Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Luka Domitrovič

Spletna aplikacija za vodenje dogodkov planinskih društev

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Mira Trebar

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kandidat naj preuči področje storitev v računalniškem oblaku, ki vključujejo večuporabniški arhitekturni vzorec pri razvoju programske opreme. Osredotoči naj se na upravljanje podatkov različnih uporabnikov, dinamične podatkovne strukture in preklapljanje med različnimi shemami. Za implementacijo aplikacije, ki bo delovala kot storitev in bo na voljo planinskim društvom za urejanje in vodenje dogodkov ter objavo na spletni strani, naj uporabi spletno ogrodje Ruby on Rails in podatkovno bazo PostgreSQL.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Program Interface	Programski vmesnik
CNAME	Canonical Name	Zapis kanoničnega imena
CRUD	Create, Read, Update and Delete	Ustvari, beri, posodobi, izbriši
CSS	Cascading Style Sheets	Kaskadne stilske predloge
DOM	Document Object Model	Objektni model dokumenta
HTML	Hyper Text Markup Language	Jezik za označevanje nadbesedila
HTTP	Hyper Text Transfer Protocol	Protokol za izmenjavo nadbesedila
IaaS	Infrastructure as a Service	Infrastruktura kot storitev
MVC	Model-View-Controller	Model-Pogled-Nadzornik
ORM	Object-Relational Mapping	Objektno-relacijska preslikava
REST	Representational State Transfer	Prenos predstavitvenega stanja
PaaS	Platform as a Service	Platforma kot storitev
PZS	Alpine Association of Slovenia	Planinska zveza Slovenije
RoR	Ruby on Rails	Ruby on Rails

SaaS	Software as a Service	Programska oprema kot storitev
SOAP	Simple Object Access Protocol	Protokol za spletne storitve
SQL	Structured Query Language	Strukturirani povpraševalni jezik
URL	Uniform Resource Locator	Enotni naslov vira
WYSIWYG	What you see is what you get	Kar vidiš to dobiš

Povzetek

Naslov: Spletna aplikacija za vodenje dogodkov planinskih društev

Avtor: Luka Domitrovič

Diplomsko delo zajema razvoj in predstavitev spletne aplikacije za planinska društva na področju Slovenije. Namen aplikacije je olajšati delo strokovnim delavcem društva pri organizaciji dogodkov, širši javnosti pa ponuditi interaktivne spletne strani planinskih društev. Glavne funkcionalnosti so enostavno ustvarjanje dinamične vsebine dogodkov, opremljene s slikami in zemljevidi, objava in prikaz na spletni strani, obveščanje članov po elektronski pošti, elektronska prijava udeležencev na dogodke ter samodejno pridobivanje podatkov za evidenco. Aplikacijo smo zasnovali po večuporabniškem arhitekturnem vzorcu gradnje programske opreme kot storitev v računalniškem oblaku. Za razvoj smo uporabili ogrodje za gradnjo spletnih aplikacij Ruby on Rails in podatkovno bazo PostgreSQL. Razvito programsko opremo bomo kot storitev ponudili v uporabo planinskim društvom.

Ključne besede: spletna aplikacija, programska oprema kot storitev, planinsko društvo, Ruby on Rails.

Abstract

Title: Web application for management of events in Alpine clubs

Author: Luka Domitrovič

The diploma thesis covers the development and presentation of a web application for alpine clubs in Slovenia, which aims to facilitate the work of the club professionals in organising events, and offer interactive websites of alpine clubs to the public. The main functionalities are the simple creation of dynamic event-related content, including images and maps, publication and display on the website, member notification by e-mail, registration of participants to events, and automated data collection for records. The application was designed, in accordance with the multi-user architectural model of software construction, as a cloud computing service. We used the Ruby on Rails web application construction framework and the PostgreSQL database for development. The developed software will be offered as a service to alpine clubs.

Keywords: web application, software as a service, alpine club, Ruby on Rails.

Poglavje 1

Uvod

Tehnologija nas obdaja na številnih področjih vsakdanjega življenja. Uporabljamo jo za doseganje boljših poslovnih rezultatov v podjetjih, vodenje državnih organizacij, v izobraževanju, komunikacijah in prav tako v prostem času. Obstajajo pa področja, kjer se razvoj sodobnih tehnoloških rešitev še ni uveljavil. To je mogoče opaziti predvsem pri neprofitnih organizacijah. Kot planinski vodnik sem vključen v delovanje neprofitne organizacije Planinsko društvo Velenje. Razlogov, zakaj so tovrstne organizacije zaostale pri razvoju digitaliziranih rešitev, je več, ključen razlog pa je pridobivanje finančnih sredstev za razvoj sodobnih digitalnih rešitev. Slovenska zakonodaja zahteva, da se v planinskih društvih vodijo evidence o dogodkih in udeležencih. Društva so ob pomanjkanju ustrezne in cenovno ugodne programske opreme prisiljena voditi evidence v papirni obliki ali pa v preglednicah na računalniku. Pri predstavitvi društva širši javnosti in komunikaciji s člani je mogoče opaziti, da je to področje slabo podprto s tehnološkimi rešitvami.

Svetovni splet se je že pred časom uveljavil kot platforma za gradnjo programske opreme. Z razvojem vse bolj kompleksne programske opreme se spreminjajo standardi, izboljšujejo programski jeziki in orodja za gradnjo. V porastu so storitve, ki temeljijo na osnovah računalništva v oblaku. Izkoriščajo prednosti, modele, principe in standarde računalniških oblakov za reševanje kompleksnih tehnoloških problemov.

Z uporabno naprednih principov računalništva v oblaku lahko izboljšamo ponudbo programske opreme za planinska društva. Pozitivne lastnosti računalništva v oblaku omogočajo, da lahko dosežemo enostavne rešitve problemov z ugodnimi finančnimi možnostmi za društva, uporabnikom pa ponudimo boljšo uporabniško izkušnjo. Porodila se je ideja o spletni aplikaciji, ki bi na eni strani omogočala društvu poenostavljeno delo in na drugi strani širši javnosti ponudila obsežnejše in zanimivejše informacije o aktivnostih, ki se na društvu dogajajo. Cilj diplomskega dela je izdelati spletno aplikacijo do točke osnovne in sprejemljive rešitve ter jo kot storitev ponuditi v uporabo planinskim društvom. Aplikacija vključuje reševanje problemov ustvarjanja interaktivnih objav dogodkov, prikaz dogodkov širši javnosti, obveščanje o novostih, omogočanje elektronske prijave na dogodke in s tem samodejno zbiranje podatkov o udeležbi. Razvoj aplikacije bo temeljil na ogrodju za gradnjo spletnih aplikacij Ruby on Rails. Pri načrtovanju aplikacije bomo uporabili različne principe in arhitekturne vzorce za izvedbo večuporabniške arhitekture.

Diplomsko delo smo razdelili na več delov. Prvi del je namenjen teoretičnemu pregledu metod računalništva v oblaku. Sledil bo opis tehnologij in orodij, ki jih bomo uporabili pri razvoju aplikacije. Preučili bomo delovanje društva ter rešitve, ki jih trenutno uporabljajo pri svojem delu. Načrtovanje in izdelava aplikacije je obsežno področje, v diplomskem delu pa se bomo osredotočili na tri ključne teme. Začeli bomo z zasnovo podatkovnega modela, nadaljevali z implementacijo večuporabniške arhitekture in končali pri realizaciji dinamičnih podatkovnih struktur. Sledil bo pregled in uporaba uporabniškega vmesnika aplikacije. V zaključku bomo predstavili uporabo aplikacije, izbrano arhitekturno zasnovo ter podali predloge za nadaljnje delo.

Poglavje 2

Pregled področja

2.1 Računalništvo v oblaku

Pojem računalništvo v oblaku (angl. cloud computing) označuje način zagotavljanja skupne rabe računalniških virov na zahtevo. Računalniška sredstva v tem primeru zajemajo strojno in programsko opremo v različnih oblikah storitev, kot so uporabniške aplikacije, shranjevanje podatkov in razvojne platforme [15]. Glede na potrebe uporabnika ločimo več vrst oblakov, ki jih poznamo kot: javni, zasebni, hibridni ter združen oblak skupnosti.

Značilnosti računalniških oblakov [6]:

- računalniški sistem na zahtevo (angl. On-demand Computing Model): organizacije ne potrebujejo svojih podatkovnih centrov, da zadovoljijo informacijsko tehnološke potrebe. Ponudniki računalniških oblakov ponujajo dostop do deljenih računalniških virov;
- avtonomija delovanja (angl. Autonomous): uporabnikom se odvzame skrb za izbiro in upravljanje strojne opreme, tehnologije, omrežja ter zaposlitev ljudi, ki bodo upravljali infrastrukturo;
- zagotovljena kakovost storitve (angl. Predefined QoS): uporabniki lahko izberejo ponudnika glede na zagotovljeno kakovost storitve, ki izpolnjuje uporabnikove tehnične zahteve;

- enostavna uporaba (angl. Easy-to-use): ponudniki računalniških oblakov ponujajo enostavno upravljanje storitev preko uporabniških in programskih vmesnikov (angl. API);

- skalabilnost (angl. Scalable): uporabniki niso omejeni s količino računalniških virov, ampak jih lahko po potrebi prilagajajo;

- cenovna dostopnost (angl. Inexpensive): uporabnikom ni potrebno postavljati svoje infrastrukture zaradi česar se izognejo stroškom razvoja infrastrukture.

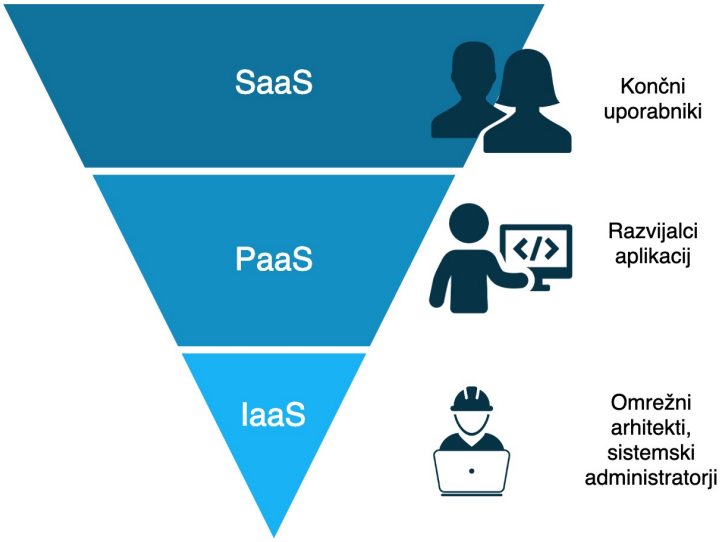
Skupna raba računalniških virov prinaša večjo možnost zlorabe podatkov. Znižanje tveganja zlorab se zagotavlja z zaščito podatkov na področjih shranjevanja, prenašanja in obdelave. Zaščita podatkov je največji izziv računalništva v oblaku. Za izboljšanje varnosti je potrebno zagotoviti avtentikacijo, avtorizacijo in nadzor dostopa do podatkov [32]. Področja, s katerimi se ukvarja zaščita podatkov v računalniškem oblaku:

- zaupnost (angl. confidentiality): programska oprema ne sme dopuščati napadov zlonamernih uporabnikov;

- integriteta (angl. integrity): jamstvo, da je uporabnikom dovoljen dostop samo do lastnih podatkov;

- razpoložljivost (angl. availability): neprekinjena dosegljivost storitev.

Glede na namen uporabe računalniškega oblaka ločimo tri modele storitev: infrastruktura kot storitev (IaaS), platforma kot storitev (PaaS) in programska oprema kot storitev (SaaS) [15, 29]. Grafični prikaz modelov storitev v oblaku se nahaja na sliki 2.1. Vsak model storitve v oblaku je namenjen svoji ciljni skupini uporabnikov.



Slika 2.1: Prikaz modelov storitev v oblaku.

2.1.1 Infrastruktura kot storitev

Infrastruktura kot storitev – IaaS (angl. Infrastructure as a Service). Ponudnik storitev uporabniku nudi procesorsko moč, podatkovni prostor, omrežja in druga osnovna računalniška sredstva. Uporabnik ima popoln nadzor nad izbiro operacijskega sistema, uporabo podatkovnega prostora, nastavitvami omrežja in programsko opremo. Pri določenih ponudnikih storitev je omenjena računalniška sredstva mogoče koristiti dinamično, kar pripomore k učinkoviti rabi virov. Uporabnik nima dostopa do infrastrukture oblaka. IaaS je primerna za sistemske administratorje, ki lahko postavijo okolje za poganjanje aplikacij. Primer ponudnika je Amazon Web Services [2].

2.1.2 Platforma kot storitev

Platforma kot storitev – PaaS (angl. Platform as a Service). Ponudnik storitev omogoča uporabnikom nalaganje aplikacij v programskih jezikih, uporabo knjižnic in orodij, podprtih s strani ponudnika. Uporabnik nima nadzora nad infrastrukturo oblaka, omrežji, strežniki, operacijskim sistemom, ampak le

nad naloženo programsko kodo. PaaS je najbolj primeren model storitve v oblaku za razvijalce aplikacij, ker omogoča hitro in enostavno objavo aplikacije na spletu. Primer takšnega ponudnika je Heroku [7].

