UNIVERZA V LJUBLJANI

FAKULTETA ZA DRUŽBENE VEDE

Kevin Mastnak

# Izdelava spletne aplikacije za promocijo vsebin preko Facebook API-ja.

Diplomsko delo

Ljubljana, 2017

UNIVERZA V LJUBLJANI

FAKULTETA ZA DRUŽBENE VEDE

Kevin Mastnak

Mentor: izr. prof. dr. Aleš Žiberna

# Izdelava spletne aplikacije za promocijo vsebin preko Facebook API-ja.

Diplomsko delo

Ljubljana, 2017

**Izdelava spletne aplikacije za promocijo vsebin preko Facebook API-ja**

Uporabniki Facebooka, katerih namen uporabe tega družbenega omrežja je ohranjanje socialnih stikov, obveščenost o dogajanju po svetu in deljenje stvari, do katerih jim je mar, so postali vedno pogostejša tarča oglaševalskih vsebin, ki se jim prikrito in neprikrito (targetirano) ponujajo, s svojimi vedenjskimi dejavnostmi pa Facebooku nevede ustvarijo ogromno oglaševalskega prihodka. Cilj diplomske naloge je s pomočjo Facebook API-ja ustvariti sistem, od katerega bodo vsi aktivni uporabniki imeli koristi, kar pomeni spremembo poti transakcij, ki sedaj potekajo od oglaševalca k Facebooku, k oglaševalcu neposredno do uporabnika; še pomembneje, tehnično izvesti celoten projekt in pridobiti potrebna programerska znanja. Ugotovil sem, da za vsak problem, na katerega sem med izdelavo spletne aplikacije naletel, že obstaja mnogo različnih pristopov reševanja tega problema, sam pa sem jih pogostokrat reševal s pomočjo implementacije tujih, že preverjenih knjižnic in praks. Menim, da mi je na koncu uspelo postaviti uporabnikom prijazno spletno aplikacijo, ki služi svojemu namenu, zanimivo pa bi jo bilo spremljati ob polnem zagonu.

**Ključne besede:** oglaševanje, uporabniška izkušnja, Python, Django, Facebook API.

**Building a Web Application for Promoting Content Using Facebook API**

The main purpose of using Facebook social network is to maintain social connections with friends and family or investigate current events in the world or share and express what matters to its users. However, Facebook users have become the centre of subliminal and targeted advertising. Their everyday activities unknowingly generate enormous amount of advertising revenue for the company. The aim of this thesis is to use Facebook API in order to create a system, which will benefit every user. This means the direction of transactions from advertisers to Facebook will shift directly to users themselves. More importantly, the focus is on gaining all the necessary programming skills for technical execution of the entire project. I have discovered that each problem I came across during the coding process has many different working solutions and approaches one can use to solve them. I followed the golden rule of programming, which emphasizes using the existent approaches. Finally, I believe I managed to build a user-friendly web application, which serves the purpose. Furthermore, it would be interesting to observe the application when used at full charge.

**Keywords:** Advertising, User Experience, Python, Django, Facebook API.

**KAZALO**

**1 UVOD ..... 5**
**2 TEORETIČNI PREGLED ..... 8**
2.1 Od spleta 1.0 do spleta 2.0 ..... 8
2.2 Spletna družbena omrežja ..... 11
2.3 Definicija družbenih omrežij ..... 11
2.4 Družbeno omrežje Facebook ..... 12
2.5 Oglaševanje na Facebooku ..... 13
2.6 Oglaševanje na moji aplikaciji ..... 15
2.7 Uporabniška izkušnja ..... 15
**3 UPORABLJENA TEHNOLOGIJA ..... 17**
3.1 Python ..... 18
3.2 Django ..... 18
3.3 HTML ..... 19
3.4 CSS ..... 19
3.5 JavaScript in jQuery ..... 19
3.6 AJAX ..... 19
3.7 SQLite in SQL ..... 20
3.8 Facebook API ..... 20
**4. IZDELAVA SPLETNE APLIKACIJE ..... 21**
4.1 Priprava delovnega okolja ..... 21
4.2 Avtentikacija in avtorizacija s pomočjo družbenega omrežja Facebook ..... 24
4.3 Model uporabnika ..... 28
4.4 Promocija vsebine ..... 30
4.5 Model objave ..... 33
4.6 Domača stran ..... 36
4.7 Dashboard ..... 42
4.8 Transakcije na osebni račun ..... 46
**5 ZAKLJUČEK ..... 46**
**6 LITERATURA ..... 48**
**PRILOGA A: Povezava na izvorno kodo projekta ..... 50**

## KAZALO SLIK

Slika 4.1: Seznam paketov nameščenih z orodjem Pip. ..... 22
Slika 4.2: Avtentikacijska stran aplikacije ..... 25
Slika 4.3: Prijava preko zunanjega ponudnika upravljanja z identitetami – Facebooka. ..... 26
Slika 4.4: Prijava preko zunanjega ponudnika upravljanja z identitetami – Facebooka. ..... 27
Slika 4.5: Uporabnikova izbira vloge ..... 27
Slika 4.6: Obrazec za promocijo vsebine. Prikaz koledarja. ..... 31
Slika 4.7: Obrazec za promocijo vsebine. Prikaz koledarja. ..... 32
Slika 4.8: Domača stran uporabnika »Clicker« (tisti, ki deli vsebino). ..... 37
Slika 4.9: Ponovni prikaz deljenja želene objave ..... 40
Slika 4.10: Osnovni prikaz statistike objav ..... 43
Slika 4.11: Podrobnejša statistika posamezne objave ..... 44
Slika 4.12: Simulacija dvigovanja sredstev iz računa (leva stran) in nakazovanje kreditov oglaševalca (desna stran) ..... 46

