Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Bogdan Golobič

Razvoj spletnega portala za študente

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Viljan Mahnič

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Razvoj spletnega portala za študente

Tematika naloge:

Izhajajoč iz predpostavke, da družbena omrežja, kot je npr. Facebook, ne pokrivajo vseh funkcionalnosti, ki jih študentje potrebujejo za izmenjavo informacij o študiju, realizirajte spletno aplikacijo, ki bo izboljšala njihovo medsebojno komunikacijo. V ta namen najprej izpeljite anketo med študenti, ki bo služila za potrditev zgoraj navedene predpostavke in identifikacijo tistih funkcionalnosti, ki so po mnenju študentov najpomembnejše. Na podlagi teh rezultatov izdelajte ustrezno rešitev. Predstavite orodja, potrebna za realizacijo, in načrt aplikacije ter opišite potek njene izdelave in funkcionalnost, ki jo nudi.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Bogdan Golobič sem avtor diplomskega dela z naslovom:

Razvoj spletnega portala za študente (angl. *Development of a students' portal*)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Viljana Mahniča,
- 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 18. avgust 2016 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ACID	atomicity, consistency, isolation, durability	atomarnost, konsistenčnost, izolacija, vzdržljivost
API	application program interface	programski vmesnik
CSRF	cross site request forgery	ponarejanje spletnih zahtev
HATEOAS	hypermedia as the engine of application state	hipermedij kot motor za stanje aplikacij
HTML	hypertext markup language	jezik za označevanje hiperteksta
HTTP	hypertext transfer protocol	protokol za prenos hiperteksta
JSON	JavaScript object notation	notacija Javascript objektov
MIT	Massachusetts Institute of Technology	
MVC	model-view-controller	model-pogled-krmilnik
MVVM	model-view-viewmodel	model-pogled-pogledmodel
ORM	object-relational mapping	objektno-relacijsko mapiranje
REST	representational state transfer	reprezentacijski prenos stanja
URL	uniform resource locator	enotni naslov vira
XML	extensible markup language	razširljivi označevalni jezik

Povzetek

Naslov: Razvoj spletnega portala za študente

V diplomskem delu je predstavljen razvoj spletne aplikacije za medsebojno komuniciranje študentov glede študija. Podrobnejše je opisana zasnovana anketa in njeni rezultati, razvojne tehnologije, zahtevane funkcionalnosti, razvojni proces in razvite funkcionalnosti aplikacije. Osredotočili smo se tudi na odzivnost spletne strani, da se lahko aplikacijo uporablja tudi na napravah z manjšimi ekrani, kot so pametne mobilne naprave. Za razvoj strežniške tehnologije je bil izbran Django, za logiko na odjemalčevi strani pa AngularJS, obogaten z Bootstrap. Celotna komunikacija med strežnikom in odjemalcem poteka preko programskega vmesnika. Za hranjenje podatkov smo uporabili podatkovno bazo PostgreSQL. Funkcionalnost aplikacije obsega registriranje in prijavljanje uporabnikov v aplikacijo, sledenje predmetom, pisanje objav, komentarjev in samolepilnih lističev. Prav tako lahko uporabnik filtrira objave po predmetih, kontekstu, vsebini, avtorju in jih doda med zazanamke. Omogočena je tudi administrativna kontrola nad vsebino podatkovne baze preko spletnega brskalnika.

Ključne besede: spletni portal, študent, filtriranje objav, sledenje predmetom, administrator.

Abstract

Title: Development of a students' portal

The thesis presents development of a web portal for students to communicate about courses and studies. Hereinafter is described a made survey and its results, used development tools, required functionality derived from survey, development process and developed application. Big focus was put on a responsive web design, so that web portal can be used on devices with smaller screen, such as smart mobile phones. Django was chosen for the server side development and AngularJS with Bootstrap for the client side. Communication between server and client is done wholly through API. Data is stored in a PostgreSQL database. On a developed web portal a user can register, log in, follow courses and write posts, comments and sticky notes. Furthermore user can filter posts by course, context, content, author and bookmark them. Enabled is also administrative control over content of database through web browser.

Keywords: web portal, student, posts filtering, following courses, administrator.

Poglavje 1

Uvod

Pred dobrimi petimi leti so študentje za medsebojno komuniciranje glede študija in študijskih obveznosti na spletu uporabljali forume. Z večanjem popularnosti Facebooka se je to postopoma preneslo nanj, vendar pa Facebook primarno ni bil namenjen temu, kar se opazi tudi pri njegovi uporabi. Daleč najbolj opazna pomanjkljivost je iskanje med objavami. Prav tako manjka konsistenčno grupiranje po študijih, njihovih letnikih in predmetih. Z uporabo Facebook skupin je to funkcionalno področje rahlo pokrito, vendar je pregled večjega števila skupin zelo nadležen in neučinkovit. V kolikor je večje število predmetov pokritih v isti skupini, je iskanje starejših objav še bolj oteženo. Prednost Facebooka je prijazen uporabniški vmesnik za objavljanje in komentiranje. Predlagana rešitev zajema prijazen uporabniški vmesnik z vsemi zgoraj navedenimi funkcionalnostmi, ki so pogrešane. Tako je namen združiti čim več dobrih funkcionalnosti forumov in Facebooka ter se izogniti vsem funkcionalnostim, ki ovirajo učinkovito uporabo. V osnovi je spletni portal namenjen študentom za komuniciranje s kolegi o opravljanju in poteku študija. V nadaljevanju naloge je opisana analiza ankete, katere cilj je ugotoviti želje potencialnih uporabnikov, ki jih pretvorimo v zahtevane funkcionalnosti. Spletni portal je bil implementiran v AngularJS za uporabniški vmesnik, Django za strežnik in PostgreSQL za podatkovno bazo, vse skupaj povezuje RESTful. Vsebina aplikacije je prilagodljiva velikosti ekrana, tako

uporabniku ni otežena uporaba na bolj mobilnih napravah. V zaključku so predstavljene tudi potencialne izboljšave in ugotovitve po končani implementaciji.

Poglavje 2

Pregled področja

Študentje se hitro prilagajajo trendom tehnoloških sprememb. Enako velja tudi pri medsebojni komunikaciji glede študija, vendar je na tržišču opaziti primanjkljaj spletnih storitev, ki bi to omogočala. Študentje tako uporabljajo večinoma samo še Facebook, kjer se ustvarjajo skupine za posamezne smeri študija, letnike, module in predmete. Facebook nima razvitih funkcionalnosti, ki bi omogočala iskanje po vsebini ali avtorju objave. Prav tako ne omogoča grupiranja znotraj skupin. Posledično je uporabnikom prepuščeno, da poskušajo znotraj vsebine objav čimbolj jasno navesti, v kakšen kontekst spada njihova objava. Tega se mnogi ne držijo. Pred uporabo Facebooka so študentje uporabljali forume (na FRI-ju je bil popularen http://fri-info.net/forum/), ampak izgleda, da sta prijaznejši uporabniški vmesnik in hitrost objavljanja/komentiranja prepričala študente za uporabo Facebooka.

2.1 Anketa

Za lažje razumevanje dejanskih želja in zahtev uporabnikov je bila pripravljena elektronska anonimna anketa na spletnem portalu Google Forms [3], ki je bila posredovana večinoma študentom, delno pa tudi dijakom, v razmerju 9 : 1. Vsi anketirani študentje so obiskovalci prve stopnje študija na Univerzi v Ljubljani. Anketa se je pošiljala osebno, torej URL povezava na

anketo ni bila nikjer javno objavljena. Tako je bila populacija kontrolirana. Vsi, ki jim je bila anketa poslana, so anketo izpolnili.
V nadaljevanju je naprej predstavljena vsebina ankete. Taki anketi, kot so jo dobili anketiranci, sledita še razlaga vprašanj in interpretacija rezultatov.

2.1.1 Vsebina

Študijsko socialno omrežje

Spoštovani,
v okviru diplomskega dela sem pripravil kratko anketo o uporabi in uporabniški izkušnji interneta in spletnih storitev za medsebojno komuniciranje študentov glede študija. Odgovori so anonimni in bodo uporabljeni le v namen analize znotraj diplomske naloge.

Pred dobrimi petimi leti so študentje za medsebojno komuniciranje glede študija in študijskih obveznosti na spletu uporabljali forume. Z večanjem popularnosti Facebooka se je to postopoma preneslo nanj, vendar pa Facebook primarno ni bil namenjen temu, kar se opazi tudi pri njegovi uporabi.
Primer: zanima te, kaj so se prejšnje leto pogovarjali glede izpita pri določenem predmetu, vendar ta predmet nima svoje skupine, tako da moraš iskati po skupini za celo smer študija, za celo leto nazaj, kjer je bilo vmes objavljenih na stotine novih, zate nerelavantnih objav.
S tem razlogom sem se odločil, da bom za diplomsko nalogo naredil enostavno socialno omrežje, namenjeno študentom za medsebojno pomoč pri študiju (in mogoče tudi šolarjem pri šolanju).

1. **Kako pogosto uporabljate forum ali Facebook skupine za komuniciranje s kolegi (sošolci) glede študija?**
   (a) Večkrat tedensko
   (b) Večkrat mesečno

(c) Nekajkrat letno

(d) Nikoli

2. **Kako pomembna je za vas vsaka izmed naslednjih funkcionalnosti Facebooka:**
(1 - zelo nepomembna, 2 - nepomembna, 3 - nevtralno, 4 - pomembna, 5 - zelo pomembna)

(a) Ocenjevanje objav in komentarjev
1 2 3 4 5

(b) Neskončno drsenje (angl. infinite scrolling)
1 2 3 4 5

(c) Urejanje profila (slike, informacije o sebi)
1 2 3 4 5

(d) Izmenjevanje zapiskov in ostale literature
1 2 3 4 5

3. **Kako pomembna vam je določena funkcionalnost, ki je Facebook nima ali pa je slabo implementirana:**
(1 - zelo nepomembna, 2 - nepomembna, 3 - nevtralno, 4 - pomembna, 5 - zelo pomembna)

(a) Iskanje po objavah
1 2 3 4 5

(b) Grupiranje predmetov po letniku in študiju
1 2 3 4 5

(c) Grupiranje objav po predavanjih, vajah, kolokvijih in izpitih
1 2 3 4 5

(d) Odsotnost smetja (angl. spam)
1 2 3 4 5

(e) Izmenjevanje zapiskov in ostale literature
1 2 3 4 5

4. **Si želite še kakšno dodatno funkcionalnost, ki ni bila omenjena?**

---

5. **Ali bi uporabljali internetno stran, ki bi vključevala zgoraj zaželene funkcionalnosti?**

   (a) Da

   (b) Ne

6. **Pripombe, misli in ideje:**

2.1.2 Opis ankete

Osnovni namen ankete je bil, da bi se ugotovilo, kako pogosto študentje in dijaki uporabljajo internet za komuniciranje glede študija med sabo in katere funkcionalnosti so najbolj zaželene. Prav tako so bili vprašani, ali bi uporabljali portal z željenimi funkcionalnostmi. Za vsako vprašanje bo v nadaljevanju opisan razlog za njegovo izbiro in pričakovani rezultati (oštevilčenje razlage vprašanj ustreza oštevilčenju vprašanj v anketi).

1. S prvim vprašanjem smo želeli izvedeti, kako pogosto ali pa če sploh anketiranec uporablja forum ali Facebook skupine za komuniciranje s kolegi glede študija. Namerno smo se odločili, da ne bomo vključili vprašanja o dnevni uporabi, kajti obiskovanje takih strani po navadi ni redno, temveč je sorazmerno s tekočimi študijskimi obveznostmi. Namen tega vprašanja je bil, da izvemo ali sploh obstaja trg, kjer bi bilo povpraševanje po naši zamišljeni spletni aplikaciji.