2.1.3 Programska oprema kot storitev

Programska oprema kot storitev – SaaS (ang. Software as a Service). Ponudnik storitve uporabniku zagotavlja celovito programsko opremo, ki rešuje domenski problem. Uporabniki so lahko v tem primeru tudi končne stranke ali pa organizacije, ki zajemajo skupino ljudi. Ponudnik poizkuša podrobno razumeti potrebe uporabnikov in načrtovati aplikacijo za ciljno skupino uporabnikov. Uporabnik ne upravlja z infrastrukturo oblaka. Onemogočen je dostop do strežnikov, omrežja, operacijskega sistema in programske kode aplikacije. Dovoljen je dostop le do vsebine preko uporabniških vmesnikov. Pogosta praksa ponudnikov programske opreme kot storitev je, da za gostovanje svoje programske opreme uporabljajo druge ponudnike PaaS. Ključne karakteristike SaaS so [8]:

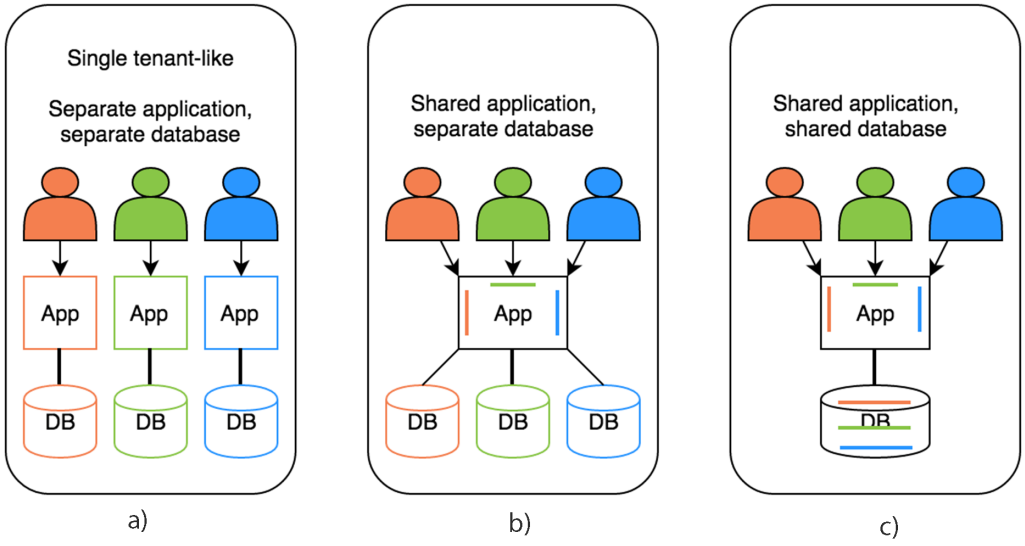
- dostopnost preko spletnega brskalnika: od uporabnikov se ne zahteva namestitve programske opreme na osebni računalnik, ampak do nje dostopajo preko spletnega brskalnika in mobilnih naprav;

- razpoložljivost na zahtevo: za pridobitev dostopa do SaaS produktov uporabnikom ni potrebno skozi prodajni proces, ampak lahko pridobijo dostop samostojno z registracijo novega računa. Uporabniški račun omogoča dostop do aplikacij kjerkoli in kadar koli;

- plačilni pogoji glede na uporabo: zaradi enostavnega procesa pridobitve novega uporabnika tudi pristopni finančni vložek ni velik. Preprosto bi lahko rekli, da uporabnik plačuje uporabnino za programsko opremo na mesečni ali letni ravni. Ko storitve ne potrebuje več, preneha s plačevanjem in izgubi pravico do uporabe programske opreme;

- minimalne zahteve IT: SaaS sistemi ne zahtevajo veliko tehničnega znanja za njihovo nastavitev in delovanje.

Programsko opremo kot storitev je mogoče zaslediti v dveh oblikah, ki ju poznamo kot: enouporabniška arhitektura (angl. single tenant) in večuporabniška arhitektura (angl. multi tenant) [31].



Slika 2.2: Pristopi k načrtovanju arhitekture [13].

Enouporabniška arhitektura

Vsak uporabnik oz. organizacija ima v lasti svojo kopijo izvorne kode aplikacije in popolnoma ločeno podatkovno bazo. Primer enouporabniškega načina načrtovanja arhitekture je prikazan na sliki 2.2.a. Uporabniki si ne delijo računalniških virov, ampak jih uporabljajo ločeno [31]. Ponudnik storitev določi količino računalniških virov, ki jih lahko posamezni uporabnik uporablja. Plačilni modeli storitev običajno od uporabnika zahtevajo, da plača dogovorjeno količino namenskih virov ne glede na izkoriščenost. Izoliranost virov prinaša pozitivne učinke v smislu varnosti, saj se izloči možnost dostopa do napačnih podatkov. Zasedenost virov enega uporabnika ne vpliva na delovanje ostalim uporabnikom. Izdelava varnostnih kopij in obnovitev

podatkov sta enostavna. Z namensko rezerviranimi viri pa so povezani tudi stroški delovanja, ki so v primerjavi z večuporabniško arhitekturo večji [24].

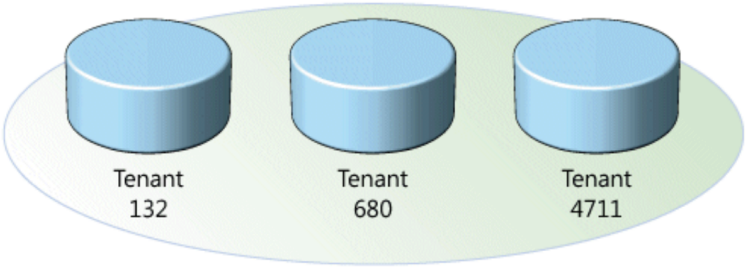
Večuporabniška arhitektura

Vsi uporabniki sistema si delijo eno kopijo izvorne kode aplikacije. Računalniški viri se porazdelijo med vse uporabnike, kar zagotavlja optimalno izkoriščenost sistema [31]. Po finančni plati je to najbolj ugodna rešitev za ponudnika kot tudi za uporabnika. Ponudniki si prizadevajo za čim bolj prijazen in enostaven začetek uporabe storitve, saj so poslovni rezultati neposredno odvisni od količine zadovoljnih uporabnikov. Vzdrževati je potrebno le eno kopijo izvorne kode, zato so popravki in vpeljava novih funkcionalnosti enostavnejši ter se nanašajo na vse uporabnike hkrati. Lastnost skupnih sprememb se slabo odraža v primeru napak, zato si ponudniki ves čas prizadevajo, da programska oprema deluje brezhibno [24]. Med podatki uporabnikov se loči na podatkovnem nivoju, kjer je potrebno zagotoviti, da vsak uporabnik lahko dostopa le do svojih podatkov. Večuporabniška arhitektura se srečuje s problemom izolacije med podatki uporabnikov in v ta namen se uporabljajo trije različni pristopi načrtovanja podatkovnega nivoja. Pristopi načrtovanja večuporabniške arhitekture:

- **Ločene podatkovne baze**

  Najbolj enostaven pristop k varovanju podatkov je, da ima vsak uporabnik svojo podatkovno bazo. Slika 2.2.b prikazuje načrtovanje večuporabniške arhitekture s pristopom ločenih podatkovnih baz. Iz slike 2.3 je mogoče razbrati, da ima vsak uporabnik samostojno podatkovno bazo. Rač-unalniški viri so deljeni na drugih nivojih aplikacije. V tem primeru mora programska oprema poskrbeti, da poveže uporabnika s pravilno podatkovno bazo in je ključno vprašanje zasebnosti podatkov zagotovljeno, saj na tak način uporabniki dostopajo le do svojih podatkov [14]. V primerjavi z enouporabniško arhitekturo je ta pristop finančno bolj učinkovit, saj so računalniški viri na aplikacijskem in logičnem nivoju

deljeni. V splošnem pa je ta pristop še vedno drag, ker mora ponudnik vzdrževati toliko podatkovnih baz kot ima uporabnikov. Pristop ločene podatkovne baze se uporablja v primerih, ko imajo uporabniki visoko politiko o varnosti podatkov in so v ta namen pripravljeni investirati več finančnih sredstev, na primer podjetja, ki hranijo medicinske in bančne podatke.

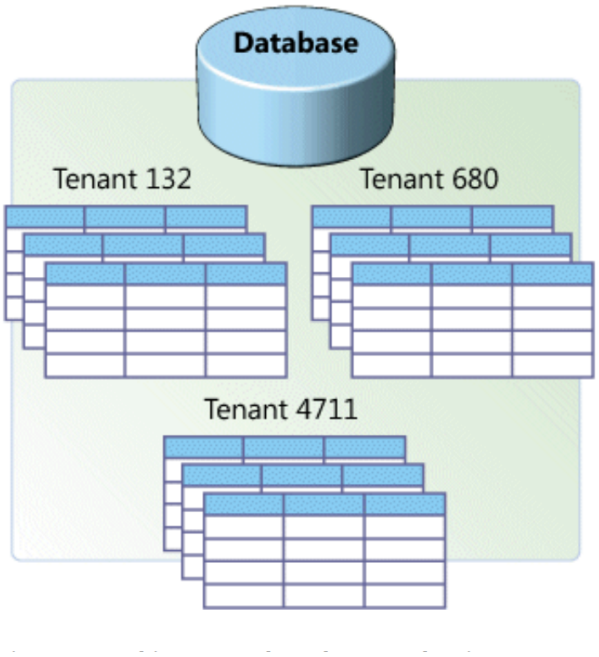


Slika 2.3: Ločene podatkovne baze [14].

- **Skupna podatkovna baza, ločene sheme**

  Drugi pristop vključuje enotno podatkovno bazo za vse uporabnike in ločene podatkovne sheme. V tem primeru podatkovna shema zajema množico tabel, ki pripadajo enemu uporabniku. Zasnova arhitekture po tem pristopu je podana na sliki 2.2.c. Slika 2.4 prikazuje skupine tabel, ki pripadajo različnim uporabnikom znotraj ene podatkovne baze. Ob registraciji novega uporabnika sistem poskrbi, da se ustvari nova skupina tabel, povezanih v shemo. V sistemu se vodi privzeta shema, iz katere se kasneje ustvarjajo kopije novih. Spreminjanje podatkovnega modela vodi v popravljanje privzete sheme in uveljavljanje sprememb na že obstoječih shemah. Ta pristop ponuja zmeren nivo izolacije podatkov za potrebe varnosti. Vidik varnostnih kopij podatkov in morebitne obnove je bolj kompleksen, saj z obnovitvijo podatkov enega uporabnika prepišemo podatke drugega uporabnika, ki so nastali v času od napake do obnovitve [14]. V praksi se takšne situacije rešujejo na način, da

se za čas obnovitve postavi ločena podatkovna baza, iz katere se ročno izvozijo podatki kritičnega uporabnika in se prepišejo nazaj v izvirno podatkovno bazo. Ta pristop se uporablja pri aplikacijah z relativno majhnim številom tabel, ker že v primeru, da imamo 100 tabel na uporabnika lahko začnemo dosegati fizične omejitve strojne opreme.

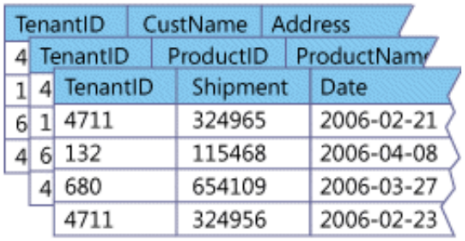


Slika 2.4: Skupna podatkovna baza, ločene sheme [14].

- **Skupna podatkovna baza, skupna shema**

  Tretji pristop vsebuje eno podatkovno bazo in vsi uporabniki uporabljajo iste tabele, kjer vsak podatek vsebuje ključ uporabnika, po katerem se uporabniki razlikujejo. Slika 2.5 prikazuje vpeljavo ključa uporabnika v posamezne tabele. Načrtovanje arhitekture po tem pristopu je mogoče predstaviti s sliko 2.2.c. Po zasedenosti računalniških virov je ta pristop najbolj učinkovit. Posledično je izolacija podatkov zelo nizka in morajo biti razvijalci pri samem razvoju aplikacije ves čas pozorni, da uporabniki nikoli ne morejo dostopati do podatkov drugih uporabnikov.

Varnostne kopije in obnovitve podatkov se izvajajo na podoben način kot pri pristopu s skupno podatkovno bazo in ločenimi shemami. Ta pristop je uporaben v primeru, ko želijo ponudniki gostiti veliko število uporabnikov z majhnimi operativnimi stroški delovanja in uporabniki ne zahtevajo najvišje možne varnosti podatkov [14].



Slika 2.5: Skupna podatkovna baza, skupna shema [14].

2.2 Tehnologije in orodja

Izbor spletnih tehnologij je temeljil na programskih jezikih in orodjih, ki omogočajo hiter in preprost razvoj. V najkrajšem možnem času želimo izdelati osnovno sprejemljivo rešitev (angl. Minimum Viable Product – MVP) in jo ponuditi planinskim društvom.

2.2.1 Ruby

Ruby je dinamičen splošno namenski programski jezik [30]. Leta 1993 ga je zasnoval in razvil japonski programer Yukihiro Matsumoto z idejo, da želi razviti programski jezik, ki bo naraven ljudem. Zaradi svoje enostavnosti, elegantne sintakse in učinkovitosti je pritegnil zanimanje številnih programerjev. Najbolj razširjen je pri razvoju spletnih aplikacij. Z leti obstoja je pridobil mnogo razvijalcev, ki neprestano skrbijo za razvoj novih funkcionalnosti in izboljšav. Je objektno usmerjen, vse v programskem jeziku je objekt, ki mu je mogoče pripisati svoje lastnosti in akcije. Ruby ima lastnost popolne fleksi-

bilnosti, dopušča poseganje in spreminjanje samega delovanja programskega jezika.

2.2.2 Ruby on Rails

Ruby on Rails (RoR) je ogrodje za izdelavo spletnih aplikacij [28]. Napisan je v programskem jeziku Ruby in deluje po principu MVC (angl. Model-View-Controller). Spodbuja arhitekturni stil REST (angl. Representational State Transfer) za spletne storitve. Razvijalci so poskrbeli, da RoR vsebuje vse potrebne gradnike za enostavno izdelavo modernih spletnih aplikacij. Temelji na dveh osnovnih principih, ki omogočata hiter razvoj in enostavno vzdrževanje aplikacij.