## KAZALO KODE

Koda 4.1: Model uporabnika ..... 28
Koda 4.2: Dodatni metodi na modelu uporabnika ..... 29
Koda 4.3: Validacija obrazca in URL polja ..... 32
Koda 4.4: Model objave in njene komponente ..... 33
Koda 4.5: Pošiljanje zahtev na Facebook API s Python knjižnico requests ..... 35
Koda 4.6: Asinhrona funkcija za deaktivacijo objave ..... 35
Koda 4.7: Razred domače strani in metoda za prikaz relevantnih objav ..... 37
Koda 4.8: Spletna šablona posamezne objave ..... 38
Koda 4.9: Javascript funkcija z AJAX opravilom sprožena z deljenjem objave ..... 40
Koda 4.10: Funkcija, ki sprejme AJAX zahtevo za deljenje objav ..... 41
Koda 4.11: Razred DetailPostView in njegove metode, ki se nanašajo na vstop v stran za statistiko posamezne objave. ..... 45

# 1 UVOD

Prvotni namen Facebook uporabnikov je socializacija in ne potrošnja, vendar so ti vedno večja tarča oglaševalskih vsebin, ki se jim prikrito ali neprikrito ponujajo. Facebook je bil zastavljen kot družbeno omrežje, vendar je s pridobitvijo ogromne količine osebnih podatkov sedaj dveh milijard uporabnikov, ustvaril mogočen oglaševalski medij, ki jim je v letu 2016 ustvaril $ 26.885 milijard prihodka, kar je slabih 36,5 % več kot v letu 2015 (Statista2017). Po podatkih tržne raziskave spletne strani eMarketer (Hern, 2015) je povprečni uporabnik Facebooka leta 2014 doprinesel podjetju $ 12,76 dohodka od oglaševanja, za leto 2017 pa ocenjuje dvig na $ 17,50. Vprašanje je, zakaj vsaj del tega zneska ne konča v uporabnikovi denarnici, ki je zanj zaslužen? Ideja izdelane aplikacije je vsaj delni poskus decentralizacije moči Facebooka in ustvarjanje dodatnih možnosti za uporabnike.

Namen diplomske naloge je s pomočjo Facebook API-ja razviti spletno aplikacijo, ki bo pritegnila uporabnike Facebooka in jim v smislu nagrajevanja za aktivnost nudila koristi. Ciljna skupina bodo tisti uporabniki, ki bi bili radi za svojo vsakdanjo uporabo Facebooka nagrajeni in tisti neuveljavljeni posamezniki (ali mlada podjetja, organizacije ipd.), ki bi radi na moji aplikaciji odkrito promovirali svojo vsebino. Ostali uporabniki bi jo prikrito delili na Facebook, s čimer bi bila njihova vsebina v primerjavi z že preverjenimi oglaševalskimi orodji, še na drug način vidna. To je interaktivna aplikacija, ki deluje kot medij med oglaševalcem in ciljno javnostjo. Facebook uporabniku omogoča, da se na spletno platformo prijavi kot navadni uporabnik oziroma kot oglaševalec. Facebook uporabnik (organizacija, podjetje, posameznik – *freelancer*, *influencer*, za katerega ni nujno, da ima ustvarjeno Facebook stran), ki se torej prijavi kot oglaševalec, ima možnost promocije svojih Facebook objav (*post-ov*). Hkrati je navadnim uporabnikom, ki delujejo kot oglaševalski medij aplikacije, omogočeno *všečkanje* in *deljenje* teh objav, za kar bodo tudi nagrajeni – najprej za namen diplomske naloge z namišljeno valuto, ki jo morebiti kasneje, v prihodnjih verzijah lahko unovčijo v splošno veljavno plačilno sredstvo, tj. denar. Prav tako pa morajo oglaševalci za promocijo svoje vsebine plačati svoj delež glede na odziv (*všečke/deljenja*), ki so ga deležni, pri čemer pa se vrednoti tudi število Facebook prijateljev, ki ga uporabniki – delilci imajo, saj je tudi od tega odvisna vidnost deljene objave. Na ta način ima oglaševalec celostni vpogled v stopnjo odziva na njegove objave in njen organski doseg.

Zavedam se, da bi bilo pred javno uporabo sistema potrebno preveriti tudi pravne in etične vidike, vendar se v diplomski nalogi osredotočam zgolj na tehnični vidik. Moje raziskovalno vprašanje je torej: »Ali je možno v Python programskem jeziku s pomočjo Facebook API-ja

izdelati spletno aplikacijo, ki bo oglaševalcem omogočala promocijo, ostalim uporabnikom pa promoviranje njihove vsebine s pomočjo Facebook operacij *všečkanja* in *deljenja*?«.

# 2 TEORETIČNI PREGLED

## 2.1 Od spleta 1.0 do spleta 2.0

Ker moja aplikacija prevzema nekatere temeljne ideje Spleta 2.0 prav tako pa tehnologije, nastale v tem času, se je na tem mestu smiselno ustaviti na začetku. Leta 1992 Tim Berners-Lee na ideji medsebojno povezanih dokumentov hiperteksta oblikuje World Wide Web, ki ga danes poznamo kot Splet 1.0 (angl. Web 1.0). Strokovnjaki ga označujejo, kot splet »samo za branje«, saj je bila vloga povprečnega internetnega uporabnika omejena samo na branje informacij, ki so mu bile predstavljene. To je najbolje predstavilo milijon statičnih spletnih strani, ki so nastale med internetnim mehurčkom Dot-com v letih 1995–2001. Za njih je bilo značilno, da je imel administrator absolutni nadzor nad prikazanimi informacijami. Strani so bile oblikovane tako, da se jih je samo bralo – z odsotnostjo interakcij. Uporabniki so bili zgolj informacijski potrošniki. Po gostitvi prve Web 2.0 konference Tim O'Reilyja se je začel trend uporabe pojma Splet 2.0, ki označuje nekatere konceptualne spremembe in so se začele pri oblikovanju spletnih strani. Tehnološko gledano do revolucionarnih sprememb ni prišlo. Spremenila se je zgolj vloga uporabnika, ki je postal center informacij in njihov generator (Lorenzo-Romero idr. 2011, 42).