2. Naslednja tri vprašanja sprašujejo po funkcionalnosti, ki jih vsebuje Facebook in prevladujejo pri uporabniški izkušnji, poleg možnosti objavljanja in komentiranja.

- Prvo izmed teh je spraševalo, ali se jim zdi pomembno, da lahko ocenjujejo objave in komentarje. Facebook omogoča ocenjevanje objav preko všečkov (angl. like). Zanimalo nas je, kako pomembno je to uporabnikom, saj te funkcionalnosti forumi nimajo, kar bi lahko bil eden od razlogov, zakaj so se uporabniki preselili iz forumov na Facebook.
- Drugo vprašanje je prisotnost neskončnega drsenja. To je funkcionalnost, ko uporabnik pride do konca strani in se mu objave dinamično dodajo na konec strani. Tako uporabniku ni potrebno naložiti nove strani. Pri tej funkcionalnosti smo bili rahlo razdvojeni, po eni strani je na prvi pogled privlačna in doda k dinamičnosti strani, po drugi strani pa uporabniku oteži iskanje starejših objav, saj mora vedno na novo drseti po strani navzdol in čakati, da naloži nove objave ter to ponavljati, dokler ne najde željene objave.
- Tretja prevladujoča funkcionalnost Facebooka je urejevanje profila. Čeprav to ničesar ne doprinese k samemu namenu naše spletne aplikacije, bi jo mogoče morali dodati, če bi pomagala privabiti večje število uporabnikov.

3. Naslednjih pet vprašanj je spraševalo anketirance po funkcionalnostih, ki jih ima Facebook slabše implementirane ali pa jih sploh nima. Izbrali smo tiste funkcionalnosti, za katere se nam je zdelo najpomembnejše, da izvemo, kaj glede njih menijo potencialni uporabniki.

   - Kako pomembna je uporabnikom možnost iskanja po objavah? To je eden izmed glavnih razlogov, zakaj se nam je zdel Facebook neprimeren za tako vrstno uporabo. Poleg tega z neskončnim drsenjem še dodatno oteži uporabnikom iskanje starejših objav.
   - Možnost grupiranja predmetov po letniku in študiju. Te funkcionalnosti Facebook prav tako ne pokriva. Sicer ima skupine, kjer se grupirajo študentje po predmetu ali študiju, vendar so zelo ne-

pregledne. Prav tako znotraj teh skupin ni omogočeno ustvarjanje podskupin.

- Možnost grupiranja objav po predavanjih, vajah, kolokvijih, izpitih in zapiskih. To bi uporabnikom omogočilo, da lahko hitro ugotovijo, v kakšen kontekst spada posamezna objava. Tega Facebook sploh ne pokriva. Tako so uporabniki začeli znotraj objav pisati kontekst, če so hoteli bolj pregledno skupino.
- Anketiranci so bili prav tako vprašani, koliko se jim zdi pomembno pomanjkanje smetja (angl. spam). Pri Facebooku so skupine večinoma zaprte, kar pomeni, da te mora skrbnik skupine sprejeti, preden lahko bereš in objavljaš. Če bi sprejeli le tiste, ki dejansko obiskujejo predmete znotraj skupine, bi to zahtevalo ogromno dela za skrbnike. Posledično se skupinam pridružujejo ljudje, ki le občasno objavijo kakšno reklamo.
- Možnost izmenjavanja zapiskov. Facebook sicer omogoča shranjevanje datotek, vendar nima učinkovitega razvrščanja in seznam datotek tako hitro postane nepregleden. Študentje se zato poslužujejo drugih spletnih aplikacij, kot je na primer Dropbox.

4. Anketiranec je imel možnost dodati svojo idejo ali predlog o funkcionalnosti, ki prej ni bila omenjena, a se mu zdi pomembna. Upali smo, da bomo tako dobili kakšno dobro idejo, ki bi lahko izboljšala spletno aplikacijo.

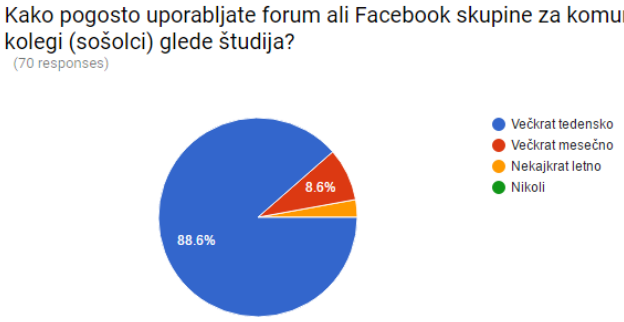
5. Zanimalo nas je, ali bi bili študentje pripravljeni uporabljati spletni portal, ki bi vključeval zgoraj zaželene funkcionalnosti.

6. Zadnje vprašanje je bilo popolnoma odprto, anketiranec je lahko napisal svoje mnenje, ideje in pripombe. Zopet je bilo postavljeno v upanju, da bomo dobili kakšno dobro pripombo, ki nam bo pomagala razviti boljšo aplikacijo.

2.2 Interpretacija rezultatov

Anketo je izpolnilo sedemdeset študentov in dijakov (profil populacije je opisan v poglavju 2.1). Dobili smo pregled nad tem, koliko od njih redno uporablja internet za komunikacijo s kolegi glede študija in katere funkcionalnosti so jim najpomembnejše (boljše ocenjene funkcionalnosti smo vzeli kot pomembnejše - bolj zaželene).
Za vsak odgovor bo predstavljen grafikon z odgovori (oštevilčenje razlage vprašanj ustreza oštevilčenju vprašanj v anketi).

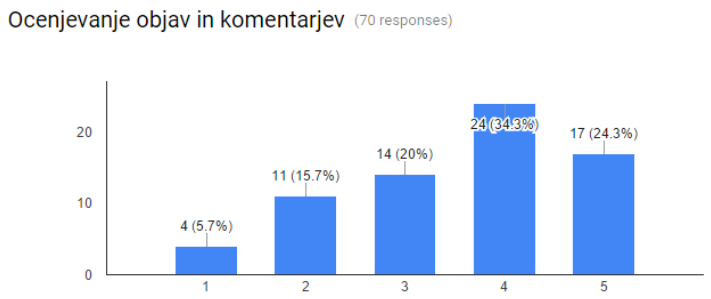
1. Pri vprašanju kako pogosto študentje uporabljajo forum ali Facebook skupine za komuniciranje s kolegi, so rezultati pokazali, da ju kar 88.6% študentov uporablja večkrat tedensko, 8.6% večkrat mesečno, 2.9% pa nekajkrat letno. (slika 2.1)



Slika 2.1: : Graf rednosti uporabe interneta za komuniciranje s kolegi glede študija v %

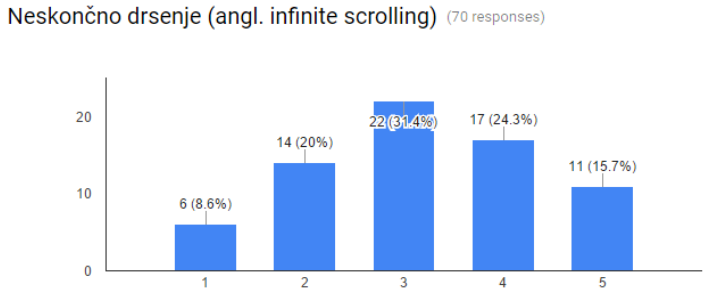
2. Facebook funkcionalnosti:

   - Iz spodaj razvidnega grafikona je opaziti, da je funkcionalnost ocenjevanja objav in komentarjev zaželena, vendar ni vitalnega pomena. Največji delež anketirancev jo je ocenilo z oceno 4 (33.3%), nato s 5 (24.3%) in za tem s 3 (20%). (slika 2.2)



Slika 2.2: Graf pomembnosti ocenjevanja objav in komentarjev v %

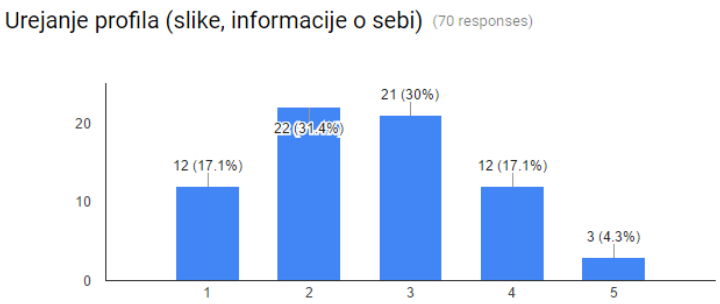
- Anketirancem se neskončno drsenje ne zdi ena od pomembnejših funkcionalnosti. Večina je odgovorila s 3 (31.4%), na drugem mestu s 4 (24.3%) in za tem z 2 (20%). (slika 2.3)



Slika 2.3: Graf pomembnosti neskončnega drsenja v %

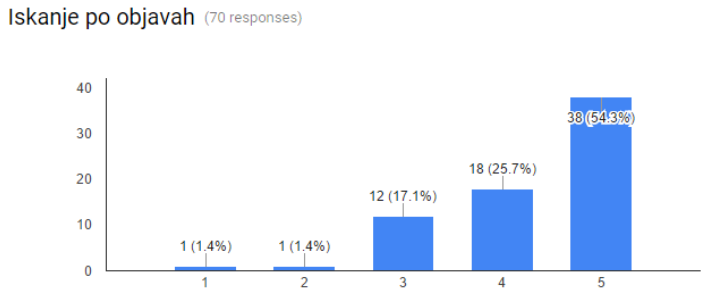
- Izmed vseh funkcionalnosti je bila najmanj zaželena možnost urejevanje profila - slike in informacije o sebi. Povsem pričakovano, saj to ničesar ne doprinese k učinkovitem izmenjevanju znanja in lahko celo odvrne pozornost od namena portala. Največ jih je to funkcionalnost ocenilo z 2 (31.4%), za tem s 3 (30%) in na trejtem mestu je enak procent odgovoril z 1 in s 4 (17.1%). (slika 2.4)

3. Funkcionalnosti, ki jih ima Facebook slabše implementirane ali pa jih sploh nima:



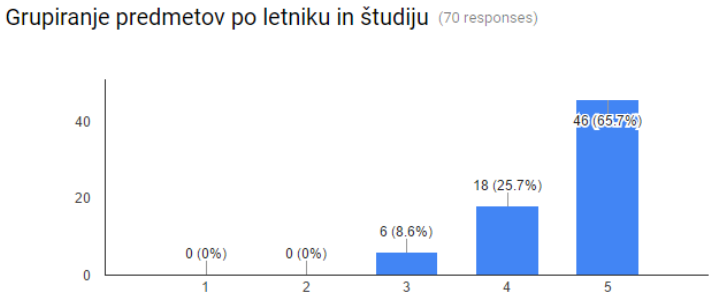
Slika 2.4: Graf pomembnosti urejevanja profila v %

- Iz ankete je bilo ugotovljeno, da je iskanje po objavah nadpovprečno zahtevana funkcionalnost, ki jo je 54.3% odgovorilo s 5, 25.7% z 4 in 17.1% s 3. (slika 2.5)