- **Dogovor pred nastavitvami** (angl. Convention over configuration): Ruby on Rails ponuja privzete nastavitve, ki so jih skrbno zbrali razvijalci ogrodja. Z upoštevanjem pravil, ki določajo poimenovanje razredov, metod, atributov in določene arhitekturne strukture, razvijalcem aplikacij prihrani veliko časa in odločitev. Razvijalci tako namenijo več časa reševanju domenskih problemov, namesto da za nov projekt vsakič izgubljajo dodaten čas za nastavitve. Pozitivni učinek tega principa se odraža na položnejši učni krivulji za nove razvijalce. Vsako nastavitev oz. dogovor je možno po potrebi tudi enostavno prilagoditi [27].
- **Ne ponavljaj se** (angl. Don't repeat yourself): ideja tega principa je, da se enaka programska koda ne ponavlja na različnih mestih aplikacije. Programski jezik Ruby omogoča visoko povezljivost med komponentami in enostavno abstrakcijo programske kode. Omogoča lažje vzdrževanje aplikacije in zmanjša verjetnost napak [27].

REST

REST (angl. Representational State Transfer) je arhitekturni stil, ki označuje standard za komunikacijo med računalniškimi napravami na svetovnem spletu. Strežnik in odjemalec delujeta ločeno in ne hranita medsebojnega stanja. V

primerjavi z alternativnimi rešitvami, kot je SOAP (angl. Simple Object Access Protocol), je REST po zmogljivosti manj zahteven in enostavnejši način za izmenjavo spletnih virov [25]. Spletne vire v tem primeru zajemajo podatki in funkcionalnosti spletne aplikacije. Nad njimi se izvajajo osnovne operacije ustvarjanja, branja, posodabljanja in brisanja podatkov ali z eno besedo CRUD. Ruby on Rails komponenta Active Resource izvaja preslikavo CRUD akcij v metode protokola HTTP (POST, GET, PUT, DELETE).

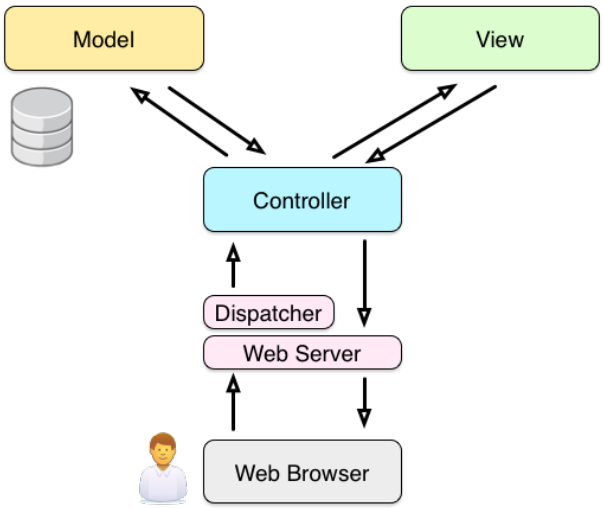
MVC

Model, pogled, nadzornik ali krajše MVC (angl. Model-View-Controller) je arhitekturni vzorec, ki aplikacijo deli na tri plasti: podatkovno, logično in predstavitveno.

- Model upravlja stanje podatkov v aplikaciji. Včasih je stanje prehodno in traja le nekaj transakcij med odjemalcem in strežnikom, včasih pa je to stanje trajno in se ga shrani v podatkovno bazo. Model ne opravlja samo naloge shranjevanja podatkov, temveč vsebuje tudi poslovna pravila, ki veljajo za te podatke in preverja njihovo veljavnost. V ogrodju Ruby on Rails model razširja osnovni razred komponente Active Record, katera implementira ORM (angl. Object-relational mapping). ORM skrbi za preslikavo elementov podatkovne baze v objekte in nazaj. Preslikovanje poteka tako, da tabele iz podatkovne baze preslika v razrede, vrstice podatkov v objekte, atribute tabel pa v atribute razredov. Active Record komponenta omogoča naslednje funkcionalnosti [22]:
  - predstavitev modelov in njihovih podatkov;
  - predstavitev relacij in medsebojne odvisnosti med podatki;
  - preverjanje veljavnosti podatkov pred hranjenjem;
  - izvedbo operacije podatkovne baze v objektno orientiranem načinu.
- Nadzornik (angl. Controller) skrbi za komunikacijo. Sprejme zahtevek in modelu posreduje vhodne podatke. Pridobljene izhodne podatke

modela pošlje naprej pogledu oz. predstavitveni plasti. Grafični prikaz delovanja vzorca MVC v ogrodju RoR je predstavljen na sliki 2.6. V Ruby on Rails ogrodju je nadzornik implementiran v komponenti Action Controller. Za procesiranje zahtevkov skrbi komponenta Action Dispatcher, ki posreduje vhodne parametre odgovorni metodi v nadzorniku. Komponenta Action Controller omogoča tudi upravljanje s sejami in preusmeritvami.

- Pogled (angl. View) skrbi za prikaz podatkov, posredovanih s strani nadzornika. Cilj pogleda je izdelati dokument HTML in ga poslati končnemu uporabniku za prikaz v spletnem brskalniku, kot je to prikazano na sliki 2.6. Za pogled je odgovorna komponenta Action View, katere glavna naloga je, da na podlagi predloge in pomožnih funkcij izdela datoteko HTML. Pomožne funkcije omogočajo enostavne operacije nad podatki s programskim jezikom Ruby. Sintaksa za izdelavo predlog ERB (angl. Embedded Ruby) omogoča vstavljanje podatkov in pomožnih funkcij v datoteko HTML.



Slika 2.6: Arhitekturni vzorec MVC v RoR [12].

Vzorec MVC narekuje strukturirano ločevanje programske kode in po-

sledično doprinese k bolj pregledni programski kodi ter hitrejšemu razvoju. Ruby on Rails vsebuje skupino različnih komponent, ki razvijalcem ponujajo odskočno desko za razvoj funkcionalnosti in s tem prihranijo čas, da jim ni potrebno vedno znova graditi temeljev za vsak nov projekt. Ostale Ruby on Rails komponente so [27]:

- Action Cable omogoča gradnjo funkcionalnosti v realnem času na podlagi protokola WebSocket [26];
- Action Mailer skrbi za pošiljanje elektronskih sporočil iz aplikacije;
- Active Job izvaja naloge procesiranja v ozadju;
- Active Storage skrbi za delo z datotekami;
- Active Support razširja standardno knjižnico jezika Ruby in nudi uporabne funkcije za delo z datumi, časom in lokalizacijo.

Programski jezik Ruby ponuja možnost vključevanja knjižnic, v Ruby svetu jih imenujejo *gem*. Za razvoj in vzdrževanje knjižnic skrbi skupnost odprtokodnih razvijalcev. Ogrodje RoR podpira idejo razvoja aplikacij s pomočjo vključevanja knjižnic, saj s tem razvijalci prihranijo čas in se bolj posvetijo reševanju domenskega problema. Knjižnice običajno zajemajo generične funkcionalnosti, ki jih je mogoče uporabiti v mnogih aplikacijah, na primer: registracija in prijava uporabnikov, nastavljanje uporabniških dostopov do vsebine, generiranje dokumentov različnih formatov.

2.2.3 Heroku

Heroku [7] je platforma kot storitev v računalniškem oblaku za gostovanje spletnih aplikacij. Uporablja 12-nivojsko metodologijo za gradnjo programske opreme [1]. Je priljubljena izbira pri razvijalcih, saj omogoča hitro in enostavno postavitev gostovanja spletnih aplikacij brez potrebne predhodne konfiguracije spletnega strežnika.

2.2.4 Bootstrap

Bootstrap [16] je odprtokodno ogrodje, ki združuje osnovne tehnologije razvoja spletnih aplikacij na strani odjemalca. Ključni prednosti ogrodja sta zagotavljanje strukturirane arhitekture za razvoj in enostavna prilagoditev za naprave z različno velikostjo zaslonov. Bootstrap je zgrajen iz tehnologij:

- HTML (angl. Hyper Text Markup Language) je označevalni jezik za izdelavo spletnih strani. Predstavlja osnovo zgradbo spletnega dokumenta oz. spletne strani, hkrati pa predpisuje tudi semantični pomen posameznih delov dokumenta;
- CSS (angl. Cascading Style Sheets) omogoča nastavljanje videza elementov HTML;
- JavaScript je skriptni programski jezik, ki se interpretira na strani odjemalca. Uporablja se predvsem za izboljšanje uporabniške izkušnje. Omogoča, da preko DOM (angl. Document-Object Model) predstavitve dokumenta dostopamo do elementov HTML in jih spreminjamo.

2.2.5 Sidekiq

Ogrodje Sidekiq [23] omogoča procesiranje opravil v ozadju. Spletna aplikacija pošlje strežniku Sidekiq navodila za izvajanje novih opravil, ki se naložijo v čakalno vrsto Redis [21]. Uporablja se v primeru izvajanja performančno zahtevnih opravil, asinhrono glede na uporabnikovo interakcijo z aplikacijo. Različni tipi opravil so razporejeni v samostojne delovne procese, posamezna opravila se izvajajo v svoji niti in s tem se doseže sočasno izvajanje z uporabo več niti na proces.

2.2.6 PostgreSQL

PostgreSQL je odprtokodni objektno-relacijski sistem za upravljanje s podatkovnimi bazami [18]. Vsebuje številne napredne funkcionalnosti, ki jih je

mogoče zaslediti v komercialnih podatkovnih bazah. Napredna funkcionalnost združevanja podatkovnih tabel v skupine, imenovane sheme [20], je pri razvoju programske opreme kot storitev po arhitekturnem vzorcu večuporabniških aplikacij še posebej koristna.

2.3 Planinska društva in obstoječe rešitve

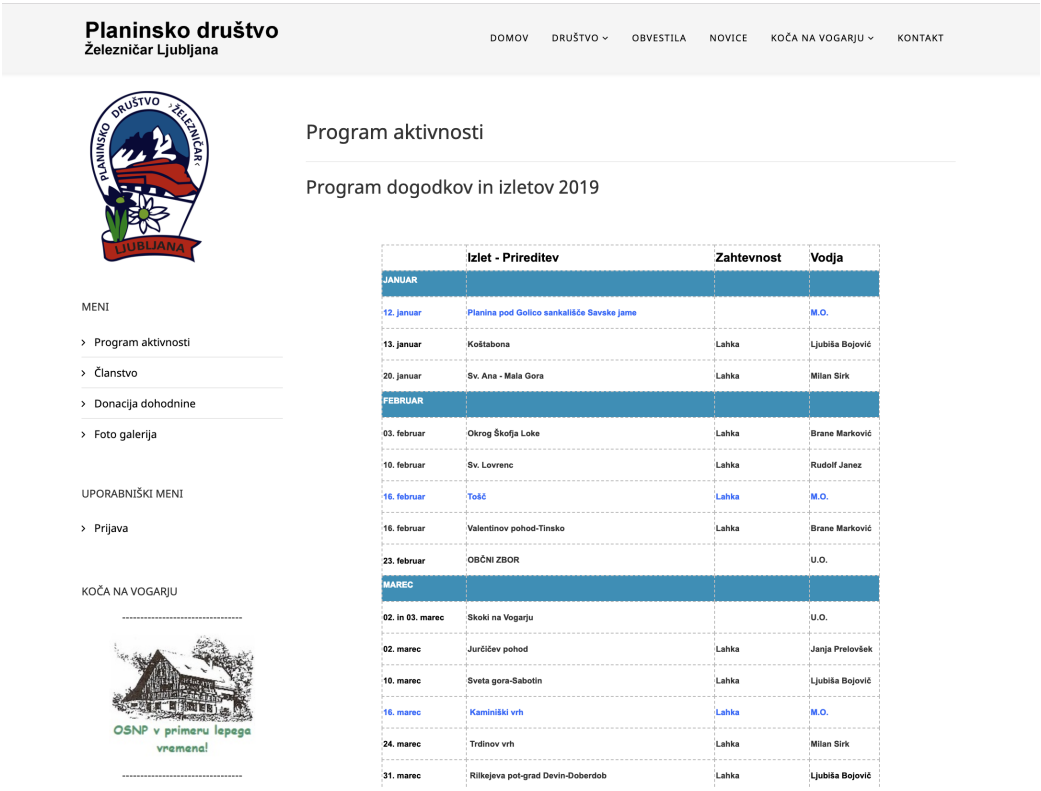
Planinska zveza Slovenije (PZS) je ena največjih in najbolj množičnih nevladnih, prostovoljskih organizacij v Sloveniji. PZS združuje 289 [33] prostovoljskih in neprofitnih planinskih društev, ki skupno vključujejo 58.413 članov. Ena od številnih aktivnosti društev je organizacija s planinstvom povezanih dogodkov. Večina organiziranih dogodkov so planinski izleti v sredogorje in visokogorje. Cilji izletov zajemajo vse različne možnosti obiskov okoliških hribov pa tudi do gorstev v tujini. Ostali dogodki, ki jih društva še organizirajo, so planinski tabori, orientacijska tekmovanja, zimovanja, izobraževanja, planinski ples. Planinsko društvo sestavljajo strokovni delavci in člani društva. Med strokovne delavce štejemo predsednika, upravni odbor, tajništvo in planinske vodnike.