> Po O'Reilly (2005) je Web 2.0 *omrežje kot platforma, ki zajema vse povezane naprave; Web 2.0 aplikacije so tiste, ki najbolj izkoristijo prednosti te platforme; pomeni nudenje programske opreme kot stalno posodobljeno storitev, ki postaja boljša s številom uporabnikov, porabljanjem in združevanjem podatkov iz več virov, vključno s posameznimi uporabniki, medtem ko zagotavlja svoje lastne podatke in storitve v obliki, ki omogoča ustvarjanje mrežnih učinkov skozi arhitekturo sodelovanja in participacije ter presega statične strani Web 1.0 z namenom zagotavljanja bogatih uporabniških izkušenj*.

Strinjamo se lahko, da je Web 2.0 vpeljal številne novosti povezane s »socialno programsko« tehnologijo, kot so aktivno sodelovanje, uporabnik kot proizvajalec vsebin, opolnomočenje množic, bogatejša uporabniška izkušnja ipd., vendar je potrebno tukaj izpostaviti to, da so te tehnologije izpeljane iz standardov Web 1.0, zato lahko potemtakem sklepamo, da je Web 2.0 zgolj izpeljava oziroma posledica izpopolnjenega spleta (Anderson 2007, 6).

V nadaljevanju bom predstavil šest idej, ki temeljijo na Tim O'Reillyjevih konceptih in kasneje Andersonovih (2007) zamislih. To so ideje o ustvarjanju nečesa večjega od globalnega

informacijskega prostora – nekaj veliko bolj družbenega; ideje, katerih sodelovanje, prispevek in skupnost so glavne vrednote refleksivnega spleta, v katerem se čudni pojavi in tipologije producirajo s strani milijarde uporabnikov, tako na mikro kot na makro ravni.

1. Posameznikova proizvodnja vsebin

»Vedno sem si predstavljal informacijski prostor dostopen vsem, ne samo za brskanje, vendar tudi za ustvarjanje« (Tim Berners-Lee v Anderson 2007, 14). Danes živimo v bolj izpostavljeni kulturi, v kateri biti opažen pomeni veliko. Dostopnost dokaj poceni telefonov, fotoaparatov in kamer je prispevala k novinarstvu državljanov na kraju dogodka. Javnost ima danes večjo vlogo k prispevku vsebin, ki jih mediji uporabljajo. Že z nekaj kliki, lahko kdorkoli objavi svojo zgodbo, sliko ali video svojim prijateljem ali širšemu svetu preko spleta. Posamezniki so začeli pisati lastne bloge, združili moči pri pisanju wikijev; skratka ovira vstopa na internet se manjša, vključitev posameznikov k prispevku in manipulaciji informacij pa posledično vpliva na oblikovanje spleta, ki ga poznamo danes (Anderson 2007, 14–15).

2. Uporaba znanja množic

Hiperpovezave so temelj spleta. Z aktivno proizvodnjo novih vsebin in spletnih strani ter njihovo povezavo ene na drugo se organično veča mreža spletnih povezav. Eden prvih primerov tega je Yahoo, ki je nastal kot katalog oziroma seznam hiperpovezav, tj. skupek tisočih in kasneje milijon uporabnikov interneta. Tudi Googlov dosežek na področju iskanja PageRank, ki je temeljil na metodi uporabe hiperpovezav in ni upošteval samo karakteristike dokumentov hiperteksta, je prispeval k boljšemu rezultatu iskanja. Prav tako eBay in Amazon, ki sta postala uspešna prav zaradi kolektivnega prispevka aktivnih uporabnikov pri ocenjevanju produktov. Njuna stran temelji na viralnem marketingu oziroma propagandnemu priporočilu med uporabniki. O'Reilly najbolje ponazori znanje množice na primeru programskega orodja Cloudmark. To je sistem za filtriranje nezaželene pošte, ki zbira podatke uporabnikov elektronske pošte in njihove odločitve. Na podlagi teh podatkov ugotavlja, ali prispela pošta spada v zaželeno ali nezaželeno kategorijo. Ideja je torej ta, da uporabniki delujejo neodvisno, vendar skupaj pridejo do »pravega odgovora« (O'Reilly 2010, 230–232).

3. Podatki v ogromnih količinah

Podatkovno upravljanje je eno ključnih kompetenc Web 2.0 podjetij. V informacijski dobi uporabniki ter nekatere spletne aplikacije, tj. stranski produkt njihove uporabe, dnevno generirajo vse več podatkov, ki so uporabljeni za nadaljnje analize. Na primer Amazon lahko

iz zabeleženih nakupnih navad potrošnika sklepa na preference ostalih uporabnikov. Enako je pri prikazovanju oglasov, ki glede na zgodovino iskanja ciljajo na določeno občinstvo. Podatki so pogosto dostopni tudi razvijalcem. Lahko jih uporabijo in medsebojno kombinirajo, čemur pravimo *»mash-up«*. Prvi takšen primer je bila Paul Rademacherjeva stran HousingMaps.com, ki je združevala spletno kartografsko storitev Google Maps ter ameriški CraigsList, ki je tedaj služil oglaševanju nepremičnin. Danes *»mash-upe«* poznamo kot odprte API-je (angl. Application Programming Interfaces). Ti podatke dojemajo kot vire, ki lahko spremenijo svojo namembnost, se preoblikujejo in ponovno uporabijo, kar predstavlja neko vrsto osvoboditve podatkov. Tukaj Brown in Duguid (v O'Reilly, 2010) opozarjata na centralizacijo podatkov, odprto pa ostaja tudi vprašanje, kdo si lasti podatke? (Anderson 2007, 18–19).

