Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Kovač

Aplikacija za upravljanje goveje črede

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ržiše, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Problem manjših slovenskih kmetij, ki se ukvarjajo z rejo govejih pitancev, je iz leta v leto težja sledljivost poteka bivanja živali na kmetiji. Posledično je ob prodaji živali skoraj nemogoče ugotoviti, ali nam je prodaja prinesla dobiček ali pa smo že dolgo pred tem začeli delati izgubo. Sledljivost od samega rojstva oziroma nakupa živali do prodaje vključuje vse veterinarske posege, vmesne preglede, opažanja, nakupe prehrambenih dodatkov in v končni fazi pripravo glavnega krmnega obroka za vsak dan. Kot rezultat naloge se pričakuje izdelava namizne aplikacije, s pomočjo katere bo na enem mestu mogoče spremljati in voditi vsa opažanja, dogodke, stroške in prihodke za vsako posamezno žival. Poleg tega naj aplikacija ponuja tudi pregled in prikaz trenutnih parametrov delovanja kmetije ter omogoča namestitev na operacijski sisteme Windows, Linux in macOS. Vsi podatki naj se shranjujejo v podatkovno bazo, dostopno v oblaku.

Zahvaljujem se vsem najbližjim, ki so odprto sprejeli mojo željo po študiju računalništva. Prav tako hvala dekletu ter prijateljem, ki so mi čas med študijem še dodatno popestrili.

Kazalo

Seznam uporabljenih kratic

DA	Desktop application	namizna aplikacija
REST	REpresentational State Transfer	način za izmenjavo podatkov v računalniških sistemih
JSON	JavaScript Object Notation	tekstovna notacija za oblikovanje JS objektov
SPA	Single Page Application	enozaslonska aplikacija
HTML	Hyper Text Markup Language	označevalni jezik za izdelavo spletnih vsebin
CSS	Cascading Style Sheets	slogovne predloge za oblikovanje spletnih vsebin
JS	JavaScript	programski jezik za ustvarjanje dinamičnih spletnih vsebin
URL	Uniform Resource Locator	enotni naslov vira
API	Application Programming Interface	aplikacijski programski vmesnik
CLI	Command-line interface	vmesnik z ukazno vrstico
NPM	Node Package Manager	upravitelj paketov Node
XML	Extensible Markup Language	razširljiv označevalni jezik
AJAX	Asynchronous JavaScript and XML	asinhrona povezava povezava jezikov Javascript in XML

Povzetek

Naslov: Aplikacija za upravljanje goveje črede

Avtor: Anže Kovač

Kmetje, ki se ukvarjajo z rejo govejih pitancev, morajo skozi celotno vzrejo živali paziti na stroške, možne simptome ali ponavljajoče se dogodke, ki bi lahko vodili k neoptimalnemu razvoju živali, kar bi posledično pomenilo manjšo prodajno ceno, od katere je kmet neposredno odvisen. Ker trenutno na trgu ne obstaja aplikacija, ki bi omogočala vodenje in upravljanje črede govejih pitancev, je namen te diplomske naloge ravno to: razviti aplikacijo, ki bo kmetu ob vsakem trenutku omogočala pregled nad podatki za celotno čredo, kot so kategorizacija živali glede na spol, pasmo in starost. Aplikacija naj bi omogočila tudi beleženje dogodkov, aktivnosti, dokumentov ter stroškov in prihodkov za vsako posamezno žival. Poleg tega nudi še možnost izpisa in prikaza trenutnih parametrov kmetije v grafični podobi za lažji dostop do najpogosteje iskanih informacij. Glavni cilj te naloge je, da se aplikacija uporabi tudi v realnem svetu in se tako z njeno pomočjo kmetom omogoči uspešnejše poslovanje.

Ključne besede: namizna aplikacija, govedoreja, beleženje stroškov, analiza, manjše kmetije.

Abstract

Title: Application for herd management

Author: Anže Kovač

The main goal of cattle breeders is optimal development of all their animals and optimization of the breeding process to increase the profit and reduce the overall costs of breeding. An important step for reaching this goal is to track every single animal, carefully analyze its development, measure its health, keep track of any unexpected costs and log recurring events. Failure to carefully monitor livestock may lead to suboptimal development of the animals, which consequently drives down the profit, on which the farmer directly depends. Since there is no software on the market that would meet the needs of logging, analysis and monitoring cattle herds and associated costs, the purpose of this bachelor's thesis was to develop a desktop application that will at any given time allow farmers to view and analyze data for their entire herd, such as categorization by gender, breed and age. Moreover, it is to enable farmers to track any events, activity, documents, costs and income for each animal, as well as graphically displaying up-to-date parameters related to the breeding process. The main objective of this thesis is to offer an application that can be used in real life, helping farmers monitor the livestock breeding process and more accurately predict costs, thus increasing their chances of success.

Keywords: desktop application, cattle breeding, cost tracking, analysis, smaller farms.

Poglavje 1

Uvod

V diplomski nalogi smo se posvetili izdelavi namizne aplikacije za upravljanje in nadzor goveje črede, ki bo kmetom omogoča vnašanje najrazličnejših aktivnosti, dogodkov, dokumentov ter stroškov in prihodkov, povezanih z vsako živaljo. Med samo implementacijo smo izdelali tudi sistem partnerjev, kjer lahko na enem mestu vidimo vse osebe oziroma družbe, s katerimi naša kmetija posluje. Prav tako smo omogočili dodajanje akcij posameznemu partnerju. V prvem delu diplomske naloge bomo opisali problem sledljivosti podatkov in se dotaknili procesa vzreje živali od nakupa oziroma rojstva do njene prodaje. Prav tako bomo opisali uporabljeno tehnologijo za izdelavo aplikacije.

1.1 Slovenske kmetije

Povprečna slovenska kmetija ima 6,8 ha [4] in se pretežno ukvarja z rejo govejih pitancev. Natančneje povprečna kmetija redi 5,6 glav velike živine (GVŽ) [9]. Problem takšne reje je, da kljub majhnemu številu živali in posledično majhnemu letnemu prihodku kmet za pripravo obrokov za zimsko obdobje potrebuje podobne stroje, infrastrukturo in število ljudi kot kmetije, kjer redijo 4- ali 5-krat večje število živali. Veliko takšnih kmetij se preživlja od stranskega zaslužka, ki običajno izhaja iz redne 8-urne službe. Razlogov,

zakaj še vedno vztrajajo pri reji pitancev, je več, največkrat slišani razlog pa je, da si je skoraj nemogoče zabeležiti in zapisati vse stroške in dogodke, ki so se zgodili skozi dve leti reje živali, ter posledično izračunati končno dobičkonosnost takšne dejavnosti.

1.2 Pregled področja

Na tem področju nisem našel podobne aplikacije, predvsem pa ne takšne, ki bi podpirala in spremljala razvoj govejih pitancev. Sistemi, ki se počasi začenjajo pojavljati, so predvsem osredotočeni na kmetije s prirejo mleka. Tudi v takšnih primerih gre večinoma za pametne ovratnice, ki spremljajo gibanje in nekatere osnovne parametre živali, niso pa to aplikacije za vodenje administracijskega dela vzreje. Programska rešitev, ki jo je moč zaslediti pri nas, Phanteon Farming, je primerna predvsem za večje kmetije in vsebuje veliko funkcionalnosti, ki jih manjši kmetje ne potrebujejo.

1.3 Struktura diplomske naloge

Diplomsko nalogo pričenjamo s pregledom področja, problema in predlagane rešitve. V nadaljevanju so opisane uporabljene tehnologije, nato pa še način, na katerega nam te tehnologije pomagajo zgraditi moderno namizno aplikacijo, ki sledi najnovejšim trendom razvoja takšnih aplikacij. V zadnjem delu je opisano delovanje aplikacije in kako vsak posamezni del pomaga razrešiti zadani problem.

Poglavje 2

Pregled problema in rešitve

2.1 Pregled problema

Vzreja govejega pitanca običajno traja do dve leti in se začne s skotitvijo, za katero so potrebna predhodna opazovanja in priprave, oziroma z nakupom teleta na kmetiji, kjer ga trenutno ne morejo preskrbeti in ga zato prodajajo. Največje tveganje se pojavlja prav v tem obdobju, ko so živali še zelo občutljive. Posledično velikokrat pride do zdravstvenih težav, ki zahtevajo budno spremljanje in včasih tudi posredovanje veterinarske službe. Vsi takšni posegi imajo velik vpliv na rast in razvoj živali, navsezadnje pa takšno zdravljene predstavlja tudi precejšen strošek. Čas intenzivnega hranjenja in razvoja poteka od tretjega meseca do drugega leta starosti. V vmesnem času so tveganja manjša. Še vedno pa prihaja do občasnih bolezni ali poškodb. Zgodi se, da ob končni prodaji živali teža in razvitost nista na želeni ravni. Na kmetiji, ki redi 40 živali, ki se izmenjajo na vsaki dve leti, si je skoraj nemogoče zapomniti vse posege, dogodke in ostale dejavnike, zato se nam je porodila ideja o izdelavi sistema, ki bi beležil podatke za vsako posamezno žival.

2.2 VOLOS

Prvotna ideja aplikacije je bila integracija s sistemom VOLOS, ki omogoča beleženje in vnašanje zakonsko predpisanih premikov živali ter branje podatkov klavnic ali nadaljnjih rejcev brez posredovanja uporabnika. Problem, na katerega smo naleteli, je bil odnos Ministrstva za kmetijstvo, gozdarstvo in prehrano, kjer so sprva obljubili aplikacijski dostop do sistema, med samo implementacijo pa so dostop zavrnili z razlogom zakonske prepovedi širjenja osebnih podatkov in nepotrebne varnostne ranljivosti zaradi odpiranja sistema javnosti.