Organizacija dogodkov zahteva veliko predpriprav. Planinski vodnik si zamisli izlet na določen cilj, za katerega mora pripraviti razpis, ki vsebuje podatke o vrsti izleta, datum, uro in kraj odhoda, težavnost, predviden čas hoje, koliko višinskih metrov bo potrebno premagati, vrsto prevoza, strošek udeležbe, število prostih mest ter opis poti in področja izleta, s katerim pritegne pohodnike. Društvo potrdi izlet in ga umesti v plan izletov društva in objavi na oglasni deski ali spletni strani društva. Do predvidenega datuma za prijavo se lahko člani društva prijavijo na izlet preko telefona ali osebno v pisarni društva. Stroške udeležbe lahko poravnajo ob prijavi v pisarni ali na dan izleta. Društvo vodi evidenco organiziranih dogodkov, seznam prijavljenih udeležencev, koordinira število mest za prevoz in vodi finančno poročilo o dogodku. Na dan izleta mora vodnik v pisarno društva po seznam udeležencev, ker ga potrebuje na samem izletu. To so osnovne funkcionalnosti,

ki jih bo morala aplikacija zajemati, da jo lahko uvedemo v delovanje društva.

Društva skrbijo za objavo razpisanih izletov. V preteklosti je bil glavni medij za objavo oglasna deska društva, kar nekatera društva uporabljajo še danes. Kasneje so društva uvedla statične spletne strani brez zaledja in podatkovne baze. Urejanje vsebine na takšni spletni strani ni mogoče brez posegov razvijalcev. V veliko primerih je to privedlo do neažurnosti vsebine. Sčasoma so se uveljavili sistemi za upravljanje vsebine (angl. content management system). Razvoj spletne strani s sistemom za upravljanje vsebine je enostaven in finančno dosegljiv za večja društva. Strokovni delavci društva lahko samostojno posodabljajo besedila na spletni strani preko urejevalnikov besedil WYSIWYG. Sistemi za upravljane vsebine odpravijo potrebo prisotnosti razvijalcev za urejanje besedil, še vedno pa je njihova uporaba zelo omejena za potrebe društva. Ne omogočajo prilagojenih vnosnih mask, uporabe zemljevida za prikaz poti na izletu, obveščanja članov preko elektronske pošte, elektronske prijave na dogodke in izdelave različnih evidenc za vodenje društva. Na tržišču ni mogoče zaslediti spletne aplikacije, ki bi ponujala ustrezne rešitve, prilagojene za različne potrebe planinskih društev.

Po pregledu obstoječih spletnih strani planinskih društev [33] smo ugotovili, da ima velika večina društev objavljen letni plan dogodkov. Pogosto smo zasledili plan dogodkov v tabelarični obliki, kot je to razvidno iz primera Planinskega društva Železničar na sliki 2.7. Predstavljeni so le osnovni podatki dogodka: datum, ime dogodka, zahtevnost in vodja. Obiskovalce spletne strani pa zanima veliko več informacij o samem dogodku. Dodatno slikovno gradivo in interaktivni zemljevidi bi obiskovalcem obogatili uporabniško izkušnjo ter pripomogli k odločitvi o udeležbi. Zaradi pomanjkanja programskih rešitev na tem področju društvom ni omogočena objava dodatnih informacij o posameznih dogodkih.



Slika 2.7: Letni plan dogodkov Planinskega društva Železničar [11].

Poglavje 3

Arhitektura in razvoj aplikacije

Z izdelavo programske opreme želimo planinskim društvom izboljšati delovni proces in pohodnikom približati informacije. Tradicionalen pristop razvoja informacijskih rešitev bi narekoval naslednje korake: najprej stopimo v stik z organizacijo in preučimo njihovo delovanje; na podlagi ugotovitev naredimo načrt rešitve in izdelamo aplikacijo; kasneje ugotovimo, da se za podobno aplikacijo zanimajo tudi ostala planinska društva ter naredimo kopijo izvorne programske kode aplikacije in jo glede na zahteve posameznega društva prilagodimo; v določenem trenutku imamo za vsako planinsko društvo svojo aplikacijo, ki teče na ločenem strežniku; pri večjem številu organizacij upravljanje s strojno in programsko opremo postane neobvladljivo in predstavlja previsoko finančno breme za večino planinskih društev.

Primernejši pristop je izbrati model gradnje programske opreme kot storitev po načinu večuporabniške arhitekture, kjer si planinska društva delijo strojno in programsko računalniško opremo. Programsko opremo, ki je v našem primeru spletna aplikacija, je potrebno zasnovati tako, da bo modularna in bo vključevala zahteve večine društev.

3.1 Opis funkcionalnosti

Po pregledu vzorčnega planinskega društva je mogoče ugotoviti, da obstajata dva tipa uporabnikov aplikacije. Prvi so strokovni delavci planinskih društev: planinski vodniki, upravni odbor, tajništvo in predsednik. Te osebe so odgovorne za vsebino in delovni proces organizacije. Drugi tip uporabnikov so širša javnost, pohodniki in člani društva. Zato je smiselno aplikacijo razdeliti na administratorski del in predstavitveno spletno stran planinskega društva.

Pomembna naloga planinskih društev je organizacija raznovrstnih dogodkov, zato bo tudi večina aplikacije zajemala z dogodki povezane funkcionalnosti. Društva organizirajo več tipov dogodkov: planinski izlet, planinski tabor, orientacijsko tekmovanje, zimovanje, izobraževanje, planinski ples in zbor. Ker ne moremo vnaprej predvideti vseh vrst dogodkov, ki jih različna društva izvajajo, je smiselno zasnovati sistem prilagodljivih predlog. Narava dogodka narekuje podatke, ki jih mora predloga vsebovati. Kot upravljavci aplikacije se želimo izogniti urejanju vsebinskega dela podatkov, saj je pri večjem številu društev obseg dela prevelik. V ta namen mora aplikacija društvom omogočati popoln nadzor in upravljanje z vsebino. Predloge je mogoče sestaviti po meri iz različnih tipov polj: besedilo, datum, slika in zemljevid. Dogodki se izdelajo na podlagi izbrane predloge. Izdelava mora biti enostavna in uporabniku prijazna.

Drugi del aplikacije zajema spletno stran za širšo javnost. Vsako planinsko društvo mora imeti samostojno spletno stran, saj obiskovalce zanima vsebina le za specifično društvo. Glavna zanimivost vsebine so prihajajoči dogodki, zato morajo biti vidni na prvi strani v obliki seznama. S klikom na dogodek v seznamu se odpre podstran z vsebino o samem dogodku.

Obiskovalci spletne strani društva imajo možnost registracije in prijave, ki omogoča, da na elektronski naslov prejmejo informacije o novo objavljenih dogodkih. S prijavo na spletno stran jim je omogočena tudi elektronska prijava na dogodek. Z elektronskim zbiranjem prijav bodo društva samodejno pridobila seznam udeležencev dogodka.

3.2 Arhitektura

Spletno aplikacijo za urejanje in prikaz dogodkov bomo izdelali v oblaku in programsko opremo ponudili kot storitev planinskim društvom. Zaradi učinkovitejše rabe računalniških virov, enostavnejšega vzdrževanja in finančno bolj ugodnega poslovnega modela za društva bomo aplikacijo zasnovali na principu večuporabniške arhitekture. Pri načrtovanju in razvoju tovrstne aplikacije je mogoče zaznati tri večje tehnološke izzive: varno ločevanje podatkov med planinskimi društvi, preklapljanje med podatkovnimi shemami in dinamične podatkovne strukture.

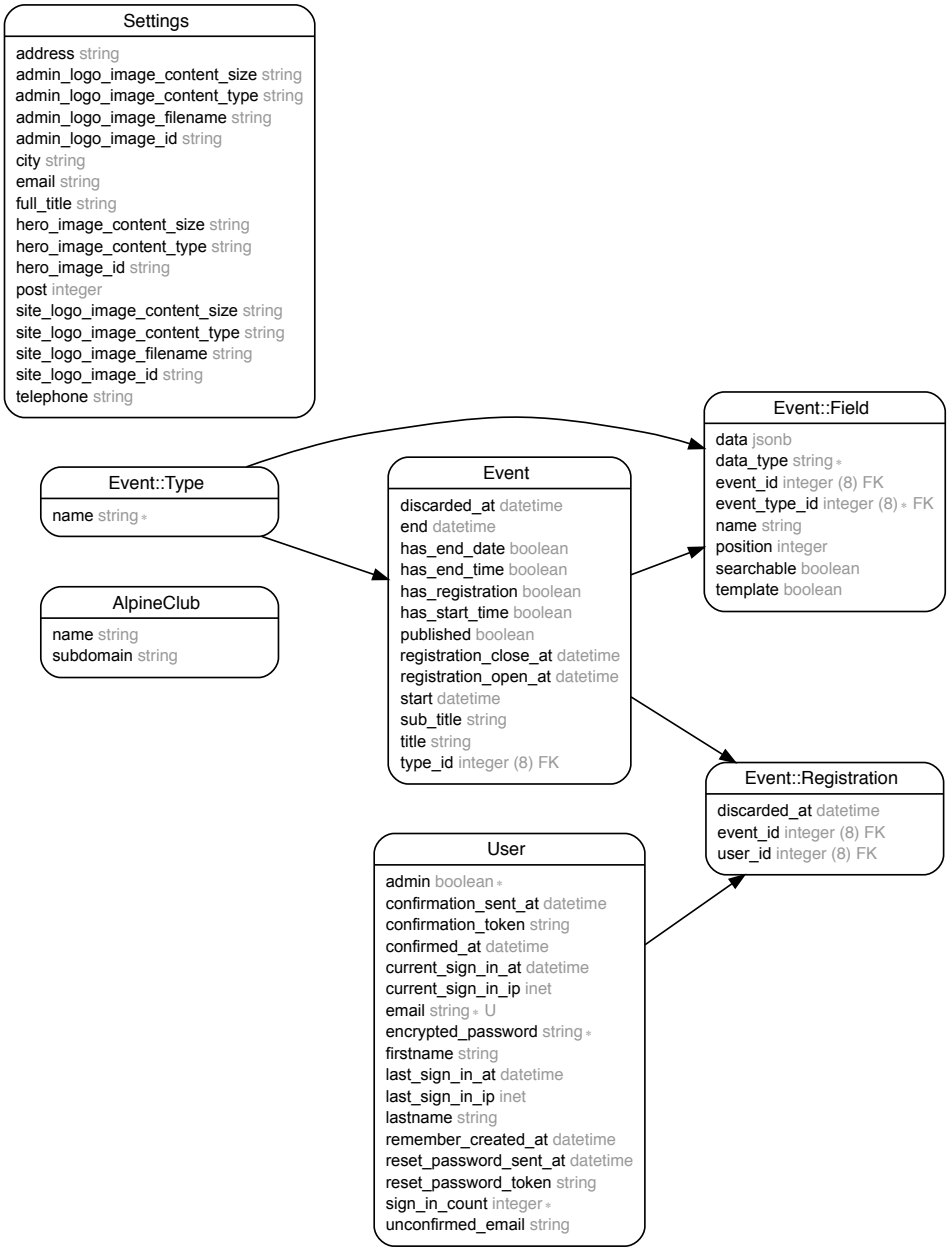
3.2.1 Ločevanje podatkov

Večuporabniško arhitekturo je mogoče zasnovati po treh različnih principih, ki so opisani v poglavju 2.1.3. V našem primeru je najbolj smiselna uporaba principa skupne podatkovne baze in ločene sheme, saj ponuja zmeren nivo izolacije podatkov za potrebe varnosti ter dobro izkoriščenost računalniških virov. Podatkovna baza PostgreSQL ima dobro podporo za delo z večjim številom podatkovnih shem. Knjižnica Apartment [3] izkorišča sheme za ločevanje skupine tabel, ki pripadajo eni organizaciji. Z zahtevkom po podatkih za specifično društvo, knjižnica Apartment poskrbi za preklop sheme. Podatkovna baza zagotavlja, da se vse poizvedbe SQL izvajajo na tabelah trenutne sheme.

Podatkovni model

Slika 3.1 prikazuje načrt podatkovnega modela za aplikacijo. Tabeli *AlpineClub* in *Settings* vsebujeta osnovne informacije o planinskih društvih. Predloge so hranjene v tabeli *Event::Type* in pripadajoča ustvarjena polja s strani strokovnih delavcev društva v tabeli *Event::Field* z zastavico template. Glavna tabela v podatkovnem modelu je *Event*, ki vsebuje polja *Event::Field*, določena v predlogi. Oba tipa končnih uporabnikov, tako strokovni delavci, kot tudi registrirani uporabniki spletne strani društva, so hranjeni v tabeli *User* in

preko povezovalne tabele *Event::Registration* je omogočena prijava na dogodke. Tabela *Settings* hrani kontaktne podatke, povezave na logotipe in naslovno sliko za posamezno društvo.



Slika 3.1: Podatkovni model aplikacije.

3.2.2 Preklapljanje med podatkovnimi shemami

Pri večuporabniški arhitekturi obstaja več načinov preklapljanja med podatkovnimi shemami. V interesu planinskih društev je, da obiskovalci spletne strani vidijo le vsebino njihovega društva, zato je najbolj smiselno ločiti že sam dostop do spletne strani. Vsako društvo dobi svojo poddomeno, na kateri obiskovalci dostopajo le do vsebine tega društva. Krovna domena aplikacije je *http://planinsko-drustvo.si*. Posamezno društvo dostopa do svoje spletne strani preko *http://ime-drustva.planinsko-drustvo.si*. V primeru, da planinsko društvo že ima svojo domeno, jo enostavno preusmeri z zapisom *CNAME*. Za zagotavljanje prikaza pravilne vsebine mora aplikacija iz URL zahtevka razbrati, do katerega društva želi obiskovalec dostopati in preklopiti na pravilno podatkovno shemo. V aplikacijo smo vključili knjižnico Apartment, ki s pravilnimi nastavitvami opravlja nalogo preklapljanja med podatkovnimi shemami na podlagi poddomene v zahtevku.

```
1 Apartment.configure do |config|
2   config.excluded_models = %w[AlpineClub]
3   config.tenant_names = -> { AlpineClub.pluck :subdomain }
4 end
5
6 Rails.application.config.middleware.use
    Apartment::Elevators::Subdomain
7 Apartment::Elevators::Subdomain.prepend
    RescuedApartmentMiddleware
```

Programska koda 3.1: Nastavitve *Apartment* knjižnice.