4. Arhitektura sodelovanja

Pojem *arhitektura sodelovanja* izraža več kot le idejo sodelovanja in uporabnikovo produkcijo vsebin, saj enako težnjo namenja arhitekturi aplikacije oziroma storitve, kjer uporabnik prispeva. Enostavno povedano, aplikacije so načrtovane tako, da izboljšujejo in olajšujejo uporabnikovo sodelovanje, hkrati pa same postajajo uporabnejše. Aplikacija ima torej vlogo pametnega posrednika, ki povezuje uporabnike ter jih opolnomoči. Primer tega je BitTorrent, tj. decentralizirana mreža nalagalcev in sejalcev, ki nudijo svojo pasovno širino in podatke drugim uporabnikom. Več kot je uporabnikov, več virov je na voljo. Ključna ugotovitev in ideja Web 2.0 je, da s številom uporabnikov storitev postane boljša. Ta koncept bazira na ideji odprtokodne programske opreme, ki spodbuja k prispevku h kodi, vpeljevanju novih zamisli in izpostavitvi podatkov za ponovno uporabo (Anderson 2007, 19).

5. Učinek omrežja

Učinek omrežja je ekonomski termin, ki opisuje rast koristi obstoječih uporabnikov neke storitve, ko se poveča število novih uporabnikov, zaradi možnega medsebojnega delovanje – interakcije. Za primer Anderson (2007) navaja vključitev novega uporabnika v telefonsko omrežje. Ne samo, da ta uporabnik pridobi na koristi, saj pridobijo tudi obstoječi uporabniki, ki lahko sedaj na telefon dobijo novega uporabnika. Podobno je pri družabnih omrežjih, kot je Facebook, ko se začne učinek omrežja stopnjevati in se ljudje začnejo zavedati rasti priljubljenosti neke aplikacije ali storitve. Ta po navadi hitro pridobi na tržnem deležu, kar lahko vodi do vklenjenega položaja uporabnika, ki uporablja ta produkt. Primer tega so Microsoftovi Office, ki so postali standard izmenjave dokumentov. Zaradi števila uporabnikov,

ki uporablja ta produkt, je za posameznika težko preiti na drugi produkt, saj zaradi kompatibilnosti ne bi več morali medsebojno izmenjati datotek (Anderson 2007, 20–21).

6. Odprtost

Z razvojem spleta se je razvil širok razpon regulatornih pravnih, politični in kulturnih pravil, ki se nanašajo na dostopnost do in pravice za upravljanje digitalnih vsebin. Ne glede na to, prevladuje na spletu še vedno vsesplošna tradicija delovanja na odprtosti, ki jo vzpodbuja Web 2.0 tehnologija, za katero so značilna načela dela z odprtimi standardi, uporabe odprtokodnega programskega orodja, proste dostopne vsebine in njena ponovna uporaba ter odprtost do inovacij nasploh (Anderson 2007, 25).

Blogi, družbene novice, wikiji (spletna mesta z javnim dostopom), družbeno označevanje in deljenje, zaznamki, spletni forumi, podcasti, spletna mesta za socialna omrežja (angl. social networking sites – SNS), RSS-ji ter ostale številne spletne storitve oziroma aplikacije, ki so bile izumljene z razvojem tehnologije in odprtih protokolov najbolje predstavijo koncept Web 2.0.

### 2.2 Spletna družbena omrežja

Družbena omrežja, kot so Facebook, Twitter, Instagram, Quora ter velika množica ostalih socialnih omrežjih, katerih število registriranih uporabnikov lahko predstavimo v stotinah milijonov, so nedvomno integrirana v naša vsakdanja življenja in navade. Njihove tehnološke značilnosti ostajajo predvsem podobne, medtem ko je večji razkorak viden pri njihovih uporabnikih, interesih, kulturi ter ozračju, ki ga uporabniki soustvarijo. Večina družbenih omrežij stremi k ohranitvi že prej vzpostavljenih socialnih stikov, medtem ko druge pomagajo povezati neznance glede na njihove skupne interese, dejavnosti, politična ter verska prepričanja ipd. (Boyd in Ellison 2008, 210).

### 2.3 Definicija družbenih omrežij

Boyd in Ellison (2008) definirata družbeno omrežje kot »spletno storitev, ki omogoča posamezniku, da si znotraj sistema ustvari javni oziroma nejavni profil, sam upravlja s komer želi deliti poznanstvo in vezi ter ima pregled nad svojimi in tujimi povezavami znotraj sistema«. V svojem delu poudarjata, da je povezovanje (angl. *networking*) in spoznavanje neznancev ena izmed lastnosti družbenih omrežij, vendar je ključna lastnost družbenih omrežij ta, da omogočajo uporabnikom svobodo izražanja, prosto pot pri samopredstavitvi ter ohranjanje že obstoječih vezi. Tudi Lorenzo-Romero in ostali (2011) menijo podobno, saj so »spletna

družbena omrežja spletne aplikacije, ki omogočajo svojim uporabnikom generiranje javnega profila, deljenje informacij, sodelovanje pri ustvarjanju vsebin, spontano sodelovanje v socialnih dejavnostih ter izražanje javnega mnenja na decentraliziran način«. Po tej definiciji bi lahko tudi mojo aplikacijo opredelili kot družbeno omrežje, čeprav v primerjavi z drugimi ne vsebuje funkcionalnosti neposrednega sporočanja med uporabniki. Dejansko deluje bolj kot vmesnik oziroma podaljšek družbenega omrežja Facebook.