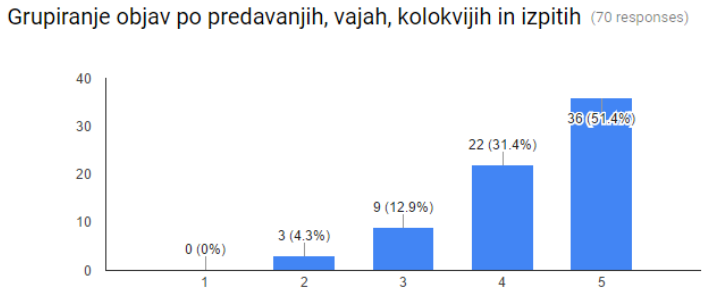


Slika 2.5: Graf pomembnosti iskanja po objavah v %

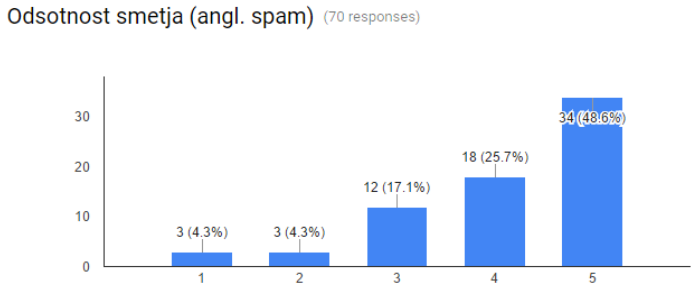
- Grupiranje predmetov po letniku in študiju je bila najboljše ocenjena funkcionalnost, saj jo je 65.7% odgovorilo s 5, 25.7% s 4 in vsi ostali s 3 (8.6%). (slika 2.6)
- Grupiranje objav po predavanjih, vajah, kolokvijih in izpitih je prav tako med bolj zaželenimi funkcionalnostmi. 51.4% je odgovoril s 5, 31.4% s 4 in 12.9% s 3. (slika 2.7)
- Anketiranci so bili prav tako vprašani, kako pomembna se jim zdi odsotnost smetja. Večina je odgovorila s 5 (48.6%), na drugem mestu s 4 (26.7%) in za tem s 3 (17.1%). (slika 2.8)



Slika 2.6: Graf pomembnosti grupiranja po študiju, letniku in predmetu v %



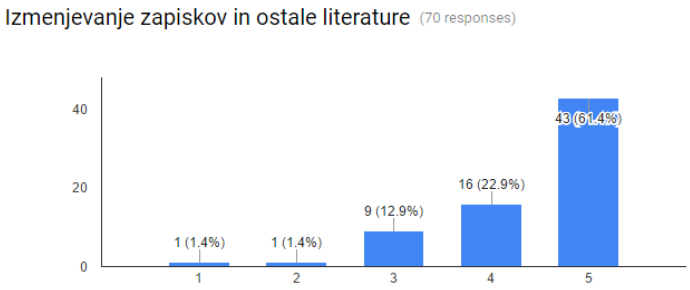
Slika 2.7: Graf pomembnosti grupiranja po predavanjih, vajah, kolokvijih in izpitih v %



Slika 2.8: Graf pomembnosti odsotni smetja v %

- Druga najbolj zaželena funkcionalnost je bila izmenjavanje zapiskov, kar 61.4% je odgovorila s 5, 22.9% s 4 in 12.9% s 3. (slika

2.9)



Slika 2.9: Graf pomembnosti izmenjevanja datotek v %

4. Večina anketirancev je na vprašanje, ali si želijo še kakšno dodatno funkcionalnost, odgovorila z ne ali pa pustila prazno. Dva predloga sta se nam zdela dobra. Prvi je zaznamki objav, ki se uporabniku zdijo pomembni, drugi pa so samolepilni lističi (angl. sticky post) za datume in snov, ki bo na kolokvijih in izpitih.

5. Pri petem vprašanju nas je zanimalo, ali bi bili študenti pripravljeni uporabljati spletni portal, ki bi vključeval zgoraj zaželene funkcionalnosti. Kar 92.9% jih je odgovorilo z *Da*, kar kaže na visoko potrebo po takem spletnem portalu. (slika 2.10)

Slika 2.10: Graf potencialnih uporabnikov v %

6. Zadnje vprašanje je bilo odprto, anketiranec je lahko napisal svoje mnenje, ideje in pripombe. Večina jih je zopet pustila prazno. Eden je izrazil željo po tem, da ne bi bilo reklam. Drugi pa: *"Uporaba temelji na tem da bi jo uporabljali tudi vsi ostali. Omrežje je lahko najboljše vendar je brez uporabnikov mrtvo."* Komentar poudari dejstvo, da so taki spletni portali odvisni od števila ljudi, ki jih uporabljajo. Ni važno, kako dobro je izdelano, če ni uporabnikov, ki bi ga uporabljali.

2.3 Ugotovitve

Iz ankete je bilo ugotovljenih kar nekaj stvari. Prvo, da študentje redno uporabljajo forume ali Facebook skupine za komunikacijo s kolegi glede študija. Razvidno je bilo tudi, da forumi in Facebook skupine ne zadovoljujejo vseh njihovih potreb. Iz dobljenih rezultatov smo se odločili, da se bomo osredotočili na naslednje funkcionalnosti:

- iskanje po objavah,
- grupiranje objav po predavanjih, vajah, kolokvijih, izpitih in zapisikih,
- grupiranje objav po študiju, letniku in predmetih,
- možnost izmenjevanja zapiskov in
- preprečevanje nezaželenih objav.

Prejeta sta bila tudi dva zelo dobra predloga za funkcionalnosti; samolepilni lističi za datume in snov, ki bo na kolokvijih in izpitih, ter možnost zaznamkov objav, zapiskov.

Poglavje 3

Pregled orodij in tehnologij

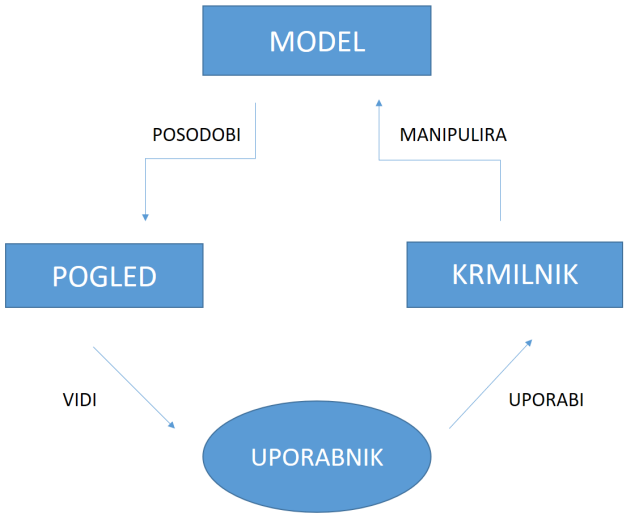
Pri izbiri tehnologije za realizacijo tehnologije je bilo predvsem pomembno vprašanje, ali omogoča, da bo spletna aplikacija odzivna. Hkrati je bilo važno tudi, da je novejša, ali bolje rečeno, aktualna. Kar je novo, ni nujno boljše od starejšega. Tretji kriterij za izbiranje je bilo spoznavanje s temi tehnologijami in učenje razvoja spletnih aplikacij, ker smo imelo zelo malo predznanja pri razvoju spletnih aplikacij. Prav tako je bilo pomembno tudi, da je možno prenesti že obstoječe znanje v izbrane tehnologije.

3.1 Django

Django je odprtokodno ogrodje za razvoj spletnih aplikacij, napisan v jeziku Python in vzdrževan s strani neprofitne organizacije Django Software Foundation. Glavno načelo je enostavnost ustvarjanja kompleksnih spletnih strani, ki temeljijo na podatkih. Poudarek je tudi na ponovni uporabi že ustvarjenih komponent, ki se jih da preprosto prenesti med projekti, hitremu razvoju in načelu *"Don't repeat yourself* [2] (ne programiraj, kar si že prej sprogramiral). Python se uporablja skozi ves projekt, tudi pri definiranju podatkovnega modela in nastavitvi aplikacije.
Sledi načelom arhitekture MVC *(Model-View-Controller)* [4]. To je način razvoja programske opreme, ki razdeli aplikacijo na tri dele: model, pogled

in krmilnik (slika 3.1). Model direktno upravlja s podatki, logiko in pravili aplikacije [5]. Pogled so izhodni podatki v kakršni koli obliki, krmilnik pa sprejme podatke in jih spremeni v ukaze za pogled in model.

Pri ogrodju Django je Model sestavljen iz ORM (*Object-Relational mapper*) [7], ki povezuje relacijsko bazo in Python-ske razrede. Pogled procesira HTTP zahteve. Krmilnik je dispečer, ki temelji na regularnih izrazih.ŒŒ Zaradi zgoraj naštetih načel je zelo enostavno svoji aplikaciji dodati funkcionalnosti preko tako imenovanih paketov (angl. Package).



Slika 3.1: Klasično sodelovanje komponent MVC arhitekture

3.2 Arhitekturni stil REST

REST (angl. Representational state transfer) je način arhitekture za razvoj povezanih aplikacij, ki so sestavljeni iz koordiniranih skupkov komponent, povezav in podatkov distribuiranga hipermedijskega sistema [10]. Poudarek je na performanci in ne na implementacijskih detajlih. Prednost REST arhitekture je tudi skalabilnost, preprostost, preglednost, prenosljivost in spre-

menljivost.
Sistem, ki sledi načelom in omejitvam REST, se imenuje *RESTful*. V večini primerov RESTful sistemi komunicirajo preko HTTP protokola s primernimi HTTP ukazi *GET, POST, PUT, DELETE, itd.* Vsak vir je predstavljen kot spletna storitev z enoznačnim naslovom URL.
Sistem se mora držati šestih načel, da velja za RESTful. V kolikor prekrši katerega koli izmed teh, ne velja več za RESTful sistem [9]. Te načela so:

- **Client-server:** Strežnik in odjemalec sta ločena. To pomeni, da odjemalcu ni pomembno, kako so podatki shranjeni in kako so pridobljeni/shranjeni. Prav tako strežnika ne zanima, kako so odjemalcu podatki prikazani ali v kakšnem stanju je seja. S tem načelom je prišlo do ločitve uporabniškega vmesnika od strežnika.

- **Stateless:** Komunikacija med strežnikom in odjemalcem je še dodatno omejena tako, da strežnik ne sme shranjevati konteksta med različnimi poizvedbami odjemalca. Torej vsako poizvedbo strežnik smatra kot neodvisno transakcijo, ki ni povezana z nobeno izmed prejšnjih.

- **Cacheable:** Odgovori strežnika se lahko shranijo v medpomnilniku odjemalca, da ni nepotrebnih redudantnih poizvedb za podatke, ki jih je odjemalec že "izvedel". To lahko precej pospeši odzivnost strani.

- **Layered system:** Strežnik običajno ne more vedeti, ali komunicira s končnim ali z vmesnim strežnikom. To omogoča lažjo skalabilnost aplikacije, da lahko promet preusmeri v skladu z obremenjenostjo.

- **Uniform interface:** To je glavna lastnost REST storitve. Uniform interface (enotni vmesnik) loči arhitekturo na dele, kar omogoči, da se vsak del razvija neodvisno od drugega. Pri tem so pomembna štiri načela:

  - Identifikacija virov
  - Manipulacija virov preko njihovih reprezentacij

  - Samo-opisna sporočila
  - Hipermedij kot motor za stanje aplikacij (*HATEOAS*)

- **Code on demand**(neobvezno): Strežnik lahko pošlje izvršilno kodo odjemalcu, kar lahko razširi funkcionalnost aplikacije, kot npr. JavaScript skripte.