2.3 Rešitev

Rešitev, za katero smo nato pripravili podatkovni model in povzetek zahtev, zajema možnost vnosa kmetije in podatkov zanjo, ljudi, ki na kmetiji pomagajo, beleženje poslovnih partnerjev in tretjih oseb ter beleženje interakcij med njimi in izbrano kmetijo. Ob tem pa nudi tudi možnost vnosa živali, urejanja in priprave podatkov ter beleženja vseh stroškov in dogodkov med dvoletno rastjo. Pri živalih ženskega spola je podprta možnost načrtovanja brejosti in vnosa podatkov o novorojenih teletih. Da bi bila aplikacija uporabniku karseda prijazna, smo vse najpomembnejše in najpogosteje iskane podatke želeli prikazati na domači strani s pomočjo grafičnih ponazoritev.

Poglavje 3

Tehnologije in programska oprema

Pri sami izdelavi aplikacije smo uporabili najnovejšo tehnologijo s področja ogrodij za izdelavo namiznih aplikacij JavaScript ter Python. Uporabljeni programski sklad izhaja iz sklada MEAN, kjer je za podatkovno bazo uporabljen PostgreSQL, kot ogrodje uporabniškega vmesnika pa Vue.js.

3.1 HTML5

HTML5 je najnovejša verzija označevalnega jezika HTML, ki je izšla oktobra 2014. HTML je nadgradnja označevalnega jezika XML, ki omogoča prikaz in oblikovanje multimedijskih vsebin. Osnovni HTML je služil kot temelj pri razvoju nadaljnjih jezikov HTML in se še vedno uporablja za osnovno arhitekturo dokumentov, s pomočjo katerih definiramo metapodatke, glavo, vsebino in nogo spletne strani. HTML5 celotno osnovo nadgradi z možnostjo vključevanja novih medijskih oblik, kot sta zvok in video, brez potrebe po uporabi dodatnih knjižnic, ki bi podatke interpretirale in jih na prijazen način prikazale uporabniku. Poleg možnosti vključevanja novih multimedijskih oblik HTML5 omogoča tudi ustvarjanje semantičnega pomena strani, ki ga določimo z uporabo značk. Tako kot njegovi predhodniki je HTML5 na-

mreč sestavljen iz značk, ki jih brskalniki in odjemalci ne prikazujejo, temveč definirajo obliko in prikaz vsebine [7].

3.2 CSS3

CSS so kaskadne slogovne predloge, ki so preko selektorjev povezane z značkami HTML in dodatno definirajo obliko posameznih elementov HTML. Osnovni CSS nam omogoča preprostejše oblikovanje elementov, kot na primer odebeljevanje besedila, barvanje ozadja ali definiranje odmika od robov. CSS3 pa vsem tem funkcionalnostim doda predvsem veliko možnosti za upravljanje z večpredstavnostnimi vsebinami. Osnovni CSS omogoča vključitev slike, nove funkcionalnosti v CSS3 pa omogočajo, da tej sliki določimo tudi prosojno procentualno prehajanje ozadij. V sklopu spletnih strani se najbolj uporabljajo funkcionalnosti animacije, zaobljenosti robov, podpora senc, možnost naprednejšega definiranja barv in pa naprednejši sistem za postavitev strani, kot je recimo sistem mreže.

3.3 JS

JavaScript je skriptni programski jezik, ki v zadnjem času dosega vse večjo popularnost in uporabo. V osnovi je bil namenjen izdelavi dinamičnih spletnih strani. V področje spletnih strani je s tem jezikom prišel tudi AJAX, ki definira dinamično osveževanje le določenega dela strani brez potrebe po osveževanju celotne strani. Danes se JavaScript primarno uporablja kot programski jezik, s katerim so zgrajena največja ogrodja svetovno znanih podjetij, kot sta Googlov AngularJS in Facebookov React. Vse več pa se JavaScript kot jezik pojavlja tudi v zalednih strežniških sistemih, kjer so do nedavnega kraljevali jeziki, kot so C, Java, PHP. Veliko popularnost JavaScript dosega tudi zaradi možnosti uporabe na mobilnih napravah, kjer se vse pogosteje uporablja koncept hibridnih mobilnih aplikacij, s pomočjo katerih moramo sprogramirati le eno verzijo aplikacije, ki nato deluje tako na sistemu Android

kot na sistemu iOS[2].

3.4 Node.js

Node.js je pogon JavaScript, s pomočjo katerega lahko podporo za JavaScript uporabljamo tudi izven brskalnikov. Temelji na pogonu V8 JavaScript, ki ga Google razvija in uporablja kot primarni pogon v brskalniku Google Chrome. Uporabnost pogona Node.js se je pokazala predvsem pri izgradnji strežnikov, kjer lahko z uporabo JavaScripta pišemo programe, ki dostopajo tudi do sistemskih virov, kar je znotraj brskalnika onemogočeno. Uporablja se pri izdelavi večine novejših spletnih aplikacij. Na njem temelji tudi orodje Electron, ki je opisano v nadaljevanju. Najbolj uporabna funkcionalnost sistema Node.js pa je upravitelj paketov NPM, kjer so shranjene vse dodatne razširitvene knjižnice. Namestitev takšne knjižnice izvedemo s preprostim ukazom »npm install vue«. Sistem NPM v ozadju samodejno generira sistem odvisnosti ter preveri, katere knjižnice na računalniku manjkajo, in jih samodejno namesti [10].

3.5 Vue.js

Vue.js je odprtokodno progresivno ogrodje JavaScript za izdelavo modernih spletnih aplikacij. Ustvarjen je bil z namenom poenostavitve in organizacije razvoja spletnih strani. V osnovi gre za ogrodje z dvosmernim referenciranjem podatkov, kar pomeni, da ogrodje za vsak podatek hrani tudi vse reference, kjer je to polje uporabljeno. V primeru posodobitve podatka na katerikoli strani se podatek samodejno osveži. Osveževanje komponent, ki so se spremenile, se dogaja 60-krat na sekundo, s čimer dosežemo gladke, za uporabnika nemoteče animacije. Glavna prednost Vue.js pred ostalimi ogrodji je njegova majhnost in število odvisnih paketov, s čimer dosežemo veliko hitrost in učinkovitost delovanja. Med ostale pomembne sestavne dele ogrodja štejejo še:

- predloge, ki jih definiramo enkrat in potem dinamično polnimo s podatki konteksta,
- sistem komponent, ki omogoča pakiranje funkcionalnosti na manjše, ponovno uporabne komponente,
- sistem prehodov in animacij, ki temelji na sistemu osveževanja 60-krat na sekundo,
- usmerjanje, ki je ključni sestavni del enozaslonskih aplikacij in omogoča izris vsebine v dokumentu glede na trenutno pot usmerjevalnika.

3.6 Electron

Electron je odprtokodno ogrodje za izdelavo namiznih aplikacij, ki ga je za potrebe izdelave urejevalnika Atom razvil GitHub, pozneje pa je iz tega naredil odprtokoden projekt, ki je dosegljiv na platformi GitHub. Glavna značilnost tega ogrodja je, da za svoje delovanje uporablja Node.js, ki na katerikoli napravi ustvari pogon JavaScript, v katerem se lahko JavaScript izvaja enako kot v brskalniku. Za razvijalca to v prvi vrsti pomeni, da je potrebno sprogramirati le eno aplikacijo, ki deluje na operacijskih sistemih Windows, Linux in macOS. Druga ključna prednost pa je, da vedno točno vemo, katero verzijo pogona V8 uporabljamo. Problem pri različnih verzijah pogona V8 je namreč, da obstaja možnost, da na brskalnikih, ki še niso bili posodobljeni, nekatere funkcionalnosti ne bodo delovale. Ogrodje v osnovi že ponuja dostop do vseh sistemskih virov, kot so datoteke, upravljanje zvoka, zajemanje zaslonskih mask in še mnogo več. Poleg samega projekta pa smo uporabili še modul simple-electron za vzpostavitev začetnega projekta.

3.7 ESLint

Je odprtokodni projekt, ki ga je junija 2013 začel Nicolas C. Zakas . Ideja modula je, da preko vnaprej definiranih pravil skrbi, da je izgled naše kode enak

v vseh komponentah. Problem, ki se pojavi s tako množično uporabo JavaScripta, je veliko različnih stilov pisanja kode, kar pa lahko vodi do neurejene in nepregledne kode. ESLint tako omogoča, da vnaprej določimo pravila, kot so število presledkov, dovoljenih znakov, načini pisanja oklepajev in novih vrstic. Modul se v praksi izkaže za zelo uporabnega, saj razvijalec v kodi ne more pustiti svojega podpisa. Prav tako modulu uporabnost povečajo vtičniki za razvojna orodja, ki pravila ESLinta preverjajo že sproti med pisanjem kode in nas opozarjajo na nepravilnosti oziroma možne izboljšave.

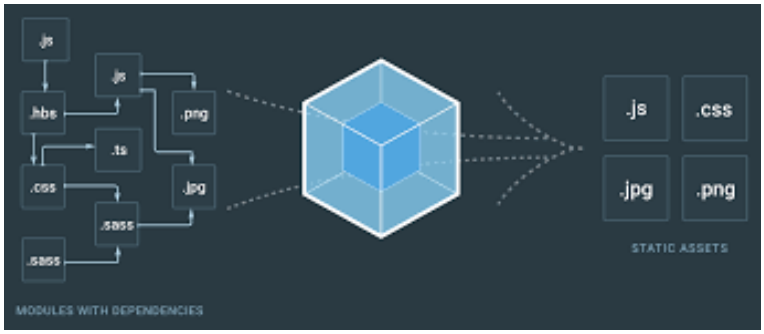
3.8 Bootrstrap4

Bootstrap je prvo ogrodje, ki združuje funkcionalnosti HTML5, CSS3 in JavaScripta za izgradnjo strani, ki se prilagajajo glede na spreminjanje velikosti okna. Prav tako so vse komponente pripravljene tako, da se primerno prikažejo tudi na mobilnih napravah. V osnovi je Bootstrap uvedel revolucionaren sistem mreže komponent HTML, ki skrbi za lažje določanje širine, velikosti in postavitve elementov na spletni strani. Skozi razvoj ogrodja se je Bootstrapu dodalo veliko število vnaprej definiranih komponent, kot so recimo gumbi ali vnosna polja z vnaprej določeno obliko in slogom.