V samemu jedru družbenih omrežij je po navadi profil, ki je povezan z ostalimi uporabniki, tj. profili sistema. Profili so opisi posameznikov oziroma njihove značilnosti, s katerimi so prepoznani na omrežju. Med včlanitvijo v družbeno omrežje je posameznik primoran izpolniti serijo vprašanj, po navadi sestavljenih iz osebnih podatkov, iz katerih se kasneje generira profil. Seveda se stopnja modifikacije profila med različnimi družbenimi omrežji razlikuje, prav tako pa njegova vidljivost. Na primer na Facebooku, lahko uporabnik obogati svoj profil z različnimi multimedijskimi vsebinami in določi, kdo bo imel vpogled v to vsebino ter njegovo omrežje prijateljev (Boyd in Ellison 2008, 211).

Po včlanitvi v družbeno omrežje je posameznik pozvan, da identificira druge, s katerimi ima povezave. Ta zveza je mnogokrat obojestranska in potrebuje potrditev z obeh strani. Označujeta pa jo besedi, kot sta prijateljstvo in kontakt. Pogosta je tudi enosmerna relacija med uporabniki in vzorniki. Takrat govorimo o občudovalcih (angl. *fans*) in sledilcih (angl. *followers*). Ključna komponenta družbenih omrežij je prikaz teh zvez oziroma razmerij med uporabniki. Lista prijateljstva po navadi vsebuje povezave na prijateljeve profile, preko katerih potekajo potovanja po socialnih omrežjih. Večina družbenih omrežij ponuja tudi mehanizme, ki omogočajo uporabnikom medsebojno komunikacijo; po navadi se to dogaja v smislu »puščanja komentarjev« ali pa ima aplikacija že vgrajene naprednejše tehnologije privatnega sporočanja. Poleg profilov, prijateljev, komentiranja in privatnih obvestil pa se družbena omrežja razlikujejo še v drugih lastnostih, tj. predvsem v tipih uporabnikov. Nekatera družbena omrežja dopuščajo deljenje fotografij ali videov, druga imajo vgrajen način *bloganja* in privatno sporočanje. Vedno bolj se uporabljajo na mobilnih platformah, ciljne skupine so pa specifične starostne, geografske, jezikovne, etnične, politične in ostale skupine (Boyd in Ellison 2008, 212–213).

### 2.4 Družbeno omrežje Facebook

Leta 2004 je družbeno omrežje pod imenom »The Facebook« ustanovil 23 letni študent psihologije na Harvardu, Mark Zuckerberg. Najprej je bil namenjen študentom Harvarda kot

omrežje za akademske namene in že 24 ur po izidu je bilo registriranih 1200 uporabnikov. Uporaba omrežja se je razširila še na ostale ameriške univerze. Avgusta 2005 se je preimenoval in za $ 200.000 je bila kupljena domena Facebook.com. Septembra 2006 so omogočili dostop registracije v družbeno omrežje Facebook vsem veljavnim naslovom elektronske pošte in od takrat naprej deluje kot brezplačna aplikacija. Večino prihodkov ustvari preko oglaševanja, njen cilj pa je dati javnosti orodje za izgradnjo skupnosti in zbližanje sveta. Danes šteje 2, 01 milijard aktivnih uporabnikov mesečno, od katerih ga po podatkih iz junija 2017 dnevno uporablja 1.32 milijard ljudi, z namenom, da ostanejo povezani s prijatelji in družino, da ugotovijo, kaj se dogaja v svetu in delijo, tj. izrazijo stvari, za katere jim je mar (Phillips, 2007; Newsroom.fb, 2017).

### 2.5 Oglaševanje na Facebooku

V zadnjem desetletju sta Google in Facebook drastično spremenila masovni marketing. Leta 2000 je Google predstavil produkt AdWords, ki je tržnikom omogočil transparenten in zanesljivejši medij, ki zagotavlja še večji vpogled v potrošnikove navade. Z njim lahko tržniki zaobidejo tradicionalne medije, kot so televizija, radio in tisk ter se neposredno osredinijo na posameznike, ki iščejo določene produkte ali storitve. Facebook, ki je sprva ustvaril sistem za promocijo blagovnih znamk kar preko načela »od ust do ust«, začne omogočati tudi kampanjsko oglaševanje. Skratka oglaševalska industrija kot spreminjajoča komunikacija s potrošniki je iz množičnega komuniciranja prešla v neposrednejše, *mikro* osredotočanje potrošnikov. Poleg »*brandinga*« je postal pomemben odziv potrošnikov ter njihov organski stik z znamko v primerjavi s plačanim oglasom. Tržnikom so se odprle možnosti izvajanja bolj nadzorovanih, vplivnih in multidimenzionalnih kampanj. Splet kot medij je omogočil tudi merjenje odziva potrošnikov. Orodja za sledljivost in optimizacijo omogočajo podroben in skoraj trenutni prikaz odzivov na oglaševalsko komunikacijo, kar dopušča ne samo višjo stopnjo transparentnosti ampak tudi višjo stopnjo vpliva na potrošnikovo vedenje. Internet v primerjavi z ostalimi mediji ponuja tudi višji nivo interakcij in posledično pristnejšo, tj. bolj osebno izkušnjo. Ker imajo uporabniki najhitrejši dostop do informacij je internet postal informacijski medij in pomemben faktor pri prepričevanju k nakupu. Tržnikom ponuja pogled, kako doseči potencialnega kupca in ga spreobrniti k nakupu, pri čemer pa vedno večjo vlogo igra upoštevanje medsebojne povezanosti uporabnikov na socialnih omrežjih. Uporabniki pri nakupu upoštevajo mnenja in priporočila neznancev, *všečke* prijateljev na Facebooku ali pa jim Google enostavno priporoči vsebino glede na zgodovino iskanja in aktivnost njihovih prijateljev (Young 2014, 2–9).