3.3 AngularJS

AngularJS je odprto-kodno (pod MIT licenco) ogrodje za razvoj spletnih aplikacij na odjemalčevi strani [1]. Vzdrževano je s pomočjo Googla in posameznih odprtokodnih razvijalcev. Poudarek je predvsem na izboljšanju razvoja in testiranja enostranskih spletnih aplikacij z arhitekturo MVVM (Model-View-Viewmodel) [14]. Arhitektura MVVM je zelo podobna arhitekturi MVC. Glavna razlika je, da pri MVC pogled (View) in krmilnik (Controller) med sabo ne moreta komunicirati. Pri MVVM krmilnik zamenja *pogled modela* (Modelview), ki omogoča, da sta pogled in pogled modela konstantno povezana in lahko komunicirata med sabo [6]. AngularJS temelji na treh že uveljavljenih spletnih tehnologijah, to so Javascript, CSS in HTML.
Glavne funkcionalnosti, ki jih AngularJS ponuja, so:

- Vključevanje odvisnosti (angl. dependncy injection), ki razvijalcu omogoča, da že napisano logiko uporabi večkrat.
- Ajax (ang. asynchronous JavaScript and XML), ki omogoča asinhrono komuniciranje odjemalca s strežnikom.
- MVVM arhitektura in s tem avtomatsko sinhronizacijo med modelom in pogledom.
- Testiranje skozi celoten razvoj spletne aplikacije.

Razlogi za izbiro AngularJS so: enostavnost razvoja interaktivnih strani, razbremenjevanje strežnika tako, da se večino logike izvaja na odjemalčevi

strani, in možnost učenja sodobne spletne tehnologije.
V času začetnega razvoja je za AngularJS 2 obstajala samo beta verzija. Posledično smo se odločili za izbiro v razvoju AngularJS 1.

3.4 PostgreSQL

PostgreSQL je odprt kodni sistem za upravljanje z objektno-relacijski podatkovnimi bazami. Razvit je s strani PostgreSQL Global Development Group, ki je sestavljen iz različnih podjetji in posameznih razvijalcev. Deluje na vseh večjih operacijskih sistemih (Linux, FreeBSD, OS X, Solaris, Microsoft Windows, ...) [8]. Je popolnoma skladen s standardom ACID (angl. Atomicity, Consistency, Isolation, Durability):

- **Atomicity:** pomeni, da se vsaka transakcija izvede v celoti. V kolikor se katerega dela transakcije ne izvede, se cele transakcije ne izvede. To vključuje tudi izpad elektrike ali sesutje sistema.

- **Consistency:** vsaka transakcija bo prinesla podatkovno bazo iz enega veljavnega stanja v drugo veljavno stanje.

- **Isolation:** rezultat vsaj dveh transakcij, ki se izvajajo sočasno, je enak, kot če bi se izvajale zaporedno.

- **Durability:** zagotavlja, da bo transakcija, ko se izvede, ostala izvedena tudi po izgubi elektrike ali sesutju sistema.

PostgreSQL popolnoma podpira tudi tuje ključe (angl foreign keys), stikanje (angl. joins), poglede (angl. views), sprožilce (angl. triggers) in shranjene procedure.
Razlog za izbiro relacijske baze namesto NoSQL baze je ogromno ponovitev podatkov v drugih tabelah preko tujih ključev. V tem primeru so relacijske baze veliko hitrejše kot NoSQL.

3.5 Programsko okolje in jeziki za implementacijo

Za razvoj spletne aplikacije so bili uporabljeni naslednji programi:

- Sublime text 3 za pisanje strežniške logike v jeziku Python [11].
- WebStorm za razvoj logike na odjemalčevi strani v jezikih Javascript, HTML in CSS [13].
- DBeaver za interakcijo s podatkovno bazo v jeziku SQL [12].
- Chrome in Firefox za testiranje aplikacije.
- Bash za direktno interakcijo z Django strežnikom.

Poglavje 4

Načrtovanje spletnega portala

Kot osnova za načrtovanje spletnega portala so bile uporabljene uporabniške zgodbe po metodologiji Scrum. Uporabniška zgodba je kratek, preprost opis funkcije iz perspektive uporabnika. Večinoma se sledi predlogi: *Kot <tip uporabnika>, si želim <funkcionalnost/cilj>, ker <razlog>.* Razlog za izbiro uporabniških zgodb je, da se lahko na kratko opiše vse zahteve in dobi pregled, kako se bo portal uporabljal.

4.1 Uporabniške zgodbe

4.1.1 Uporabnik

Kot navaden uporabnik se lahko registriram s svojim študentskim elektronskim naslovom, da si lahko ogledam vsebino spletnega portala. Kot prijavljen uporabnik, lahko:

- dodam/odstranim predmete, ki jim želim slediti, da se mi nove objave, vezane na te predmete, pokažejo na domači strani,
- si ogledam objave znotraj vseh predmetov, ki jim sledim, ali pa samo po posameznem predmetu,
- si ogledam objave znotraj konteksta (predavanja, vaje, izpit, kolokvij, zapiski),

- ocenim objavo s *+1* ali *-1*, da lahko drugi uporabniki hitro vidijo kvalitetne objave,
- pri vsaki objavi izberem, na kateri predmet se veže in ali spada med predavanja, vaje, izpite, kolokvije ali zapiske,
- v kolikor objavim objavo, ki spada pod zapiske, lahko dodam URL povezavo do zapiska, da imajo ostali uporabniki možnost predogleda zapiskov znotraj objave,
- komentiram pod posamezno objavo,
- objavo shranim med zaznamke, da si jo lahko hitro pogledam, ne da bi jo vsakič na novo iskal,
- filtriram objave po vsebini ali avtorju, da lažje najdem želene objave,
- shranim samolepilni listič (angl. sticky note), da ne pozabim pomembnih stvari,
- spremenim geslo in uporabniško ime.

4.1.2 Administrator

Kot administrator lahko:

- izbrišem posamezno objavo,
- dodam/odstranim predmete, letnike in študije,
- uporabniku onemogočim prijavo, če objavlja neprimerno vsebino.

4.2 Načrtovanje podatkovnega modela

V osnovi imamo uporabnika s petimi funkcijami, ki najbolj vplivajo na podatkovni model:

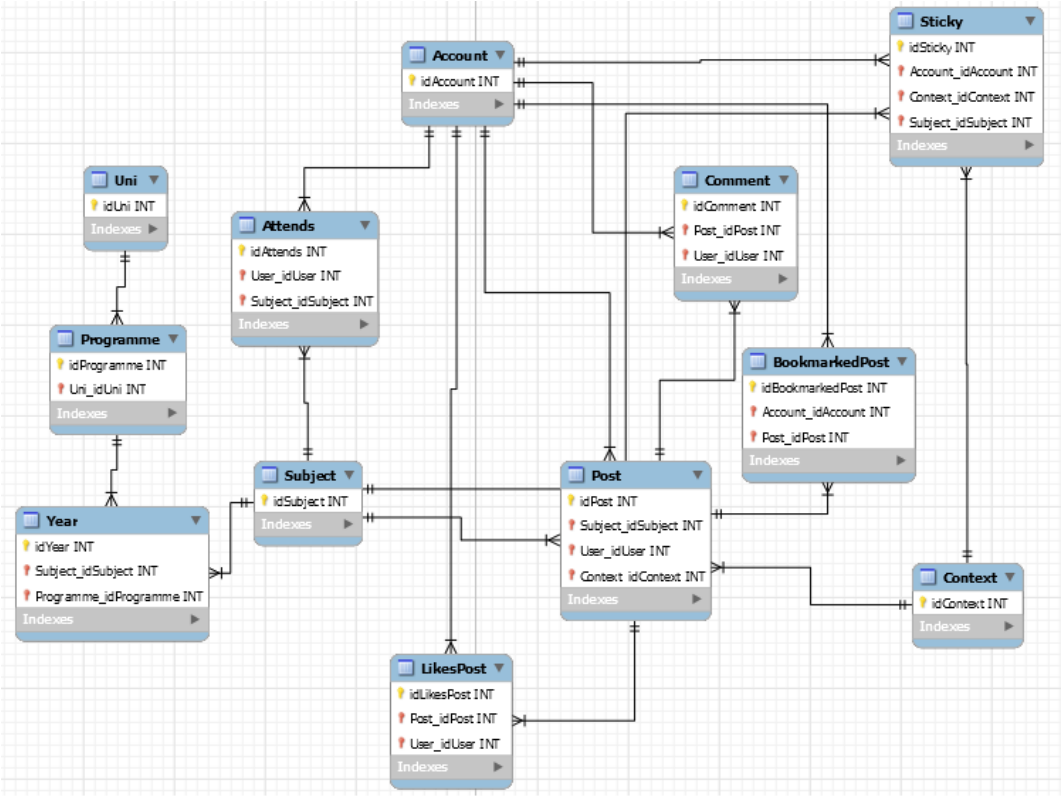
- Sledi predmetom.
- Piše objave. Za vsako objavo je treba določiti tudi, pod kateri predmet in kontekst spada.
- Komentira pod posamezno objavo. Pod posamezno objavo lahko komentira tudi večkrat.
- Oceni objavo. Vsako objavo lahko oceni le enkrat, vendar oceno lahko kasneje spremeni ali odstrani.
- Shrani samolepilni listič.

Tabela za predmete je vezana na tabelo za študijske smeri preko tabele za letnike. Znotraj tabele za letnike je atribut, ki predstavlja številko letnika. Prednost te implementacije je, da lahko predmet spada tudi pod več kot en letnik. Tako lahko predmeti, ki so si zelo podobni, vendar se izvajajo v drugem letniku ali študijski smeri, spadajo pod en in isti predmet. To omogoča, da si lahko večje število študentov pomaga med sabo. Glede na zgoraj naštete točke je bil sestavljen naslednji podatkovni model. (slika 4.1).

4.3 Podatkovni model

Kot je razvidno iz slike 4.1, je potrebno 12 tabel za delovanje aplikacije. Vsa imena so v angleščini, za lažje razumevanja baze osebam, ki ne razumejo slovenskega jezika, če bi take osebe začele razvijati aplikacijo. Te tabele so:

- Account (Uporabnik)
- Uni (Fakulteta)
- Programme (Študijska smer/program)
- Year (Letnik)
- Subject (Predmet)



Slika 4.1: Diagram podatkovnega modela

- Attends (Sledi)
- Sticky (Samolepilni listič)
- Post (Objava)
- Comment (Komentar)
- LikesPost (Ocena objave)
- BookmarkedPost (Objava z zaznamkom)
- Context (Kontekst objave)

Vsaki tabeli Django sam definira atribut identifikator, ki ga ni potrebno eksplicitno definirati.

Osnova aplikacije je možnost pisanja objav, ki so shranjeni v tabeli *Post*. Struktura objave je samoumevna. Potrebni atributi so:

- author (tuji ključ uporabnika, ki je ustvaril objavo),
- content (tekstovna vsebina objave),
- link (URL povezava na zapisek),
- context (tuji ključ konteksta objave, ki jo je izbral uporabnik pri ustvarjanju),
- subject (tuji ključ predmeta objave, ki jo je izbral uporabnik pri ustvarjanju),
- likes (vsota vseh ocen te objave),
- created_at (datum in ura, ko je bila objava ustvarjen),
- updated_at (datum in ura, ko je bila objava spremenjena, trenutno uporabnik še nima te funkcionalnosti).

V tabeli *Context* so samo našteti možni konteksti. V trenutni verziji aplikacije jih je 6: splošno (angl. general), predavanja (angl. lectures), vaje (angl. practices), kolokvij (angl. colloquiums), izpit (angl. exams) in zapiski (angl. notes). Konkteksti omogočajo lažje filtriranje in razvrščanje objav.

