski]. Dosegljivo: http://developer.android.com/intl/zh-CN/reference/android/database/sqlite/SQLiteOpenHelper.html. [Dostop 2015].

[27] „Android PDF document“. [Elektronski] Dosegljivo: https://developer.android.com/reference/android/graphics/pdf/PdfDocument.html. [Dostop 2015].

[28] „Android PDF document graphics“ [Elektronski]. Dosegljivo: https://developer.android.com/reference/android/graphics/pdf/package-summary.html. [Dostop 2015].

[29] „Android API,“ [Elektronski]. Dosegljivo: https://developers.google.com/maps/documentation/android-api/. [Dostop 2015].

[30] „ZXing,“. [Elektronski] Dosegljivo: https://github.com/zxing/zxing [Dostop 2015].

[31] „Gson,“. [Elektronski]. Dosegljivo: https://github.com/google/gson. [Dostop 2015].

[32] „MPAndroidChart,“ [Elektronski]. Dosegljivo: https://github.com/PhilJay/MPAndroidChart. [Dostop 2015].

[33] „Joda Time,“ [Elektronski]. Dosegljivo: http://joda-time.sourceforge.net. [Dostop 2015].

[34] „Google Chart API,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/Google_Chart_API. [Dostop 2015].

[35] „Cron Jobs,“ [Elektronski]. Dosegljivo: https://en.wikipedia.org/wiki/Cron. [Dostop 2015].

[36] S. Moritsugu, Practical UNIX,, 2000.