Z uporabo ogrodja Bootstrap lahko oblikujemo spletno stran, ne da bi pred tem kreirali kakršenkoli slog CSS.

3.9 Webpack

Webpack je prav tako odprtokodno orodje, ki je v osnovi namenjeno združevanju najrazličnejših datotek v arhitekturi našega projekta v JavaScriptu. Z najrazličnejšimi odvisnostmi, do katerih pridemo zaradi uporabe sistema paketov NPM, lahko naš projekt zelo hitro postane zbirališče najrazličnejših datotek različnih vrst. Glavni problem, ki ga to orodje rešuje, je končna priprava aplikacije za testiranje in kasnejšo predajo v uporabo. Kot primer lahko vzamemo spletno aplikacijo, ki ima vključen modul za validacijo vnosnih polj.



Slika 3.1: Ponazoritev prikaza delovanja Webpack orodja

Ta modul za svoje delovanje potrebuje še modul za preverjanje tipov vrednosti. Za pravilno delovanje aplikacije zato NPM shrani oba modula. Problem pa nastane pri prenosljivosti aplikacije, kjer se moramo ukvarjati z velikim številom datotek, sploh če uporabljamo več modulov. Webpack na podlagi vsebine datoteke združi v po eno za vsak tip. Bistvena prednost je ta, da se po uporabi Webpacka ukvarjamo samo s približno desetimi datotekami, pred uporabo pa je lahko teh datotek tudi nekaj sto [11].

3.10 Babel

Ker se specifikacije okolja JavaScript vsako leto dopolnjujejo z novimi funkcionalnostmi in obliko pisanja kode, morajo avtorji spletnih brskalnikov te spremembe redno umeščati v svoje pogone JavaScript. Zato se velikokrat zgodi, da zadnje verzije različnih brskalnikov ponujajo različno podporo za različne dele specifikacije. To predstavlja oviro za razvijalce, ki morajo skrbeti, da uporabljajo samo funkcionalnosti, ki so na voljo v vseh brskalnikih. Babel pa je orodje za pretvorbo JavaScripta, ki lahko uporablja vse specificirane funkcionalnosti njegove trenutne verzije in jih pretvori v starejšo obliko, oziroma za nekatere funkcionalnosti uporabi knjižnice, ki nadomestijo manjkajočo funkcionalnost. S tem si pri razvoju aplikacij omogočimo, da lahko uporabimo najnovejše zmožnosti jezika JavaScript neodvisno od tega, v ka-

terem brskalniku se bo potem ta koda izvajala. Ker bo Babel poskrbel za primerno pretvorbo v starejšo verzijo, ki jo podpira večina brskalnikov, bodo vse funkcionalnosti, uporabljene v programski kodi, vedno delovale [1].

3.11 Git

Git je trenutno najbolj uporabljan in poznan odprtokodni sistem za kontrolo verzij. Gre za distribuiran sistem, pri katerem je poudarek na hitrosti, podpori nelinearnim tokovom razvoja programske opreme in integriteti podatkov. Najpogosteje se sistem Git danes uporablja za kontroliranje trenutnih verzij programske kode [3]. Ker Git kot sistem sam po sebi ne omogoča deljenja kode z drugimi uporabniki, ampak je zasnovan kot lokalni sistem, potrebujemo spletno ali kakršnokoli drugo gostovanje, ki omogoča uporabo, pregledovanje, urejanje in administracijo ter sinhronizacijo. Med najbolj znanimi ponudniki takšnih rešitev so Github[1],Bitbucket[2] ter GitLab[3].

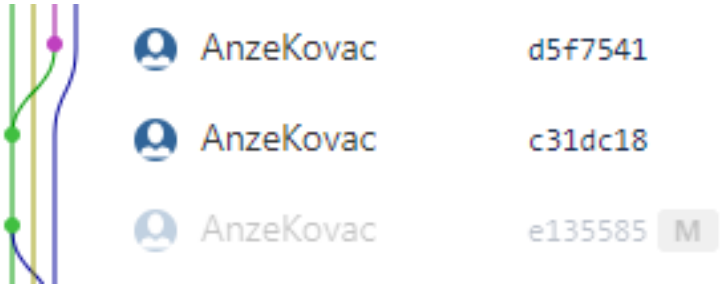
3.11.1 Uporaba

Za samo uporabo Gita se je skozi čas razvilo več slogov. Najpogosteje uporabljeni med njimi se imenuje Git-Flow in temelji na uporabi dveh glavnih vej:

- produkcijska veja
- razvijalska veja

Na glavno vejo pride le verzija kode, ki je bila skozi razvojni proces potrjena in pregledana s strani osebe, zadolžene za določen del funkcionalnosti, ali osebe, katere glavni cilj je zagotavljanje kakovosti. Poleg dveh glavnih vej pa se lahko v procesu kreirajo še manjše veje, na katerih se razvijajo posamezne funkcionalnosti ali popravki kode. S tem dosežemo, da lahko več

[1] `https://github.com`
[2] `https://bitbucket.org`
[3] `https://about.gitlab.com`



Slika 3.2: Primer razvejevanja

ljudi hkrati dela na istem izvoru kode, potem pa združijo nove funkcionalnosti nazaj v enotno verzijo. Skozi celoten proces imamo pregled nad tem, kaj točno se je v datotekah spremenilo. Prav tako lahko vidimo opombe oziroma komentarje, ki jih je razvijalec vnesel ob kreiranju nove verzije kode v sistemu. Za uporabo Gita lahko izbiramo med ukazno vrstico, kjer celoten proces vodimo skozi vnaprej definirane ukaze, ali pa uporabimo orodje z grafičnim vmesnikom, ki nam ponuja vizualni pregled nad procesom.

3.12 Heroku

Heroku je podjetje pod okriljem Salesforcea, ki ponuja svojo platformo kot storitev (angl. platform as a service). Nudi nam možnost gostovanja aplikacijskih strežnikov, napredno analitiko prometa in pregledovanje parametrov delovanja. Poleg tega omogoča tudi gostovanje podatkovne baze. Privzeta podatkovna baza je PostgreSQL, ki je obenem tudi najbolj optimiziran za delovanje na platformi. Heroku brezplačnim uporabnikom omogoča 1000 ur izvajanja na aplikacijskih strežnikih na deljeni arhitekturi [6].

3.12.1 Konfiguracija

Sama konfiguracija aplikacijskega strežnika se izvede v dveh delih. Začne se z izbiro predkonfiguriranega okolja za izvajanje kode, na primer Python. Po izbiri okolja storitev omogoča, da sistem povežemo s svojo kontrolo verzij. Ob

vsaki spremembi, ki se zgodi v kontroli verzij, se aplikacijski strežnik zgradi na novo in začne ponujati najnovejšo verzijo naše programske kode. Med datotekami v kontroli verzij se mora nahajati tudi konfiguracijska datoteka, ki jo Heroku prebere in njej primerno konfigurira okolje za zagon kode.

3.12.2 Dodatne funkcionalnosti

Platforma pa poleg gostovanja aplikacijskega strežnika in podatkovne baze omogoča tudi časovno definirano izvajanje akcij, s čimer lahko ob želenih urah sprožimo določene akcije v naši aplikaciji. Prav tako platforma omogoča naprednejše konfiguriranje procesa izgradnje aplikacije iz konfiguracijske datoteke. Vseskozi so nam na voljo tudi vtičniki, ki jih razvijajo člani skupnosti Heroku.

3.13 Python

Python je interpretacijski, visokonivojski in objektno orientiran programski jezik, ki svojo razširjenost dosega v prvi vrsti z zelo preprostim semantičnim slogom pisanja programske kode. Zato se Python uporablja za veliko različnih namenov, med drugim za:

- razvoj spletnih aplikacij in zalednih sistemov, zaradi česar smo ga uporabili tudi v tej diplomski nalogi,
- znanost in podatkovno rudarjenje,
- izobraževalne namene, predvsem poučevanje programiranja,
- izdelavo uporabniških vmesnikov.

Obstajata dve verziji Pythona, ki med seboj nista kompatibilni – predvsem zaradi sintaktičnih razlik. V nalogi smo uporabili verzijo 3.6 skupaj z orodjem za kreiranje virtualnih okolij virtualenv, ki zadržuje vse naknadne namestitve paketov in knjižnic znotraj posameznega projekta in jih ne namešča globalno na naš računalnik.

3.14 Flask

Flask je mikro spletno ogrodje, namenjeno za uporabo s programskim jezikom Python, ki omogoča izdelavo celotnih spletnih aplikacij. Največkrat se v uporabi pojavi kot aplikacijski strežnik REST. Za svoje delovanje in obdelavo spletnih zahtevkov privzeto uporablja vgrajeni strežnik Werkzeug. V našem primeru je bil za zagon na platformi Heroku uporabljen modul Gunicorn, ki se od Werkzeuga razlikuje predvsem po karakteristikah delovanja, pa tudi po varnostnih funkcionalnostih [5].