Vsaka objava ima lahko nato več komentarjev, ki so shranjeni v tabeli *Comment*. Struktura komentarja je zelo podobna strukturi objave. V tabeli se shranjujejo naslednji atributi:

- author (tuji ključ uporabnika, ki je ustvaril objavo),
- post (tuji ključ objave, pod katero spada komentar),
- content (tekstovna vsebina komentarja),
- created_at (datum in ura, ko je bil komentar ustvarjen),
- updated_at (datum in ura, ko je bil komentar spremenjen,t renutno uporabnik še nima te funkcionalnosti).

Ker imajo uporabniki možnost oceniti objavo (z ocenama 1 in -1), morajo biti ocene shranjene v posebni tabeli, ki zagotavlja, da ima lahko uporabnik samo eno oceno za eno objavo. Uporabnik lahko to oceno spremeni ali odstrani. To se shranjuje v tabeli *LikesPost* z naslednjimi atributi:

- author (tuji ključ uporabnika, ki je dal oceno),
- post (tuji ključ objave, katero je uporabnik ocenil),
- like (ocena objave),
- created_at (datum in ura, ko je bila ocena ustvarjena),
- updated_at (datum in ura, ko je bila ocena spremenjena).

Vsak uporabnik lahko objavo doda med zaznamke. Zaznamki omogočajo uporabniku, da objavo doda na svoj seznam, ki je ločen od seznama objav, ki so vezana na predmet. To se shranjuje v tabeli *BookmarkedPost* z naslednjimi atributi:

- account (tuji ključ uporabnika),
- post (tuji ključ objave),
- created_at (datum in ura, ko je bil zaznamek ustvarjen).

Uporabnik lahko v aplikaciji tudi shrani večje število samolepilnih lističev, ki se shranjujejo v tabeli *Sticky*. Po strukturi rahlo spominjajo na strukturo objave, ki je poenostavljena z atributi:

- author (tuji ključ uporabnika, ki je ustvaril objavo),
- content (tekstovna vsebina),
- context (tuji ključ konteksta objave, ki jo je izbral uporabnik pri ustvarjanju),
- subject (tuji ključ predmeta objave, ki jo je izbral uporabnik pri ustvarjanju),
- seqnum (zaporedna številka lističa, ki je znotraj vsakega uporabnika unikatna),
- created_at (datum in ura, ko je bila objava ustvarjen),
- updated_at (datum in ura, ko je bila objava spremenjena).

Do spletne aplikacije lahko dostopajo le registrirani uporabniki, ki so shranjeni v tabeli *Account*. S to tabelo je nadomeščena privzeta implementacija uporabnika v Django aplikacijah. Tabela Account vsebuje atribute:

- email (email uporabnika),
- username (uporabniško ime uporabnika),
- password (implicitno definiran preko Django privzete funkcionalnosti za gesla, ki zagotovi shranjevanje preko zgoščevalne funkcije z dodano soljo),

- is_admin (v kolikor je uporabnik administrator, lahko dostopa do *admin* strani),
- is_banned (v kolikor je uporabnik *banned*, mu je onemogočena prijava).

Vsaka objava je vezana na en predmet. V tabeli *Subject* so shranjeni študijski predmeti. Vsak predmet ima naslednje podatke:

- name (ime predmeta),
- abb (kratica za predmet),
- created_at (datum in ura, ko je bil predmet ustvarjen),
- updated_at (datum in ura, ko je bil predmet spremenjen).

Seznam fakultet je shranjen v tabeli z imenom *Uni*. Atributi so:

- name (ime fakultete),
- abb (kratica za fakulteto),
- city (mesto, kjer se nahaja fakulteta),
- country (država, kjer se nahaja fakulteta),
- created_at (datum in ura, ko je bil zapis ustvarjen),
- updated_at (datum in ura, ko je bil zapis spremenjen).

V tabeli *Programme* so shranjene študijske smeri (študijski programi). Vsaka študijska smer spada pod eno fakulteto. Posledično so potrebni atributi:

- uni (tuji ključ fakultete),
- name (ime smeri študija),
- level (stopnja študija),

- abb (kratica za študij),
- created_at (datum in ura, ko je bil zapis ustvarjen),
- updated_at (datum in ura, ko je bil zapis spremenjen).

Tabeli *Programme* in *Subject* sta povezana skupej s tabelo *Year*. . Predmet spada v študij tako, da mu je dodeljen letnik. Posledično se lahko en predmet razvršča v več študijskih programov in v različne letnike. Razlog za tako odločitev je, da so si nekateri predmeti enaki, čeprav spadajo pod različne študije, tako jih je enostavno grupirati brez ustvarjanja dodatne logike. Atributi v tabeli *Programme* so:

- year (številka leta),
- prog (tuji ključ študija),
- subject (tuji ključ predmeta),
- created_at (datum in ura, ko je bil zapis ustvarjen),
- updated_at (datum in ura, ko je bil zapis spremenjen).

Katerim predmetom uporabnik sledi je shranjeno v tabeli *Attends* z naslednjimi atributi:

- account (tuji ključ uporabnika, ki sledi predmetu),
- subject (tuji ključ predmeta, ki mu uporabnik sledi),
- created_at (datum in ura, ko je bil zapis ustvarjen),
- updated_at (datum in ura, ko je bil zapis spremenjen).

Poglavje 5

Potek in izdelava spletne aplikacije

5.1 Programski vmesnik

Komunikacija med strežnikom in odjemalcem poteka preko strežniškega programskega vmesnika (angl. application program interface), zato smo najprej začeli delati na njem. Podatki se pošiljajo v formatu JSON. Potrebno je bilo definirati vse končne točke, za katere bo odjemalec potreboval podatke. Primer odgovora za eno objavo odjemalcu je na sliki 5.1. Razširitev *Django REST Framework* za Django omogoča, da se lahko na efektiven način definira končne točke. Omogoča tudi koreninske končne točke, to je primerljivo z ugnezdeno poizvedbo. Če vzamemo konkretni primer, */account/1/* vrne vse informacije o uporabniku z identifikatorjem. Prav tako lahko z URL-jem */account/1/attends/* pridobimo vse predmete, ki jim ta uporabnik sledi (v kolikor ne sledi nobenemu, vrne prazno).
Glavne končne točke za aplikacijo so:

- account (uporabnik) s podtočkami:
  - attends (predmeti, ki jim sledi)
  - unattends (predmeti, ki jim ne sledi)

```
"content": "Predavatelj je rekel, da kdor se predavanj ne bo udeleževal bo moral nujno na ustni izpit",
"context": 2,
"likes": [
    {
        "id": 27,
        "author": 3,
        "post": 34,
        "like": 1,
        "created_at": "2016-06-13T13:05:27.634362Z",
        "updated_at": "2016-06-13T13:05:27.641778Z"
    }
],
"comments": [
    {
        "id": 92,
        "author": {
            "id": 3,
            "email": "test@test.si",
            "username": "Janez Kovač",
            "created_at": "2016-04-18T16:02:27.189570Z",
            "updated_at": "2016-04-18T16:02:27.189584Z"
        },
        "post": 34,
        "content": "Test123",
        "likes": 0,
        "created_at": "2016-06-13T13:03:47.685703Z",
        "updated_at": "2016-06-13T13:03:47.691677Z"
    },
    {
        "id": 91,
        "author": {
            "id": 21,
            "email": "pabldomain@gmail.com",
            "username": "Bogdan Golobič",
            "created_at": "2016-04-20T08:55:32.761791Z",
            "updated_at": "2016-05-19T16:19:48.862633Z"
        },
        "post": 34,
        "content": "Hvala!",
        "likes": 0,
        "created_at": "2016-06-13T10:49:07.315700Z",
        "updated_at": "2016-06-13T10:49:07.319017Z"
    }
],
"created_at": "2016-05-19T11:58:01Z",
"updated_at": "2016-05-19T11:58:01Z"
```


Slika 5.1: Primer odgovor strežnika pri zahtevku po objavah

  - bookmarksposts (vse objave, ki jih ima pod zaznamki)
  - attendsposts (objave predmetov, ki jim sledi)

- posts (objave) s podtočkami:
  - comments (komentarji objave)
  - contexts (objave določenega konteksta)
  - likes (ocene) s podtočko:
    - likes (ocene posameznega uporabnika)

- subjects (predmeti) s podtočkamo
  - posts (objave posameznega predmeta s podtočko
    - contexts (objave posameznega predmeta s posameznim kontekstom)

- contexts (vsi konteksti)

- stickys (vsi samolepilni lističi) s podtočko:
  - account (lističi posameznega uporabnika)

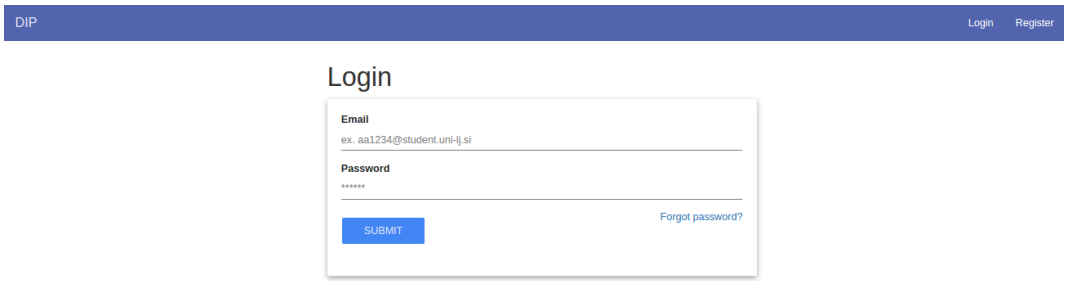
5.2 Uporabniški vmesnik za običajnega uporabnika

Spletno aplikacijo lahko uporabljajo le prijavljeni uporabniki. Da se lahko uporabnik prijavi, se mora najprej registrirati. Registracija je možna le s študentskim elektronskim naslovom. Razlogov za to odločitev je več. Kot prvo je študentom pomembno, da imajo dostop le študentje. Tako uporabnikom ni treba skrbeti, da bi lahko objave bral kdorkoli. Enako so uporabniki dosegli pri Facebooku z uporabo zaprtih Facebook skupin. Vsebino teh skupin si lahko videl šele, ko ti je skrbnik skupine dovolil dostop. Tako je bilo vsaj deloma nadzorovano, kdo dostopa do vsebine. Razlog za tako

Slika 5.2: Registracija

odločitev je tudi, da se skoraj popolnoma izognemo nezaželenim objavam. Vsak študent ima le en elektronski naslov na študij, ki ga obiskuje (večina študentov obiskuje le enega). V kolikor bi ta elektronski naslov uporabljal za širjenje nezaželenih objav, mu je prijava lahko onemogočena. Ker so ti elektronski naslovi na osebo zelo omejeni, bi nepridipravu hitro omejili dostop z zaprtjem vseh elektronskih naslovov, ki jih ima.