Zaradi visokega dosega občinstva in interaktivne narave so družbena omrežja postala najpomembnejše orodje za oglaševalske aktivnosti. Večina podjetij uporablja družbeni profil kot srce svojih oglaševalskih kampanj, preko njega objavljajo vsebino in posebne ponudbe, vabijo uporabnike na dogodke, jih naprošajo za ocenitev produktov ali storitve ipd. Skratka družbene profile uporabljajo kot generatorje odgovorov na svoja raziskovalna vprašanja. Facebook specifično uporabljajo oglaševalci za promocijo lastne znamke preko Facebook Ads (podobno kot oglasna pasica), Apps (aplikacij, iger ipd.) in Facebook Pages (»*fan page*«) – spletna stran znotraj Facebooka namenjena predstavitvam posameznikov, podjetij ali organizacij, ki lahko vsebuje integrirane aplikacije, prostor za razprave ali celo spletno trgovino. Uporabniki Facebooka lahko postanejo privrženci določenega »*fan page*« in se z njim povežejo. Vsak »*fan page*« vsebuje tudi »*steno*«, na katero objavlja/-jo (pogosto tudi privrženci) vsebino (tekst, video, slike, ankete). Vsakdo se lahko potem na to vsebino odzove z nadaljnjim deljenjem, izražanjem svojega mnenja v komentarjih ali enostavnim *všečkom*. Večina tržnikov se objavljanja oglaševalskih vsebin poslužuje dnevno, da bi potrošnika ohranili aktivnega in zainteresiranega za njihov produkt oziroma storitev. Posledično se aktivnost potrošnikov s sodelovanjem poveča, kar privede do nenamernega posrednega oglaševanja uporabnikov samih, ki zagotovijo, da je oglaševalsko sporočilo vidno še ostalim. Na primer z objavo novih »superg« (športnih copatov) vzbudijo željo in potrebe svojih prijateljev (Muntinga in drugi 2012, 121–127; Rauschnabel in drugi 2012, 153). Po podobnem principu deluje tudi moja aplikacija, kjer uporabniki lahko delijo njim sorodne vsebine neposredno na svoj Facebook zid, pri čemer ne bo razvidno, da gre za oglaševano vsebino.

Prednost Facebooka v primerjavi z ostalimi ponudniki oglaševalskega prostora je ta, da ima o svojih uporabnikih največ osebnih in demografskih podatkov. Že ob registraciji profila vpiše uporabnik v polja podatke o starosti, spolu, lokaciji, zakonskemu stanu, poklicu ipd. Med samo uporabo spletne platforme se uporabnik včlanjuje v razne interesne skupine, postane oboževalec določenih »*fan pagov*«, s tem posledično sporoča svoja zanimanja in navade. Seveda se ti podatki nadalje uporabljajo za boljše določanje ciljne skupine oglaševalcev za svoje oglase (Damjan 2016, 36). Moja aplikacija temelji na včlanitvi z že ustvarjenim Facebook računom, zato nekatere osebne podatke tudi sam (seveda s strinjanjem uporabnikov) zapišem v bazo podatkov in jih po potrebi uporabljam, da bi izboljšal funkcionalnost same aplikacije.

## 2.6 Oglaševanje s pomočjo moje aplikacije

Oglaševalci vedno več sredstev namenjajo za promocijo svojih produktov na družbenih omrežjih, družbena omrežja pa neprestano razvijajo nova orodja za prikaz čedalje več oglaševalskih vsebin in spodbujajo uporabnike k potrošnji, pri čemer uporabljajo naše osebne podatke. Ljudje se ne zavedamo, da uporaba Facebooka ni brezplačna ampak se monetizacija izvede v smislu zagotavljanja podatkov. Jaron Lanier (2013) je mnenja, da je internet, kot demokratično orodje postal zbiralnik bogastva manjšine, ki ima v lasti »največje« računalnike, tj. zbiralnike podatkov. Pravi, da družbena omrežja pod pretvezo, da so zastonj, prikrito izkoriščajo uporabnike (glasbenik sam pa je bil bolj slišan samo ob plačani promociji svoje vsebine). Ideja moje aplikacije je vsaj delni poskus decentralizacije moči Facebooka in nagraditi uporabnika za njegovo uporabo. Aplikacija deluje na podoben način kot Facebook, toda oglaševalec za vsako svojo deljeno objavo na Facebook plača znesek, ki bo odvisen od števila Facebook prijateljev uporabnika, ki jo je delil. Posebnost tega sistema je, da oglaševalec ta znesek plača neposredno (brez posrednikov) delivcu. Uporabnik moje aplikacije deluje kot oglaševalski medij, ki bo s pomočjo orodja Facebook, natančneje njegovega API-ja s svojim delovanjem (*deljenjem* in *všečkanjem*), promoviral njemu všečne vsebine. Ta dva pojma se večkrat pojavljata med nalogami, hkrati pa sta temeljni operaciji moje aplikacije, zato ju bom zaradi jasnosti poskušal definirati. Pojma sta se začela uporabljati na Facebooku kot dejanji, s katerima vzpostavimo interakcijo z vsebino. *Všečkanje* je funkcionalnost, ki omogoča uporabnikom, da izrazijo svojo podporo do specifične tematike, objave, slike, statusa ali »*fan paga*«. *Deljenje* pa je dejanje, ki ima v primerjavi z *všečkanjem* večjo težo, saj uporabnik s klikom na gumb deli vsebino na svoj osebni profil (»Facebook zid«), kjer ga lahko vidi vnaprej določena publika.

