Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žiga Kerec

Razvoj spletne aplikacije za organizacijo dela v manjših skupinah

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Aleš Smrdel

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi razvijte spletno aplikacijo, ki bo omogočala organizacijo skupinskega dela prijavljenih uporabnikov. Aplikacija naj bo zasnovana tako, da bo omogočala delitev prijavljenih uporabnikov na manjše skupine, pri tem pa lahko posamezen uporabnik sodeluje tudi v več skupinah. Aplikacija naj omogoča manipulacijo skupin, kar vključuje: dodajanje, brisanje in spreminjanje nastavitev skupine in manipulacijo uporabnikov, kar vključuje: dodajanje uporabnika, brisanje uporabnika in spreminjanje nastavitev uporabnika. Ker je aplikacija namenjena organizaciji dela znotraj skupine, mora aplikacija omogočati tudi določitev nalog posameznemu uporabniku ter spremljanje izvrševanja določenih nalog. Za boljšo komunikacijo in lažjo organizacijo dela implementirajte tudi sistem za komunikacijo med člani skupine. Prav tako pa implementirajte tudi sistem za izmenjavo datotek med člani skupine, ki omogoča nalaganje datotek na strežnik in prenos datotek s strežnika. Pri implementaciji uporabite tehnologije na strani strežnika in na strani odjemalca, ki bodo omogočale hitro in odzivno delovanje aplikacije. Pri razvoju aplikacije pozornost posvetite tudi primernemu prikazu na različnih napravah, oziroma za različne velikosti zaslonov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Žiga Kerec sem avtor diplomskega dela z naslovom:

Razvoj spletne aplikacije za organizacijo dela v manjših skupinah

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Aleša Smrdela,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 2. septembra 2015 Podpis avtorja:

Hvala mentorju doc. dr. Alešu Smrdelu za mentorstvo in strokovno pomoč. Hvala vsem mojim bližnjim za podporo in zaupanje v času študija.

Kazalo

kratica	angleško	slovensko
HTML	HyperText Markup Language	jezik za označevanje hiperteksta
CSS	Cascading Style Sheets	kaskadne stilske podloge
Ajax	Asynchronous JavaScript and XML	asinhroni JavaScript in XML
XML	Extensible Markup Language	razširljiv označevalni jezik
SASS	Syntactically Awesome Stylesheets	sintaktično odlične stilske podloge
SVG	Scalable Vector Graphics	umerljiva vektorska grafika
XUL	XML User Interface Language	XML jezik za uporabniški vmesnik
DOM	Document Object Model	objektni model dokumenta
XHTML	Extensible HyperText Markup Language	razširljiv jezik za označevanje hiperteksta
AMP	Apache, MySQL, PHP	Apache, MySQL, PHP
MVC	Model-View-Controller	model-pogled-krmilnik
ORM	Object-relational mapping	objektno relacijsko preslikavanje
SQL	Structured Query Language	strukturirani povpraševalni jezik
RGB	Red, Green, Blue	rdeča, zelena, modra
API	Application Programming Interface	aplikacijski programski vmesnik
HTTP	HyperText Transfer Protocol	protokol za prenos hiperteksta
URL	Uniform Resource Locator	enolični krajevnik vira
CSV	Comma-separated values	vrednosti ločene z vejico
SPA	Single-page Application	enostranska aplikacija
JSON	JavaScript Object Notation	JavaScript objektni zapis

Povzetek

Cilj diplomske naloge je bil razvoj spletne aplikacije, ki olajša organizacijo in izvajanje skupinskega dela. Namejena je članom manjših ekip in jo lahko opišemo kot komunikacijsko orodje za boljšo organizacijo skupine. Aplikacija ponuja storitve za neposredno komunikacijo preko diskusij, deljenje datotek in razdelitve tekočih opravil. Strežniški del aplikacije je nastajal v ogrodju Django programskega jezika Python, osprednji del pa v jezikih JavaScript, HTML in CSS. Aplikacija stilsko sledi načelu odzivnega dizajna in je primerna za prikaz tudi na zaslonih manjših dimenzij. V prvem delu diplomske naloge so opisane uporabljene tehnologije, v drugem delu pa razvoj in uporabniška izkušnja spletne aplikacije.

Ključne besede: spletna aplikacija, Django, MySQL, odzivni dizajn.

Abstract

The purpose of this bachelor thesis was the development of an web application that would enable a smoother organization and prosecution of team work. It is intended to be used by members of smaller groups and can be described as a tool of communication that will improve the group's organization. The application offers services for direct communication through discussions, file sharing and the division of current tasks. The server part of the application was created in the framework Django in the Python programming language, where as the frontend was created in the JavaScript, HTML and CSS languages. The application follows the responsive design and is also suitable for display on screens of smaller dimensions. This bachelor thesis consists of two main parts. The first part of the thesis is dedicated to the description of the used technologies and the second part of the thesis deals with the development and the user experience of the web application.

Keywords: web application, Django, MySQL, responsive design.

Poglavje 1

Uvod

V zadnjih letih opažamo vse večjo rast mladih (*startup*) podjetij, po večini sestavljenih iz mladih in kreativnih ljudi, ki želijo na takšen ali drugačen način spremeniti življenje posameznikov in nam z zanimivimi inovacijami olajšati vsakdanjik. Za uspešno delo vsakega podjetja pa je potrebna dobra, hitra in kakovostna medsebojna komunikacija. Seveda industrija ponuja veliko različnih in kakovostnih komunikacijskih rešitev (Skype [1], elektronska pošta, Facebook [2], ...), vendar so le redke povsem osredotočene na produktivnost in delo v skupini. Poleg funkcij pomembnih za delo, ponujajo namreč tudi storitve, ki so namenjene zgolj zabavi uporabnikov. Ob brskanju po spletu in iskanju podobnih rešitev, sem naletel na aplikacijo Slack [3], ki uporabnikom omogoča izoliran komunikacijski kanal, ustvarjen z namenom povečanja produktivnosti in učinkovitejše komunikacije skupine. Istoimensko podjetje je eno izmed zelo hitro rastočih podjetij Silicijeve doline.

Aplikacija Slack je sicer dobro zasnovana, vendar sem hotel implementirati svojo različico in dodati še kakšno podrobnost, ki izboljša uporabniško izkušnjo. Ker se mi zdi, da je delo v skupinah s sproščeno komunikacijo bolj stimulativno, sem se odločil izdelati lastno spletno aplikacijo, ki implementira podobne storitve, kot jih ponuja Slack in dodaja nekaj novih uporabnih rešitev. Moja spletna aplikacija imenovana *Teamsy* je namenjena organizaciji dela manjših skupin, kjer se registrirani uporabnik lahko pri-

ključi obstoječi skupini ljudi ali ustvari novo skupino ljudi. Povezanost in komunikacijo skupine pa obsegajo diskusije, deljenje datotek in razdelitve določenih nalog. Ponuja tudi hiter statistični pregled zapisanih objav, nalog in števila uporabnikov znotraj določene skupine. Skozi celoten razvoj sem aplikacijo poskušal pisati z mislijo na različne dimenzije zaslonov, tako da se vsebina strani prilagaja glede na tip naprave (mobilnik, tablični računalnik, prenosnik, ...). Ravno odziven dizajn odpira možnosti za kasnejše nadgradnje funkcij in inovativnejšne mobilne verzije aplikacije. Diplomska naloga je razdeljena na dva dela. V prvem delu opisujem uporabljene tehnologije in knjižnice ter razvojna orodja, ki sem jih uporabljal pri izdelavi spletne aplikacije, v drugem delu pa opisujem sam razvoj in delovanje spletne aplikacije ter uporabniško izkušnjo.

Poglavje 2

Uporabljene tehnologije, ogrodja in razvojna orodja

2.1 Tehnologije

Razvoj spletne aplikacije je razdeljen na dva dela, to sta razvoj ospredja (vidnega dela aplikacije) in razvoj zaledja (povezave s podatkovno bazo in priprave podatkov za prikaz na zaslonu). Podobno kot vse ostale spletne aplikacije je tudi ospredje moje aplikacije sestavljeno z označevalnim jezikom HTML (*Hyper Text Markup Language*, jezik za označevanje hiperteksta), in je oblikovano s CSS (*Cascading Style Sheets*, kaskadne stilske podloge) [4]. Za animacije in odzivnost spletne aplikacije delno skrbi CSS3 (najnovejša različica jezika CSS), v veliki večini pa programski jezik JavaScript. Zaledje aplikacije je spisano v programskem jeziku Python, ki skrbi za povezovanje s podatkovno bazo in za dodajanje, branje, popravljanje ter brisanje podatkov v podatkovni bazi, v mojem primeru v podatkovni bazi MySQL [5]. Za dinamično posredovanje podatkov pa skrbi tehologija Ajax (*Asynchronous JavaScript and XML*, asinhroni JavaScript in XML).

2.1.1 HTML

Hyper Text Markup Language ali krajše HTML se je v širši javnosti pojavil leta 1991 [6] s pojavitvijo prvih spletnih brskalnikov. Zaradi svoje inovativnosti in enostavnosti za uporabo je postal standardiziran označevalni jezik za izdelavo spletnih strani. Sintaksa je sestavljena iz vnaprej predpisanih HTML značk, ki brskalnikom narekujejo prevajanje kode in pomagajo pri izpisu vsebine na zaslon. Trenutno najnovejša verzija označevalnega jezika HTML5 ponuja nekaj novih značk, ki olajšajo programerjevo delo, prav tako pa pohitrijo delovanje spletnih aplikacij. Velik napredek nove verzije in prilagoditve zahtevam ter potrebam svetovnega spleta so vidni v predstavitvi multimedijskih vsebin, na primer videa, katerega prikaz je veliko enostavnejši in ne zahteva več dodatnih vtičnikov, ki so pogosto lahko omejevali hitro delovanje spletne aplikacije. Vse popularnejši element - platno (HTML Canvas), pa omogoča izris 3D elementov, grafik in animacij znotraj spletne aplikacije [7]. Za podobne učinke je v prejšnjih verzijah skrbel vtičnik Adobe Flash [8] ali katera izmed drugih, časovno potratnih knjižnic. Še vedno pa so daleč najbolj uporabljene značke tiste, ki so bile predstavljene v prejšnji verziji (4.01) iz začetka tisočletja, predvsem zaradi dejstva, da se njihova uporaba v novi verziji ni bistveno spremenila.

2.1.2 CSS

CSS (*Cascading Style Sheets*, kaskadne stilske podloge) je jezik za določanje stilov spletnih strani. Poleg razporeditve elementov po strani brskalnika nam omogoča še stilske popravke vsebine elementov, v najnovejši produkcijski različici CSS3 pa še animiranje določenih HTML elementov. CSS3 torej že posega na področje JavaScripta, pri nekaterih enostavnejših animacijah (rotacije, translacije in skaliranje) zaradi svoje enostavnosti in hitrosti prekaša tudi JavaScriptovo knjižnico jQuery [9]. Tehnologija je bila prvič predstavljena leta 1994 [10] zaradi zahtev po formatiranju izpisa HTML strani, ustvarjen pa je bil tudi z namenom ločevanja vsebinske in stilske sekcije HTML doku-

mentov. Do HTML elementov dostopamo v kodi napisani v jeziku CSS. Dostopamo lahko preko tipov ali preko atributov HTML elementov. Selektorji sledijo prioritetnemu vrstnemu redu, bodisi glede na pomembnost selektorja, bodisi glede na mesto v HTML hirearhiji. Stile lahko definiramo na treh nivojih: medvrstično (*inline*), na nivoju dokumenta in na nivoju spletišča. Največji problem omenjenega jezika je nezmožnost uporabe spremenljivk ali zank, ki so temeljni sestavni deli vseh modernih programskih jezikov. Naprednejša SASS (*Syntactically Awesome Style Sheets*, sintaktično odlične stilske podloge) [11] in LESS [12] (knjižnica za naprednejšo obdelavo CSS datotek) zajemata tudi te prvine pravih programskih jezikov in omogočata prevajanje delov kode v CSS datoteke, berljive vsem spletnim brskalnikom. Te novosti s pridom izkoriščajo naprednejše knjižnice za oblikovanje spletnih strani. CSS ne ureja le HTML zapisov, njegove funkcionalnosti lahko namreč izkoristimo še na raznih XML (*Extensible Markup Language*, razširljiv označevalni jezik) [13], SVG (*Scalable Vector Graphics*, umerljiva vektorska grafika) [14] ali XUL (*XML User Interface Language*, XML jezik za uporabniški vmesnik) [15] zapisih.

2.1.3 JavaScript

Pomemben del posamezne spletne aplikacije, ki teče v brskalniku na uporabnikovem računalniku, je ponavadi napisan v programskem jeziku JavaScript. JavaScript je objektno usmerjen programski jezik, ki omogoča dinamične funkcionalnosti spletnih strani. Prvič je bil predstavljen leta 1995 [16], izdan pa je bil izpod spretnih prstov programerjev, pionirja spletnih brskalnikov - Netscape. JavaScript uporablja za manipulacijo HTML dokumentov konvencijo DOM (*Document Object Model*, objektni model dokumenta) [17]. DOM je neodvisen od platforme in predstavlja interakcijo z objekti na HTML, XML ali XHTML (*Extensible HyperText Markup Language*) [18] strani. XHTML ima enak namen kot HTML, vendar je usklajen s sintakso XML. V zadnjih letih uporaba JavaScripta še naprej raste predvsem zaradi nadgradnje obstoječih in razvoja novih knjižnic za programiranje spletnih

strani, ki omogočajo izvajanje asinhronih zahtevkov, kot sta na primer AngularJS [19] in React [20]. Te knjižnice nam ponujajo razvoj po tako imenovanem SPA (*Single Page Application*, enostranska aplikacija) modelu programiranja. SPA model ponuja tekočo uporabniško izkušnjo, spletne aplikacije pa še bolj približa lokalnim računalniškim aplikacijam. Spletna stran se iz strežnika prenese le enkrat, za nadaljno funkcionalnost pa skrbi JavaScript na uporabnikovem računalniku [21]. JavaScript se uporablja za dinamično spreminjanje stilov, razne izračune, poslovno logiko ali shranjevanje podatkov. V HTML dokument lahko vključimo ločene JavaScript skripte, lahko pa kodo pišemo neposredno v dokument znotraj za to določenih značk.