Ob registraciji ima uporabnik možnost vpisati svoj študentski elektronski naslov in uporabniško ime (slika 5.2). Uporabniško ime bo potem tisto, s katerim bo prepoznan znotraj aplikacije. Priporočljiva je uporaba formata *ImePriimek* ali *PriimekIme* za lažje prepoznavanje uporabnikov. Seveda je uporabniku omogočena prosta izbira imena, v kolikor ima le alfanumerične znake. Ob uspešni registraciji je na uporabnikov elektronski naslov poslano sporočilo z geslom. Geslo je dolgo 8 znakov in je sestavljeno iz naključnih alfanumeričnih znakov. Za shranjevanja gesla poskrbi Django, da mu je dodana sol in je poslan skozi zgoščevalno funkcijo za varno shranjevanje. S



Slika 5.3: Prijava

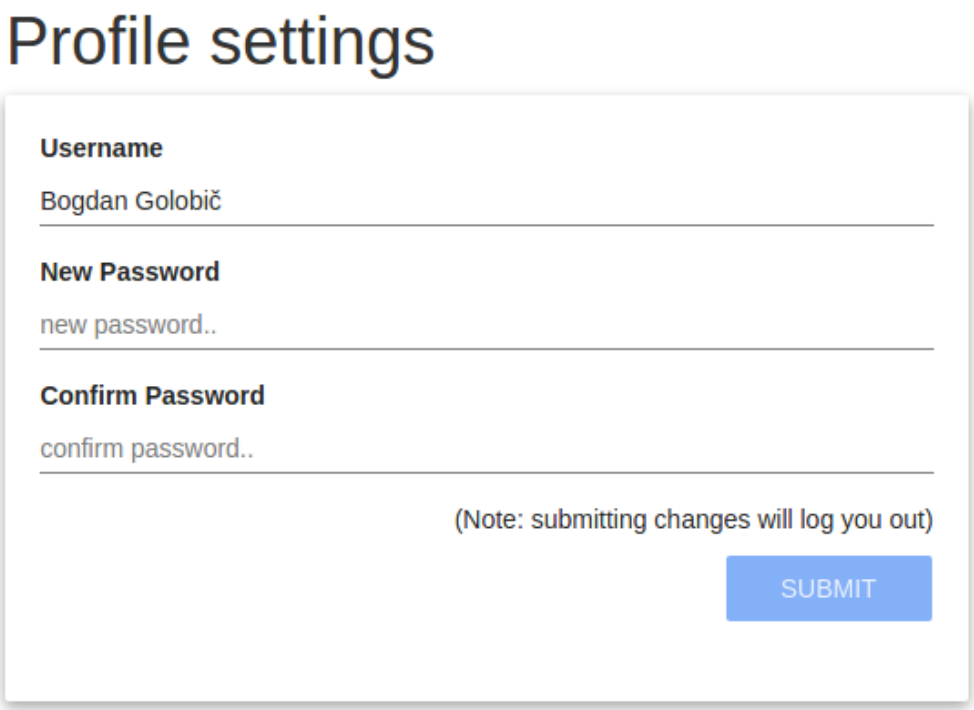
tem je zagotovljeno, da je oseba, ki se prijavlja, res lastnik vpisanega naslova. S poslanim geslom je uporabniku omogočena prijava v aplikacijo (slika 5.3). Seveda ima prijavljen uporabnik za tem možnost, da spremeni geslo, če mu dodeljeno ne ustreza. Hkrati lahko spremeni tudi uporabniško ime, s tem se ustrezno spremeni ime pri vseh njegovih prejšnjih objavah (slika 5.4).

Osnovne informacije o uporabniku se nato shranijo v piškotek (angl. cookie). Hkrati se shrani tudi piškotek CSRF (angl. Cross-Site Request Forgery), ki onemogoča ustvarjanja ponarejenih zahtevkov, za katerega skrbita Django in AngularJS sama. Le na odjemalčevi strani je bilo potrebno nastaviti:

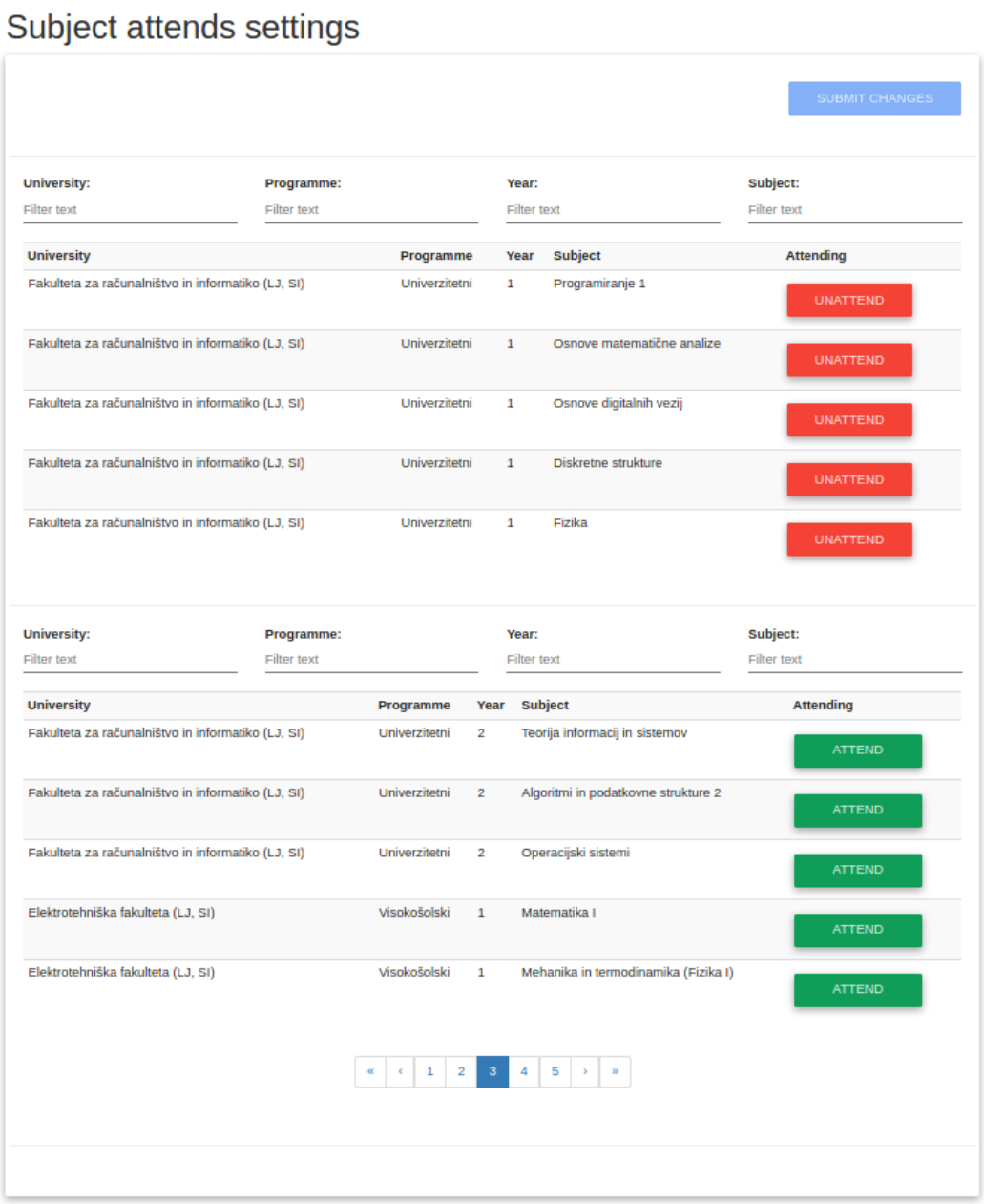
```
1 angular.module('dip').run(function ($http) {
2     $http.defaults.xsrfHeaderName = 'X-CSRFToken';
3     $http.defaults.xsrfCookieName = 'csrftoken';
4 });
```

Listing 5.1: Nastvaitev CSRF piškotka

Da lahko uporabnik učinkovito uporablja aplikacijo, je bila ustvarjena stran, kjer lahko dostopa do izbire predmetov, ki jim sledi. Sestavljena je iz dveh tabel (slika 5.5). Vsebina prve tabele so predmeti, ki jim sledi. Temu primerno so v drugi tabeli predmeti, ki jim ne sledi. Ker je takih predmetov ogromno, je možno filtrirati predmete, ki jim uporabnik ne sledi po



Slika 5.4: Sprememba gesla in uporabniškega imena



Slika 5.5: Primer nastavljanja sledenja predmetom

študiju/smeri, letniku in imenu predmeta. Uporabi lahko en filter ali pa poljubno število filtrov hkrati. Prav tako so predmeti, ki jim ne sledi, straničeni za lažje iskanje. Da ne pride do nerazumevanja, se predmet ob izbiri, da mu uporabnik sledi, odstrani iz tabele predmetov, ki jim ne sledi, in doda v tabelo, ki jim sledi. V obratni smeri se tabeli obnašata enako. Spremembe se ne shranijo takoj, da se ne obremenjuje strežnika, v kolikor se uporabnik zmoti. Posledično mora uporabnik po spremembi vsebin tabel pritisniti na gumb Save changes (Shrani spremembe).

Takoj po shranitvi spremembe se osveži seznam izbranih predmetov v stranski orodni vrstici (angl. side bar). To je omogočeno preko dispečerjev (angl. dispatcher) in poslušalca dogodkov (angl. event listener), ki omogočata, da lahko kontrolerja med sabo komunicirata preko *$rootScope*.

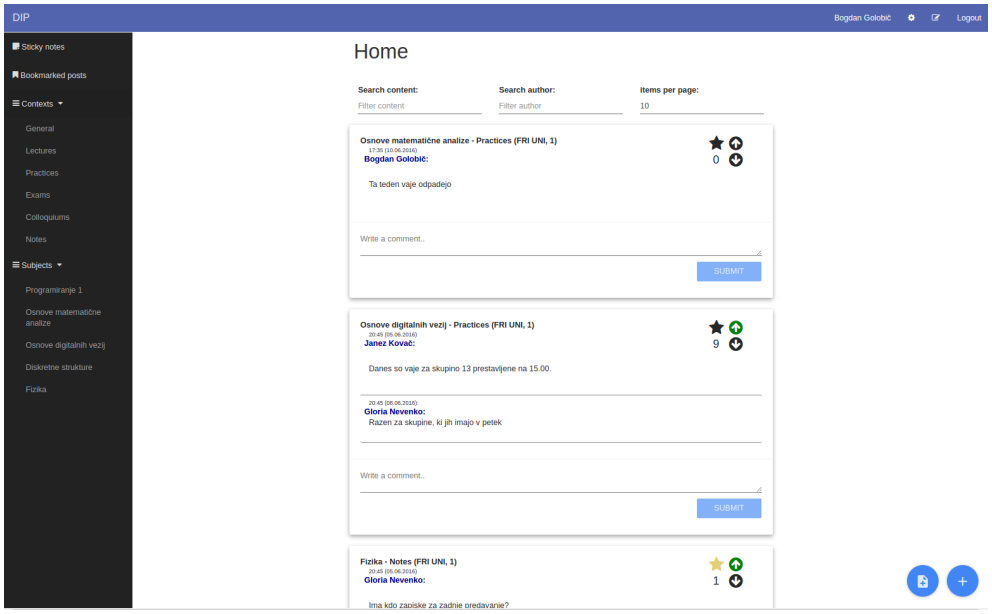
```
1 function attendChangeSaved(data, status, headers, config) {
2     Snackbar.show("Updating attend successful!");
3     $rootScope.$broadcast('attends-changed');
4 }
```

Listing 5.2: Dispečer

```
1 $scope.$on('attends-changed', function(event, args) {
2     Profile.getAttends().then(attendsSuccessFn,
          attendsErrorFn);
3 });
4 function  attendsSuccessFn(data, status, headers, config) {
5     Profile.attends = data.data;
6     $scope.subjects = Profile.attends;
7 }
8 function attendsErrorFn(data, status, headers, config) {
9     Snackbar.error('Major error reading attends!');
10 }
```

Listing 5.3: Poslušalec dogodka

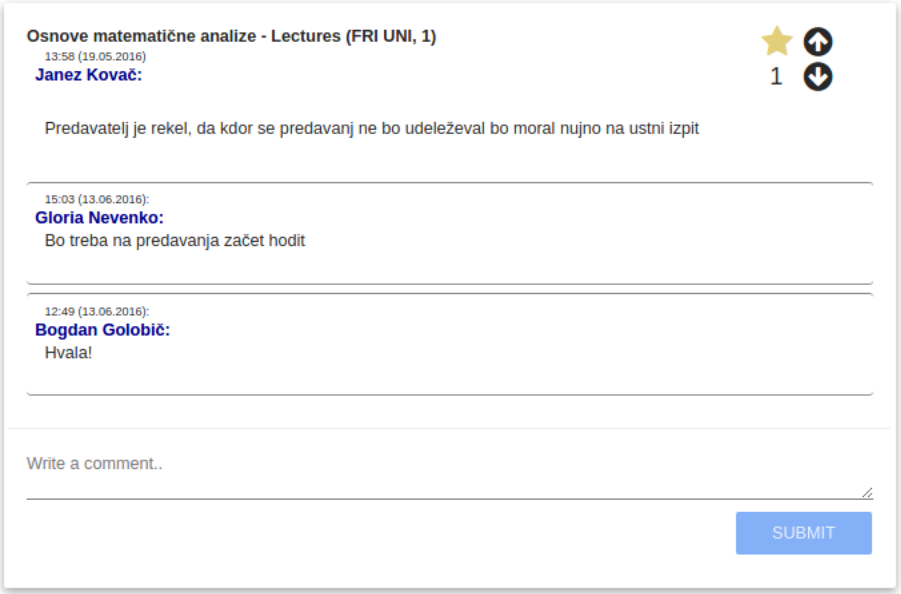
S tem je omogočena popolna funkcionalnost stranske orodne vrstice, hkrati pa je le-ta osvežena po vsaki spremembi in ni potrebe, da bi uporabnik ponovno osvežil celotno stran.