Poglavje 4

Zasnova aplikacije

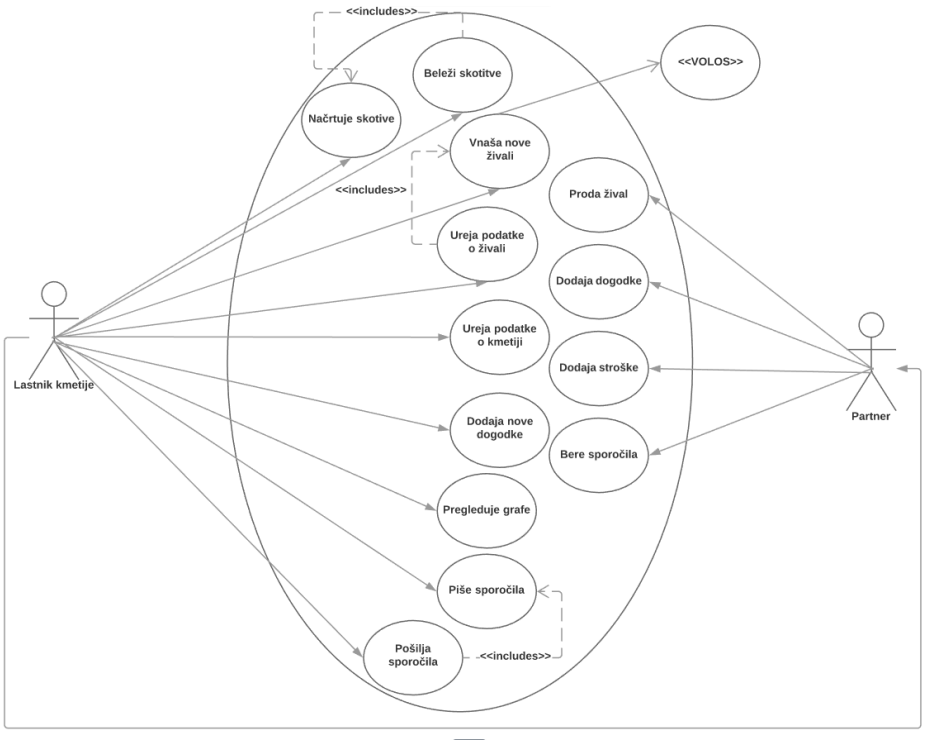
Sledeče poglavje opisuje način zajema podatkov, pripravo diagrama primerov uporabe in podatkovnega modela.

4.1 Diagram primerov uporabe

Pred samim načrtovanjem uporabniških vmesnikov in podatkovne baze smo najprej razdelili funkcionalnosti sistema glede na različne primere uporabe. Ugotovili smo, katere tipe uporabnikov naj aplikacija omogoča in do katerih opcij v sistemu naj imajo dostop. Vse skupaj smo realizirali s pomočjo diagrama primerov uporabe. Iz diagrama (Slika 4.1) lahko vidimo, da ima uporabnik, ki se v sistem prijavlja kot partner, omejena dovoljenja, medtem ko ima lastnik kmetije popoln nadzor.

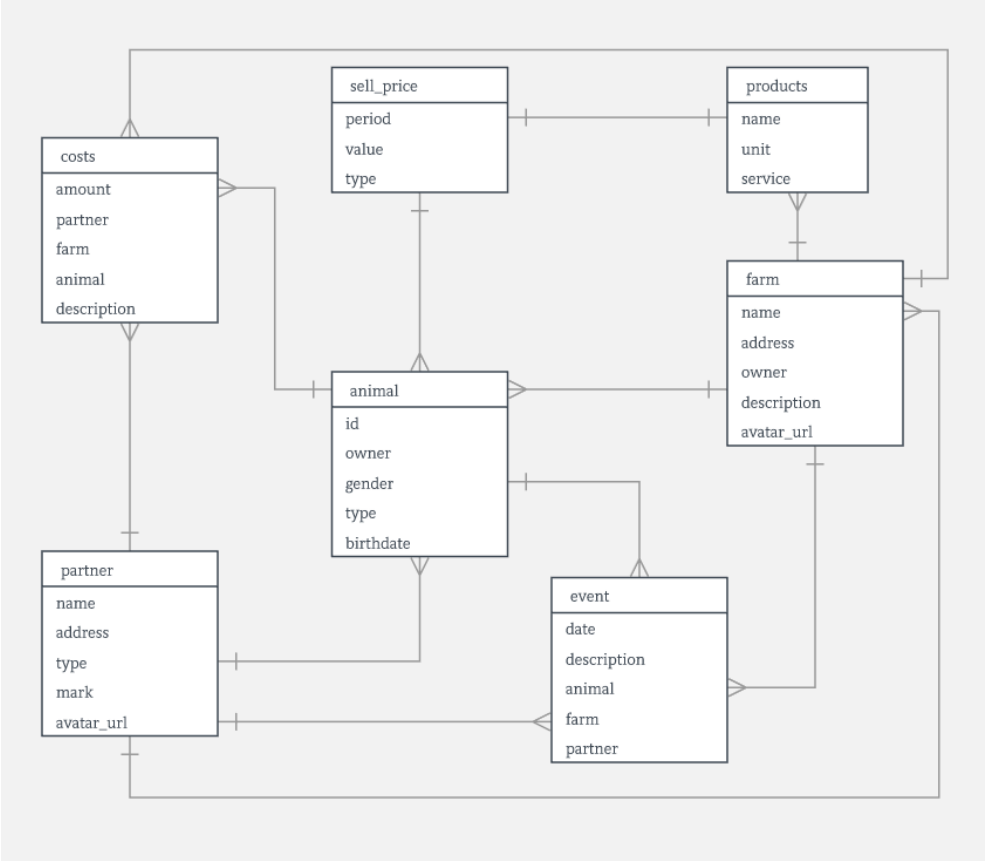
4.2 Podatkovni model

Podatkovni model aplikacije je bil zgrajen na osnovi zahtev, ki so se pokazale skozi načrtovanje uporabniških vmesnikov, ter potrebnih podatkov za prikaz in analizo. Uporabili smo podatkovno bazo PostgreSQL z neposrednim gostovanjem na platformi Heroku. Glavna prednost gostovanja aplikacijskega strežnika in podatkovne baze na isti platformi je možnost deljenja poveril-



Slika 4.1: Diagram primerov uporabe

nic, ki jih ponuja ogrodje. S tem preprečimo ročno vnašanje uporabniških imen in gesel neposredno v kodo. Podatkovni model sestoji iz sedmih entitet. Ena od glavnih podatkovnih zbirk vsebuje glavne podatke o posamezni živali: njeno ime, številko, datum rojstva, pasmo, trenutnega lastnika in stanje živali. Pomožni zbirki, ki hranita podatke o dogodkih in stroških, sta poleg entitete živali ključni tabeli za delovanje aplikacije. Hranita namreč vse podatke, dogodke in stroške, ki so se zgodili skozi vzrejo živali. Ker gre za aplikacijo, namenjeno za uporabo na več različnih kmetijah, vsi zapisi vsebujejo tudi podatek, kateri kmetiji pripada. Posledično lahko vsak kmet vidi le svoje podatke. Entiteta produktov in prodajnih cen hraniti podatke o storitvah, ki jih kmetija ponuja, njen opis, način zaračunavanja in ceno na enoto. Ta entiteta pa je povezana s podatki o prodaji, kjer se hrani vrednost, tip in datum prodajnega dogodka. Entiteta partnerja hrani podatke partner-



Slika 4.2: Entitetno-relacijski model

jev, morebitno pripadnost kmetiji skozi zaposlitveno razmerje, tip partnerja ter njegovo oceno in opombe.

4.3 Arhitektura

Celotna aplikacija ima popolnoma ločen zaledni del in del za namizno aplikacijo, namenjen prikazu in obdelavi podatkov.

4.3.1 Zaledni del

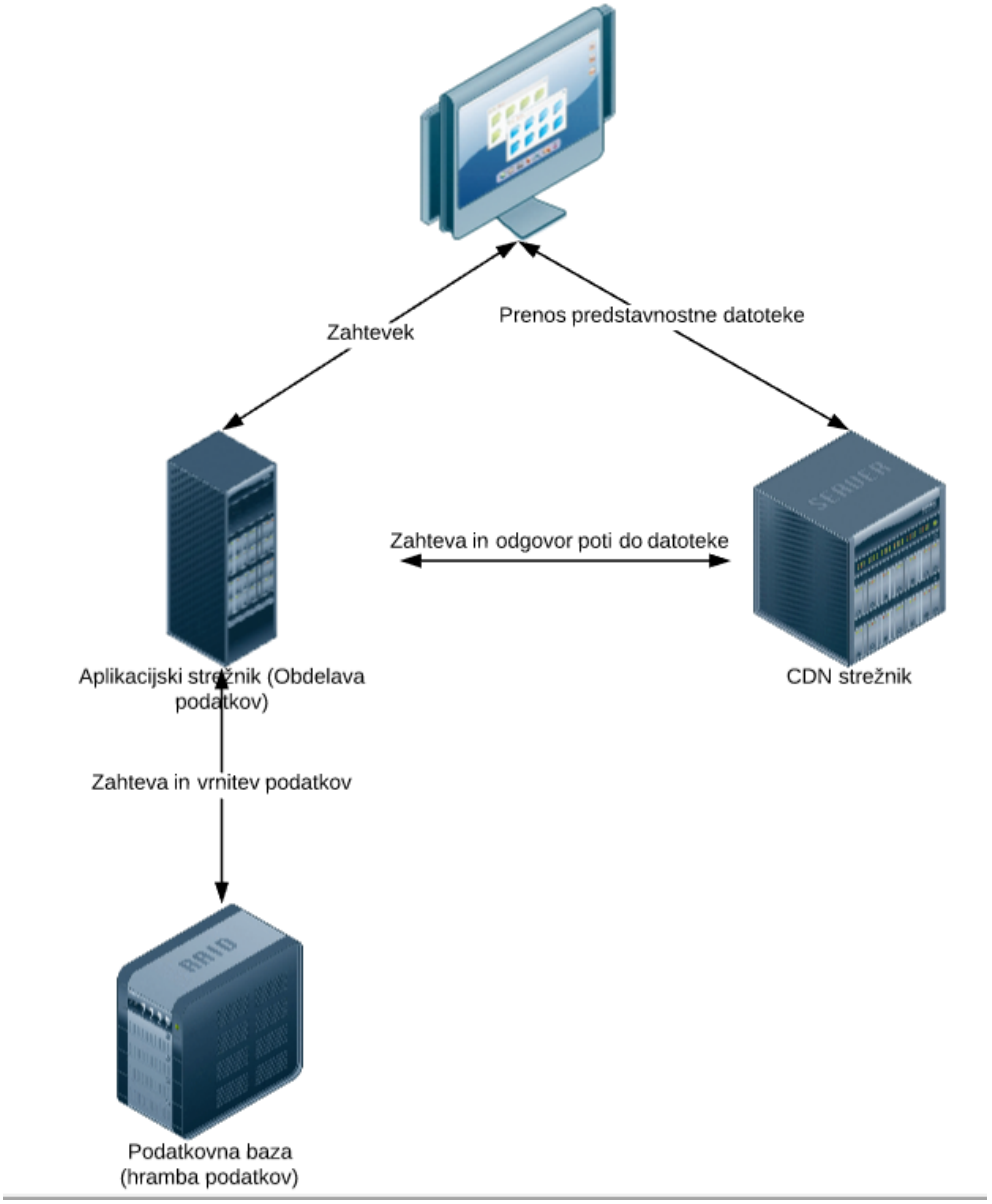
Zaledni del aplikacije sestoji iz treh med seboj neodvisnih storitev. Prva storitev je aplikacijski strežnik z gostovanjem na platformi Heroku. Aplikacijski strežnik teče v okolju Linux in poganja Python verzije 3.6. Za namestitev dodatnih paketov, s pomočjo katerih strežnik sprejema in obdeluje zahtevke, se uporablja pip in datoteka requirements.txt, ki je del projekta iz sistema za kontrolo verzij. Strežnik za procesiranje zahtev uporablja paket Gunicorn in se ob zagonu poveže s podatkovno bazo. Poverilnice za vzpostavitev povezave pridobi iz sistemskih konfiguracijskih datotek, ki se v kodo vnesejo dinamično. S tem dvignemo raven varnosti, ker svojih gesel in prijavnih podatkov ne hranimo v čistopisu. Aplikacijski strežnik na naslovu URL, ki ga pripravi Heroku, odpre in ponudi celoten aplikacijski vmesnik REST. Vsa komunikacija med namizno aplikacijo in strežnikom poteka z notacijskimi objekti v JavaScriptu (JSON) in je zaščitena z vnaprej definiranim ključem.