2.1.4 Python

Python je visokonivojski programski jezik s širokim spektrom uporabe [22]. Je objektno usmerjen, njegove ključne prednosti pa so berljivost, enostavnost in estetski videz kode. Python ne uporablja zavitih oklepajev kot veliko sorodnih jezikov, namesto tega ločnice med funkcijami in zankami nakažemo s presledki ali tabulatorjem. Začetki uporabe in nastanek segajo v leto 1989, prva uporabniško podprta različica pa je izšla leta 2000 [23]. Python je programski jezik, ki je še vedno v razvoju, z vsako novo različico pa so vidne predvsem hitrostne in varnostne izboljšave. Prav tako je odličen jezik za začetnike, saj omogoča enostavno in postopno učenje ter pisanje enostavnih programov brez potrebe po znanju celotnega programskega jezika. Python razvija in vzdržuje ekipa pod vodstvom Guida van Rossuma, prvotnega pisca in arhitekta Python programskega jezika. Čeprav je objektno usmerjen, je mogoče pisati programe tudi v neobjektnem proceduralnem načinu. Obstaja več vrst produkcijskih implementacij, med drugimi `CPython`, `Jython`, `IronPython`, `PyPy` [24] itn. CPython je klasična in največkrat uporabljena implementacija, ki jo poznamo kot Python.

2.1.5 Ajax

Ajax (*Asynchronous JavaScript And XML*, asinhroni JavaScript in XML) sicer ni le ena tehnologija, ampak je skupek večih različnih tehnologij, ki omogočajo asinhrone klice v zaledje aplikacije. Tradicionalne spletne strani podatke pridobivajo s pomočjo zahtevkov na strežnik, na rezultate pa mora uporabnik počakati, preden se lahko njegova uporabniška izkušnja nadaljuje. Ajax zahtevke za shranjevanje ali pridobivanje podatkov izvaja asinhrono v zaledju, pri tem pa spletna stran ni blokirana, tako da uporabnik lahko nemoteno nadaljuje z uporabo spletne aplikacije. Rezultati strežniškega procesiranja so tako vidni brez osveževanja celotne strani, komunikacija s strežnikom pa se tipično sproži preko XMLHttpRequest (XHR) objekta, spisanega v programskem jeziku JavaScript [25]. Podatki se po večini prenašajo v obliki JSON (*JavaScript Object Notation*, JavaScript objektni zapis) ali XML sporočil, JavaScript pa s pomočjo DOMa skrbi za dinamično prikazovanje pridobljenih podatkov na spletni strani. Tehnologijo v veliki meri izkoriščajo prej omenjene knjižnice za razvoj asinhronih aplikacij (AngularJS, ...), ki v zakulisju aplikacij prenašajo podatke z Ajax klici.

2.1.6 MySQL

Prva izdaja drugega najbolj uporabljenega sistema za upravljanje relacijskih podatkovnih baz MySQL se je pojavila leta 1995 [26], razvili pa so jo programerji na Švedskem. Je zelo popularna izbira programerjev spletnih aplikacij in je centralna komponenta LAMP in ostalih AMP strežnikov (Apache, MySQL, PHP). MySQL relacijsko podatkovno bazo uporabljajo med drugimi tudi giganti spletnih storitev, kot so Google, Facebook, Twitter, Flickr in Youtube [27]. Za manipulacijo podatkov uporabljamo povpraševalni jezik SQL (*Structured Query Language*, strukturirani povpraševalni jezik za delo s podatkovnimi bazami). Najvišji nivo MySQL strukture predstavlja podatkovna baza, sestavljena iz dvodimenzionalnih tabel. Za hitrejše in učinkovitejše pridobivanje podatkov je pomembna pravilna definicija polj v posameznih tabe-

lah, saj nam normalizirana struktura MySQL omogoča učinkovito delovanje na manjših in tudi velikih sistemih.

2.1.7 Git

Git je odprtokodni sistem za verzioniranje izvorne kode [28]. Poenostavljeno lahko rečemo, da je sistem za varnostno shranjevanje programske kode, razvit pa je bil za namene podpore razvoja jedra operacijskega sistema Linux. Razvil ga je Linus Torvalds leta 2005, danes pa ga množično uporabljajo programerji za različne vrste projektov. Podpora porazdeljenemu programiranju praktično neomejenemu številu programerjev na enem ali večih projektih, je ena glavnih lastnosti in prednosti sistema. Omogoča enostavno vejenje in končno združevanje dela večih programerjev na istem projektu, datoteki ali celo funkciji [29].

2.2 Ogrodja

Zaradi potrebe po hitrejšem in enostavnejšem razvoju, se v zadnjih letih pojavlja čedalje več ogrodij in knjižnic za delo v zgoraj omenjenih programskih jezikih.

2.2.1 Django

Odprtokodno visokonivojsko ogrodje Django [30], spisano v programskem jeziku Python, omogoča hiter razvoj, implementira čist in pragmatičen oblikovni značaj, kar poudarja praktičnost uporabe ogrodja in lepše strukturirano kodo. Njegove glavne prednosti so predvsem hitrost, varnost in dinamičnost pri prehodih na večje sisteme. Vgrajenih ima že veliko funkcij (na primer avtentikacija uporabnika), ki jih je v ostalih programskih jezikih in ogrodjih potrebno definirati že pred začetkom pisanja same aplikacije, varnostne funkcije pa poskrbijo za obrambo pred najpogostejšimi spletnimi napadi. Implementira arhitekturo MVC (*Model-View-Controller*,

model-pogled-krmilnik), ki aplikacijo razbije v tri pomembne, med seboj povezane enote. Dostop do podatkov, implementacija in manipulacija le teh je za programerja veliko lažja, če uporablja ogrodja, ki delujejo po principu MVC. Poudarja učinkovito delo v skupini, saj omogoča ločevanje uporabniškega vmesnika (View), katerega razvoj poteka povsem ločeno, in zaledja aplikacije. Tako programerjem, ki skrbijo za uporabniško izkušnjo aplikacije, ni potrebno čakati na delujočo verzijo strežniškega dela. Django je bil predstavljen leta 2003, njegovo strukturo pa prav tako uporablja zavidljivo število spletnih strani, med njimi: Pinterest, Instagram, Mozilla, The Washington Times, ... [31]. Djangova komponenta za podatkovne baze ORM (*Object Relational Mapper*, objektno relacijsko preslikavanje) omogoča povezavo med podatkovnim modelom in podatkovno bazo. Podpira veliko različnih sistemov za upravljanje relacijskih podatkovnih baz (med drugimi tudi MySQL), preklapljanje med njimi pa je zelo enostavno. Django vsebuje enostaven in nezahteven spletni strežnik za razvoj in testiranje, ponuja pa tudi pregleden izpis napak, ki lahko služi kot uporaben razhroščevalnik. Ogrodje sicer ne ponuja interne knjižnice za delo z Ajax zahtevki, implementacija zunanje knjižnice pa ni pretirano zahtevna [32].

2.2.2 Semantic UI

Knjižnic na področju oblikovanja in izgleda spletnih aplikacij je na tržišču že zelo veliko. Za Semantic UI [33] sem se odločil zaradi modernega videza in uporabnih funkcij, ki sem jih ocenil kot prednost pri razvoju spletne aplikacije tega tipa. Semantic UI ponuja CSS in JavaScript datoteke s prednapisanimi funkcijami in oblikovnimi piškotki, uporabljena koda pa je izjemno dobro dokumentirana. Omogoča tudi prevajanje po delih, kjer si programer lahko izlušči in prilagodi le del ogrodja, ki ga potrebuje v svoji aplikaciji, poleg klasičnih funkcionalnosti pa ponuja tudi dodatke, na primer ikone, ki jih moramo v konkurenčnih ogrodjih po večini naknadno uvoziti. Implementiran mrežni sistem olajša programiranje prilagajanja aplikacije na različne širine zaslonov. S pravilno uporabo HTML elementov in CSS selektorjev namreč

določimo, kako se bodo elementi na strani premikali ob morebitnem oženju vidnega polja. Knjižnica je prosto dostopna in odprtokodna.

2.2.3 jQuery

jQuery [34] je knjižnica, ki jo velikokrat omenjamo v povezavi s programskim jezikom JavaScript. S svojo hitrostjo, kompaktnostjo in naborom funkcij je prepričala milijone programerjev, ki so jo vpletli v svojo vsakodnevno rutino. Ponuja funkcionalnosti, ki jih lahko izvedemo z le eno vrstico kode, ki v JavaScriptu terjajo na stotine vrstic nepreglednega programa [35]. Ogrodje je bilo predstavljeno leta 2006 [36], ponuja pa enostavnejšo navigacijo po dokumentu, manipulacijo z DOM elementi, animiranje HTML objektov in implementacijo Ajax klicev. jQuery knjižnica je prosto dostopna na spletu v obliki JavaScript datoteke, ki jo pripnemo v željeni HTML dokument. Podpira jo večina modernih internetnih brskalnikov.

2.3 Razvojna orodja

Pojem razvojno orodje označuje skupek programske opreme, ki programerju pomaga pri razvijanju aplikacij. Po večini so to urejevalniki besedil, ki nam z označevanjem rezerviranih besed, ki pripadajo uporabljenemu programskemu jeziku ali dopolnjevanju pogosto uporabljenih stavkov olajšujejo delo in na koncu občutno skrajšajo razvojni čas. Med razvojna orodja pa prav tako spada programska oprema, ki na grafični način prikazuje stanje določenega dela aplikacije v razvojnem ciklu (na primer orodja za grafični prikaz podatkovnih baz).

2.3.1 PyCharm

Za programiranje celotnega zaledja in delno ospredja aplikacije sem uporabil razvojno orodje PyCharm [37]. Aplikacijo v množici drugih profesionalnih razvojnih orodij izdeluje češko podjetje JetBrains [38], ki ponuja ene izmed

najboljših in najzaneslivejših razvojnih orodij na trgu. PyCharm poleg barvnega označevanja sintakse in avtomatskih popravkov ponuja tudi napreden razhroščevalnik za lažjo detekcijo napak v kodi. Pregleden način prikaza strukture delovnega okolja pa omogoča lažje navigiranje po uporabljenih datotekah.

2.3.2 MySQL Workbench

Aplikacijo MySQL Workbench [39] namenjeno delu s podatkovnimi bazami sem uporabil predvsem za modeliranje podatkovnega modela. Delo z MySQL Workbenchom je enostavno, saj poteka v grafičnem okolju. Uvoz, izvoz in sinhronizacija podatkovnega modela s podatkovno bazo pa prav tako poteka brez večjega napora. Največja prednost orodja je inovativno risanje podatkovne baze, brez pisanja SQL ukazov za ustvarjanje tabel in uvažanja podatkov. Relacije so enostavno vidne, implementacija popravkov pa hitra.

2.3.3 Sequel Pro

Sequel Pro [40] je okolje, ki izgleda in funkcionira zelo podobno kot MySQL Workbench, vendar ne ponuja risanja/modeliranja podatkovnega modela. V zameno na veliko enostavnejši način prikaže strukturo baze in podatke v njej. Aplikacijo sem uporabljal predvsem za spremljanje dogajanja v podatkovni bazi med izvajanjem aplikacije ter za dodajanje, brisanje in spreminjanje testnih podatkov. Je zelo uporabno orodje v fazi testiranja.

2.3.4 Orodja za razvijalce brskalnika Chrome

Zagotovo najboljši prijatelj vseh spletnih razvijalcev pa so orodja za razvijalce brskalnikov Chrome [41] in Firefox [42]. Sam sem zaradi dobrega poznavanja največ uporabljal orodja za razvijalce brskalnika Google Chrome. Orodja ponujajo takojšnji pregled kode ob prikazu strani, prav tako omogočajo spreminjanje in dodajanje CSS nastavitev. Vključujejo tudi konzolo za nadzor JavaScript skript, ki prikazujejo in izpisujejo napake, ki se ob zagonu, ali

določenem dogodku pojavijo v aplikaciji. Uporabimo jih lahko tudi za kontrolne izpise ali detekcije napak prenosa datotek iz strežnika. Orodja za razvijalce so vgrajena tudi v vse druge modernejše spletne brskalnike.

2.3.5 Ostala uporabna orodja

Seveda sem pri svojem delu občasno uporabljal še kakšno orodje, ki mi je olajšalo delo za katero izmed specifičnih opravil. Na spletu prosto dostopna aplikacija Adobe Kuler [43] ponuja enostavno in pregledno izbiranje barv, zapis slednjih pa je viden v šestnajstiškem ali zapisu po komponentah RGB (*red, green, blue*). Kuler ponuja prikaz komplementarnih barv, barvnih triad ipd., za hitrejše in učinkovitejše oblikovanje spletne aplikacije. Pri oblikovanju prototipa in nekaterih elementov na strani sta bili v veliko pomoč profesionalni orodji za oblikovanje točkovne in vektorske grafike: Adobova Photoshop [44] in Illustrator [45]. Obe orodji omogočata napredne funkcije za obdelavo slikovnega in tekstovnega materiala.

Brackets [46] je odprtokodno prosto dostopno orodje za razvoj spletnih aplikacij. Orodje je napisano v programskem jeziku JavaScript, z nalaganjem uporabnih dodatkov, ki ponujajo pametno predikcijo sintakse, barvanje rezerviranih besed ipd., pa lahko občutno povečamo svojo produktivnost.

Spletno orodje Bitbucket [47] je aplikacija, ki nam omogoča enostaven pregled Git repozitorija. Zastonjska verzija omogoča hkratno delo do pet uporabnikov, ki lahko delajo na istem projektu. V aplikaciji hranimo še pomembne informacije o tekočem projektu, vodimo wiki stran in izdajamo testne in končne verzije. Za verzioniranje kode s sistemom Git sem poleg vgrajenega terminala (ukazna vrstica na Unix sistemih) uporabljal še grafično orodje SourceTree [48]. S pomočjo slednjega je kloniranje repozitorija na Bitbucketove strežnike enostavno in pregledno, Bitbucket pa že privzeto ponuja poglobljeno integracijo z omenjenim orodjem.