Slika 5.6: Uporabniški vmesnik za domačo stran

Na domači strani so navedene objave vseh predmetov, ki jim uporabnik sledi. Preko stranske orodne vrstice lahko te objave filtrira po predmetu in kontekstu (slika 5.6). Možnosti so: samo po predmetu, samo po kontekstu ali pa po predmetu in kontekstu hkrati. Prav tako lahko dostopa do objav, ki jih je dodal med zaznamke. Z namenom, da si lahko uporabniki delijo linke do posameznega predmeta s ali brez konteksta, se ta filter doda v URL (primer: primer.com/subject/26/context/6/), kjer so številke identifikatorja predmeta ali konteksta. Premikanje znotraj strani med predmeti in konteksti ne zahteva osveževanja strani. Vsaka poizvedba je asinhrona in uporabniku stran izgleda veliko bolj odzivna in dinamična. Prav tako lahko uporabnik znotraj teh filtrov filtrira še po vsebini in avtorju objave.

Znotraj posamezne objave so nanizani potrebni podatki, da uporabnik ugotovi, kam spada. Tako je vsaka objava posamezna entiteta, za katero uporabnik ne potrebuje drugih informacij, da razume kontekst objave. Najprej je napisan predmet in kontekst in za tem v oklepaju študij in letnik (slika

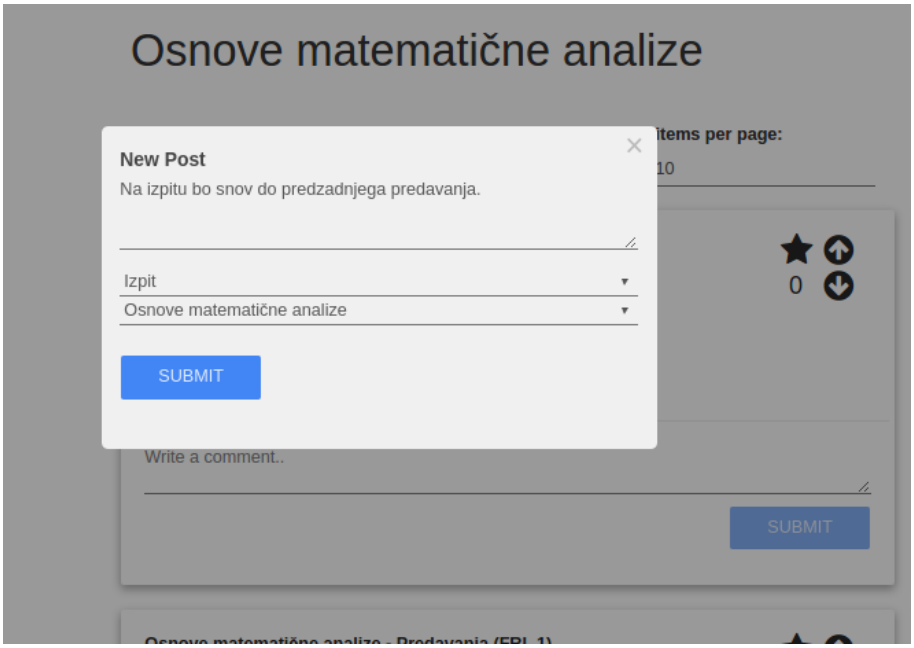


Slika 5.7: Uporabniški vmesnik posamezno navadno objavo

5.7). Pod tem je ura in datum objave in spodaj še avtor. Na desni strani je rezultat končne vsote ocen objave, prav tako ima uporabnik možnost, da oceni objavo in jo doda med zaznamke. Pod vsem tem je vsebina objave. V kolikor spada objava pod zapiske in ima veljavno URL povezavo do zapiska, ki je lahko v formatu PDF, .doc, .docx, .xls, .xlsx, .ppt ali .pptx, se interaktivni predogled do tega zapiska prikaže (slika 5.8). Uporabnik lahko tako pregleda celoten dokument znotraj te objave, lahko pa jo odpre v novem zavihku ali pa prenese na svojo napravo. Nato so nanizani vsi komentarji. Komentar je sestavljen iz ure in datuma komentarja, avtorja in vsebine. Seveda ima uporabnik možnost tudi sam dodati komentar. Dodajanje komentarjev je zopet dinamično in ni potrebno osvežiti strani, da se komentar prikaže.

Novo objavo lahko uporabnik ustvari tako, da pritisne na desni gumb izmed dveh, ki se vedno nahajata desno spodaj, ta gumb je videti na sliki 5.6. Po pritisku se pojavi nov obrazec (slika 5.9), kjer uporabnik lahko vpiše vsebino objave, izbere pod kateri predmet spada (predlaga tiste, ki jim uporabnik sledi) in pod kateri kontekst spada. V kolikor uporabnik izbere opcijo

Slika 5.8: Uporabniški vmesnik posamezno objavo z zapiski

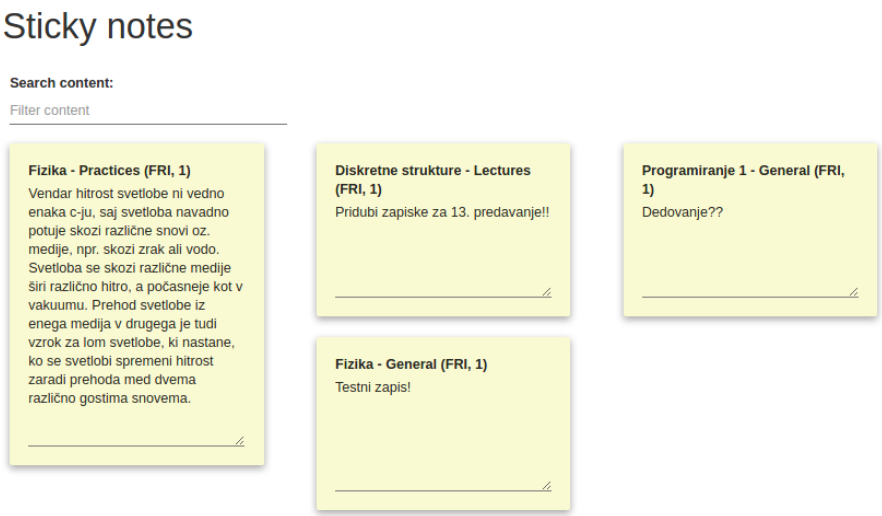


Slika 5.9: Uporabniški vmesnik za novo objavo

zapiski (angl. notes), se mu pojavi novo vnosno polje, kjer lahko vpiše URL povezavo do zapiska. Vsa tri vnosna polja morajo biti izpolnjena, drugače objave ni možno potrditi. Objava se takoj prikaže, brez potrebe po ponovni osvežitvi strani. V kolikor je prišlo do napake pri shranjevanju objave, ta objava izgine iz uporabniškega vmesnika.

Ob pritisku na levega od dveh gumbov, ki se nahajata desno spodaj (slika 5.6), si uporabnik lahko naredi novi samolepilni listič. Odpre se podoben obrazec kot pri novi objavi. Enako lahko vpiše vsebino, izbere kontekst in predmet. Ta ustvarjen listič se pojavi na uporabniškem vmesniku na seznamu lističev (slika 5.10). Znotraj tega uporabniškega vmesnika lahko uporabnik spreminja vrstni red lističev, ki se shrani med različnimi sejami uporabnika in spreminja vsebino že ustvarjenih. Prav tako lahko vsak listič izbriše, če to želi. Razporejanje lističev se prilagaja njihovi velikosti, da ne pride do prekrivanja.

Pri razvijanju spletne aplikacije smo imeli v mislih različne velikosti za-



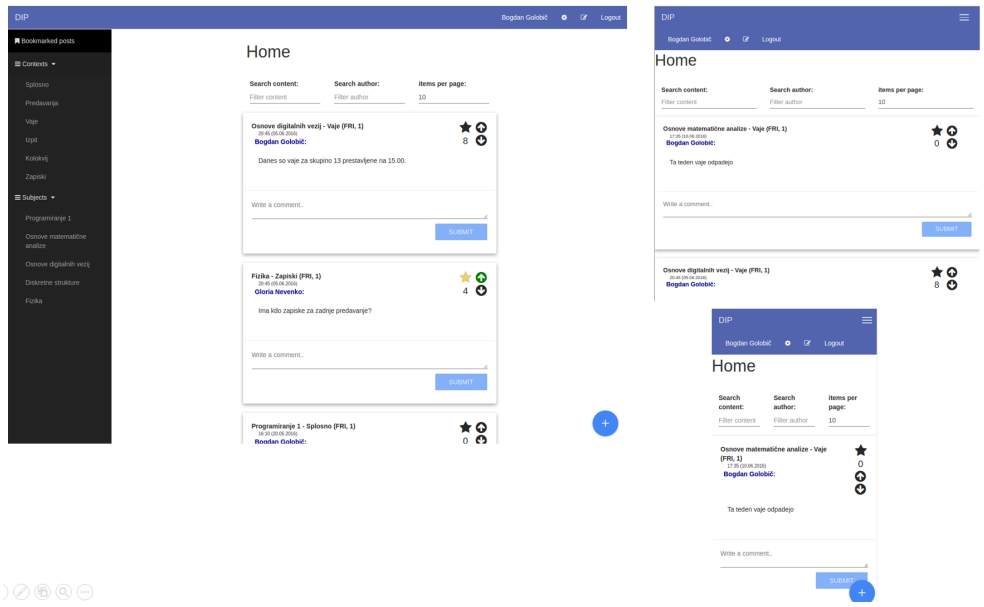
Slika 5.10: Uporabniški vmesnik za seznam samolepilnih lističev

slonov. S tem so mišljene dimenzije zaslonov tablic in telefon, kot seveda tudi osebnega računalnika (slika 5.11). Vse več ljudi dostopa do internetnih vsebin preko bolj mobilnih naprav z manjšimi zasloni. S tem namenom je bilo uporabljeno okolje Bootstrap, ki zaslon razdeli na 12 stolpcev (vsak stolpec ima lahko tudi svoje podstolpce za še bolj detajlirano oblikovanje). S tem se tvori mrežni izgled (angl. grid). Prikaz elementa je odvisen od širine zaslona (število pikslov). Bootstrap razdeli zaslone v štiri skupine:

- skupina *col-xs-** do širine 768 pikslov,
- skupina *col-sm-** do širine 970 pikslov,
- skupina *col-md-** do širine 1200 pikslov,
- skupina *col-lg-** nad 1200 piksli.