Gre za idejni projekt, ki verjetno ne bo nikoli zaživel, toda zaradi namena diplomske naloge se transakcije izvajajo z namišljeno valuto brez kakršnekoli vrednosti.

## 2.7 Uporabniška izkušnja

Privabljanje novih uporabnikov ter njihova nadaljnja ohranitev pri uporabi aplikacije je tesno povezana z uporabniško izkušnjo, zato je na tem mestu smiselno povedati še nekaj o tem pojmu, ki se večkrat zamenjuje z uporabnostjo. Broz in Sulčič (2009) *uporabnost* (angl. *usability*) definirata kot: »kakovostno značilnost, ki izraža enostavnost uporabe vmesnikov za interakcijo med človekom in napravami«. Med te naprave spada vse: od vmesnikov programske opreme, spletnih rešitev do razporeditve tipk na tipkovnici mobilnega telefona ter skupek lastnosti

kuhinjskega mešalnika. Pojem *uporabniška izkušnja* (angl. *user experience* – UX) jasne in poenotene definicije nima, saj vsak avtor opredeli, kaj vse predstavlja. Sicer izhaja s področja uporabe računalniške tehnologije, znotraj domene interakcije med ljudmi in računalniki. Nielsen Norman Group, tj. ena vodilnih strokovnih skupin na področju kakovosti uporabe spletnih tehnologij uporablja naslednjo definicijo (v Broz in Sulčič, 2009): »Uporabniška izkušnja obsega vse vidike interakcije uporabnika s podjetjem, njegovimi storitvami in izdelki«. Broz in Sulčič (2009) ugotavljata, da lahko pojem uporabniške izkušnje v veliki meri opredelimo kot »psihološko in fenomenološko kategorijo, ki jo lahko opišemo kot doživeto kakovost in zadovoljstvo, ki jo je imel uporabnik v interakciji s sistemom, organizacijo, izdelki ali storitvami«. Vsekakor se pojma uporabnost in uporabniška izkušnja prepletata, saj se pri obeh srečujemo z interakcijo dveh kompleksnih sistemov uporabnikov in računalnikov. Uporabnost se po navadi razume kot sposobnost uporabnika, da z rečjo uspešno opravi nalogo, medtem ko uporabniška izkušnja označuje širši pogled ter zajema uporabnikovo celotno interakcijo z rečjo kot tudi njegove misli, počutja in zaznavanje, kar je rezultat te interakcije (Tullis in Albert 2013, 5).

Med razvijanjem produkta razvijalci večino časa namenijo čemu bo produkt namenjen in kaj bo počel. Mnogokrat je zanemarjeno vprašanje o uporabniški izkušnji oziroma na kakšen način produkt to počne. Uporabniška izkušnja se ne nanaša na jedro delovanja produkta ali storitve temveč na vprašanje, kako produkt ali storitev delujeta navzven, ko prideta v stik z uporabnikom. Ko nekdo vpraša, kakšen je občutek uporabe nekega produkta in ali je enostaven za uporabo, sprašuje po uporabniški izkušnji. Kadar in če večina uporabnikov pomisli na dizajn produkta, pomisli na estetsko privlačnost ali njegovo uporabnost, tj. ali produkt izvaja funkcije, ki so mu pripisane. Dizajn uporabniške izkušnje gleda preko funkcionalnosti (»Gumb sproži pričakovan odziv.«) in privlačnosti produkta (»Gumb je privlačne oblike in teksture.«) in se pogosto sprašuje še o njegovem kontekstu (»Ali je premajhen, za tako pomembno funkcijo? Ali je na pravem mestu v primerjavi z ostalimi gumbi, ki jih bo uporabnik uporabljal?«). Uporabniška izkušnja je bistvenega pomena za vse produkte in storitve. S kompleksnostjo produkta je še težje zagotoviti dobro uporabniško izkušnjo, zato je še posebej pri spletnih aplikacijah, kjer je uporabnik prepuščen samemu sebi brez navodil za uporabo, pomembno, da uporabnik ne obtiči in se v najslabšem primeru nikoli ne vrne na naše spletno mesto (Garrett 2011, 4–12).

Praksa razvoja učinkovite uporabniške izkušnje se imenuje uporabniško-orientiran dizajn (angl. *user-centered design*). Koncept je relativno enostaven – osredotoča se na uporabnika pri

vsakem koraku razvoja produkta, vendar je lahko njegova implikacija presenetljivo kompleksnejša. Vsaka uporabniška izkušnja bi morala biti rezultat razvijalčevih zavestnih odločitev. To pomeni, da morajo imeti razvijalci skozi celoten proces v mislih vsako možno uporabnikovo delovanje in razumeti njihova pričakovanja (Garrett 2011, 17).

Zavedam se pomembnosti vidika uporabniške izkušnje, zato sem med razvijanjem aplikacije poskušal delovati nepristransko. Postavil sem se v vlogo uporabnika, posledično pa sem dostikrat sprejemal odločitve z njegove perspektive. Kako dobro mi je uspelo zagotoviti uporabniško izkušnjo v tej točki, ne morem izmeriti, niti ni moj namen.