Strežnik se za multimedijske vsebine, kot so profilne slike, povezuje na strežnik za hrambo multimedijskih vsebin Amazon S3. Ob zahtevku za nalaganje slike aplikacijski strežnik odpre sejo in pridobi dostopne pravice in poverilnice, ki ji potem posreduje nazaj namizni aplikaciji, ki opravi prenos slike. Ko namizni odjemalec poda zahtevek za pridobitev slike, aplikacijski strežnik iz baze prebere shranjeni naslov, na katerega je bila naložena datoteka, in ga vrne odjemalcu. Ta preko pridobljenega naslova prenese multimedijske vsebine s strežnika Amazon S3. Načrtovanje podatkovnega modela na ravni aplikacije smo uredili preko predmetov v Pythonu, ki ji zna modul za komunikacijo med Pythonom in PostgreSQL pretvoriti v spremembe podatkovnega modela na fizični osnovi. Prav tako se ob nadgradnjah modelov podatki prenesejo na novejšo oblika zapisa [8].

4.3.2 Odjemalec

Namizna aplikacija, ki predstavlja odjemalec s strežnika, pridobiva podatke s pomočjo zahtevkov HTTPS na vnaprej definirane končne točke v aplikacij-

skem vmesniku REST. S kombinacijo klicev na vmesnike odjemalec poskrbi za branje, pisanje in posodabljanje vseh podatkov v aplikaciji. Za klice v zaledni sistem se uporablja modul »axios«, ki poskrbi za primerno obdelavo poslanih in prejetih zahtevkov.



Slika 4.3: Arhitektura aplikacije

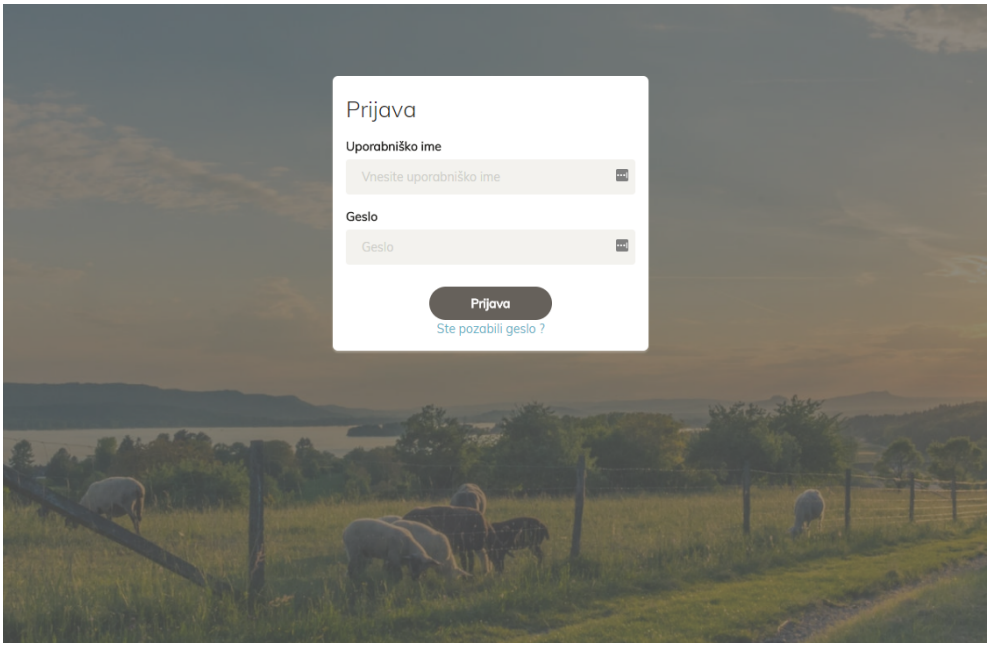
Poglavje 5

Predstavitev aplikacije

Ker gre za spletno namizno aplikacijo, za pravilno delovanje potrebujemo delujočo internetno povezavo. V trenutni fazi aplikacija ne podpira načina brez dostopa do interneta. Sama aplikacija je zgrajena kot namizna aplikacija za vse tri večje namizne operacijske sisteme in jo odpremo s klikom na zagonsko ikono.

5.1 Dostop

Ob odpiranju sistema se uporabniku najprej prikaže prijavno okno. Aplikacija trenutno deluje tako, da se je za uporabo in pridobitev uporabniškega imena in gesla potrebno dogovoriti z administratorjem sistema, ki na podlagi zahteve ustvari nov vnos za kmetijo in pripadajoče uporabnike, ki se lahko nato prijavijo v okolje povezane kmetije. Sistem omogoča naknadno spremembo ali obnovitev gesla, prav tako si mora uporabnik ob prvi prijavi zamenjati geslo, novo geslo pa se potem varno shrani v podatkovno bazo. Uporabnik je lahko član le enega kmetijskega okolja in ima dostop le do podatkov, ki so bili ustvarjeni za trenutno kmetijo. Za dostop do podatkov o drugih kmetijah se mora prijaviti z uporabniškim imenom in geslom, ki pripadata drugi kmetiji.



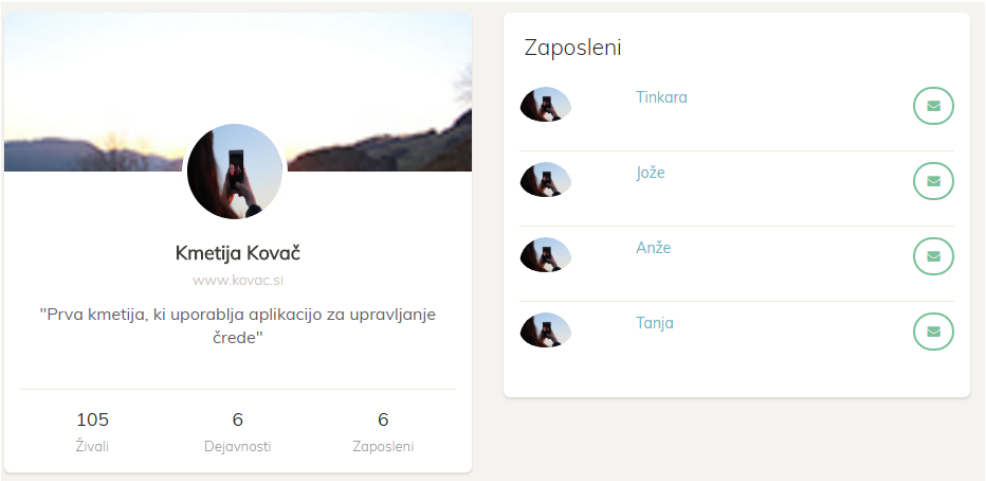
Slika 5.1: Prijavno okno

5.2 Domača stran

Po uspešni prijavi uporabnika v okolje kmetije se odpre pogled na domačo stran aplikacije (Slika 5.2), kjer so prikazani najbolj ključni podatki in trenutni status kmetije. Tako lahko uporabnik že s pogledom na osnovno ploščo vidi najpogosteje iskane podatke. V zgornji vrstici sta uporabniku vedno na voljo prikaz pregleda za trenutno okno in prikazovalnik obvestil, ki so namenjena takojšnjemu obveščanju uporabnikov o stvareh, na katere morajo biti pozorni. Trenutno se uporabniku kot obvestila izpisujejo sporočila drugih uporabnikov in opomniki o pričakovanih telitvah. V prvi vrstici glavnega okna aplikacije se uporabniku prikazujejo trenutni podatki o kmetiji. Uporabniku se takoj prikaže podatek o številu trenutno aktivnih živali na posestvu, izračun stroškov in prihodkov za prejšnji dan ter podatek o številu dogodkov, na katere mora biti danes še posebej pozoren. Uporabniku so v vrstici za obvestila na voljo podrobnejši podatki o preteklih dogodkih in dogodkih, ki so načrtovani za današnji dan. Skrajno desno se prikaže še število