Z uporabo teh CSS skupin pri predlogah je lažje kontrolirati izgled.

Prav tako je bil pri razvijanju aplikacije velik poudarek na čim prijetnejši uporabniški izkušnji. Cilj je bil, da uporabnik nikoli ne rabi osvežiti strani,

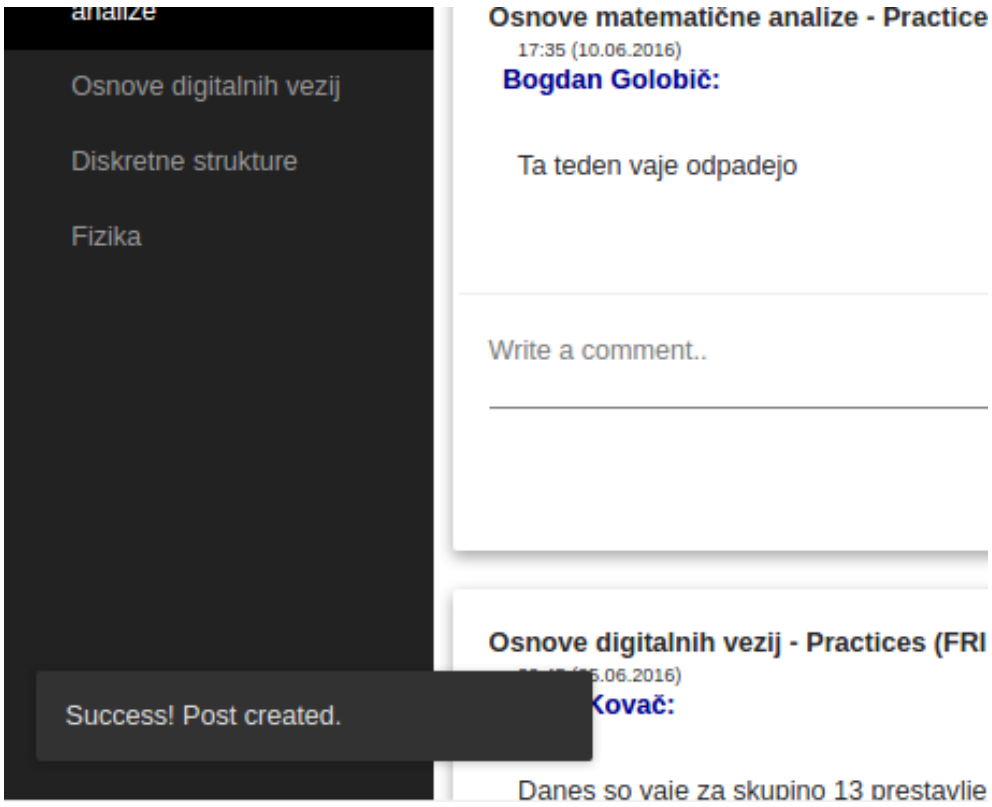


Slika 5.11: Uporabniški vmesnik pri različih širinah ekrana

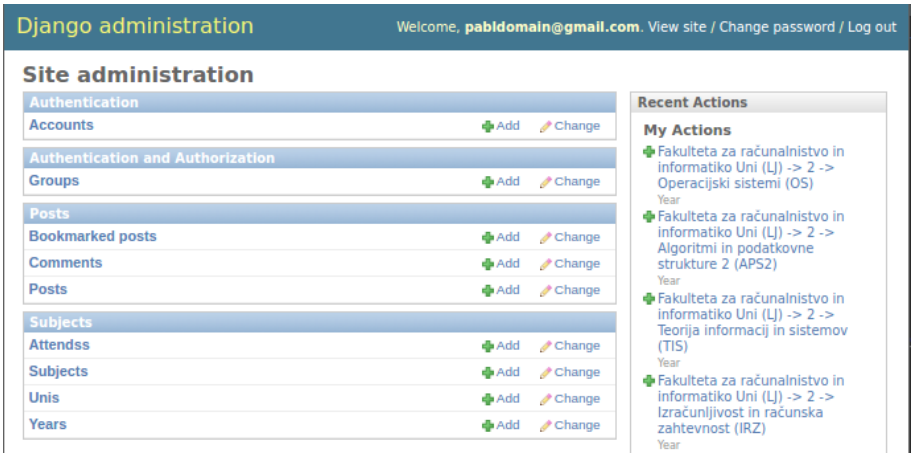
da bi opazil ustvarjene spremembe. Vsaka akcija sproži reakcijo, ki se prikaže na ekranu. Tako se uporabnik nikoli ne rabi spraševati, ali se je akcija izvedla uspešno. Nekaj primerov je bilo že opisanih v prejšnjih odstavkih tega poglavja. Drug način za dosego tega cilja je sporočanje uporabniku preko majhnega pojavnega okna (angl. snackbar), ki se prikaže v kotu in sporoči uporabniku ali je bila akcija izvedena uspešno ali neuspešno (slika 5.12).

5.3 Uporabniški vmesnik za administratorja

Za uporabniški vmesnik administratorja se uporablja privzeti uporabniški vmesnik, ki ga ponuja Django. Znotraj njega ima administrator dostop do podatkovne baze, torej lahko vsebino spreminja, dodaja in odstranjuje (slika 5.13). Uporaba tega vmesnika je lažja kot uporaba uporabniškega vmesnika, ki bi se neposredno povezal z bazo. Razlog je, da je lahko znotraj definicije podatkovnih modelov (tabel) v Djangu dodana definicija za privzeto "poimenovanje" tujega ključa. Namesto da izpiše vrednost identifikatorja pri tujem



Slika 5.12: Primer sporočanja preko pojavnega okna ob uspešno ustvarjeni objavi



Slika 5.13: Uporabniški vmesnik za administratorja

ključu, izpiše definirane vrednosti za povezano vrednost. Na primer: namesto številke tujega ključa za uporabnika z vrednostjo 13 se izpiše vrednost: BogdanGolobič, bg0000@student.uni-lj.si. Primer definicije:

```
1     return u'author: %s post:%s: %s' % (self.author, self.
          post, self.like)
```

S tem je omogočeno lažje kontroliranje vsebine.

Poglavje 6

Sklepne ugotovitve

V sklopu diplomske naloge je bila razvita spletna aplikacija, ki študentom omogoča medsebojno komuniciranje glede tekočega študija preko spleta, kot alternativa Facebooku in forumom. Ideja je prišla iz slabih lastnih izkušenj uporabe le-teh, saj niso bili razviti v ta namen. Posledično bi lahko bila namenska rešitev veliko prijaznejša uporabniku in bolj praktična. Vendar pa lastne izkušnje ne izražajo vedno dejanskega mnenja populacije. S tem namenom smo se odločili izvesti anonimno anketo, namenjeno študentom in dijakom. Izkazalo se je, da imajo ostali študentje in dijaki enake izkušnje kot mi. Hkrati imajo želje po novih funkcionalnosti, določene implementirane pa ovirajo uporabo. Več kot 92% anketiranih bi uporabljalo spletno aplikacijo, ki bi zajemala spremembe, opisane v anketi.

V okviru razvoja smo se morali soočiti z veliko problemi, saj se z večino tehnologij prej nismo še nikoli srečali. Problemi so se z branjem dokumentacije in odgovorov na vprašanja ostalih razvijalcev z enakimi težavami rešili. Največ dela je zahteval razvoj funkcionalnosti na odjemalčevi strani, to je pri razvoju v AngularJS. Razlog za to je tudi, da se večino programske logike izvaja pri odjemalcu in ne na strežniku. Vsa komunikacija med strežnikom in odjemalcem poteka preko strežniškega API-ja, razvitega v Djangu.

Razvita spletna aplikacija vsebuje kar nekaj pomanjkljivosti, ki bi jih bilo treba odpraviti, če bi hoteli, da se aplikacija spravi v realni svet. Pred-

vsem bi bilo potrebno odpraviti vse varnostne luknje. Sedaj lahko uporabnik napiše objavo za kateri koli predmet, če pošlje HTTP zahtevek v pravilni obliki, s pravimi JSON podatki, saj strežnik ne preverja vsebine prejetih objav. Enako velja tudi za komentarje. Dodali bi lahko tudi več funkcionalnosti. Ena izmed razširitev bi lahko bil ogled kateregakoli predmeta, tudi če mu ne slediš. Dobra razširitev bi bila tudi možnost shranjevanja zapiskov in ostalih datotek znotraj same aplikacije. Tako bi ta spletna aplikacija lahko postala celovit sistem za študente glede tekočega študija.

Možna dodatna rešitev, ki bi lahko bila razvita boj ločeno, bi bila lahko tudi mobilna aplikacija. Čeprav se vsebina strani prilagaja širini ekrana, bi lahko z mobilno aplikacijo omogočili hitrejše in varnejše delovanje na mobilnih napravah. Prav tako bi lahko razvili funkcionalnosti, ki uporabljajo možnost lokalnega shranjevanja podatkov na uporabnikovi napravi.

Literatura

[1] AngularJS dokumentacija. Dosegljivo: `https://docs.angularjs.org/api`, 2005. [Dostopano: 13. 5. 2016].

[2] Django dokumentacija. Dosegljivo: `https://docs.djangoproject.com/en/1.8/`, 2005. [Dostopano: 20. 4. 2016].

[3] Googlov spletni portal za ankete. Dosegljivo: `https://docs.google.com/forms/`. [Dostopano: 2. 3. 2016].

[4] Adrian Holovaty and Jacob Kaplan-Moss. *The Definitive Guide to Django: Web Development Done Right, Second Edition.* Apress, Berkely, CA, USA, 2nd edition, 2009.

[5] MVC. Dosegljivo: `https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller`. [Dostopano: 26. 5. 2016].

[6] MVVM. Dosegljivo: `http://stackoverflow.com/questions/667781/what-is-the-difference-between-mvc-and-mvvm`. [Dostopano: 26. 5. 2016].

[7] ORM. Dosegljivo: `http://hibernate.org/orm/what-is-an-orm/`. [Dostopano: 27. 5. 2016].

[8] PostgreSQL dokumentacija. Dosegljivo: `https://www.postgresql.org/docs/`. [Dostopano: 27. 4. 2016].

[9] RestPrinciple. Dosegljivo: http://www.codeproject.com/Articles/283550/Implementing-important-principles-of-REST-using. [Dostopano: 27. 5. 2016].

[10] Leonard Richardson and Mike Amundsen. *RESTful Web APIs*. O'Reilly Media, 1st edition, 2013.

[11] Urejevalnik besedila SublimeText3. Dosegljivo: https://www.sublimetext.com/3. [Dostopano: 2. 4. 2016].

[12] DBeaver upravitelj podatkovnih baz. Dosegljivo: http://dbeaver.jkiss.org/. [Dostopano: 2. 5. 2016].

[13] WebStorm razvijalno okolje. Dosegljivo: https://www.jetbrains.com/webstorm/. [Dostopano: 5. 5. 2016].

[14] Ken Williamson. *Learning AngularJS: A Guide to AngularJS Development*. O'Reilly Media, 1st edition, 2015.