Programska koda 3.1 se nahaja v inicializacijski datoteki knjižnice Apartment. V nastavitvenem bloku smo knjižnici določili, da naj se model *AlpineClub* ne vključuje v kopije privzete podatkovne sheme, saj vsebuje podatke vseh gostujočih planinskih društev. S pomočjo funkcije *lambda* se ustvari poizvedba, kjer se določi vir, iz katerega knjižnica črpa podatke o shemah gostujočih društev. Z naslednjim delom programske kode se aplikaciji nastavi

vmesna programska koda (angl. middleware) znotraj knjižnice, imenovana *Elevator*, ki poskrbi za preklop podatkovne sheme glede na prebrano poddomeno v zahtevku. Za vmesno programsko kodo smo naredili razširitev klica funkcije in v primeru, da *Elevator* ni našel sheme, skladne z zahtevano, obiskovalca preusmeri na spletno stran za napake s kodo 404. S takšnim načinom preklopa med društvi moramo zagotoviti, da zahtevki za spletno stran vsebujejo poddomeno, sicer bo zahtevek na *http://planinsko-drustvo.si* preusmerjen na stran za napake. Za zagotovitev pravilnega delovanja smo na nivoju RoR usmerjevalnika *Action Dispatcher* uvedli omejitve za poti, določene v datoteki *routes.rb*. Programska koda 3.2 prikazuje omejitev poti.

```
1 Rails.application.routes.draw do
2   constraints(SubdomainConstraint) do
3     root to: "events#index"
4     resources :events
5     ...
6   end
7 end
```

Programska koda 3.2: Omejitev poti aplikacije v datoteki *routes.rb*.

Pravila omejitev iz bloka programske kode 3.3 zahtevajo, da zahtevki vsebujejo poddomeno. Izločijo se zahtevki, ki vsebujejo poddomeno *public*, ker je to privzeta shema in *www*, ker je rezervirana beseda za preusmeritev.

```
1 class SubdomainConstraint
2   def self.matches?(request)
3     request.subdomain.present? &&
4     ["www", "public"].exclude?(request.subdomain)
5   end
6 end
```

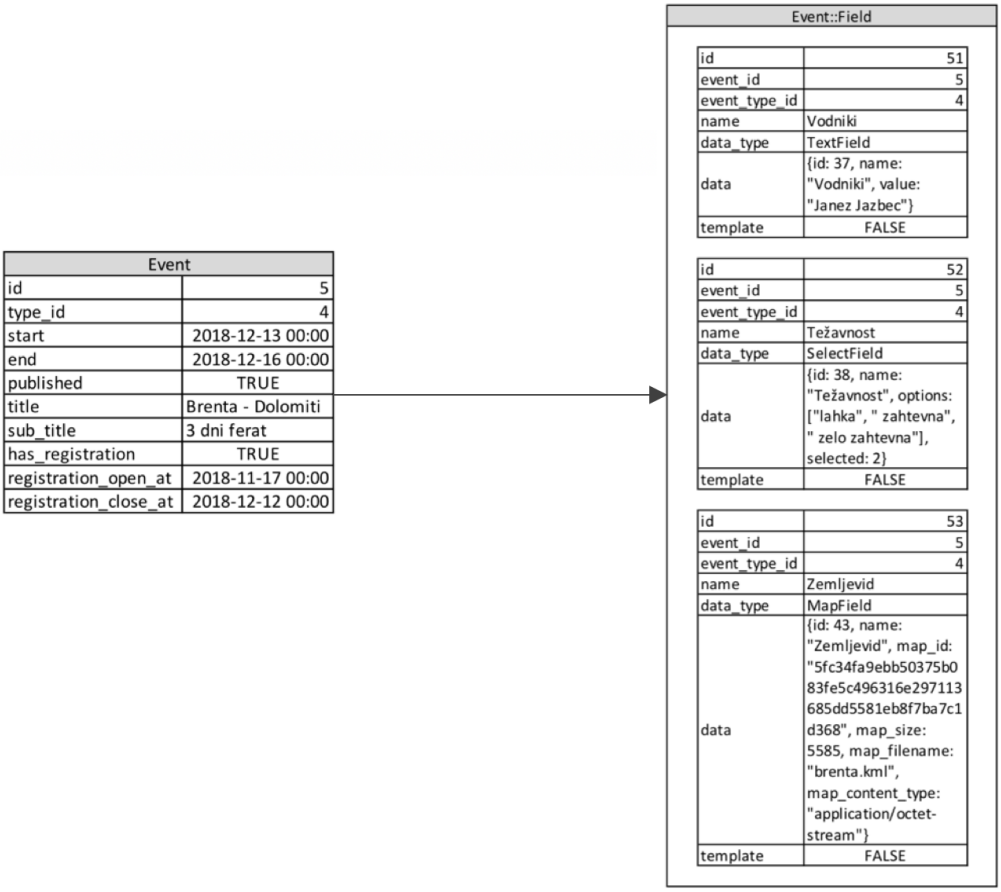
Programska koda 3.3: Pravila za omejitev zahtevkov.

3.2.3 Dinamične podatkovne strukture dogodkov

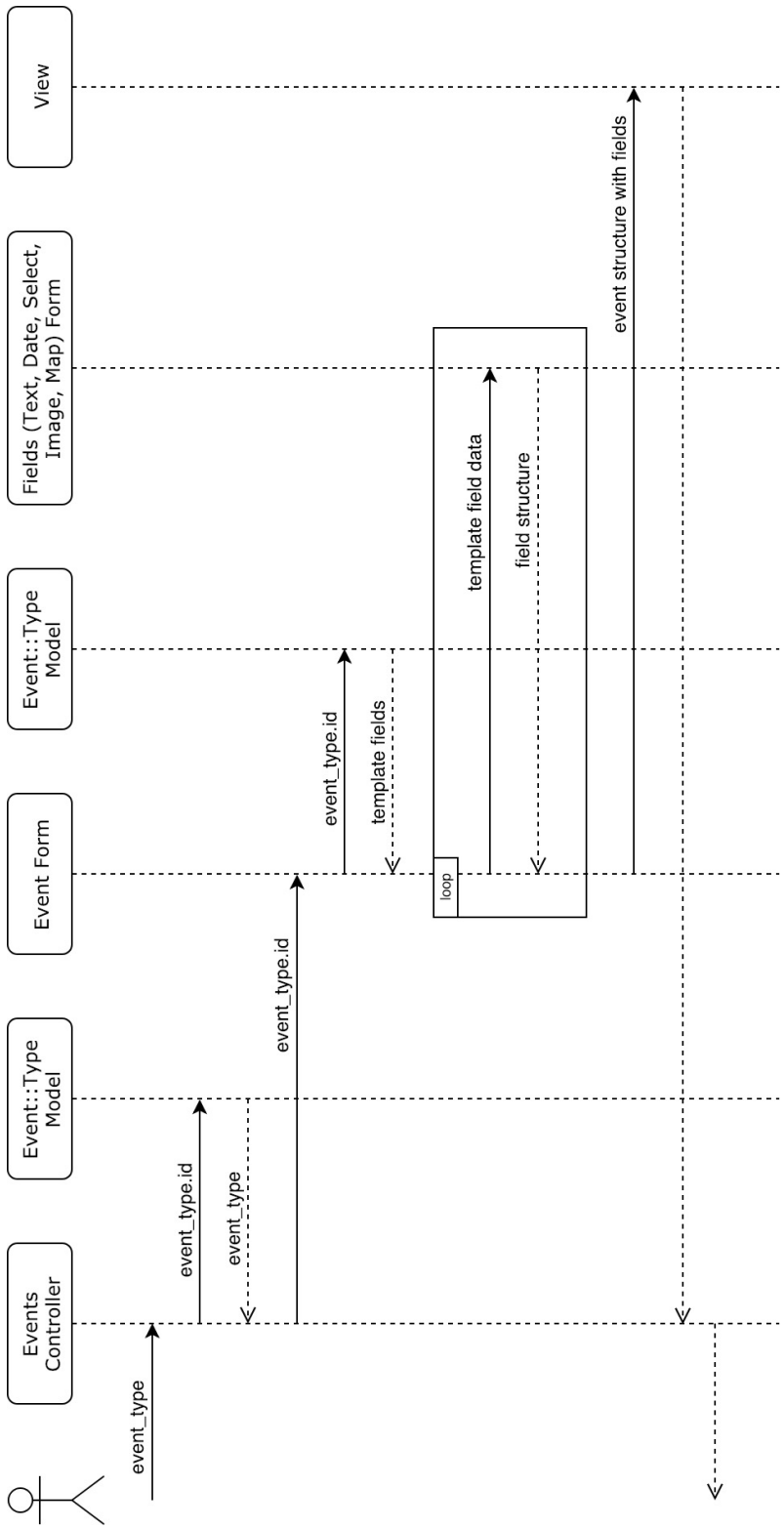
V modelu programska oprema kot storitev (SaaS) je cilj ponudnika izdelati aplikacijo, ki lahko zajame čim večjo populacijo potencialnih strank. Aplikacija mora biti narejena tako, da uporabnika ne omejuje pri njegovem delu, ampak omogoča toliko mero prilagodljivosti, da bo koristna za njegove potrebe. Aplikacija za planinska društva v ta namen vsebuje predloge dogodkov, ki društvom ponujajo tako prilagajanje vsebine, kakor tudi strukturo posamezni vsebini. To pomeni, da se lahko strukture in vsebine dogodkov razlikujejo pri vsakem od gostujočih društev brez dodatnih posegov razvijalcev. Sestavljanje predloge je videti tako, da uporabnik v predlogo dodaja nova polja. Izbira lahko med tipom polja: besedilo, datum, polje z izbiro, sliko ali zemljevidom. Vsako polje poljubno poimenuje in določi opcijske atribute, na primer možne izbire pri polju z izbiro. Ime predloge se shrani v model *Event::Type*, vsa pripadajoča polja pa v *Event::Field* z označbo, da gre za polja, ki pripadajo predlogi.

Na sliki 3.2 je primer podatkovne strukture dogodka. Vsako polje dogodka v tabeli *Event::Field* ima poleg osnovnih podatkov, ki opisujejo polje, tudi vrstico *data*. V tej vrstici je shranjen dinamičen del strukture, ki definira vsebino in nastavitve za prikaz polja. Relacijske baze zahtevajo, da se strukture podatkov določijo pri načrtovanju. Za podatke, kjer struktura vnaprej ni znana, je uporaba otežena, saj zahteva ustvarjanje veliko relacij, za vsako spremembo ali dodajanje novega tipa polja pa je potreben poseg v strukturo podatkovne baze. Za enostavno hranjenje dinamičnih struktur podatkov smo uporabili PostgreSQL podatkovni tip JSON (JavaScript Object Notation) [19].

Slika 3.3 prikazuje sekvenčni diagram interakcije med programskimi razredi in komponentami aplikacije za prikaz vnosne maske za izdelavo novega dogodka. Za ustvarjanje novega dogodka strokovni delavec društva izbere predlogo, za katero želi ustvariti nov dogodek. Zahtevek se pošlje na *Event Controller* z identifikacijsko številko izbrane predloge. Od *Event::Type Model* se pridobi več podatkov o izbrani predlogi. Izvajanje se nadaljuje s klicem *Event Form* razreda, ki specificira strukturo za vnosno masko dogodkov. *Event*



Slika 3.2: Primer podatkovne strukture dogodka.

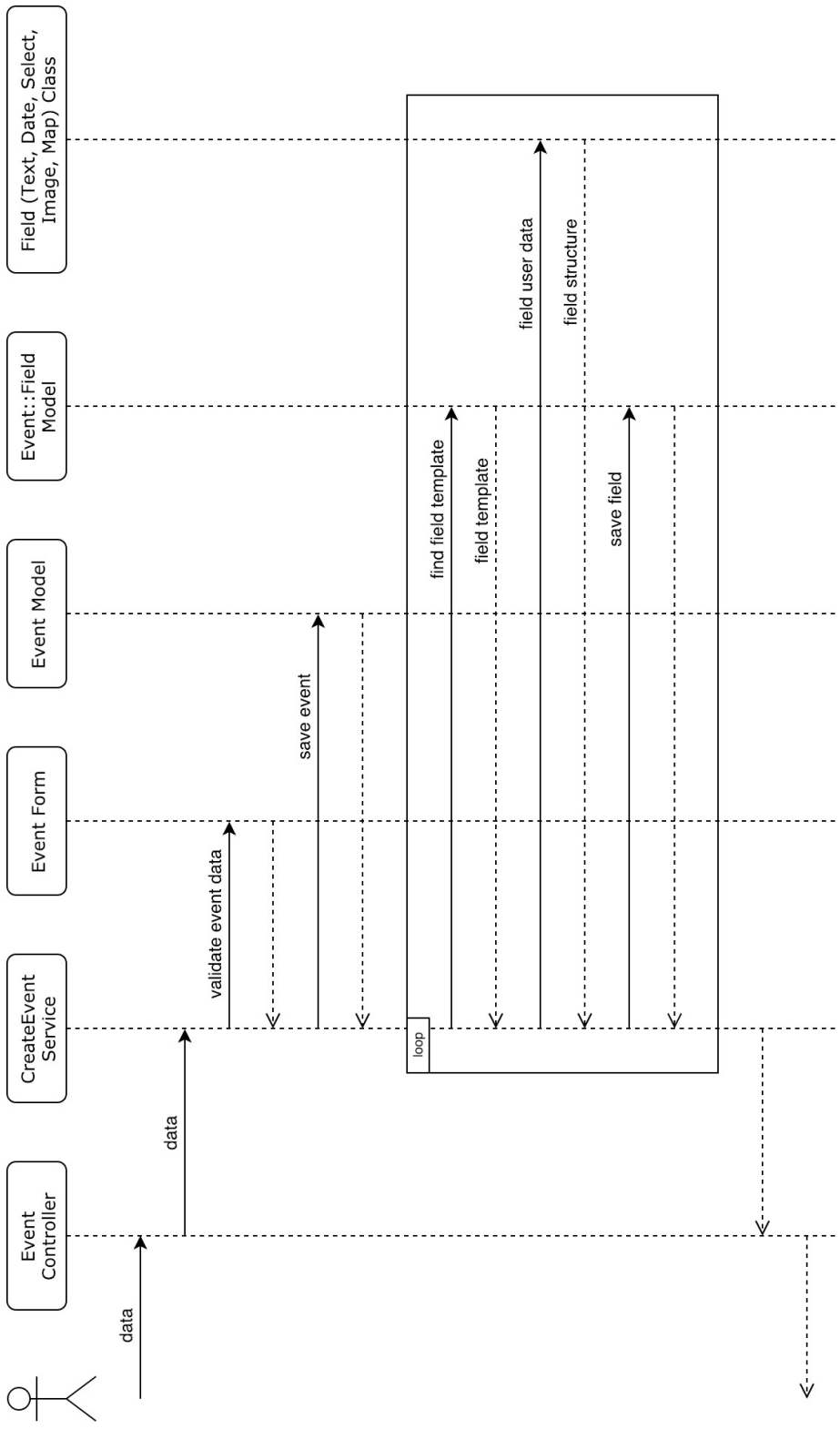


Slika 3.3: Sekvenčni diagram za izdelavo vnosne maske dogodkov.