Slika 5.2: Domača stran

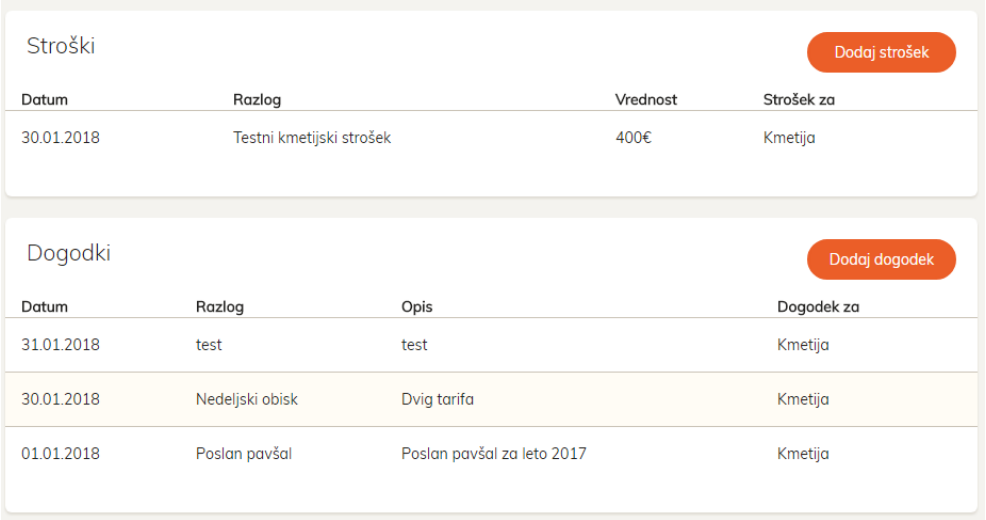
neprebranih sporočil, na katere lahko odgovori. Drugi glavni poudarek strani sta grafa, ki prikazujeta trenutno poslovanje kmetije. Graf na levi strani prikazuje razdelitev prihodkov glede na tip izvora. Sam graf prikazuje razdelitev prihodkov glede na prodajo živali in prejem klavne premije, prodajo surovin, med katere lahko umeščamo žita, lesno biomaso ali ostale izdelke, ki jih lahko kot surovino uporabijo naši partnerji ali stranke. V zadnjo kategorijo storitev spadajo vsi prihodki iz naslova opravljenih storitev, med katere lahko štejejo prihodki strojnih uslug za druge kmetije, fizična pomoč pri opravilih ter druga pomoč in svetovanje. Graf na desni prikazuje trenutno mesečno prodajo živali, pri kateri je upoštevana le prodaja bikov in telet. Krav v tem izračunu ne upoštevamo, ker le redko pride do prodaje in to ni vir prihodka, na katerega se kmetija zanaša. Pri izračunu mesečne vrednosti se prikazuje odkupna vrednost klavnic za določen mesec. Iz podatkov na Sliki 5.2 lahko hitro opazimo, da poletni meseci niso najprimernejši za prodajo živali, ker lahko takrat iztržimo bistveno manj kot recimo ob koncu leta.



Slika 5.3: Domača stran kmetije

5.3 Urejanje podatkov o kmetiji

Drugi zavihek, ki ga najdemo v vertikalnem navigacijskem meniju na levi, je kmetija in tu najdemo vse podrobnosti o kmetiji. Stran je razdeljena na dva večja dela: zgornji del prikazuje podrobnosti in je obenem tudi maska za urejanje podatkov, spodnji del pa prikazuje vse podatke, ki so vezani na kmetijo. Prvi levi stolpec prikazuje kartico kmetije z vsemi podatki, vključno z določeno prikazno sliko. Na desni strani imamo prikazano tabelo zaposlenih, za katere se sploh pri družinskih kmetijah izkaže, da so navadno člani družine, v nekaterih primerih pa so to tudi zunanji sodelavci. Pri vseh uporabnikih imamo možnost pošiljanja sporočil, pri čemer se nam ob kliku na gumb odpre pogovorno okno, kamor vpišemo želeno sporočilo. Osrednji del služi kot prikazovalnik podrobnosti, obenem pa je tudi urejevalnik podatkov. Vse spremembe, ki jih naredimo, lahko shranimo s klikom na gumb »Posodobi podatke«. Ker je aplikacija zgrajena na ogrodju JavaScript z dvosmernim povezovanjem podatkov, so vse spremembe, ki jih vpisujemo, takoj vidne na kartici kmetije. Spodnji del strani kmetije predstavljajo podatki, ki so vezani na trenutno kmetijo. Prvi razdelek so stroški, ki so vezani neposredno na kmetijo in ne na živali. Pri vzreji se velikokrat pojavijo stroški, ki jih



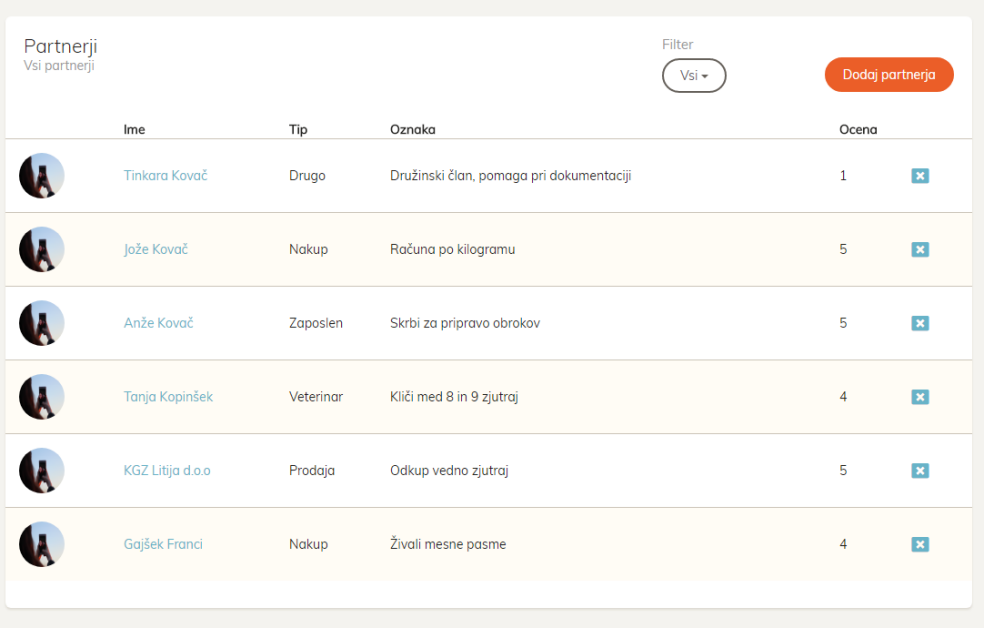
Slika 5.4: Podatki vezani na kmetijo

uporabnik ne more povezati z določeno živaljo, ampak ta strošek pripada celotni kmetiji. Med takšne stroške spadajo:

- strojne storitve, ki so jih za nas opravile druge kmetije ali strojni krožki,
- stroški, povezani s pripravo obrokov, med katere spadajo nakupi surovin, prehranskih dopolnil in mineralov,
- izdatki, ki so nastali ob nakupu potrošnega materiala (rokavice, zaščitni čevlji, oblačila).

Vse ostale stroške lahko kmet zapiše neposredno na partnerja ali na točno določeno žival. S tem doseže preglednost sodelovanja z določenim partnerjem ali izdatkov, ki jih ima zaradi posamezne živali.

Drugi povezani razdelek so dogodki, vezani na kmetijo, in ti v največji meri služijo kmetovi osebni evidenci. Uporabnik si lahko kot dogodke zabeleži raznorazne aktivnosti, ki so se zgodile na kmetiji in za katere želi v prihodnosti videti točen datum.



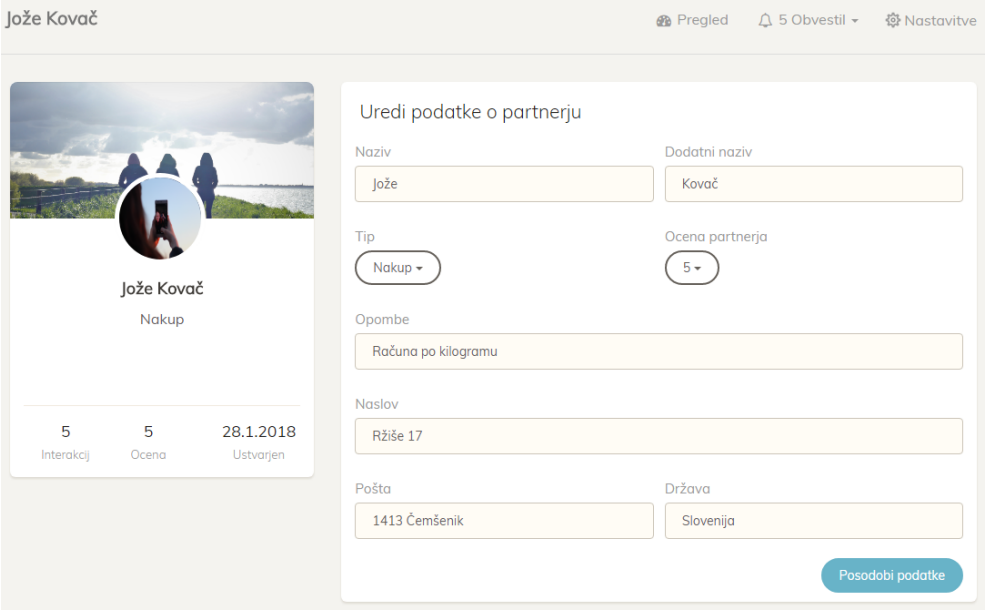
Slika 5.5: Prikaz poslovnih partnerjev kmetije

5.4 Partnerji

Razdelek partnerjev prikazuje vse ljudi, ki so na kakršenkoli način povezani s kmetijo. Med partnerje lahko uporabnik vnese:

- ljudi, ki imajo na kmetiji dodeljene vloge ali zgolj pomagajo pri fizičnih opravilih,
- poslovne partnerje, s katerimi posluje glede prodaje živali,
- ljudi in kmetije, pri katerih je možnost nakupa dodatnih živali v primeru, da je število rojstev na domači kmetiji nezadostno,
- vse ostale ljudi, ki so povezani s kmetijo (veterinarji, inšpektorji, svetovalci).