Poglavje 3

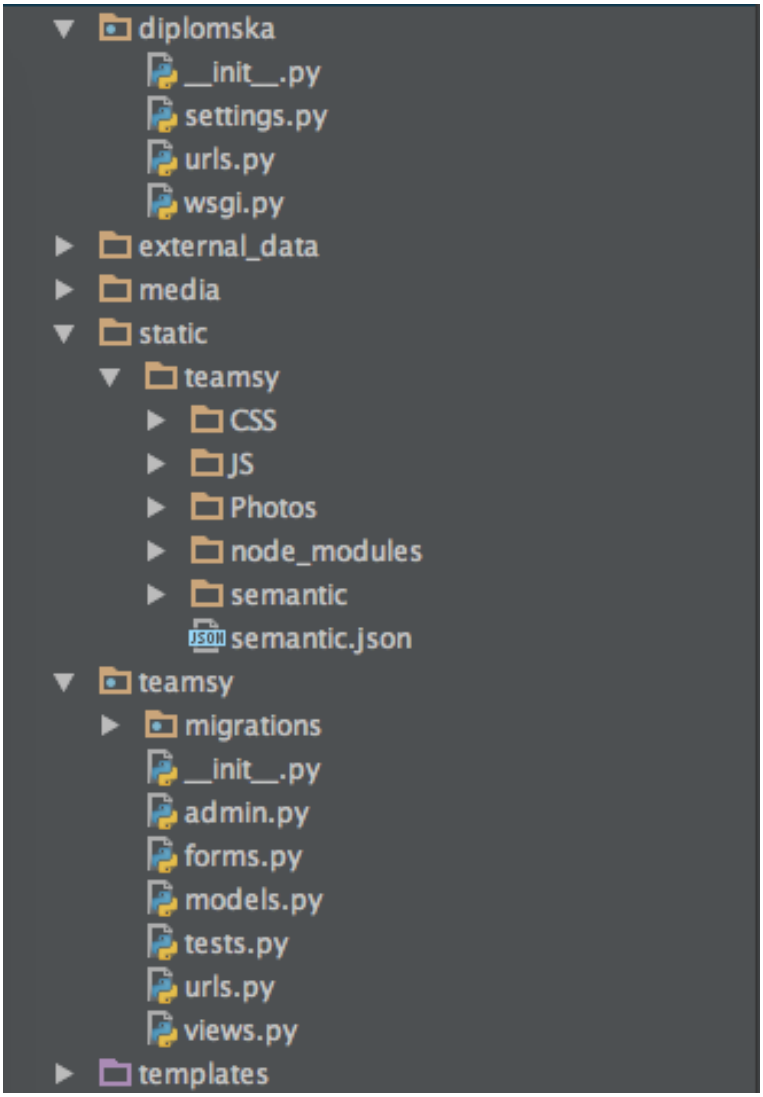
Razvoj spletne aplikacije

Za popolno delovanje aplikacije je bilo potrebno razviti izgled in zaledje aplikacije. Uporabil sem tehnologije opisane v prejšnjem poglavju, Pythonovo ogrodje Django za zaledje, za ospredje aplikacije pa sem uporabil HTML, CSS in JavaScriptovo knjižnico jQuery. Večina razvoja je potekala v okolju PyCharm. Pri pisanju in oblikovanju uporabniškega vmesnika sem upošteval dobre prakse razvoja uporabniške izkušnje. Principi, kot so odziven uporabniški vmesnik, uporaba ikon namesto besedila, enostaven izgled s poudarkom na vsebini in tipografiji, so pomembni za kakovostno uporabniško izkušnjo, ki uporabnika prepriča v nadaljnjo uporabo.

3.1 Vzpostavitev okolja in struktura map

Pred začetkom razvoja aplikacij v programskem jeziku Python je potrebno prenesti in namestiti njegov prevajalnik. Ob kreiraju novega projekta znotraj aplikacije PyCharm se dodeli privzeta hierarhična struktura map, standardna za vse aplikacije, ki temeljijo na Djangu. Znotraj korenske mape, poimenovane "diplomska"(naslov projekta), najdemo istoimensko mapo v kateri se nahajajo osnovne konfiguracijske datoteke. V podmapi, poimenovani "teamsy"(naslov aplikacije), najdemo datoteke ključne za pravilno delovanje zaledja spletne aplikacije, med drugim tudi datoteki z modeli in kontrolerji.

Ospredje aplikacije, razbito na več različnih HTML datotek, se nahaja v mapi "templates". Mapa "media" služi kot vsebovalnik za datoteke prenesene na strežnik, mapa "static"pa vsebuje zunanje datoteke CSS in JavaScript ter grafike uporabljene v aplikaciji. Opisana struktura je prikazana na sliki 3.1.



Slika 3.1: Struktura map za projekt, ki temelji na Djangu in jo ustvari ogrodje PyCharm.

3.2 Modeliranje relacijske podatkovne baze

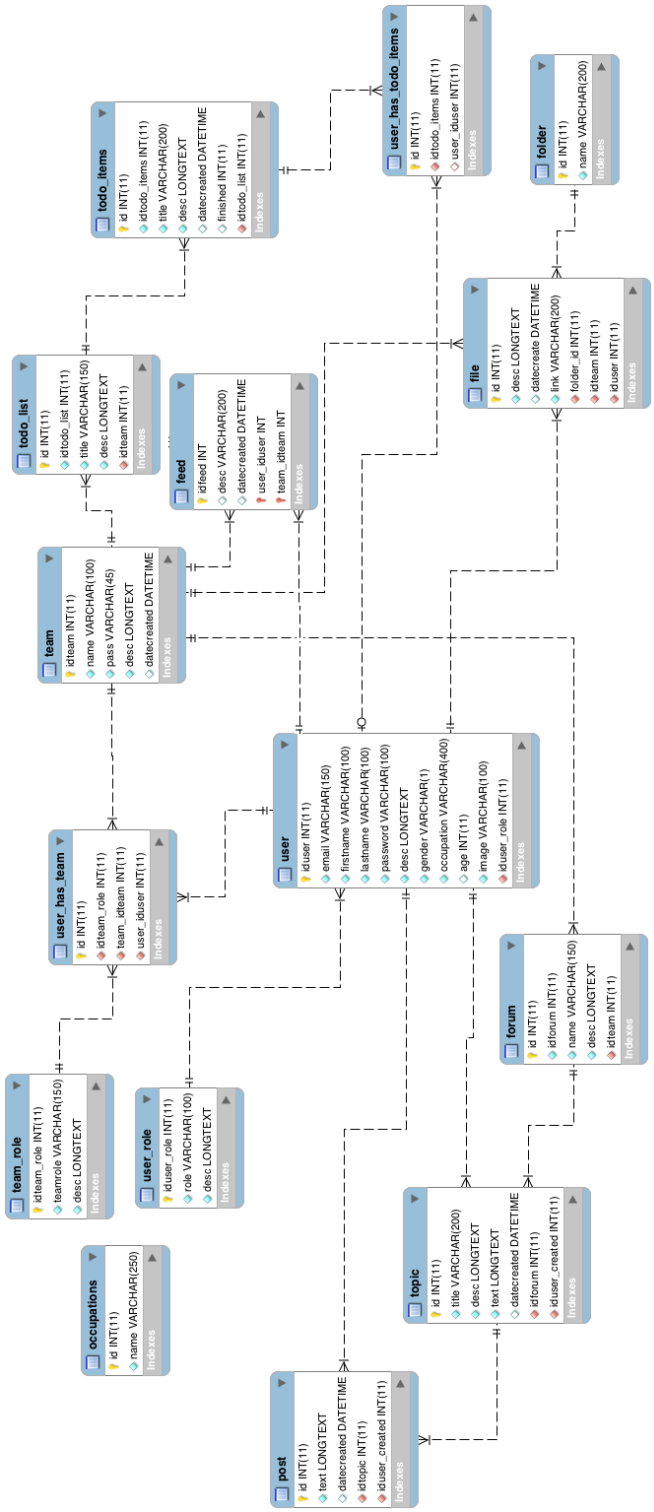
Model podatkovne baze je sestavljen iz štirinajstih po meri ustvarjenih tabel in desetih Djangovih avtomatsko generiranih tabel. Struktura tabel je bila prvotno ustvarjena v orodju MySQL Workbench in prenesena v lokalno podatkovno bazo. Django je z obratno migracijo podatkov poskrbel za generiranje ustreznih modelov. Morebitne popravke je tako možno udejanjiti na dva načina: s popravki, ki jih naredimo na vizualno narisanih tabelah podatkovne baze in ponovnim prenosom ter obratno migracijo ali s popravki v datoteki `models.py` in migracijo modelov v tabele podatkovne baze. Zaradi hitrejšega razvoja sem izbral slednje. Končna različica strukture podatkovne baze je prikazana na sliki 3.2 in vsebuje naslednje tabele:

- `user`: Vsebuje podatke o registriranem uporabniku.
- `team`: Vsebuje podatke o ustvarjeni skupini uporabnikov.
- `user_has_team`: Vmesna tabela tabel `user` in `team`, ustvarjena zaradi medsebojne relacije več na več (many-to-many[1]).
- `user_role`: Vsebuje uporabniške vloge na nivoju aplikacije.
- `team_role`: Vsebuje vloge na nivoju posamezne ekipe uporabnikov.
- `occupations`: Tabela z vsebovanim (preddefiniranim) seznamom poklicev.
- `forum`: Posamezna vrstica tabele pripada vsaki skupini, vsebuje pa osnovne podatke o forumu za določeno skupino uporabnikov.
- `topic`: Vsebuje podatke o temi razprave (v aplikaciji poimenovano: discussion). Povezana s tabelo `forum` preko relacije ena na več (one-to-many[2]).

[1]Many-to-many relacija: ena ali več vrstic tabele je lahko povezana z nobeno, eno ali več vrsticami druge tabele. Ustvarjene povezave so shranjene v vmesni tabeli.

[2]One-to-many relacija: ena vrstica tabele je lahko povezana z eno ali več vrsticami druge tabele.

- `post`: Vsebuje tekst in ostale podatke posameznega sporočila v razpravi. Povezava ena na več s tabelo `topic`.
- `todo_list`: Podobno kot `forum`, vsebuje podatke o seznamu opravil za določeno skupino uporabnikov.
- `todo_items`: Tabela z vsebovanim seznamom opravil.
- `user_has_todo_items`: Vmesna tabela tabel `user` in `todo_items`, ustvarjena zaradi medsebojne relacije več na več.
- `file`: Vsebuje podatke o naloženi datoteki. Povezava ena na več s tabelo `folder` je opcijska.
- `folder`: Vsebuje podatke o morebitnih mapah datotek.
- `feed`: Vsebuje podatke o dogajanju v posamezni skupini (pridruženi člani, ustvarjene diskusije...)

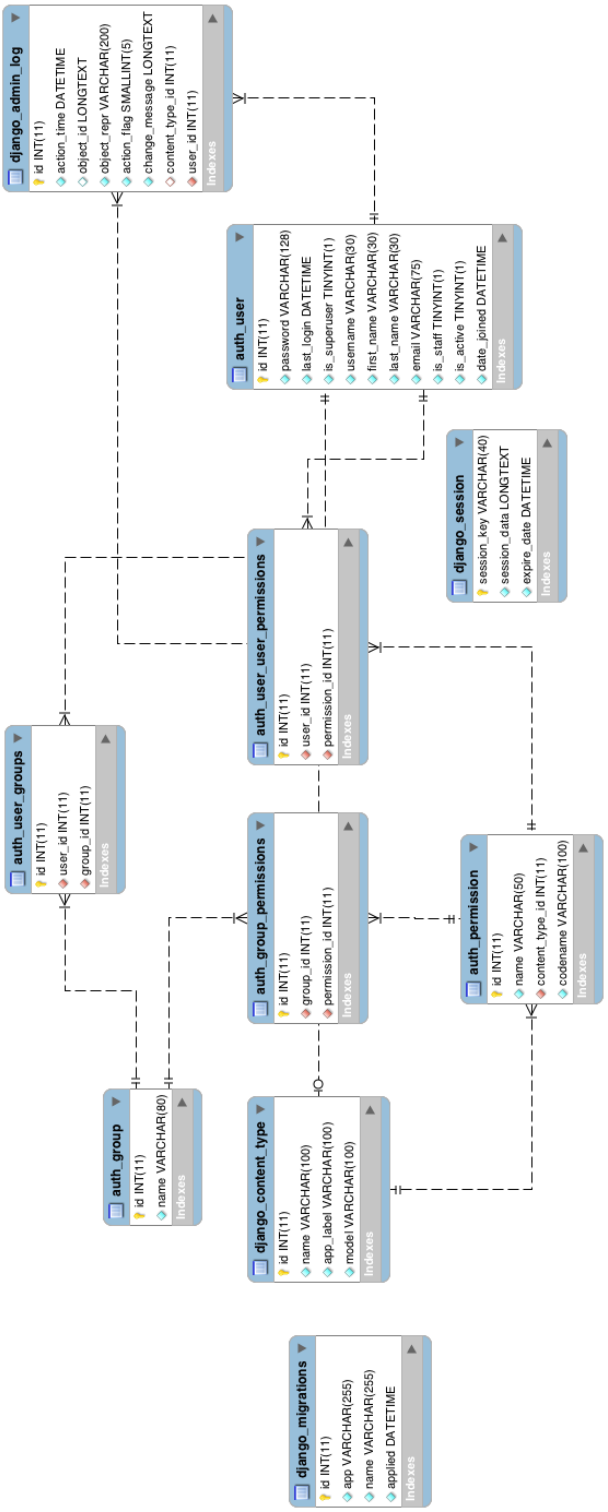


Slika 3.2: Model relacijske podatkovne baze narisan v orodju MySQL Workbench. Tabele služijo shranjevaju uporabniških podatkov.

Djangov interni migracijski sistem poleg tabel, ki jih definira programer, avtomatsko doda še nekaj svojih (struktura na sliki 3.3). Tabele služijo predvsem namenu avtentikacije in dodeljevanju uporabniških vlog, vendar ne omogočajo veliko maneverskega prostora, zato se veliko programerjev odloči za lastno implementacijo. Avtomatsko generirane tabele so:

- `auth_group`: Vsebuje opis skupine uporabnikov.
- `auth_group_permissions`: Vsebuje opise dovoljenj za določeno skupino.
- `auth_permission`: Vsebuje opise dovoljenj za določenega uporabnika.
- `auth_user`: Vsebuje podatke o registriranem uporabniku.
- `auth_user_groups`: Vmesna tabela, ki preko relacije več na več povezuje tabeli `auth_group` in `auth_user`.
- `auth_user_user_permissions`: Vmesna tabela, ki preko relacije več na več povezuje tabeli `auth_user` in `auth_permission`.
- `django_admin_log`: V tabelo se beležijo aktivnosti uporabnika v aplikaciji.
- `django_content_type`: Vsebuje podatke o vsebini aplikacije, ki jo razvijamo.
- `django_migrations`: Hrani časovne pečate opravljenih migracij podatkovne baze.
- `django_session`: Hrani podatke o sejah prijavljenih uporabnikov.

Programer se odloči, če bo uporabljal zgoraj naštete avtomatsko generirane tabele ali pa bo spisal svoje. Uporaba prej definiranega sistema je lahko zelo uporabna, vendar je to odvisno tudi od namena implementacije.



Slika 3.3: Model avtomatsko generiranih tabel ogrodja Django. Tabele služijo shranjevanju podatkov za avtentikacijo in dodeljevanje uporabniških vlog. Uporaba teh tabel je opcijska.

3.3 Struktura ogrodja Django in MVC

Pythonovo ogrodje Django deluje po arhitekturnem principu MVC (*Model-View-Controller*). Vsako izmed komponent programiramo, predvsem zaradi boljše preglednosti, v ločene datoteke.