Form je običajen Ruby razred (angl. Plain Old Ruby Object), ki razširja razred *ActiveModel::Model* za uporabo validacije in *Virtus.model* za določitev atributov dinamične strukture. *Event Form* pridobi vsa polja iz predloge. Za vsako polje iz predloge se glede na tip polja pridobi razred za izdelavo strukture. Vsak določen tip polja (besedilo, datum, polje z izbiro, slika, zemljevid) je definiran v svojem običajnem Ruby razredu. Razredi za gradnjo strukture posameznih polj prav tako razširjajo razred *ActiveModel::Model* in *Virtus.model*. Zgrajeno prazno strukturo dogodka *Event Form* pošlje v *View*, kjer se izdela HTML dokument z vnosno masko. Izdelan HTML dokument se posreduje odjemalcu kot odgovor na zahtevek in brskalnik prikaže vsebino.

Sekvenčni diagram na sliki 3.4 prikazuje potek interakcij med razredi potem, ko uporabnik vnese vse podatke za ustvarjanje novega dogodka in ko ti v obliki zahtevka prispejo do *Event Controller*. Ustvarjanje novega dogodka zahteva veliko poslovne logike in shranjevanje podatkov v več različnih tabel. Z implementacijo tovrstne poslovne logike bi po vzorcu MVC priporočil kršili pravilo vitkih nadzornikov (angl. controller) in enostavnih modelov. Namesto tega smo ustvarili storitveni razred (angl. service object) *CreateEvent Service*, ki vsebuje vso poslovno logiko za ustvarjanje novega dogodka in dodaja nove zapise podatkov v različne podatkovne tabele. *CreateEvent Service* najprej ustvari nov dogodek s klicem modela za dogodke. Kot povratne informacije se za novo ustvarjen dogodek vrne identifikacijska številka, ki bo služila kot referenca za ustvarjanje polj dogodka. Za vsako polje dogodka se najprej poišče predloga polja v tabeli *Event::Type Model* ter se naredi kopija. Vnesene podatke uporabnika se pošlje pripadajočemu razredu (*TextField*, *DateField*, *SelectField*, *ImageField*, *MapField*) za pripravo dinamične strukture podatkov polja. Kopiji polja se pripne dinamična struktura s podatki in polje se zapiše v *Event::Field Model* z referenco na dogodek. Uporabnika se obvesti o uspešnem vnosu novega dogodka.

Za tip polja zemljevid se zahteva od uporabnika, da naloži datoteko v formatu *kml* (angl. Keyhole Markup Language) [5], kjer so shranjene točke za izris začrtane poti na zemljevidu. Datoteka *kml* uporablja strukturo



Slika 3.4: Sekvenčni diagram interakcije razredov pri ustvarjanju novega dogodka.