Stran v privzetem načinu prikazuje vse partnerje, pri katerih imamo možnost filtriranja glede na njihov odnos z nami. Za vsakega partnerja



Slika 5.6: Kartica partnerja ter urejevalnik podrobnosti

si uporabnik lahko beleži tudi dodatne opombe, ki ga takoj opomnijo na specifike pri poslovanju s to osebo. Prav tako lahko uporabnik partnerjem določi oceno, ki služi lažjemu internemu odločanju pri konfliktnih situacijah ali izrednih dogodkih.

V zgornjem desnem kotu imamo gumb za ustvarjanje novega uporabnika, ki odpre pogovorno okno, kamor uporabnik vpiše vse želene podatke, glavni med njimi sta seveda naziv in klasifikacija partnerja. Pri vsakem partnerju imamo tudi možnost izbrisa v primeru, da z njim ne bomo več sodelovali.

S klikom na partnerja odpremo celovit pregled našega sodelovanja z njim.

5.4.1 Pogled poslovnega partnerja

Kartica partnerja sledi celostni oblikovni zasnovi aplikacije, tako da na levi strani najdemo združen hitri pregled podatkov s tem partnerjem, na desni strani pa ima uporabnik možnost urejanja podrobnosti.

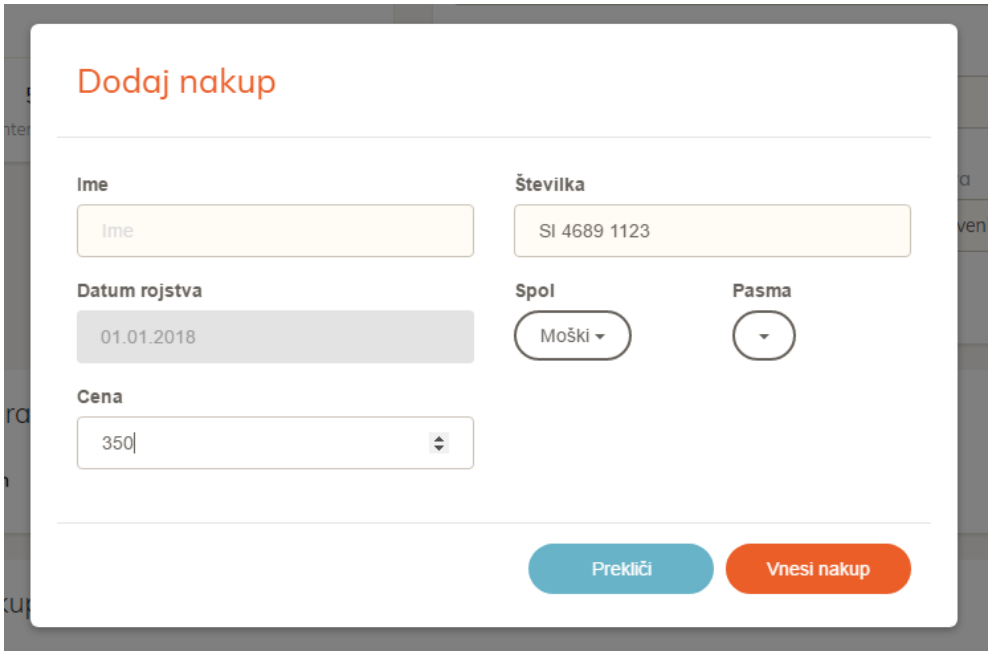
Ključnega pomena pri partnerju so naše interakcije z njim in že opravljeni

posli. Zato prva podsekcija povezanih podatkov prikazuje seznam vseh dogodkov s tem partnerjem. Celotna zasnova modula za interakcije je zastavljena tako, da ustreza tako fizičnim osebam kot podjetjem, s katerimi kmet posluje. Pri dodajanju nove interakcije si lahko uporabnik zabeleži katerikoli dogodek, ki se je zgodil v povezavi s to osebo oziroma podjetjem. V primeru veterinarja si lahko zabeleži datum obiska in storitve, ki jih je opravljal, v primeru obiska inšpektorja pa si lahko zabeleži datum obiska, katere stvari so bile kontrolirane in pomanjkljivosti, ki so bile ugotovljene ob pregledu. Za vsako interakcijo ima uporabnik možnost izbire, ali gre za nek splošen dogodek ali je ta dogodek vezan na določeno žival. V primeru, da se pri dodajanju interakcije izbere tudi žival, se ta dogodek zabeleži tudi v pogledu živali, ki si ga bomo ogledali v nadaljevanju. Pri razdelku stroškov ima uporabnik možnost vnosa splošnih stroškov za posamezno žival, kot so recimo cena veterinarskega posega, nakup dodatnih vitaminov in podobno. Prav tako lahko v primeru, da gre za partnerja, ki je klasificiran kot prodajalec živali, vnese nakup (Slika 5.7). Ob tem mora uporabnik vnesti identifikacijsko številko živali, njeno ime, pasmo in datum rojstva. Avtomatsko se v sistemu kreira nova žival, za katero se zapišejo vsi vneseni podatki, prav tako pa se nakup pojavi med stroški za to žival.

5.5 Živali

Pogled živali vsebuje za kmeta najbolj ključne podatke, zato je na maski poskrbljeno za čim večjo preglednost in možnost filtriranja po različnih kategorijah. Prav tako se na tej maski uporabniku izpišejo pred kratkim prodane živali oziroma živali, ki so zaradi izrednih dogodkov odšle s kmetije.

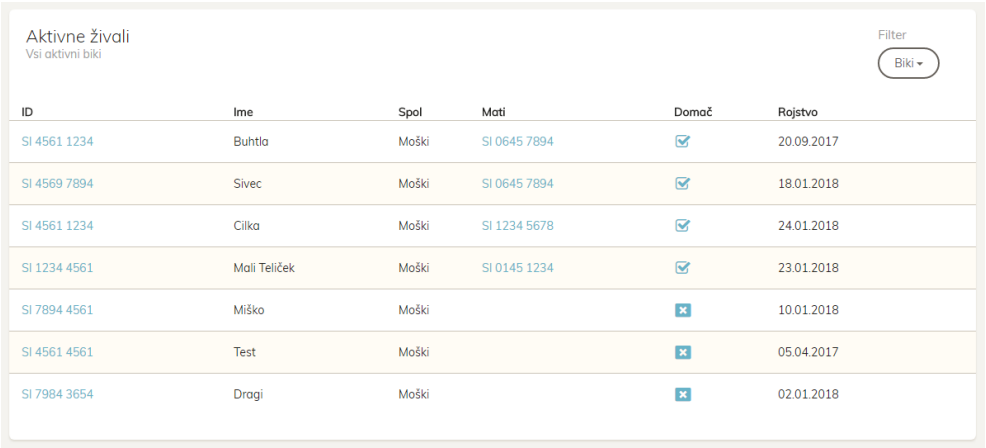
Za vsako žival takoj vidimo njen spol, starost in številko, obenem pa imamo prikazan tudi podatek o tem, ali je bila žival rojena na naši kmetiji ali je bila kupljena. Pri živalih, rojenih na naši kmetiji, imamo ob tem še podatek o materi. Klik na posamezno žival odpre njene podrobnosti.



Slika 5.7: Pogovorno okno za vnos nakupa

5.5.1 Pogled živali

Pri vsaki živali imamo tako kot pri vseh subjektih na kmetiji urejen zbran prikaz podatkov in možnost urejanja. Pri pogledu živali je specifično, da ima prikazan datum rojstva, prav tako pa imamo možnost izbire dodatnega klasifikatorja pasme, s pomočjo katerega lahko kmet hitro ugotovi, katere pasme so za njegov način reje bolj primerne in katere manj. Posebnost pri pogledu živali je tudi gumb »Prodaja«, ki sproži postopek prodaje za žival. Uporabniku se odpre okno, kamor vnese podatek o odkupni ceni, za katero se je dogovoril z odkupovalcem živine, in težo živali. Če kmet teh podatkov še nima, lahko s postopkom prodaje počaka do takrat, ko od partnerja prejme potrditev prodaje, ki mora vsebovati odkupno ceno in težo, na podlagi česar sistem samodejno izračuna prihodek od prodaje in ga vnese na prikaz živali. Prav tako ima uporabnik možnost izbire partnerja, kar mu omogoča pregled nad številom opravljenih transakcij s partnerjem. Aplikacija omogoča tudi hitro navigacijo v sistem VOLOS, kjer je potrebno urediti tudi dokumentacijo



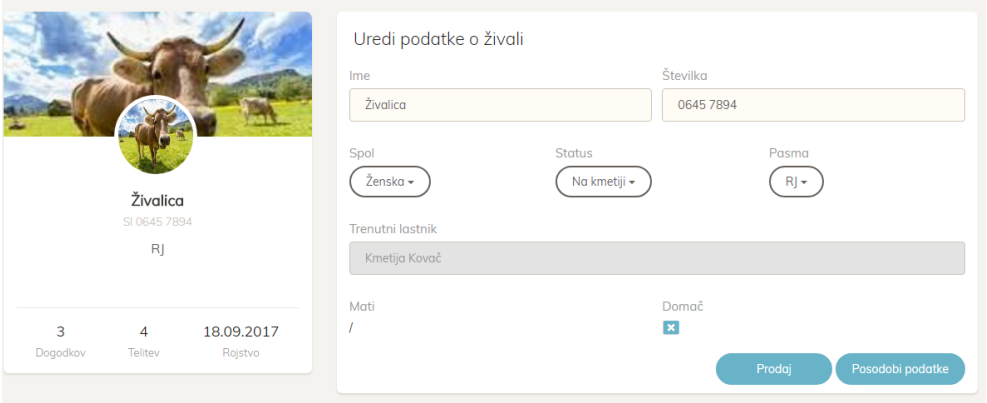
Slika 5.8: Prikaz živali z možnostjo filtriranja

za prihod in odhod. Prav tako pa se podatek o odkupni ceni shrani v sistem za ta dan. V primeru, da vnesemo več različnih odkupnih cen, se bo pri mesečnem prikazu grafov izpisala povprečna vrednost.