3.3.1 Model

Model je objekt, predstavlja pa vir informacij o podatkih, ki se nahajajo v podatkovni bazi. Vsebuje polja, karakteristike in funkcije bistvene za predstavitev podatkov, shranjenih v bazi. Praviloma vsak model pripada istoimenski tabeli v podatkovni bazi. Model je Pythonov razred, ki pripada zbirki `django.db.models.Model`, vsak njegov atribut pa predstavlja polje v neki tabeli podatkovne baze. Django omogoča avtomatsko zgeneriran API (Application Programming Interface, aplikacijski programski vmesnik) za dostop do podatkovne baze [49]. Modele pišemo v obliki Pythonovih razredov v datoteko `models.py` z uvozom ustrezne zbirke.

3.3.2 Pogled

Komponenta pogled predstavlja to, kar vidimo na zaslonu kot spletno aplikacijo. V Djangu skrbijo za generiranje HTML strani, ki se pošlje odjemalcu, kjer se prikaže v brskalniku, predloge imenovane *Templates*. Vsebujejo statično HTML kodo in nekaj značk spisanih v posebni sintaksi, ki brskalniku narekujejo, kako naj izpisuje dinamične podatke. Uporaba predlog ni obvezna, vendar je priporočljiva zaradi dobre integracije z zalednim sistemom. Dinamične podatke predstavljamo z Djangovim programskim jezikom za predloge, imenovanim *Django Template Language*, ki poleg izpisa podatkov, ponuja še nekaj uporabnih funkcionalnosti, na primer zanke in pogojne stavke. HTML datoteke so shranjene v podmapi “templates/teamsy”.

3.3.3 Kontroler

Privzeto Django za shranjevanje kode kontrolerjev pripravi datoteko `views.py`, ki s svojim imenom nekoliko zmede začetnika. Situacija je sicer logična, saj kontrolerji znotraj omenjene datoteke poganjajo pripadajočo HTML predlogo (pogled). Kontoler je funkcija, ki kot parameter sprejme spletno zahtevo, vrne pa odgovor, ki je lahko predstavljen v obliki HTML vsebine strani, preusmeritve, napake, slike, XML dokumenta, itd. Odgovor je lahko tudi prazen, torej funkcija ne vrne ničesar. Ustrezni preusmeritvi dodamo še kontekstno spremenljivko (tipično polje, seznam) s pridobljenimi podatki, pripravljenimi za prikaz. Kontroler vsebuje kodo, ki izvrši poslovno logiko, potrebno za generiranje odgovora, ne vsebuje pa kode za DOM manipulacijo HTML elementov. Kontolerje lahko po želji pišemo tudi v druge datoteke, vendar je v tem primeru potrebno ustrezno nastaviti poti. Django poleg funkcijskih kontrolerjev implementira še razredne kontrolerje. Ti ponujajo boljšo strukturiranost pri pisanju korenske datoteke in omogočajo uporabo določenega kontrolerja za več HTML predlog.

3.3.4 Usmerjanje

Usmerjanje je pomemben del koncepta MVC, saj povezuje predloge in razdrobljene dele zaledja aplikacije. Django nima posebnih omejitev glede lokacije kode usmerjanja, privzeta datoteka za programiranje povezav pa je `urls.py`. Pri pripravi okolja moramo upoštevati uvoz paketov iz `django.conf.urls` in funkcij iz naše datoteke `views.py`. Procesiranje prejete zahteve poteka po naslednjem postopku [50]:

1. Ob prejemu HTTP zahteve (`HttpRequest`), privzeto, če v atributu `urlconf` HTTP zahteve ni specificirano drugače, Django uporabi modul `URLconf`, nastavljen v spremenljivki `ROOT_URLCONF` v nastavitvah aplikacije.

2. V navedenem modulu Django poišče spremenljivko `urlpatterns`, katere vsebina je seznam `django.conf.urls.url()` instanc.

3. Django iterira čez seznam instanc in se ustavi pri prvi, ki se ujema s povezavo iz zahteve.

4. Ob ujemanju prejete povezave in regularnega izraza navedenega v eni izmed vrstic seznama povezav, se pokliče ustrezen kontroler iz datoteke `views.py`.

5. Če ni nobenega ujemanja, se v kateri koli izmed točk algoritma sproži izjema. Django pokliče ustrezen kontroler za rokovanje napak.

3.3.5 Obrazci

Pri programiranju naprednih spletnih aplikacij s potrebnim uporabniškim vnosom uporabljamo spletne obrazce oziroma forme. V HTML označevalnem jeziku spletni obrazec predstavlja skupek elementov znotraj elementa, ki ga definira značka `<form>`, omogoča pa interakcijo uporabnika s strežniškim zaledjem aplikacije. Komunikacija obrazcev s strežnikom poteka le preko metod `GET` in `POST`. Tipično se `POST` uporablja za zahteve, ki spreminjajo stanje sistema (spreminjanje stanja podatkovne baze), `GET` pa naj bi se uporabljal le za zahteve, ki stanja sistema ne spreminjajo. `GET` podatke pretvori v znakovni niz, ki se pojavi v `URL` (*Uniform Resource Locator*, enolični krajevnik vira) naslovu aplikacije. Ni primeren za vnos gesel, uporabniških imen, zaupnih podatkov ali prenosa daljših besedilnih sporočil. V MVC sistemih obrazci praviloma ne predstavljajo ločenih komponent, ampak so opisani kot HTML značke.

Django s svojim ločenim sistemom za upravljanje spletnih obrazcev poskrbi za večjo varnost, podatke pa prestrukturira tako, da so pripravljeni za upodobitev. V osrčju sistema deluje Pythonov razred `Form`, ki opisuje delovanje in izgled spletnega obrazca. Na podoben način, kot Djangov `Model` preslika svoje atribute s polji podatkovne baze, se polja razreda `Form` preslikajo v HTML elemente `<input>` [51]. Obrazce in njegove komponente pišemo kot Pythonove razrede v datoteko `forms.py`, v katero je predhodno potrebno vključiti knjižnico `django.forms`. Za ustrezen prikaz na zaslon v

kontrolerju ustvarimo instanco na željen obrazec iz prej navedene datoteke in jo dodamo kontekstni spremeljivki za prenos v predlogo. Za pretvorbo v ustrezne HTML značke uporabimo `Django Template Language`.

3.3.6 Nastavitve

Vse pomembne nastavitve projekta pišemo v datoteko `settings.py`, med drugim definicije poti do map s predlogami (Templates), statičnimi datotekami (Static Files - CSS, JavaScript, slikovne datoteke) ter pot do mape "media". V isti datoteki je shranjena tudi povezava s podatkovno bazo. V odseku kode:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME':  'mydb',
        'USER': 'root',
        'PASSWORD': 'root',
        'HOST': '127.0.0.1',
        'PORT': '8889',
    }
}
```

je predstavljen realen primer povezave s podatkovno bazo MySQL, v isti seznam pa lahko dodamo več povezav na raznovrstne podatkovne baze, tako da preklapljanje med njimi poteka na enostaven način. V zgornjem primeru je aplikacija povezana na lokalno podatkovno bazo `mydb` in je dostopna na vratih 8889.

3.4 Osnovna stran, prijava in registracija

3.4.1 Izgled in uporabniška izkušnja

Prva stran razvite aplikacije nam prikazuje le osnovne podatke o samem produktu. Na sliki 3.4 vidimo pozdravni zaslon s sliko, naslovom in sloganom spletne storitve, opazimo pa še obrazec za registracijo in nekaj gumbov.

Slika 3.4: Osnovna stran razvite spletne aplikacije.

Ospredje aplikacije je zgrajeno s HTML elementi, pripeti atributi (identifikatorji in razredi), ki sledijo principom programiranja v ogrodju Semantic UI, pa skrbijo za pravilen izris na zaslonu. Zbirka ogrodja Semantic UI ime-

novana *Grid*, služi kot osnovni vsebovalnik strani in pripomore k lažjemu programiranju odzivne spletne aplikacije. Zbirka *Grid*, ki jo definiramo kot skupek razredov `ui grid` v HTML elementu `<div>`, dokument razdeli v mrežo elementov. Stolpce in vrstice ponovno pišemo kot HTML značke `<div>`, definiramo pa jih z ustreznimi razredi `ui column` za stolpce in `ui row` za vrstice. Osnovna zgradba zbirke *Grid* je predstavljena na sliki 3.5. Vsak izmed pravokotnih elementov ima pripet razred `column`, obe vrstici pa sta vsebovani, vsaka v svojem elementu s pripetim razredom `row`. Elementa v zgornji vrstici poleg omenjenih razredov vsebujeta še razreda `left floated` in `right floated`, ki narekujeta postavitve elementov, spodnji elementi pa imajo s pomočjo razredov `three wide`, `eight wide` in `five wide` definirane še širine, katerih širina se mora sešteti v 16 enot, da zapolnijo celotno širino starševskega objekta. Semantic UI ponuja veliko dodatnih (neobveznih) razredov, ki jih lahko dodamo osnovnim konstruktom. Med najbolj uporabne sodijo `stackable` (elementi se ob manjšanju širine zaslona hierarhično zlagajo eden na drugega), `centered` (vsebina elementa je poravnana na sredino) in `wide` (direktno določanje širine polja, brez ponovne uporabe jezika CSS).

Slika 3.5: Osnovna zgradba zbirke Grid.

3.4.2 Prijava in registracija

Funkciji za prijavo in registracijo sta opisani z dvema različnima obrazcema. Registracijski obrazec od uporabnika zahteva osnovne podatke za kreiranje uporabniškega računa. Ob pravilno vnesenem imenu, priimku, naslovu elektronske pošte in geslu pa zaledje aplikacije pripravi ter shrani nov vnos v tabelo uporabnikov. V odseku kode:

```
class UserRegistrationForm(forms.Form):
  firstname = forms.CharField(
    widget=forms.TextInput(attrs={'placeholder': 'First name'})
  )
  lastname = forms.CharField(
    widget=forms.TextInput(attrs={'placeholder': 'Last name'})
  )
  email = forms.CharField(
    widget=forms.EmailInput(attrs={'placeholder': 'Email'})
  )
  password = forms.CharField(
    widget=forms.PasswordInput(attrs={'placeholder': 'Password'})
  )
```

je prikazan razred `UserRegistrationForm`, ki kot atribut prejme instanco razreda `Form`. Razred `CharField` je Pythonov razred, ki v ogrodju Django opisuje tekstovno polje. V atribut `widget` zapišemo ustrezen razred, iz katerega bo nastalo vnosno polje (HTML element `<input>`). Razred `forms.EmailInput` se bo, na primer v predlogah (s pomočjo jezika *Django Template Language*) tako preslikal v HTML element `<input id="id_email name="email" type="email">`. Atribute lahko definiramo ročno (npr. `placeholder` v zgornjem primeru), `id`, `name` in `type` pa se dodelijo avtomatsko. Prijavni obrazec od uporabnika zahteva le vnos elektronskega naslova in ustreznega gesla.

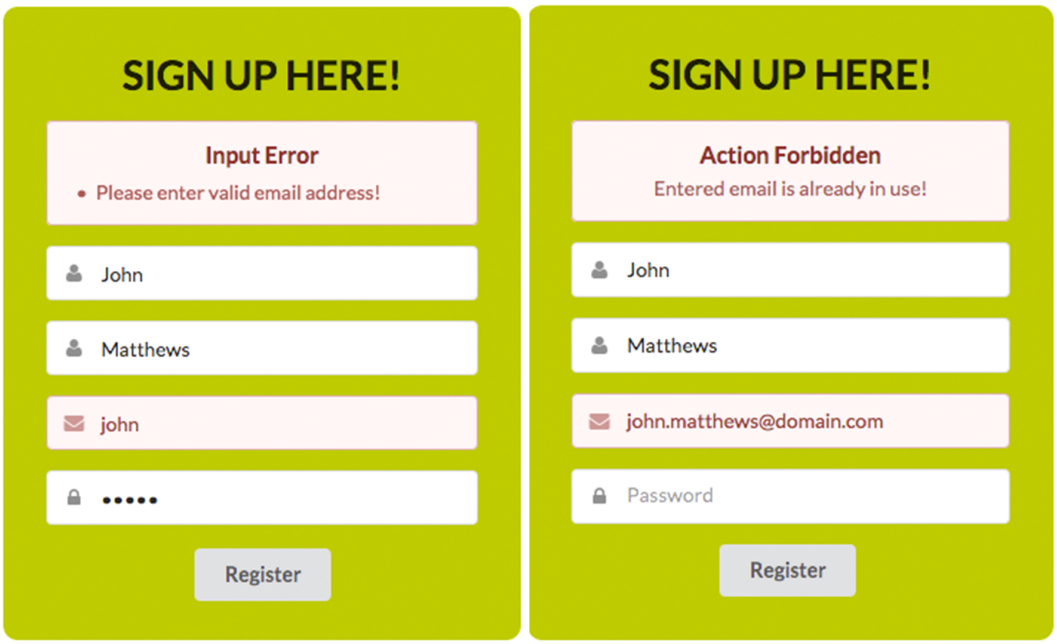
3.4.3 Validacija

Zaradi večje varnosti in boljše uporabniške izkušnje je preverjanje uporabniškega vnosa velikega pomena pri programiranju spletnih aplikacij. Pri večini uporabniških vnosov sem uporabljal dvojno validacijo. Na odjemalčevi strani (t.i. *Client-side Validation*) za preverjanje uporabniškega vnosa skrbi JavaScriptova knjižnica jQuery, podkrepljena z validacijskimi razredi iz ogrodja Semantic UI. Preprečuje prazne vnose, nepravilne vnose ali vnose neustrezne

dolžine, še preden se podatki prenesejo v zaledni del aplikacije. Izsek kode:

```
email: {
        identifier: 'email',
        rules: [{
             type: 'empty',
             prompt: 'Please enter your email!'
        },{
             type: 'email',
             prompt: 'Please enter valid email address!'
        }]
}
```

opisuje validacijo na polju za vnos elektronske pošte. Polje ne dovoljuje praznih vnosov ali vnosov, ki sintaktično ne sledijo definiciji elektronskega naslova. Če katero izmed pravil ustavi nadaljevanje izvajanja, se izpiše ustrezna napaka (`prompt`). Slika 3.6 (levo) prikazuje okno za napako, ki se pojavi ob odkriti napaki na strani odjemalca. Strežniška validacija (t.i. *Server-side Validation*) se sproži šele po uspešno izvedenem preverjanju na strani odjemalca. Potrjevalna koda teče na strežniku, preverja pa pridobljene vnose in jih primerja s podatki v podatkovni bazi (podvojeni elektronski naslov, nepravilno geslo). Ob najdeni nepravilnosti se izpiše ustrezna napaka. Slika 3.6 (desno) prikazuje okno za napako, ki se pojavi ob odkriti napaki na strani strežnika.



Slika 3.6: Levo: prikaz neustreznega vnosa elektronskega naslova, napako sproži validacija na strani odjemalca. Desno: Izpis napake ob že obstoječem elektronskem naslovu, validacija na strani odjemalca ne ustavi delovanja, stežniško preverjanje zapisov v bazi sproži napako.

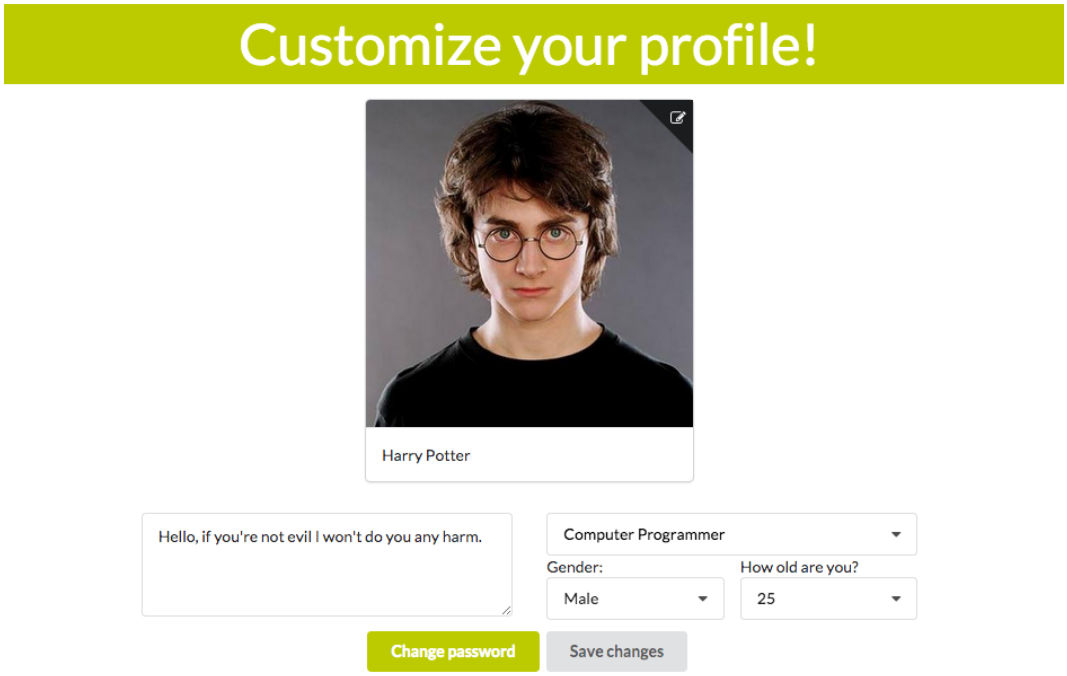
3.4.4 Seja in sejna spremenljivka

Po uspešno izvedenem preverjanju vnosa in prijavi uporabnika v aplikacijo, se ustvari sejna spremenljivka, v katero shranimo pomembne podatke o prijavljeni osebi. Spremenljivka služi kot shramba za podatke, do katerih lahko dostopamo med različnimi HTTP (*Hypertext Transfer Protocol*, protokol za prenos hiperteksta) zahtevki (ne glede na to, na kateri podstrani se nahajamo znotraj določene seje). Uporaba sej je pomembna, saj je HTTP protokol, ki ne shranjuje stanja [52]. Ob uspešni prijavi s klicem `request.session['email'] = email` v sejno spremenljivko shranimo trenutno prijavljenega uporabnika. Do podatkov, shranjenih v sejnih spremenljivkah, dostopamo s klicem funkcije `get()`, v prej opisanem primeru:

`request.session.get('email')`.

3.5 Stran za urejanje uporabniškega profila

Po uspešni registraciji novega uporabnika nas aplikacija preusmeri na stran za urejanje uporabniškega profila (slika 3.7), kjer imamo možnost prilagajanja naših osebnih podatkov. V procesu registracije aplikacija od nas zahteva le ime, priimek in elektronski naslov, tu pa dobimo še možnost spreminjanja privzetega avatarja, izbranega poklica, spola, starosti in kratkega opisa.



Slika 3.7: Postavitev strani za urejanje uporabniškega profila.

3.5.1 Spustni meniji in izbira poklica

Spustni meniji v aplikaciji so sicer prikazani kot običajni HTML elementi, definirani s skupkom značk `<select>` in `<option>`. Meni za izbiro poklica

pa nam poleg navadne izbire omogoča tudi uporabniški vnos in sprotno iskanje ustrezne izbire, zato uporaba standardne HTML značke ni bila mogoča. Prikazana koda v jeziku HTML:

```
<div class="default text" id="occupation_text">
  Select your occupation
</div>
<div class="menu">
  {% for occ in occupations %}
    <div class="item" data-value=''>{{ occ.name }}</div>
  {% endfor %}
</div>
```

opisuje spustni meni za izbiro poklica. Zanka *for* iterira čez vse, iz baze pridobljene poklice, ki smo jih s pomočjo seznama `occupations` in kontekstne spremenljivke iz kontrolerja (funkcija v datoteki views.py) podali ustrezni predlogi. Ob izbiri želenega poklica se privzeto besedilo v znački z identifikatorjem `occupation_text` zamenja z naslovom izbranega poklica. Omogočeno je tudi iskanje in avtomatsko dopolnjevanje po dolgem seznamu. Iskanje rezultate filtrira glede na ujemanje vpisanega besedila in besedila v katerikoli izmed značk `<div>` z razredom `item`, ali besedila v atributu `data-value` značke `<div>` z razredom `item`. Rezultati se ne prenašajo iz podatkovne baze, ampak skrbi za filtriranje zadetkov jQueryeva funkcija v knjižnici Semantic UI. Ker značka, v kateri prikazujemo uporabniško izbiro, ni vnosno polje in ne deluje z obrazci jezika HTML, se ob spremembi spustnega menija, uporabnikova izbira prepiše v skrito vnosno polje, ki ob potrditvi obrazca prenese podatke do zaledja aplikacije. Možnosti izbire poklicev so vnaprej definirane, zapisane pa so v datoteki tipa CSV (*Comma-Separated Values*, vrednosti ločene z vejico). Prikazana strežniška funkcija

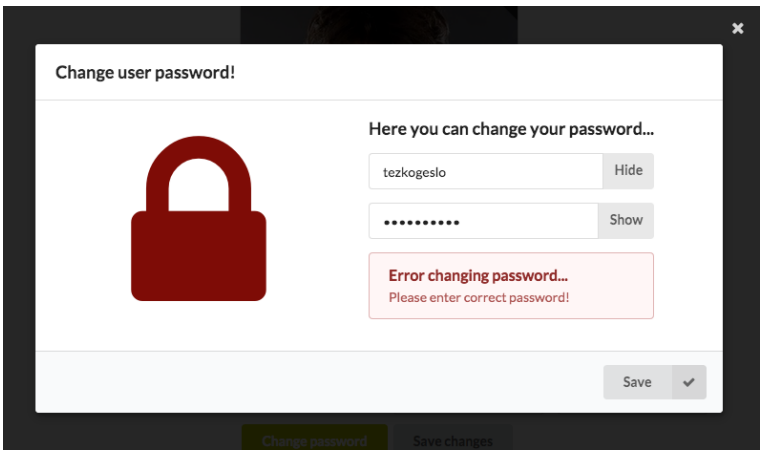
```
def add_occupations(occupations_data):
    with open(occupations_data) as f:
        for line in f:
```

```
line = unicode(line, 'utf-8')
occ = Occupations(name=line)
occ.save()
```

kot parameter sprejme ustrezno datoteko tipa CSV, jo odpre, prebere in vrstico po vrstico zapiše v tabelo `occupations` v podatkovno bazo.

3.5.2 Sprememba gesla

Obrazec za spremembo gesla (slika 3.8) se ob kliku na ustrezen gumb prikaže kot pojavno okno v sredini ekrana. Komunikacija s podatkovno bazo poteka preko Ajax klica, ki vnesena podatka prenese v zaledni sistem. Strežnik najprej preveri ustreznost vnosa dosedanjega gesla (zgornje vnosno polje) in ga ob ujemanju zamenja z novim. Ob neustreznosti gesla strežnik vrne napako, ki se izpiše na zaslonu (prikaz na sliki 3.8). Zaradi uporabe Ajax klica, komunukacija poteka tekoče in brez obnavljanja celotne strani.



Slika 3.8: Obrazec za spremembo gesla. Gumba *Show* in *Hide* omogočata prikaz ali maskiranje vnosnega polja. Komunikacija s strežnikom poteka preko Ajax klica.

Prikaz in maskiranje uporabniškega vnosa izmenično sprožimo s klikom na gumba *Show/Hide*. Gumba sta definirana kot HTML elementa, vsebujeta

pa atribut `data-toggle`, ki opisuje trenutno stanje gumba (*Show* ali *Hide*). Delovanje opisuje funkcija napisana v programskem jeziku JavaScript, z uporabo knjižnice jQuery:

```
(function($) {
  $.fn.showPass = function(button, input) {
    if(button.attr('data-toggle') == '0') {
      input.get(0).type = 'text';
      button.attr('data-toggle', 1);
      button.html('Hide');
    } else {
      input.get(0).type = 'password';
      button.attr('data-toggle', 0);
      button.html('Show');
    }
  }
})(jQuery);
```

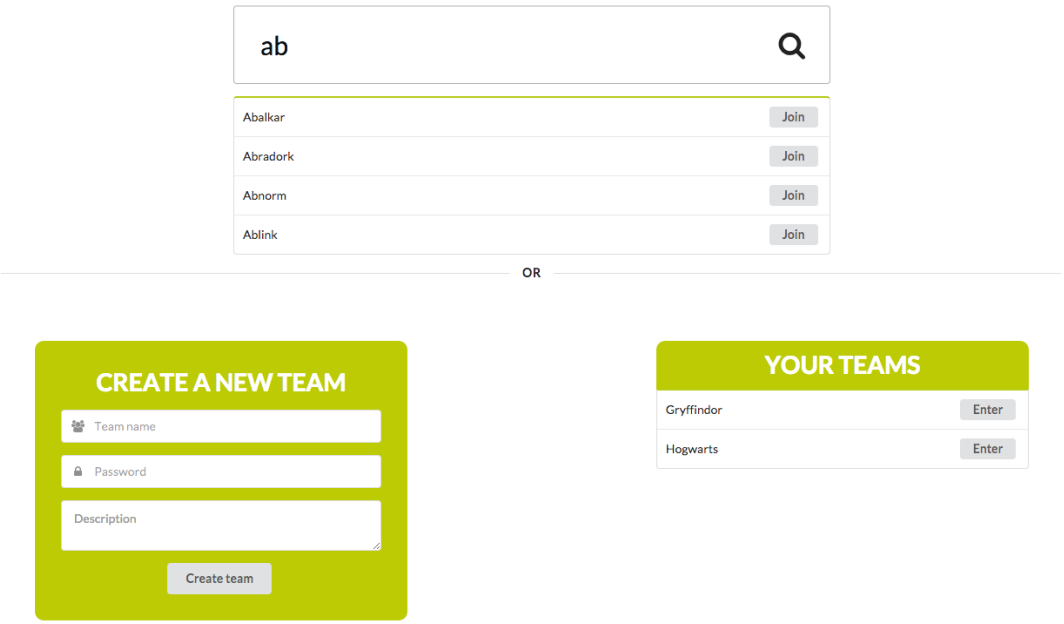
Prikaz/maskiranje izvajamo s spreminjanjem tipa vnosnega polja iz tekstovnega polja (`<input type='text'>`) v polje rezervirano za vnos gesel (`<input type='password'`) in obratno.

3.6 Osnovna stran izbire skupin

To je podstran, ki nam omogoča iskanje in vstop v obstoječe skupine ali pa ustvarjanje novih. Slika 3.9 prikazuje podstran za delo s skupinami. Na spodnjem delu strani najdemo obrazec za dodajanje novih skupin in seznam skupin, v katere je trenutno prijavljen uporabnik že vključen. Vrhnji predel strani vsebuje tekstovno vnosno polje, ki omogoča iskanje po vseh ustvarjenih skupinah. Komunikacija s podatkovno bazo, podobno kot na prejšnjih podstraneh, poteka preko Ajax klica. Zaradi slednjega je prikaz rezultatov hitrejši, sprotni izpis le teh pa nam omogoča funkcija, ki opisuje omenjeni Ajax klic:

```
$(‘#id_search’).keyup(function() {
  var text = $(this).val().toLowerCase();
  $.ajax({
    url: ‘/search/’,
    type: ‘post’,
    data: {
      ‘search’: text,
      ‘csrfmiddlewaretoken’: $(‘input[name=csrfmiddlewaretoken]’).val()
    },
    success: function(data) {
      $(‘#search_results’).html(data);
      if (text == ‘’) {
        $(‘#search_results’).html(null);
      }
    },
    failure: function(data) {
      alert(‘Search failed! Try again.’);
    }
  });
});
```

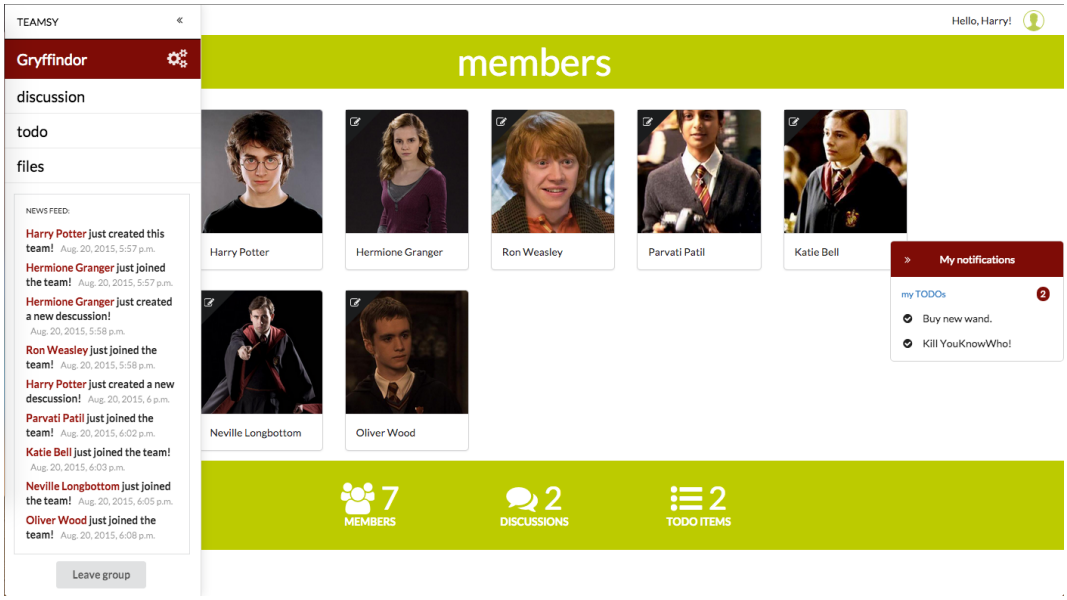
Povezava s strežniškim zalednjem se sproži ob vsakem *keyup* dogodku, torej po vsaki vpisani črki v iskalno polje. Ustrezen krmilnik (v zgornjem primeru imenovan `search`) vzpostavi povezavo s podatkovno bazo in poišče morebitne zadetke. Poizvedovanje po nizih v podatkovni bazi lahko v ogrodju Django zapišemo z le enim stavkom: `Team.objects.filter(name__icontains=inputed)` (spremenljivka `inputed` vsebuje iskalni niz), njegovo delovanje pa je podobno SQLovemu ukazu `LIKE`, ki v nizih, shranjenih v bazi, išče podani podniz. Kontroler podatke vrne kot HTTP odgovor (`HttpResponse(data)`), ki ga zgoraj opisana koda ulovi s funkcijo v parametru `success`. Na tem mestu lahko podatke formatiramo in jih izpišemo na ustrezno mesto na strani.



Slika 3.9: Del podstrani za iskanje, dodajanje ali pridružitev skupinam. Pod iskalnim poljem vidimo sprotni izpis rezultatov iskanja, spodaj levo se nahaja obrazec za dodajanje novih skupin, desno pa seznam skupin v katere je včlanjen trenutno prijavljen uporabnik.

3.7 Osnovna stran skupine

Osnovna stran skupine, prikazana na sliki 3.10, je ena najpomembnejših podstrani v celotni spletni aplikaciji. Ta pogled je iztočnica za povezovanje in interakcijo s člani skupine, saj omogoča takojšnji pregled članov in nekaj osnovne statistike (število članov, odprtih diskusij in nalog, vezanih na člane skupine), sestavljen pa je iz večih komponent. Slednje se, glede na vlogo prijavljenega uporabnika, nekoliko spreminjajo. Integrirani sta namreč dve vrsti uporabniških vlog (na nivoju posamezne skupine): *admin* (admnistrator, skrbnik skupine) in *user* (navadni uporabnik skupine). Kreator skupine privzeto pridobi skrbniške pravice nad skupino, kasneje pridruženi člani pa pridobijo le pravice navadnega uporabnika.



Slika 3.10: Osnovna stran skupine. Levi rob: stranska vrstica z navigacijskim menijem, oknom za prikaz tekočih dogodkov in gumbom za odjavo iz skupine. Desni rob: okno z obvestili prijavljenega uporabnika. Prijavljen je administrator skupine z možnostjo za dostop do nastavitev skupine in posameznega uporabnika.

3.7.1 Stranska vrstica in okno z obvestili

Stranska vrstica (*sidebar*) služi kot navigacijski meni in vsebuje najpomembnejše povezave za dostop do raznih funkcionalnosti. Na sliki 3.10 je prikazan primer, ko je v aplikacijo prijavljen uporabnik s skrbniškimi pravicami, kar mu omogoča dostop do nekaj dodatnih funkcionalnosti. V stranski vrstici pod običajnim navigacijskim menijem najdemo okno z novicami, v katerega se avtomatsko vpisujejo vsi pomembni dogodki aplikacije, in služi kot dnevnik za hiter pregled dogajanja. Ob kreiranju skupine, razprave, ali nove naloge se v tabelo *feed* v podatkovno bazo shranijo osnovni podatki o obvestilu: oseba odgovorna za določeno akcijo, v kateri skupini se je akcija zgodila, časovni pečat in opis akcije. Podatkovni model tabele *feed*, je istoimenski Pythonov

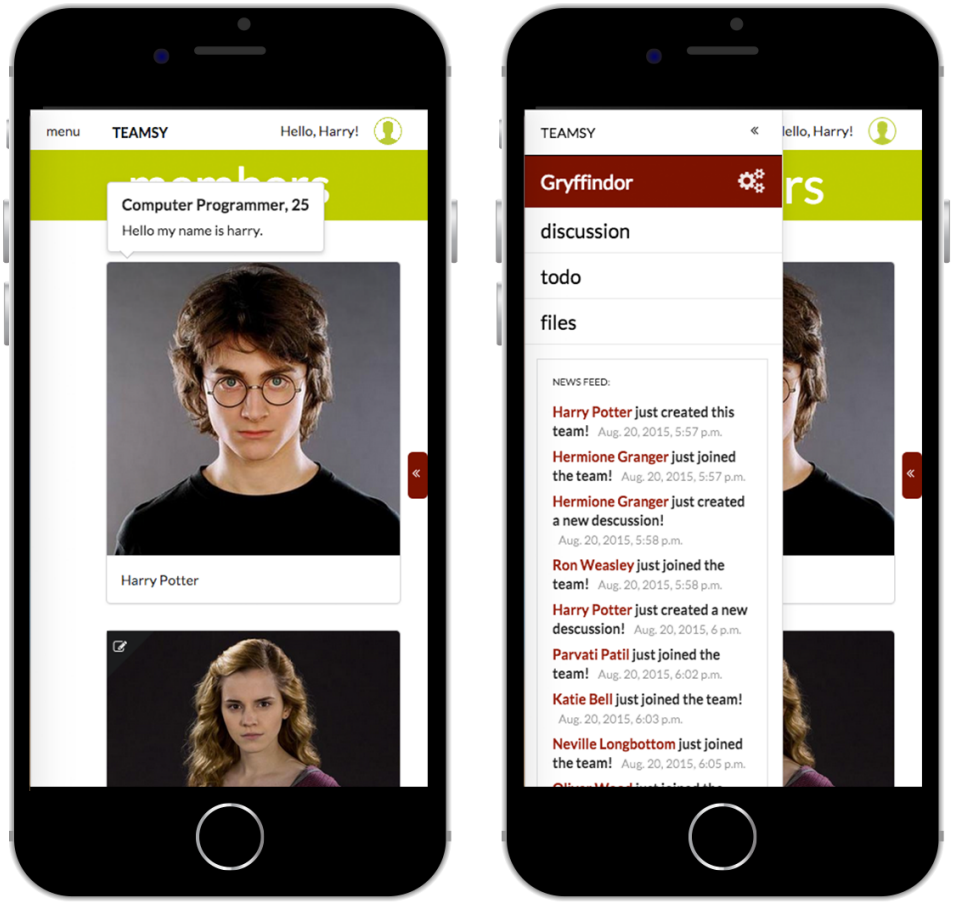
objekt v datoteki *manage.py* in skrbi za povezavo s tabelo v podatkovni bazi:

```
class Feed(models.Model):
  desc = models.CharField(max_length=500)
  user_iduser = models.ForeignKey(User, db_column='user_iduser')
  team_idteam = models.ForeignKey(Team, db_column='team_idteam')
  datecreated = models.DateTimeField(blank=True, null=True)
```

Okno z obvestili se nahaja na desnem robu zaslona in je posebej prilagojeno za vsakega prijavljenega člana. V oknu so prikazani zapisi o tekočih nalogah, dodeljenih prijavljenemu uporabniku. Prikazujejo se le neopravljene naloge, okno pa je vidno samo, če le te obstajajo. Obe komponenti, stranska vrstica in okno z obvestili, sta sprogramirani tako, da sledita načelu odzivnega dizajna aplikacije. Spodaj opisana funkcija skrbi za pravilno zapiranje obeh komponent (podobno izgleda tudi funkcija za odpiranje obeh komponent), glede na širino okna spletnega brskalnika :

```
(function($) {
  $.fn.closeSidebar = function() {
    $('#main_sidebar').hide('slide');
    $('#user_notifications').hide('slide');
    $('#open_notifications').show('slide');
  }
})(jQuery);
```

Za detekcijo dogodka spremembe širine brskalnikovega okna poskrbi jQueryeva funkcija: `$(window).on('resize', function() {...}`, minimalna širina zaslona pri kateri sta komponenti še odprti pa je 770px (*px*, zaslonska točka ali piskel). Na mobilnih napravah imamo dostop do (privzeto skritih) komponent, to sta stranska vrstica in okno z obvestili, preko za to dodeljenih gumbov, kot je to prikazano na sliki 3.11.

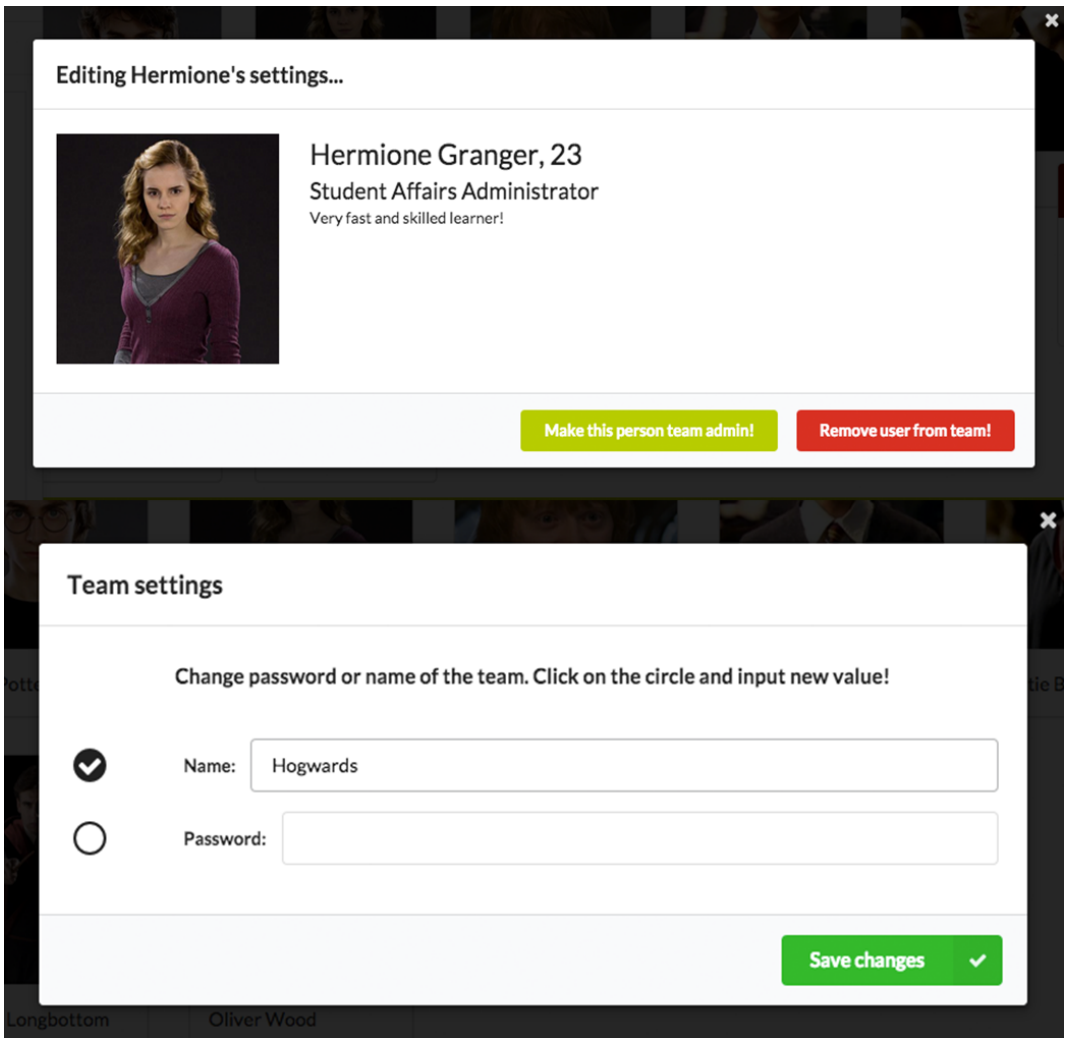


Slika 3.11: Prikaz aplikacije na mobilni napravi, kjer se vidi vloga odzivnega dizajna aplikacije. Levo: osnovna stran skupine, ob postavitvi miške na sliko se odprejo dodatne informacije o dotičnem članu skupine. Desno: ob kliku na gumb *menu* se odpre stranska vrstica, ki je sicer privzeto skrita v mobilnem pogledu.

3.7.2 Privilegiji administratorja

Prej omenjeni vrsti uporabniških vlog se najbolje odražata na opisani osnovni strani skupine. Uporabnik s skrbniškimi pravicami ima namreč dostop do nastavitev skupine, kar mu omogoča spreminjanje gesla in imena skupine, odstranjuje pa lahko tudi preostale člane ali jim dodeljuje administratorske

pravice, tako kot je to prikazano na sliki 3.12.

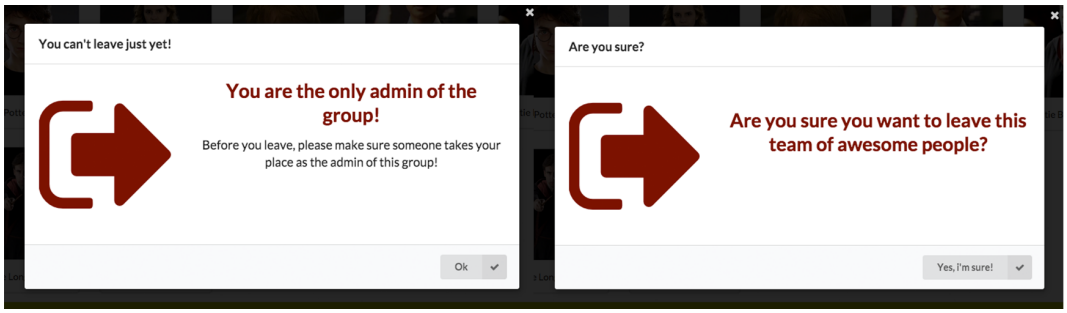


Slika 3.12: Pogovorni okni, ki ju vidi le skrbnik skupine. Zgoraj: spreminjanje nastavitev člana skupine, kot sta dodeljevanje administratorskih pravic ali odstranjevanje iz skupine. Spodaj: nastavitve skupine, kot sta spreminjanje imena ali gesla skupine. Pred vnosom nove vrednosti je polje za vnos potrebno omogočiti s klikom na ustrezen gumb.

Aplikacija ob kliku na gumb z nastavitvami uporabnika vedno pokliče enako pogovorno okno, njegova vsebina pa se prenese sproti (asinhroni klic)

iz strežnika. Nastavitve vsakega uporabnika so torej predstavljene s sliko, imenom in kratkim opisom.

Pojavno okno za spreminjanje nastavitev člana skupine je pomembno v navezi z gumbom za trajni odhod iz skupine. Slika 3.13 prikazuje dva možna primera za trajni odhod iz skupine. V situaciji, ko želi skupino zapustiti uporabnik s skrbniškimi pravicami (v primeru, da je edini s tovrstnim naborom pravic), mora prej enemu izmed preostalih članov dodeliti vlogo administratorja (slika 3.13 levo). Če odhajajoči ni edini skrbnik, ali ima zgolj uporabniške pravice, lahko skupino zapusti nemoteno (slika 3.13 desno).



Slika 3.13: Pojavni okni, prikazani ob kliku na gumb za odhod iz skupine. Levo: namenjeno skrbniku, če je edini uporabnik z administratorskimi pravicami. Desno: namenjeno vsem uporabnikom, ki niso edini administratorji in želijo zapustiti skupino.

3.7.3 Osnovni prikaz in odzivnost strani

Prej opisane komponente so le del osnovnega prikaza strani skupine. Najvidnejši del so kartice z avatarji uporabnikov skupine. Število včlanjenih ljudi v skupino ni omejeno, torej se stran vertikalno podaljšuje glede na število prikazanih kartic (ne več kot pet v eni vrstici). Naslednja koda, ki je sicer obdana z zanko *for*, ki iterira čez vse včlanjene uporabnike skupine (`member`), opisuje prikaz ene izmed kartic.