standarda XML (angl. eXtensible Markup Language) [17], kot je to prikazano na primeru programske kode 3.4. Pri slikah se prav tako zahteva, da uporabnik naloži datoteke. Za upravljanje z datotekami smo uporabili knjižnico oz. gem Refile [10]. Zaradi prostorskih omejitev platforme kot storitev Heroku je potrebno naložene datoteke uporabnikov hraniti na ločenem spletnem prostoru. Knjižnica Refile omogoča enostavno integracijo s ponudnikom infrastrukture kot storitev AWS S3 [4]. Za hranjenje datotek uporabnikov na spletu smo nastavili, da se datoteke samodejno prenesejo na spletni prostor AWS S3. V podatkovno bazo se shrani povezava do datoteke, ki se uporabi pri prikazu vsebine na spletni strani.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
    xmlns:gx="http://www.google.com/kml/ext/2.2"
    xmlns:kml="http://www.opengis.net/kml/2.2"
    xmlns:atom="http://www.w3.org/2005/Atom">
<Document>
  <name>brenta.kml</name>
  <Placemark>
    <name>brenta</name>
    <LineString>
      <coordinates>
        10.90089391634183,46.2148450808832,
        10.90179970825667,46.21235097123489,
        ...
      </coordinates>
    </LineString>
  </Placemark>
</Document>
</kml>
```

Programska koda 3.4: Primer *kml* datoteke.

3.3 Varnost in zasebnost podatkov

Ogrodje Ruby on Rails ima vgrajene varnostne sisteme pred najpogostejšimi napadi, kot so vrivanje SQL, XSS (angl. cross-site scripting) in CSRF (angl. cross-site request forgery). Med razvojem se nam ni bilo potrebno posebej ukvarjati z zaščito pred napadi. Za avtentikacijo uporabnikov smo uporabili knjižnico *Devise* [9]. Knjižnica implementira vnosne maske in poslovno logiko za registracijo in prijavo uporabnikov ter poskrbi za varno hranjenje šifriranih gesel za dostop. Z uporabo standardnih praks ogrodja RoR in knjižnice *Devise* smo zagotovili zaščito podatkov na področju zaupnosti v računalniškem oblaku.

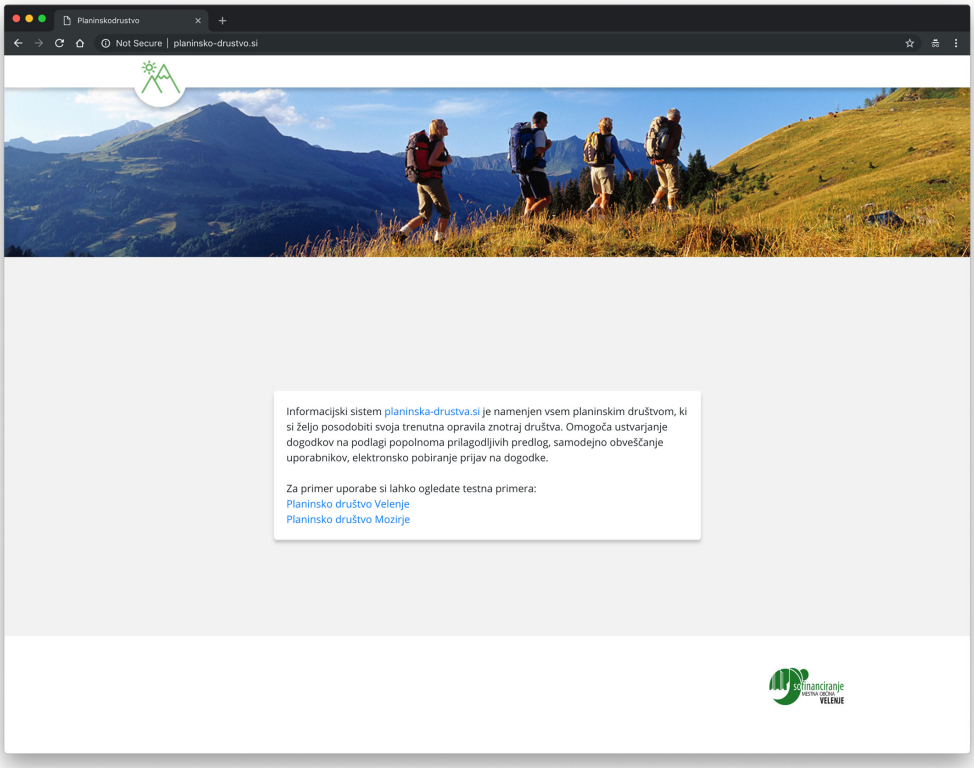
Integriteta podatkov se zagotavlja tako, da knjižnica Apartment preklaplja med shemami organizacij. Princip ena podatkovna baza in ločene sheme omogoča dobro izoliranost podatkov. Uporabniki ene organizacije lahko dostopajo le do svojih podatkov.

Za gostovanje aplikacije smo uporabili platformo kot storitev Heroku. Področje zanesljivosti v računalniškem oblaku je tako odvisno od kakovosti zunanje storitve.

Poglavje 4

Delovanje aplikacije

Za prikaz delovanja aplikacije smo vzpostavili testno okolje za dve vzorčni društvi, to sta Planinsko društvo Velenje in Planinsko društvo Mozirje. Podatki dogodkov in oseb so za potrebe prikaza izmišljeni. Na spletnem naslovu *http://planinsko-drustvo.si* se prikaže domača stran aplikacije, kot je to prikazano na sliki 4.1. Domača stran vsebuje kratko predstavitev projekta in povezave na vzorčne primere planinskih društev. V aplikaciji imamo na voljo administratorski del za strokovne delavce in spletno stran društva za obiskovalce.

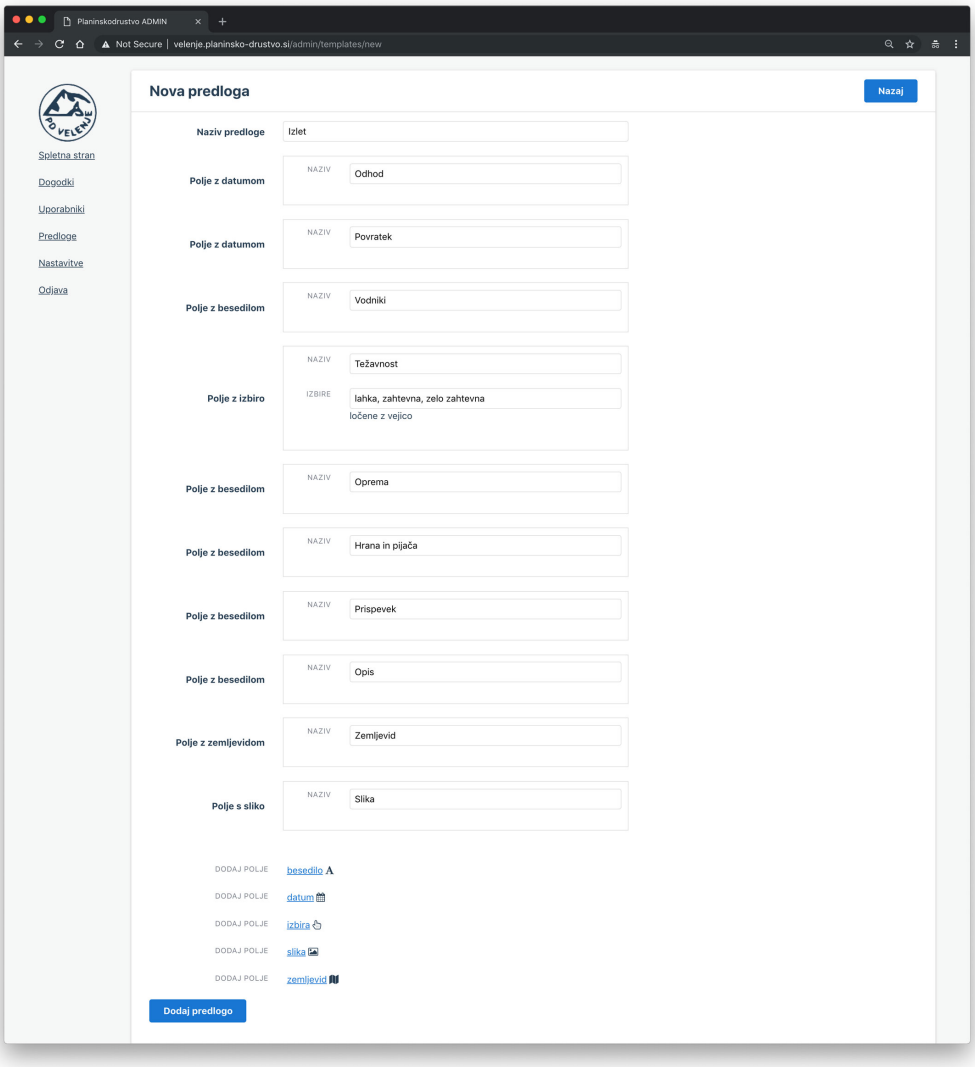


Slika 4.1: Domača stran aplikacije.

4.1 Administratorski del aplikacije

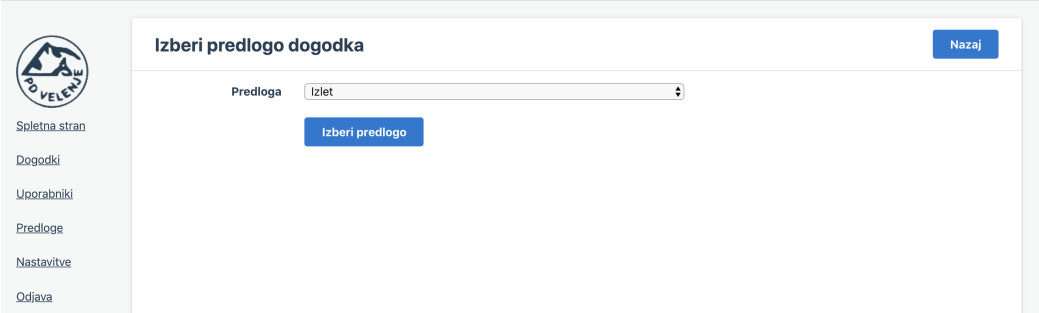
Do administratorskega dela aplikacije imajo pravico dostopati le strokovni delavci planinskega društva. Za primer Planinskega društva Velenje je dostop omogočen na naslovu *http://velenje.planinsko-drustvo.si/admin*. Ob vzpostavitvi društva je potrebno ustvariti predloge za različne tipe dogodkov. Primer ustvarjanja predloge za izlet je prikazan na sliki 4.2. Dogodek tipa izlet mora vsebovati informacije o odhodu ter povratku, zahtevnosti poti, odgovorni osebi za izlet, potrebni opremi in prispevku za udeležbo. Pri izdelavi predloge je možno dodati še polje za opis poti, polje za pripenjanje zemljevida poti in slike, da obiskovalce pritegne k prijavi na izlet. Strokovni delavec ima pri izdelavi predloge na voljo pet različnih tipov vsebine: besedilo, datum, polje

z izbiro, sliko in zemljevid.



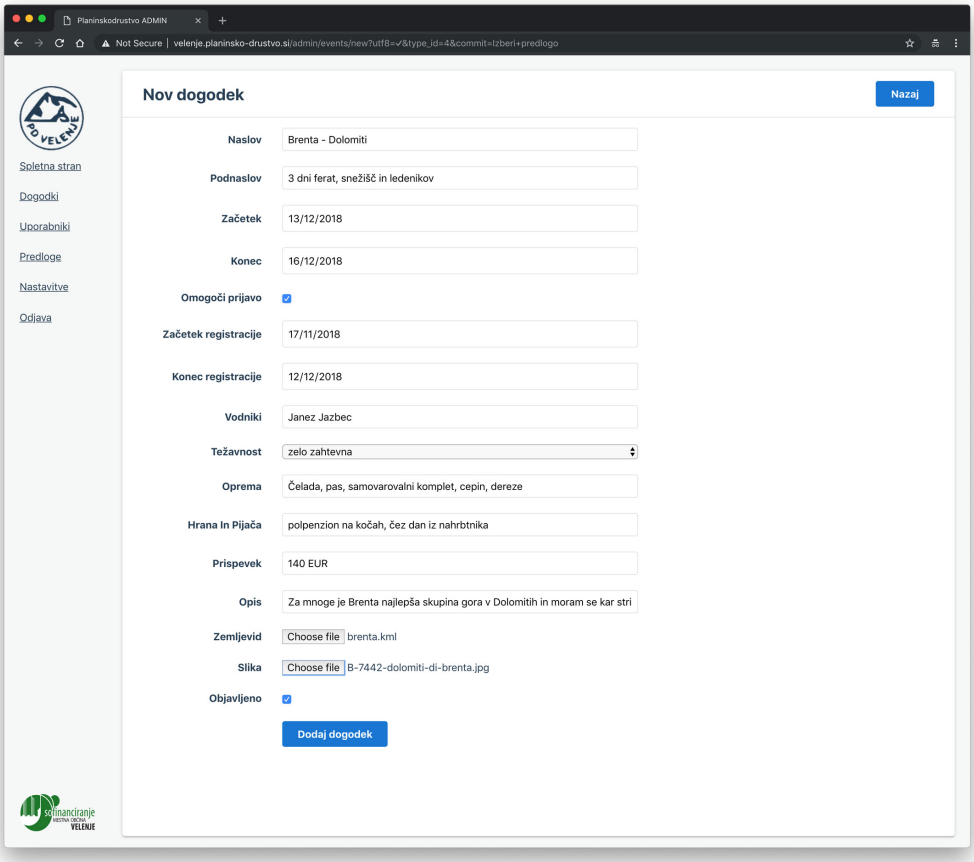
Slika 4.2: Dodajanje nove predloge.

Pred ustvarjanjem novega dogodka je potrebno izbrati predlogo za tip dogodka, ki se vnaša, kot je to prikazano na sliki 4.3.



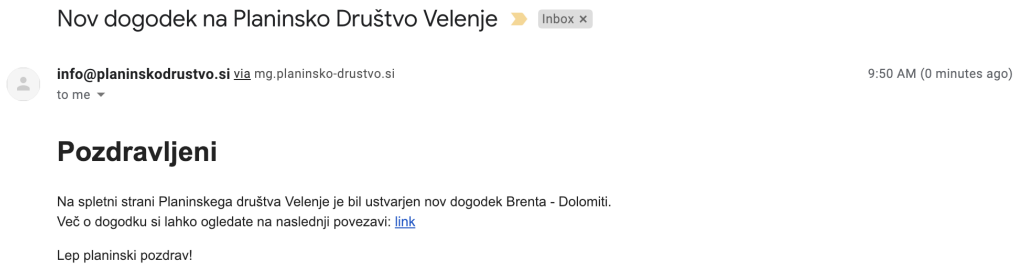
Slika 4.3: Izbira predloge pri dodajanju novega dogodka.

Dodajanje novega dogodka poteka z izpolnjevanjem vnosne maske. Na sliki 4.4 je prikazana izpolnjena vnosna maska za vzorčni izlet v Dolomite.



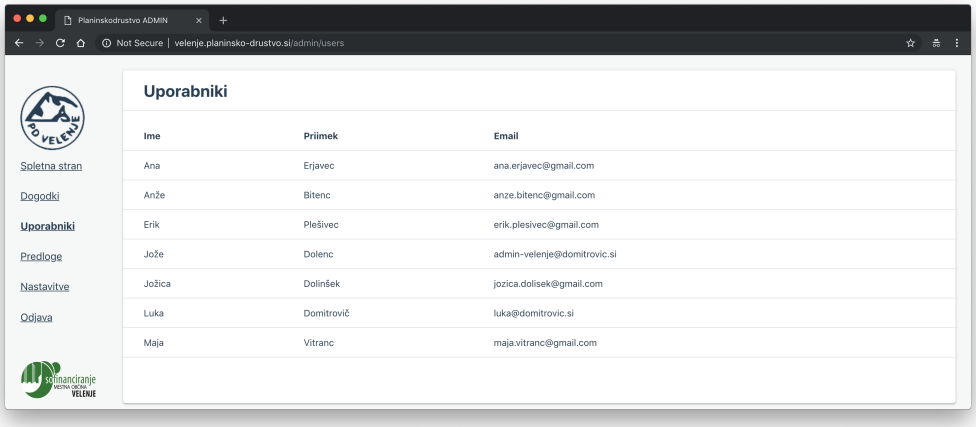
Slika 4.4: Dodajanje novega dogodka.

Ob objavi dogodka se uporabnikom pošlje obvestilo v obliki elektronskega sporočila, prikazanega na sliki 4.5. Uporabnike se prijazno povabi k udeležbi.



Slika 4.5: Elektronsko sporočilo uporabnikom ob objavi dogodka.

Strokovni delavci društva imajo možnost pregleda in urejanja vseh registriranih uporabnikov, kar je razvidno na sliki 4.6.



Slika 4.6: Seznam uporabnikov aplikacije društva.

Uporabniki se lahko elektronsko prijavijo na dogodke in s tem društvo avtomatsko pridobi seznam udeležencev za posamezen dogodek. Primer seznama udeležencev dogodka je prikazan na sliki 4.7.

PRIJAVLJENI UDELEŽENCI

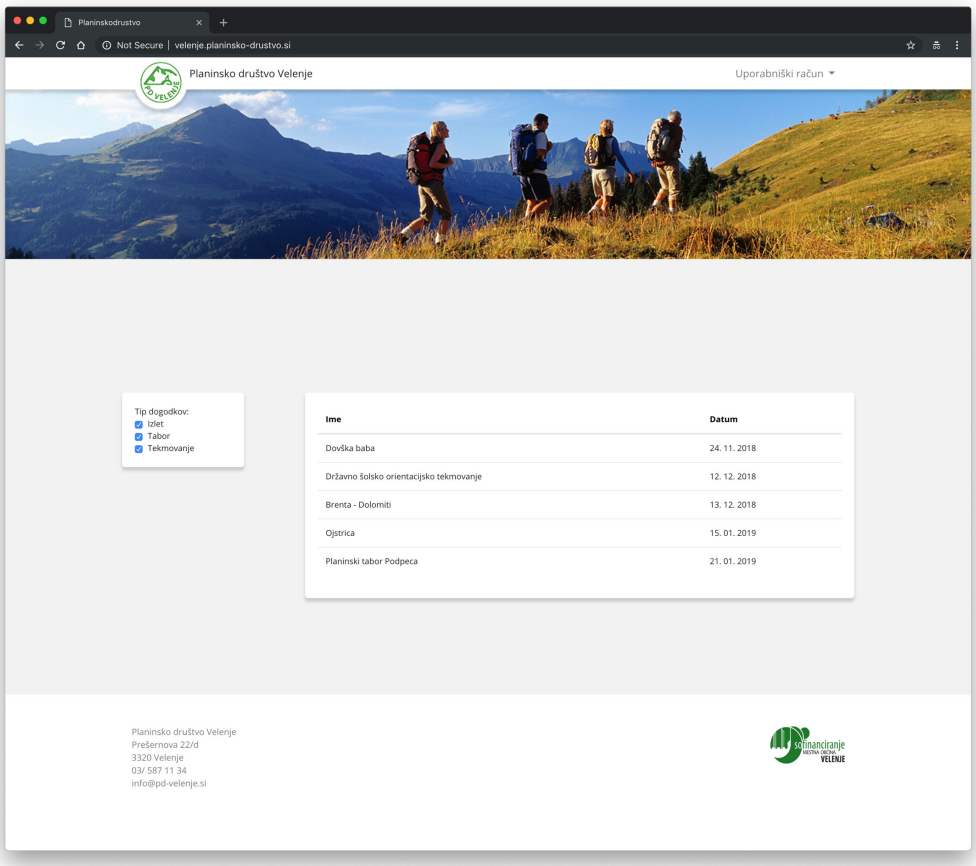
Ime	Priimek	Datum prijave	
Luka	Domitrovič	18. 11. 2018 11:01	Odjavi
Anže	Bitenc	18. 11. 2018 11:01	Odjavi
Jožica	Dolinšek	18. 11. 2018 11:01	Odjavi
Ana	Erjavec	18. 11. 2018 11:01	Odjavi
Erik	Plešivec	18. 11. 2018 11:01	Odjavi
Maja	Vitranc	18. 11. 2018 11:01	Odjavi

Slika 4.7: Pregled seznama prijavljenih udeležencev na dogodek.

Med nastavitvami društva je mogoče nastaviti logotip, naslovno sliko in osnovne podatke društva, ki bodo prikazani na domači strani društva.

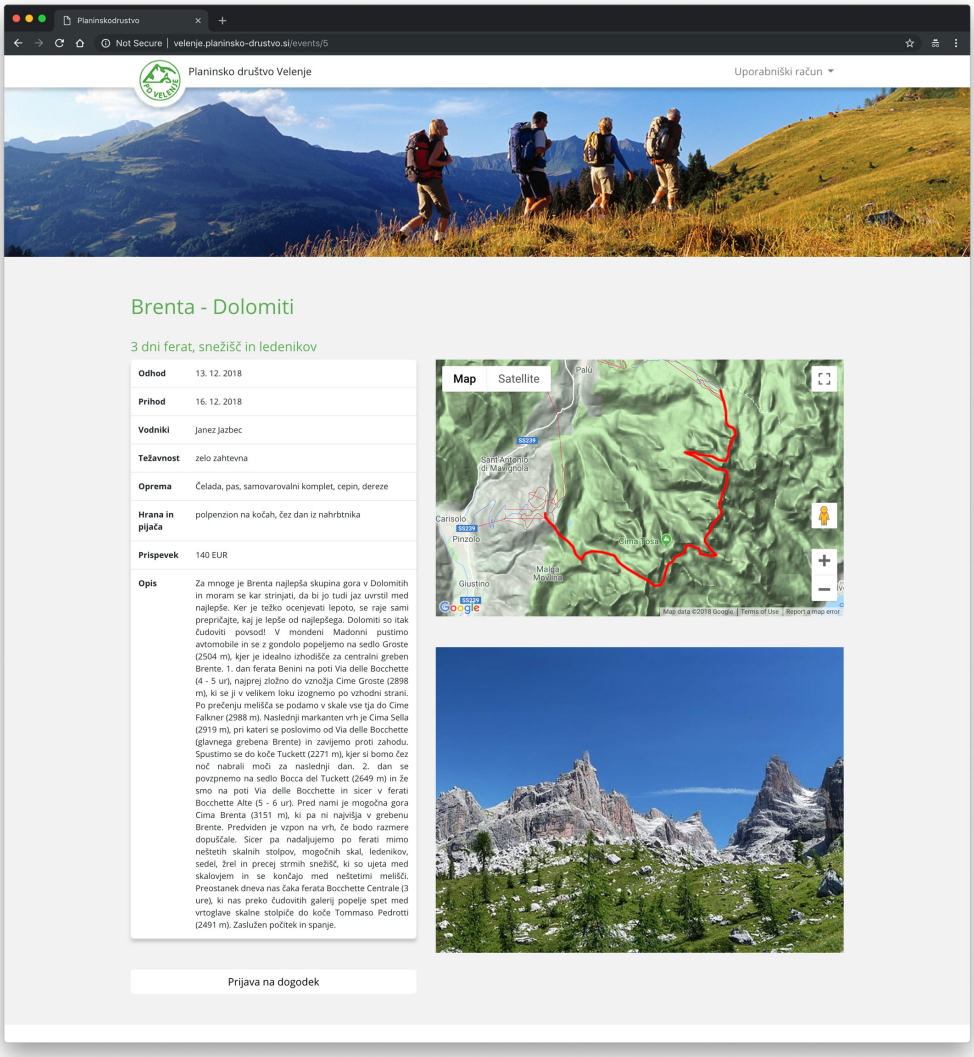
4.2 Spletna stran planinskega društva

Obiskovalci lahko do spletne strani dostopajo preko poddomene posameznega društva, na primer *http://velenje.planinsko-drustvo.si* ali preusmeritve na svojo domeno društva. Domača stran vzorčnega planinskega društva je vidna na sliki 4.8. Stran zajema logotip, naslovno sliko, seznam dogodkov, ki ga je mogoče filtrirati po različnih parametrih. V nogi strani so izpisani osnovni podatki društva, naslov in kontaktni podatki. Obiskovalec ima možnost registracije, s katero si ustvari svoj uporabniški račun. S prijavo v spletno aplikacijo je možna elektronska prijava na različne dogodke. Vsak uporabnik aplikacije lahko za svoj uporabniški račun ureja podatke pod zavihkom Uporabniški profil.



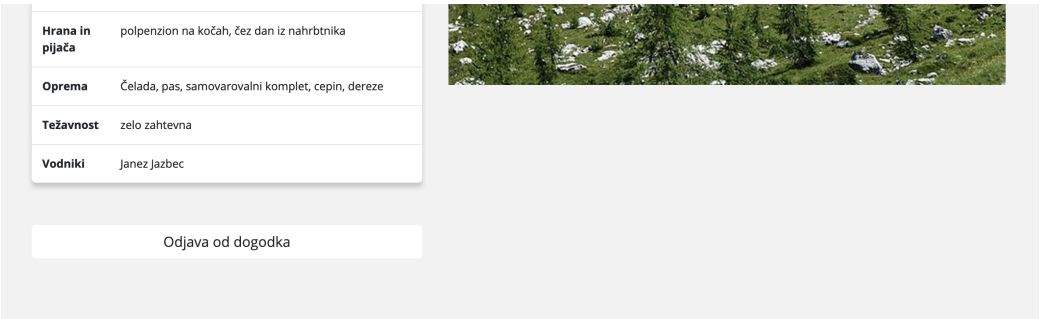
Slika 4.8: Domača stran Planinskega društva Velenje.

Na podstran o posameznem dogodku lahko obiskovalci dostopajo s klikom na dogodek v seznamu. Podstran dogodka vključuje vse podrobne informacije o dogodku. Primer izleta v Dolomite je prikazan na sliki 4.9. Vsak tip vsebine ima prilagojen prikaz, polja z besedilom, datumom in izbirami, ki tvorijo tabelo informacij o dogodku na levi strani podstrani. Slike in zemljevid začrtane poti pa se prikažejo na desni strani podstrani dogodka. V mobilnem pogledu se vsebina prilagodi za prikaz na manjšem zaslonu.



Slika 4.9: Podstran dogodka.

Prijavljeni uporabniki imajo možnost odjave od dogodka s klikom na gumb, ki je prikazan na sliki 4.10.



Slika 4.10: Odjava od dogodka.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo razvili spletno aplikacijo, namenjeno planinskim društvom in širši javnosti, ki jo zanimajo organizirani dogodki društev. Namen aplikacije je posodobiti in izboljšati najpomembnejše procese v planinskih društvih ter informacije o aktivnostih prikazati na svetovnem spletu.

Aplikacija vsebuje štiri glavne aktivnosti, ki pripomorejo k izboljšanju trenutnih procesov na društvih in obveščanju uporabnikov:

- Enostavna izdelava dogodkov z dinamično strukturo vsebine. Društvo se ne omejuje na pripravljene obrazce, ampak aplikacija omogoča popolno prilagodljivost strukture vsebini podatkov. Integracija interaktivne vsebine, kot so slike in zemljevid poti, obiskovalcem bolj slikovito in nazorno predstavi dogodek.
- Avtomatizirana objava dogodkov na spletni strani društva. Vsebino lahko društva sama predstavijo širši publiki brez posegov razvijalcev.
- Obveščanje članov društva o aktualnih dogodkih.
- Avtomatiziran in prijazen način elektronske prijave na dogodke ter obdelava podatkov o njihovih udeležencih.

Aplikacija ne zahteva obsežnega vzdrževanja, saj spremenjena programska koda na enem mestu vpliva na spremembe pri vseh društvih. V sklopu