5.5.2 Dodatne opcije

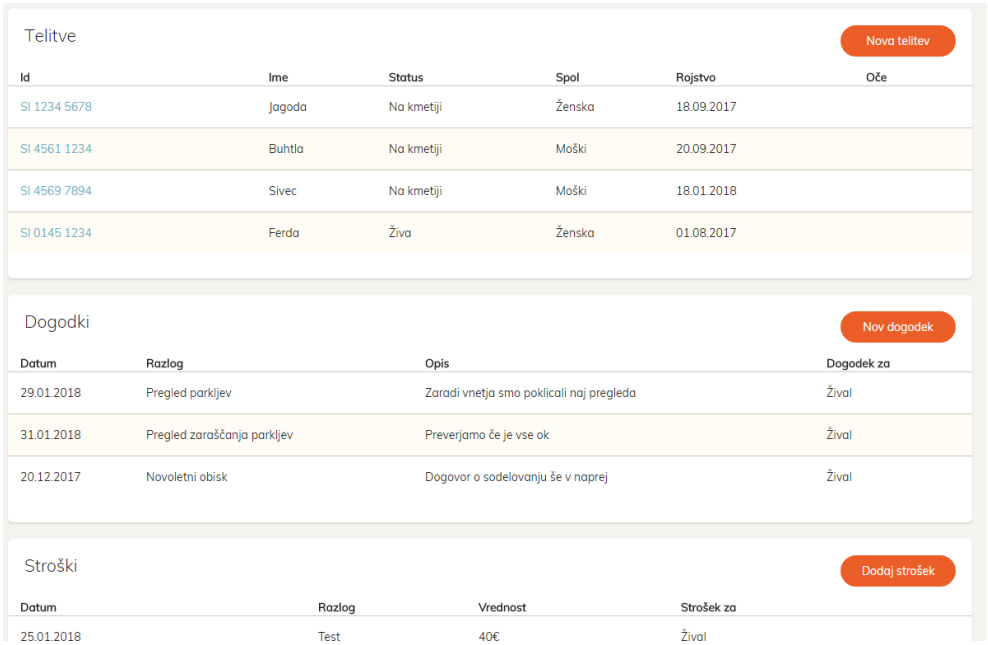
Pri vsaki živali je razdelek za povezane podatke povsem prilagojen. V prvi vrsti se razdelek prilagodi glede na spol živali, kar uporabniku pri kravah omogoči vnos in planiranje telitev, kar pri bikih in teletih ni na voljo. Prav tako se glede na starost živali omogočijo različne funkcionalnosti, kot so dodajanje cepljenja pri živalih, starejših od šestih mesecev.

Pri dodajanju telitev za živali ženskega spola, starejših od pol leta, ima uporabnik dve možnosti, in sicer v primeru, da se je telitev že zgodila, da vnese že oštevilčeno žival, njeno ime, spol in pasmo. Če pa se telitev še ni zgodila, lahko uporabnik vnese približen datum kotitve, ki mu ga je sporočil veterinar. Aplikacija en teden pred predvideno kotitvijo začne kmeta opozarjati, naj bo na to žival še posebej pozoren. Ko dejansko pride do skotitve, se lahko ta opomnik uredi v zapis oštevilčene živali. Prav tako mogoče popraviti datum rojstva, ki ni vedno devet mesecev od začetka brejosti. Za



Slika 5.9: Kartica živali

vsako žival imamo možnost neposrednega vnosa dogodkov in stroškov, med katerimi se jih precej generira avtomatično z uporabo sistema. S tem kmetu omogočimo pregled nad res vsemi dejavnostmi, ki so se zgodile za to žival od samega začetka – bodisi od nakupa bodisi od skotitve, tudi v primeru, da gre za žival z daljšo življenjsko dobo.



Slika 5.10: Kartice s povezanimi zapisi živali

Poglavje 6

Nadaljnji razvoj

Ker se aplikacija trenutno uporablja na naši kmetiji, se je skozi uporabo pokazalo veliko različnih scenarijev za nadaljnji razvoj, ki v prvotni verziji še niso bili pokriti.

6.1 Integracija s sistemom VOLOS

Cilj te diplomske naloge je bil že od začetka priprava namizne aplikacije za dostop do sistema VOLOS, ki hrani podatke iz centralnega registra govedi Slovenije. Pri prvotnem povpraševanju so nam na ministrstvu na prošnjo za povezovanje odgovorili pritrdilno. Med samo implementacijo aplikacije pa se je izkazalo, da ministrstvo ni pripravljeno urediti programskega dostopa do svojih podatkov, kljub vsem predlogom avtentikacije. Zato želja po izboljšanju uporabniške izkušnje, ki jo je uporabnik deležen na portalu VOLOS, ostaja zaenkrat še neizpolnjena.

6.2 Mobilna aplikacije

Ker gre za namizno aplikacijo, v trenutni obliki aplikacije ni možnosti, da bi do podatkov dostopali preko mobilne naprave, kar se je izkazalo kot zelo potrebna funkcionalnost. Na sami lokaciji hleva je pogosto potrebno preveriti

številko posamezne živali in njene simptome. Tu pa nastane problem zaradi namizne aplikacije. Trenutno smo v sistem dodali kaskadne sloge za način tiskanja, kar omogoča tisk podatkov neposredno iz aplikacije. Z mobilno verzijo pa bi rešili še problem prenosljivosti podatkov. Aplikacija bi bila odlična tudi pri primerjavi različnih parametrov med kmetijami za različne uporabnike.

6.3 Povezovanje med kmetijami

Ker je za aplikacijo izrazilo zanimanje kar nekaj okoliških kmetij, se je zelo hitro pojavila tudi ideja za medsebojno primerjavo in sodelovanje. Primerjavo bi rešili z možnostjo označitve nekega podatka kot javnega, s čimer bi postal viden vsem uporabnikom omenjene aplikacije. Javna bi bila tudi kartica kmetije, kjer bi lahko uporabniki našli posamezno kmetijo in jo kontaktirali. Pri tem se odpreta še dve ideji, in sicer možnost interne tržnice med kmetijami, kjer bi lahko kmetije objavljale, katere izdelke, storitve oziroma surovine imajo na voljo za prodajo in v obratni smeri bi lahko kmetije iskale in opravljale nakupe od drugih. Uredila bi se tudi sekcija povsem javnih podatkov, ki bi bili dostopni na javni strani, kjer bi lahko posameznik iskal med kmetijami in njihovimi storitvami.

6.4 Dostop partnerjev do sistema

V luči zmanjševanja administrativnih bremen za kmete je cilj v prihodnosti aplikacijo dodelati na takšen način, da bodo lahko partnerji s svojim dostopom dodajali dogodke, stroške in prihodke neposredno na naše živali in kmetije. Takšna implementacija seveda zahteva tudi naprednejši sistem pravic, ki bi ga bilo potrebno implementirati vzporedno.

6.5 Postopek registracije in kreiranje nove kmetije

Ker je bila aplikacija prvotno namenjena za uporabno na eni kmetiji, je proces kreiranja novega uporabnika in kmetije še vedno ročen. Ob porastu povpraševanja se je pojavilo veliko administrativnega dela, zato bi bil postopek lahko z avtomatskim kreiranjem kmetije in uporabnikov veliko hitrejši.

Poglavje 7

Zaključek

Cilj te diplomske naloge je bil poenotenje in poenostavitev shranjevanja in nadziranja vseh procesov, ki potekajo na kmetiji zaradi vzreje govejih pitancev. Od prvotno zadanega cilja je ostala neuresničena le popolna integracija s sistemom VOLOS. Edino, kar nam je uspelo tu, so hitre povezave, ki odprejo točno določeno stran, kjer je mogoče takoj začeti z vnosom spremembe. Prav tako je bil cilj naloge, da se aplikacija realno uporabi na vsaj eni kmetiji, kar nam je več kot uspelo z uporabo aplikacije na Kmetiji Kovač in še z dodatnim povpraševanjem sosednjih kmetij. Vsi deli aplikacije so sestavljeni tako, da jih je mogoče na preprost način nadgraditi in spremeniti. Celoten mehanizem objave sprememb kode za zaledno logiko je avtomatiziran, kar pomeni, da se mora razvijalec posvetiti le dodajanju novih sprememb, kar bo še povečalo željo po nadaljevanju razvoju te aplikacije, ki lahko prispeva k večji konkurenčnosti slovenskega kmetijstva na globalnem trgu.

Literatura

[1] Babel. Dosegljivo: `https://babeljs.io/`, 2017. [Dostopano: 15. 11. 2017].

[2] Tim Bray. The javascript object notation (json) data interchange format. 2017.

[3] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[4] Statistika strukture kmetijskih gospodarstev. Dosegljivo: `http://ec.europa.eu/eurostat/statistics-explained/index.php?title=Farm_structure_statistics/sl&oldid=352097`, 2015. [Dostopano: 4. 1. 2018].

[5] Miguel Grinberg. *Flask web development: developing web applications with python*. "O'Reilly Media, Inc.", 2014.

[6] Neil Middleton and Richard Schneeman. *Heroku: Up and Running: Effortless Application Deployment and Scaling*. "O'Reilly Media, Inc.", 2013.

[7] Mark Pilgrim. *HTML5: Up and Running: Dive into the Future of Web Development*. "O'Reilly Media, Inc.", 2010.

[8] Wuwei Shen and Shaoying Liu. Formalization, testing and execution of a use case diagram. In *International Conference on Formal Engineering Methods*, pages 68–85. Springer, 2003.

[9] Struktura kmetijskih gospodarstev. Dosegljivo: `http://www.stat.si/StatWeb/News/Index/6208`, 2016. [Dostopano: 4. 1. 2018].

[10] Stefan Tilkov and Steve Vinoski. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.

[11] Webpack. Dosegljivo: `https://webpack.js.org/`, 2017. [Dostopano: 13. 11. 2017].