```
<div class="column card_image_container">
```

```
  <div class="ui fluid card">
    <div class="image"
              data-title="{{ member.user_iduser.occupation }},
                                {{ member.user_iduser.age }}"
              data-content="{{ member.user_iduser.desc }}">
      {% if team_role_logged.idteam_role.teamrole == 'admin' %}
        {% if member.user_iduser != user %}
          <div class="ui left black corner label">
            <i class="edit icon"
                      data-user="{{ member.user_iduser.iduser }}"
                      data-team="{{ team.idteam }}">
            </i>
          </div>
        {% endif %}
      {% endif %}
      {% if not member.user_iduser.image %}
        <img src="/static/teamsy/Photos/avatar.jpg">
      {% else %}
        <img src="{{ member.user_iduser.image.url }}"
                  class="profile_photo_members">
      {% endif %}
    </div>
    <div class="content">
      <p style="overflow: scroll;">
        {{ member.user_iduser.firstname }} {{ member.user_iduser.lastname }}
      </p>
    </div>
  </div>
</div>
```

Pogojni stavki *if* vključujejo napredne možnosti za skrbnike skupine in omogočajo prikaz privzete slike, četudi uporabnik nima nastavljenega svojega avatarja.

Ob zmanjševanju zaslona se kartice razbijejo iz vrste in zložijo ena na drugo, za kar poskrbi sistem mrežne postavitve (*Grid system*) ogrodja Semantic UI.

3.8 Strani za diskusije, tekoče naloge in deljenje datotek

Strani za diskusije, tekoče naloge in deljenje datotek so podstrani, ki predstavljajo ključne funkcionalnosti spletne aplikacije. Ponujajo interakcijo z ostalimi člani skupine in omogočajo posameznikovo aktivno sodelovanje s celotno ekipo.

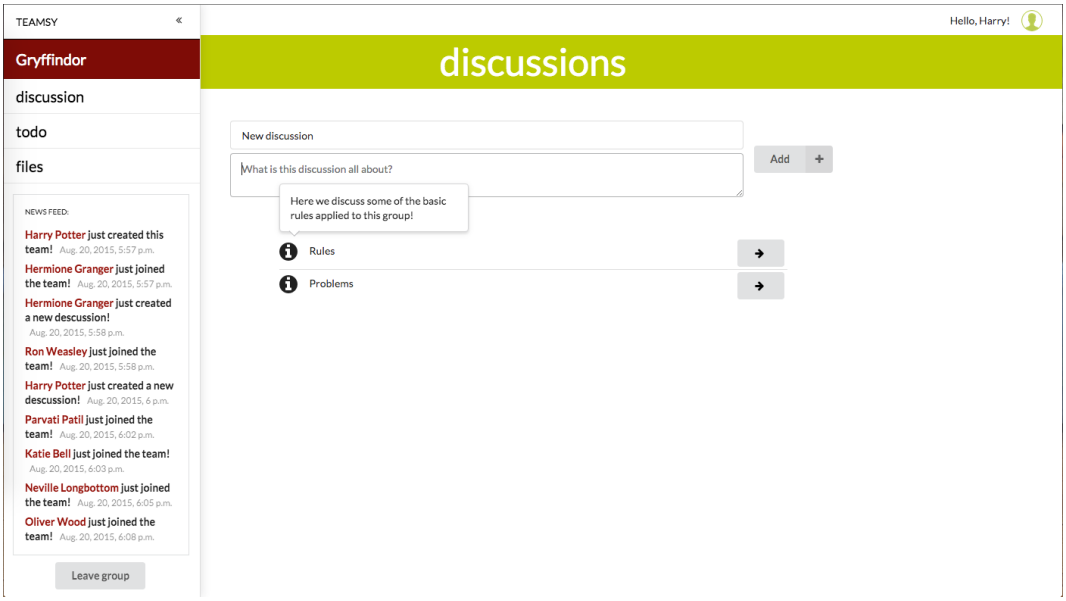
3.8.1 Stran za diskusije

Uporaba klasičnih forumov z leti čedalje bolj upada, vendar ideja in zasnova storitev za medsebojno komunikacijo ostajata podobni. Komunikacijski sistem je v aplikaciji *Teamsy* implementiran kot forum v nekoliko modernejši preobleki. Na sliki 3.14 je prikazana osnovna stran komunikacijskega sistema. V zgornje tekstovno vnosno polje vpišemo naslov diskusije, ob začetku pisanja pa se odpre še tekstovno polje za vnos opisa teme. Ob kliku na gumb za dodajanje se nova tema doda na spodnji seznam. S klikom na desno usmerjeno puščico, pa nas aplikacija preusmeri v klepetalnico. Prenos podatkov iz vnosnih polj na strežnik poteka preko zahtevka `POST`.Spodnja koda prikazuje del funkcije, ki v ustreznem krmilniku skrbi za shranjevanje podatkov v podatkovno bazo.

```
if request.method == 'POST':
  form_discussion = AddNewDiscussion(request.POST)
  if form_discussion.is_valid():
    title = request.POST.get('title')
    desc = request.POST.get('desc')

    discussion = Topic()
```

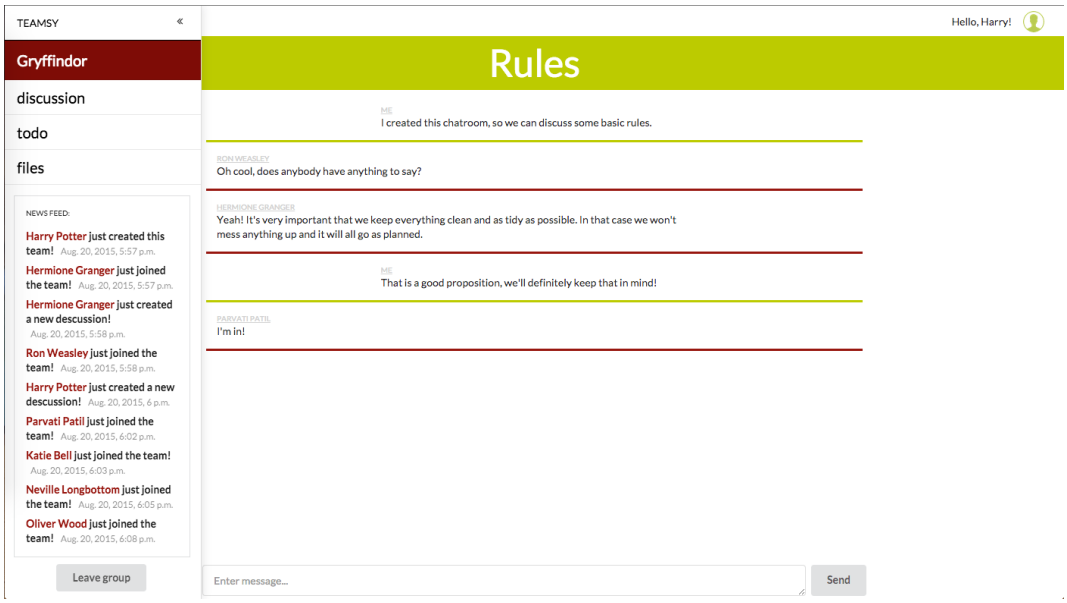
```
        discussion.title = title
        discussion.desc = desc
        discussion.iduser_created = user_logged
        discussion.idforum = Forum.objects.get(idteam=team)
        discussion.save()

        # Na tem mestu se nahaja še koda za shranjevanje vnosa
        # v tabelo feed.
else:
  form_discussion = AddNewDiscussion()
```

Prikazana funkcija najprej preveri, če gre za zahtevo `POST`, nato pa še, če je dotičen obrazec veljaven. Če sta oba pogoja izpolnjena, se ustvari nova instanca objekta Topic, katerega vsebina se na koncu s funkcijo `save()` shrani v podatkovno bazo.



Slika 3.14: Osnovna stran podstrani za medsebojno komunikacijo. Ob postavitvi miške na ikono za informacije se prikaže opis teme.

Klepetalnica, ki je prikazana na sliki 3.15, je oblikovana v stilu aplikacij za tekstovno komunikacijo, ki jih poznamo iz mobilnih telefonov. Shranjevanje sporočil poteka preko Ajax klica, ki se sproži ob pritisku na gumb *Send*, ali na tipko *Enter* na tipkovnici, če vnosno polje ni prazno. Zaradi boljše preglednosti so sporočila prijavljenega uporabnika zamaknjena v desno in podčrtana z zeleno črto. Ostala sporočila so rdeče podčrtana in imajo levo poravnavo.

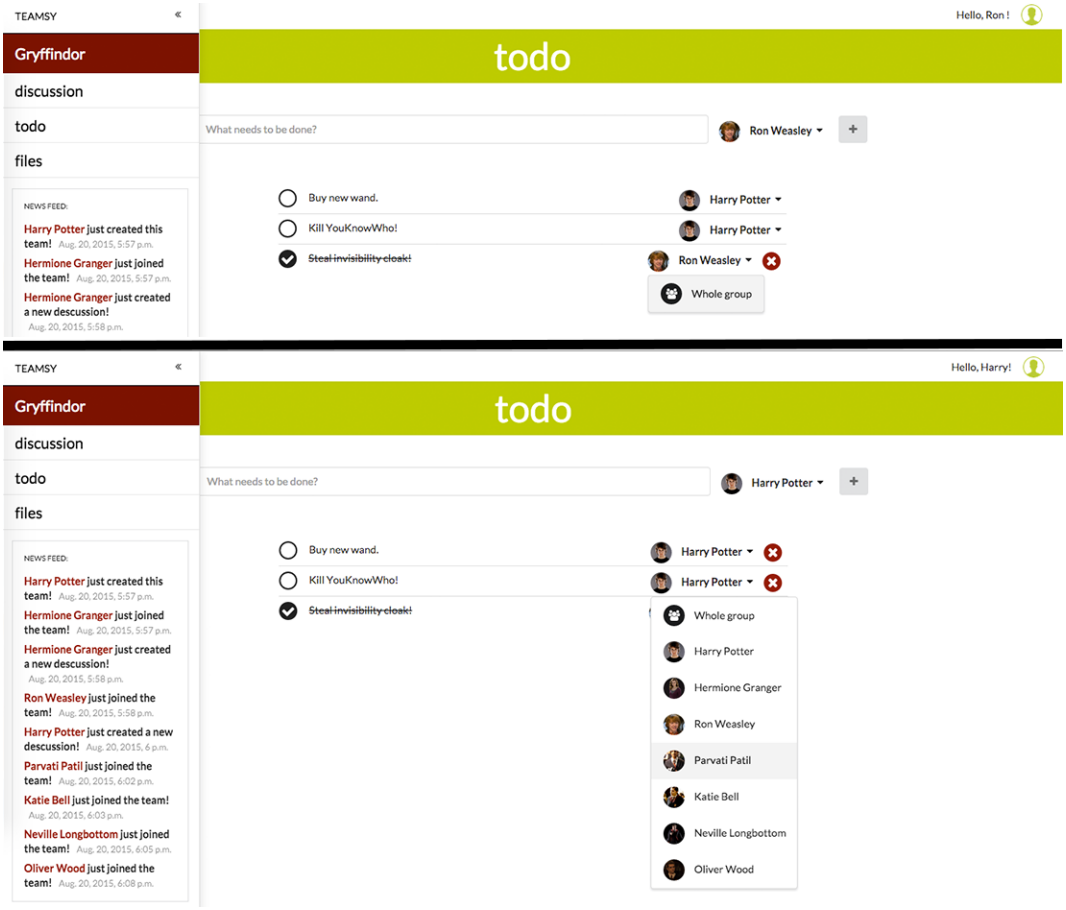


Slika 3.15: Klepetalnica teme *Rules*. Sporočila prijavljenega uporabnika so zaradi boljše preglednosti zamaknjena v desno stran. Nad sporočilom se nahajajo še ime in priimek avtorja ter časovni pečat sporočila.

3.8.2 Stran za dodeljevanje tekočih nalog

Podstran za dodeljevanje tekočih nalog, prikazana na sliki 3.16, je zamišljena kot seznam vseh tekočih opravil ekipe. Funkcionalnosti obsegajo običajno dodajanje in odstranjevanje opravila, prav tako pa lahko opravilo označimo kot opravljeno, kar je vidno tudi na seznamu. Na tej podstrani ponovno nastopijo uporabniške vloge na nivoju skupine. Skrbnik ima namreč pravico

razdeljevanja nalog med člane skupine, obenem pa lahko že ustvarjenim nalogam spreminja osebo, odgovorno za zaključek določenega opravila. Vsako opravilo lahko odstrani in označi kot opravljeno, ne glede na odgovorno osebo (slika 3.16 spodaj). Navaden uporabnik teh pravic nima, možnost brisanja in označevanja določene naloge pa ima le pri opravilih, za katere je sam odgovoren (slika 3.16 zgoraj).



Slika 3.16: Podstran za prikaz tekočih nalog. Zgoraj: prikaz namenjen navadnemu uporabniku, ki ima omejitve pri dodajanju, brisanju in spreminjanju nalog (spreminja lahko le tiste, za katere je odgovoren). Spodaj: prikaz namenjen skrbniku skupine, nima omejitev funkcionalnosti.

Vse spremembe se izvajajo sproti, brez osveževanja strani, vzporedno pa

se izvajajo tudi spremembe v podatkovni bazi. Komunikacija s strežnikom poteka preko Ajax klica. Menjava ikone in prečrtan tekst, ob kliku na prazen krog, sta implementirana s kodo knjižnice jQuery. V programski kodi se ob detekciji miškinega klika spreminjajo CSS atributi ustreznim elementom.

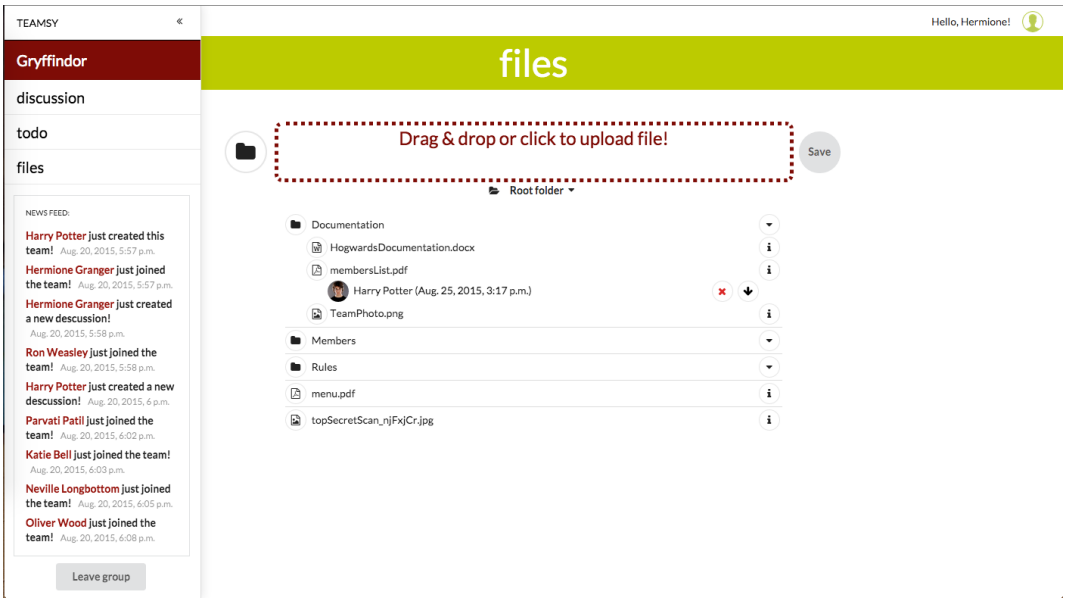
3.8.3 Stran za deljenje datotek

Kljub veliki poplavi oblačnih storitev, ki omogočajo številne napredne možnosti za upravljanje in manipulacijo z datotekami, sem zaradi ideje o popolni povezanosti in uporabi le enega orodja, vključil še svojo, sicer nekoliko okrnjeno različico sistema za deljenje datotek. Sistem omogoča enonivojsko strukturo map, nalaganje pa poteka preko vnosnega polja tipa *file*. Zato, da lahko datoteko uspešno kopiramo na strežnik, moramo napisati vrstico `file = models.FileField(upload_to=file_name)` v razred `File`. Ta razred se nahaja v datoteki `models.py`. Vrstica se iz modela v bazo preslika v polje tipa *VARCHAR*, omogoča pa shranjevanje poti do datoteke v to polje podatkovne baze. Lokacijo datoteke definiramo v atributu `upload_to`. V predhodno navedenem primeru pot do datoteke definira funkcija `file_name`:

```
def file_name(instance, filename):
  return '/'.join(['team', str(instance.idteam), filename])
```

Datoteke se kopirajo na strežnik v mapo "media/team", ki ji sledi podmapa z imenom ustrezne ekipe, kateri pripadajo naložene datoteke. Primer poti: `media/team/Gryffindor/`.

Datotečni sistem je na odjemalcu prikazan kot seznam map in dokumentov. V zgornjem delu slike 3.17 vidimo vnosno polje za izbiro datotek (s funkcijo povleci in spusti), na desni in levi strani pa gumba za potrditev (shranjevanje) in dodajanje nove mape. Pod vnosnim poljem se nahaja še spustni seznam obstoječih map, v katerem izberemo mapo, ki bo služila kot korenska mapa naložene datoteke.



Slika 3.17: Podstran za izbiro datotek za nalaganje in prenos datotek. Na vrhu strani se nahaja vnosno polje za izbiro datotek, gumba za dodajanje mape in prenos izbrane datoteke na strežnik, ter spustni seznam za izbiro korenske mape. V sredini strani opazimo odjemalčev datotečni sistem z razširjeno mapo *Documentation* in prikazanimi podrobnostmi ene izmed datotek.

V seznamu so najprej prikazane obstoječe mape, nato še datoteke v korenski mapi. Vsebino mape razkrijemo s klikom na ikono ob desnem robu seznama, podrobnosti vsake datoteke pa se prikažejo po kliku na ikono *info* (prav tako ob desnem robu seznama). Ikone datotek se prilagajajo glede na vrsto (končnico) posamezne datoteke. V prikazu podrobnosti naložene datoteke se nahajajo avatar, ime in priimek nalagatelja ter časovni pečat prenosa datoteke na strežnik, ob desnem robu pa še ikoni za izbris in prenos datoteke v lokalno okolje.

Poglavje 4

Sklepne ugotovitve

V okviru diplomskega dela sem razvil aplikacijo za organizacijo dela v manjših skupinah imenovano *Teamsy*. Aplikacija sledi najnovejšim smernicam za razvoj spletnih aplikacij, kot tudi principom odzivnega dizajna za nemoten prikaz na različnih dimenzijah zaslonov, kar omogoča dober prikaz tudi na vse bolj popularnih mobilnih platformah. Diplomsko delo sem razdelil na dva dela. Teoretični uvod predstavlja uporabljene tehnologije, ogrodja in orodja ter izpostavlja pomembnejše podrobnosti za boljše razumevanje drugega dela naloge, ki opisuje delovanje in strukturo aplikacije. Prikazanih je tudi nekaj odsekov kode in posnetkov zaslona za boljšo vizualno predstavitev produkta bralcu.

4.1 Zaključki

Ugotovil sem, da je za razvoj spletne aplikacije potrebno predvsem dobro načrtovanje, da kasnejše delo poteka gladko in brez nepotrebnih zadržkov. Jasna strategija in dobro zastavljeni cilji so pot do sproščenega dela in uspešno izvedenega projekta. Ob razvoju aplikacije sem poglobil znanje tehnologij, ki sem jih že poznal in spoznal nekaj novih principov spletnega programiranja. Prebiranje dobro napisane dokumentacije tehnologij pa mi je poleg praktičnih izkušenj zagotovilo še pomembno teoretično znanje. Velik napre-

dek v razvoju spletnih aplikacij predstavlja princip razvoja SPA (*Single Page Application*), ki temelji na asinhronih strežniških klicih. Za prenos podatkov med ospredjem in strežniškim zaledjem sem v večini primerov uporabljal Ajax klice, ki pa ob velikem številu funkcij pisanih v ogrodju jQuery, postanejo nekoliko nepregledni. Ajax klice sem uporabljal zaradi prosojnosti komunikacije s strežnikom. Uporabnik namreč ne opazi, kdaj se izvrši kakšen klic na strežnik ali prenos podatkov, rezultati pa so vidni takoj in brez osveževanja celotne strani. Težave z nepreglednostjo funkcij rešujejo knjižnice, kot so AngularJS, React in podobne, katerih uporaba je zaželjena v prihodnjih projektih. Ena boljših lastnosti razvite aplikacije je enostavnost uporabe, ki je pri organizacijskih orodjih še posebej pomembna. Uporaba ikon omogoča hiter pregled funkcij, profilne slike uporabnikov pa naredijo aplikacijo prijaznejšo in atraktivnejšo. V izboljšani različici svoje spletne aplikacije bi želel dodati še več preglednih statističnih elementov in izboljšan sistem za obvestila. Uporabniki bi tako imeli hitrejši dostop do pomembnih informacij že na osnovni strani posamezne skupine. Na sistemih, ki potencialno lahko pridobijo veliko število uporabnikov, je prav tako pomembna čim boljša izolacija zalednega dela in ospredja aplikacije, saj to omogoča učinkovitejše in hitrejše reševanje nastalih problemov. Spoznal sem, da je delo s spletnimi tehnologijami zanimivo in raznoliko, saj obstaja veliko različnih poti do enakega cilja.

4.2 Nadaljni razvoj

Spletna aplikacija je po mojem mnenju dobro zasnovana, ponuja funkcionalnosti, ki lahko postanejo nepogrešljivi del vsakdana članov manjših skupin ali celo manjših podjetij. Nepogrešljivost tovrstnih aplikacij se je izkazala kot dejstvo z velikim uspehom konkurenčnih produktov (na primer Slack), vendar pa bi bilo potrebno, za tržni uspeh razvite aplikacije, vložiti še nekaj časa in sredstev v nadaljnji razvoj. Ideja v računalniškem svetu ni tuja, vendar menim, da so novi, konkurenčni produkti vedno dobrodošli za kakovostnejšo

in dostopnejšo uporabniško izkušnjo. Izboljšana različica bi tako lahko ponujala naprednejše delo z datotekami ali celo integracijo že obstoječih oblačnih storitev Google Drive [53], Dropbox [54], Microsoft OneDrive [55], ipd. Poleg klepetalnic bi lahko implementiral še individualne pogovore članov skupine ali celo medsebojno komunikacijo vseh uporabnikov aplikacije, neodvisno od pripadnosti neki skupini. Za prijavo v aplikacijo pa bi lahko uporabnik, poleg specifičnega računa, ki ga ponuja *Teamsy*, uporabil še na primer prijavo s Facebook računom. Nadaljnje delo na aplikaciji bi potekalo z mislijo na izboljšanje uporabniške izkušnje in še boljšo vizualno predstavitev informacij. V okviru tega bi aplikacijo tudi prevedel v več jezikov, saj trenutna verzija omogoča vmesnik le v angleškem jeziku. Vzporedno z razvojem je potekalo tudi testiranje aplikacije, in sicer na večjih zaslonih prenosnikov in namiznih računalnikov, ter manjšem zaslonu pametnega telefona. Cilj diplomske naloge je bil, da spletno aplikacijo razvijem do poskusne (*beta*) faze, kar mi je tudi uspelo. Temeljito testiranje, predvsem strežniškega zaledja, pa je eden pomembnejših procesov razvoja predvsem tistih spletnih aplikacij, ki ciljajo na veliko število uporabnikov.

Literatura

[1] Skype. [Online]. Dosegljivo:
https://www.skype.com/

[2] Facebook. [Online]. Dosegljivo:
https://www.facebook.com/

[3] Slack. [Online]. Dosegljivo:
https://slack.com/

[4] J. Duckett "HTML & CSS design and build websites", 2011.

[5] MySQL. [Online]. Dosegljivo:
https://www.mysql.com/.

[6] HTML. [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/HTML.

[7] HTML Canvas. [Online]. Dosegljivo:
http://www.w3schools.com/html/html5_canvas.asp.

[8] Adobe Flash. [Online]. Dosegljivo:
http://www.adobe.com/si/products/flashplayer.html.

[9] Cascading Style Sheets. [Online]. Dosegljivo:
https://css-tricks.com/myth-busting-css-animations-vs-JavaScript/.

[10] Cascading Style Sheets. [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/Cascading_Style_Sheets.

[11] Syntactically Awesome Style Sheets. [Online]. Dosegljivo: http://sass-lang.com/.

[12] LESS. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Less_(stylesheet_language).

[13] Extensible Markup Language. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/XML.

[14] Scalable Vector Graphics. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Scalable_Vector_Graphics.

[15] XML User Interface Language. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/XUL.

[16] R. Nixon. "Learning PHP, MySQL, JavaScript, CSS & HTML5", *Third Edition*, str. 2–35, 2014.

[17] Document Object Model. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Document_Object_Model.

[18] XHTML. [Online]. Dosegljivo: http://xhtml.com/en/xhtml/reference/.

[19] AngularJS. [Online]. Dosegljivo: https://angularjs.org/.

[20] Reach. [Online]. Dosegljivo: http://facebook.github.io/react/.

[21] Single Page Application. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Single-page_application.

[22] About Python. [Online]. Dosegljivo: https://www.python.org/about/.

[23] Python. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Python_(programming_language)

[24] A. Martelli. "Python In A Nutshell, A Desktop Quick Reference", *2nd Edition*, str. 3–12, 2006.

[25] T. A. Powel. "Ajax, The Complete Reference", str. 3–10, 2008.

[26] MySQL. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/MySQL.

[27] MySQL. [Online]. Dosegljivo: https://www.mysql.com/why-mysql/

[28] Git. [Online]. Dosegljivo: https://git-scm.com/

[29] J. Loeliger, M. McCullough "Version Control with Git", *2nd Edition*, str. 2–6, 2012.

[30] Django. [Online]. Dosegljivo: https://www.djangoproject.com/

[31] Django. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Django_(web_framework)

[32] S. Jaiswal, R. Kumar. "Learning Django Web Development", str. 3–100, 2015.

[33] Semantic UI. [Online]. Dosegljivo: http://semantic-ui.com/

[34] jQuery. [Online]. Dosegljivo: https://jquery.com/

[35] D. S. McFarland. "JavaScript & jQuery the missing manual", *Third Edition*, str. 15–17, 2014.

[36] jQuery. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/JQuery

[37] PyCharm. [Online]. Dosegljivo:
https://www.jetbrains.com/pycharm/

[38] JetBrains. [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/JetBrains

[39] MySQL Workbench. [Online]. Dosegljivo:
https://www.mysql.com/products/workbench/

[40] Sequel Pro. [Online]. Dosegljivo:
http://www.sequelpro.com/

[41] Google Chrome. [Online]. Dosegljivo:
https://www.google.com/chrome/browser/desktop/index.html/

[42] Mozilla Firefox. [Online]. Dosegljivo:
https://www.mozilla.org/sl/firefox/new/

[43] Adobe Kuler. [Online]. Dosegljivo:
https://color.adobe.com/create/color-wheel/

[44] Adobe Photoshop. [Online]. Dosegljivo:
http://www.photoshop.com/

[45] Adobe Illustrator. [Online]. Dosegljivo:
http://www.adobe.com/si/products/illustrator.html

[46] Brackets. [Online]. Dosegljivo:
http://brackets.io/

[47] Bitbucket. [Online]. Dosegljivo:
http://brackets.io/

[48] SourceTree. [Online]. Dosegljivo:
https://www.sourcetreeapp.com/

[49] Django Models. [Online]. Dosegljivo:
https://docs.djangoproject.com/en/1.8/topics/db/models/

[50] URL dispatcher. [Online]. Dosegljivo:
https://docs.djangoproject.com/en/1.8/topics/http/urls/

[51] Working with forms. [Online]. Dosegljivo:
https://docs.djangoproject.com/en/1.8/topics/forms/

[52] Hypertext Transfer Protocol. [Online]. Dosegljivo:
https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol/

[53] Google Drive. [Online]. Dosegljivo:
http://www.google.com/drive/

[54] Dropbox. [Online]. Dosegljivo:
https://www.dropbox.com/

[55] Microsoft OneDrive [Online]. Dosegljivo:
https://onedrive.live.com/about/sl-si/