diplomskega dela smo razvili osnovno in sprejemljivo rešitev, ki jo lahko ponudimo potencialnim strankam. Aplikacija je primerna za srednja in velika planinska društva po Sloveniji, kar je približno polovica vseh registriranih društev pri Planinski zvezi Slovenije. Zastavljen poslovni model vključuje mesečno naročnino za vsako gostujoče društvo in majhen zagonski strošek. Z ugodnim finančnim planom in nizkimi začetnimi stroški smo približali digitalizacijo procesov društvom, ki si ne morejo privoščiti razvoja samostojne aplikacije. Širši javnosti pa smo ponudili sodoben in interaktivni način dostopa do informacij.

Predstavljena in izdelana arhitekturna zasnova zadošča za gostovanje potencialnega števila društev, ki jih je mogoče doseči v Sloveniji. Pri večjem številu gostujočih organizacij se pojavi problem prevelikega števila shem v podatkovni bazi. Začeli bi se soočati s težavami pri izdelavi varnostnih kopij, saj bi lahko čas za izdelavo dnevne varnostne kopije presegel pričakovano in obvladljivo število ur.

Omenjeno težavo je mogoče rešiti na dva načina:

- Gostujoče organizacije razporediti na več vzporednih strežnikov podatkovne baze ali tako imenovano horizontalno skaliranje. V konkretnem primeru bi na primer 100 društev hranili v eni podatkovni bazi, naslednjih 100 v drugi itd. Aplikacija tako ne bi preklapljala le med shemami, ampak tudi med strežniki podatkovnih baz. Knjižnica Apartment vsebuje podporo za shranjevanje podatkov organizacij na različnih strežnikih tako, da implementacija te rešitve ne bi zahtevala pomembnih sprememb in veliko dodatnega časa za razvoj.

- Pri res velikem številu gostujočih organizacij ter posledično velikem obsegu vzdrževalnih del, zaradi mnogo strežnikov podatkovnih baz, pa princip arhitekture skupna podatkovna baza in ločene sheme ne bi bil več primeren. V tem primeru bi bilo bolje izbrati princip skupna podatkovna baza in skupna shema večuporabniške arhitekture. Vendar je pri razvoju tega principa ključnega pomena varnost in izolacija podatkov med organizacijami. V izogib težavam prezgodnje optimizacije za razvoj

najosnovnejšega sprejemljivega produkta nismo izbrali principa skupna podatkovna baza in skupna shema.

Spletna aplikacija je bila kot projekt prijavljena na Javni razpis za dodelitev sredstev za spodbujanje podjetništva v Mestni občini Velenje. Sredstva so bila namenjena za sofinanciranje stroškov razvoja in zagona projekta. V prihodnosti bodo stroški vzdrževanja in morebitnih nadgradenj financirani iz naslova prihodkov mesečnih naročnin gostujočih planinskih društev.

Idej za izboljšave in nadgradnje je precej, predvsem bi se osredotočili na tiste, ki bodo predstavljale največjo dodano vrednost spletne aplikacije.

Spletna stran planinskega društva bi lahko poleg dogodkov vključevala še dodatne funkcionalnosti, kot so:

- informacije o društvu, njegovi zgodovini in odsekih, ki ga sestavljajo;
- informacije o planinskih kočah in poteh, ki jih društva vzdržujejo;
- cenik članarine za tekoče leto;
- galerije slik iz preteklih dogodkov.

Nadgradnja za opisane rubrike bi zahtevala razvoj vnosnih mask ter urejevalnika besedil v administracijskem delu in prikaz vsebine na spletni strani društva.

Implementacija ponudnika za sprejemanje plačil bi bila tudi ena od nadaljnjih ugodnosti za uporabnike. Ob elektronski prijavi bi lahko udeležencem ponudili tudi možnost, da poravnajo strošek udeležbe na dogodku kar preko spleta. Prav tako bi lahko izvedli elektronsko plačevanje letne članarine in razširili rubriko uporabniki v administratorskem delu aplikacije.

Trenutno imajo vsi strokovni delavci vpogled nad vsemi vsebinami v administracijskem delu aplikacije. S širjenjem rubrik in uvajanjem plačil bi bilo potrebno vzpostaviti hierarhijo uporabniških vlog in omejevanje dostopa do različnih vsebin.

Integracija s socialnimi omrežji bi lahko ob objavi dogodka omogočala samodejno objavo kratkega povabila, opremljenega s slikami in povezavo na stran dogodka.

Za informativne namene bi lahko naredili pregled preteklih dogodkov in omogočili nalaganje na dogodku posnetih fotografij vsem udeležencem dogodka.

Pri širjenju izven meja Slovenije bi morali izvesti internacionalizacijo uporabniškega vmesnika.

Literatura

[1] 12 nivojska metodologijo za gradnjo programske opreme. https://12factor.net/. [Online; Dostopano 6. marec 2019].

[2] Amazon web services (aws). https://aws.amazon.com/. [Online; Dostopano 6. marec 2019].

[3] Apartment gem. https://github.com/influitive/apartment. [Online; Dostopano 6. marec 2019].

[4] Aws s3. https://aws.amazon.com/s3/. [Online; Dostopano 6. marec 2019].

[5] Datoteka kml. https://developers.google.com/kml/documentation/kml_tut. [Online; Dostopano 6. marec 2019].

[6] Demystifying cloud computing. http://static1.1.sqspcdn.com/static/f/702523/10181434/1294788395300/201101-Hassan.pdf?token=E3irkn5DtMUsCnP86VZZ9d9nn40%3D. [Online; Dostopano 6. marec 2019].

[7] Heroku ponudnik gostovanja. https://dashboard.heroku.com/. [Online; Dostopano 6. marec 2019].

[8] An introduction to cloud computing concepts. http://www.secc.org.eg/RECOCAPE/SECC_Tutorials_An%20Introduction%20to%20Cloud%20Computing%20Concepts.pdf. [Online; Dostopano 6. marec 2019].

[9] Knjižnica devise. https://github.com/plataformatec/devise. [Online; Dostopano 6. marec 2019].

[10] Knjižnica refile. https://github.com/refile/refile. [Online; Dostopano 6. marec 2019].

[11] Letni plan dogodkov planinskega društva Železničar. http://www.pdzeleznicar.si/program-aktivnosti. [Online; Dostopano 6. marec 2019].

[12] Mastering automation in cloudforms 4.2 and manageiq euwes. https://pemcg.gitbooks.io/mastering-automation-in-cloudforms-4-2-and-manage/content/peeping_under_the_hood/chapter.html. [Online; Dostopano 6. marec 2019].

[13] Multi-tenant applications using spring boot, jpa, hibernate and postgres. https://tech.asimio.net/2017/01/17/Multitenant-applications-using-Spring-Boot-JPA-Hibernate-and-Postgres.html. [Online; Dostopano 6. marec 2019].

[14] Multi[U+2010]tenant data architecture. http://ramblingsofraju.com/wp-content/uploads/2016/08/Multi-Tenant-Data-Architecture.pdf. [Online; Dostopano 6. marec 2019].

[15] The nist definition of cloud computing. http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf. [Online; Dostopano 6. marec 2019].

[16] Ogrodje bootstrap. http://getbootstrap.com/. [Online; Dostopano 6. marec 2019].

[17] Označevalni jezik xml. https://www.w3.org/XML/. [Online; Dostopano 6. marec 2019].

[18] Postgresql. https://www.postgresql.org/. [Online; Dostopano 6. marec 2019].

[19] Postgresql json datatype. https://www.postgresql.org/docs/current/datatype-json.html. [Online; Dostopano 6. marec 2019].

[20] Postgresql schemas. https://www.postgresql.org/docs/9.1/ddl-schemas.html. [Online; Dostopano 6. marec 2019].

[21] Redis. https://redis.io/. [Online; Dostopano 6. marec 2019].

[22] Ruby on rails active record. https://guides.rubyonrails.org/active_record_basics.html. [Online; Dostopano 6. marec 2019].

[23] Sidekiq. https://sidekiq.org/. [Online; Dostopano 6. marec 2019].

[24] Single tenant vs multi-tenant: What's the difference? https://www.bmc.com/blogs/single-tenant-vs-multi-tenant/. [Online; Dostopano 6. marec 2019].

[25] Some trends in web application development. https://www.researchgate.net/publication/4250861_Some_Trends_in_Web_Application_Development. [Online; Dostopano 6. marec 2019].

[26] Websocket. https://en.wikipedia.org/wiki/WebSocket. [Online; Dostopano 6. marec 2019].

[27] Obie Fernandez. *The Rails 5 Way*. Addison-Wesley Professional, 2017.

[28] David Heinemeier Hansson. Ogrodje ruby on rails. http://rubyonrails.org/. [Online; Dostopano 6. marec 2019].

[29] Michael Kavis. *Architecting The Cloud*. Wiley, 2014.

[30] Yukihiro Matsumoto. Programski jezik ruby. https://www.ruby-lang.org/en/. [Online; Dostopano 6. marec 2019].

[31] Robert Michon. *The Complete Guide to Software as a Service: Everything you need to know about SaaS*. CreateSpace Independent, 2017.

[32] R. Velumadhava Rao and K. Selvamani. Data security challenges and its solutions in cloud computing.

[33] Planinska Zveza Slovenije. Seznam planinskih društev v sloveniji. `https://www.pzs.si/drustva.php`. [Online; Dostopano 6. marec 2019].