# 3 UPORABLJENA TEHNOLOGIJA

V tem poglavju bom predstavil tehnologije, ki sem jih uporabil pri izdelavi spletne aplikacije.

### 3.1 Python

Python je tolmačitveni interaktivni objektni programski jezik, ki ga je leta 1990 razvil Nizozemec Guido van Rossum. Python vsebuje module, razrede, dinamično tipizira podatkovne tipe, prestrega izjeme, samodejno upravlja s pomnilnikom, je prenosljiv med operacijskimi sistemi in temelji na strukturirani in objektno orientirani računalniško programski paradigmi. Zaradi enostavne sintakse in dinamičnega tipiziranja, kar pomeni, da tipa spremenljivke ni potrebno eksplicitno deklarirati oziroma se lahko sproti spreminja, velja Python za preprost in pogosto prototipski jezik, ki je pri izvajanju malce počasnejši od statično tipiziranih jezikov, kot so Java, C++ in C#. Python se pogosto uporablja kot skriptni jezik, velik poudarek pa ima na integraciji z drugimi programskimi jeziki ter orodji (docs.python, 2017). Izbral sem ga zato, ker imam z njim največ izkušenj, hkrati pa zadostuje vsem nalogam, ki sem jih zastavil.

### 3.2 Django

Django je brezplačno odprtokodno spletno ogrodje (angl. *web framework*), napisano v programskem jeziku Python. Uporablja se pri ustvarjanju kompleksnejših spletnih strani, ki temeljijo na podatkovni bazi. Spodbuja hitro razvijanje in čist, pragmatični dizajn. Vključuje veliko dodatkov, ki se lahko uporabijo pri pogostih spletno-razvojnih nalogah (avtentikacija, administracija vsebine – CMS, URL shema, abstraktni API za podatkovno bazo ipd.). Je varen, saj preprečuje napade z vrivanjem SQL stavkov, odporen je na križno izvajanje skriptov (angl. *cross-site scripting – XSS*), CSRF-je (angl. *cross-site request forgery*) in »*ugrabitve klikov*« (angl. *clickjacking*). Gre za vsestransko spletno ogrodje, saj ga podjetja, organizacije in vladni uradi uporabljajo v različne namene, tj. od sistemov za upravljanje vsebin, socialnih omrežij, do platform za prikazovanje in obdelavo podatkov. Je fleksibilen in lahko sočasno sprejme veliko število zahtev. Temelji na arhitekturi MVT (*Model-View-Template*), katere bistvo je objektno-relacijsko mapiranje (angl. *object-relational mapper – ORM*), ki deluje kot posrednik med podatkovnimi modeli (definirani kot Python razredi) in relacijsko podatkovno bazo (»*Model*«), sistem za pošiljanje podatkov in procesiranje HTTP zahtev s spletnimi šablonami (angl. *template*). To predstavlja »*View*« ter šablone same (»*Template*«). So definirane, kako bodo predstavljeni podatki (Django, 2017). Na podlagi navedenih razlogov ter lastnih izkušenj je bilo z njim ogrodje Django naslednja najbolj logična izbira.

### 3.3 HTML

HTML (angl. *HyperText Markup Language*) je označevalni jezik in osnoven gradnik spleta, saj opiše in definira vsebino spletne strani. »*HyperText*« se nanaša na povezave (angl. *links*) znotraj spletne strani ali med spletnimi stranmi, kar predstavlja temeljni aspekt spleta, »*markup*« pa anotira tekst, slike in ostale prikaze znotraj spletnega brskalnika. HTML »*markup*« vsebuje elemente oziroma značke (angl. *tags*), kot so <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, , , <img> in druge, s katerimi ustvarimo strukturo in semantično ureditev dokumenta (MDN, 2017).

### 3.4 CSS

CSS (angl. *Cascading Style Sheets*) so jezikovne stilske predloge, namenjene predstavitvi HTML ali XML dokumenta. To so predloge, ki določajo videz spletne strani, tj. pisavo, barve, velikost, razmak med elementi, animacije in ostale dekorativne značilnosti. Delujejo tako, da se sklicujejo na *id* (lahko ga ima eden element) ali *class* (lahko ga ima več elementov) atribute, katere dodajamo HTML elementom (MDN, 2017).

### 3.5 JavaScript in jQuery

JavaScript (JS) je objektni programski jezik, ki deluje v uporabnikovem brskalniku in dovoljuje implementacijo interaktivnih in kompleksnih stvari na spletni strani, tj. dinamični prikaz vsebine, upravljanje multimedije, animacija slik ter ostalih elementov ipd. Vzporedno z JavaScriptom se pogosto uporablja jQuery, tj. majhna a hitra JavaScript knjižnica, ki se osredotoča na poenostavitev izvajanja DOM (angl. *Document Object Model*) manipulacije, AJAX (angl. *asynchronous JavaScript and XML*) klicev in opravljanja dogodkov (angl. *event handling*) (MDN, 2017).

### 3.6 AJAX

AJAX (angl. *asynchronous JavaScript and XML*) termin skovan leta 2005. Skoval ga je Jesse James Garrett in opisuje »nov« pristop za izgradnjo interaktivnih ter vsebinsko bogatih spletnih strani. Je skupek vseh zgoraj omenjenih tehnologij, ki skupaj poskrbijo, da se vsebina ali izgled spletne strani posodobi, brez da bi bilo potrebno osvežiti stran. Izmenjava podatkov se izvede asinhrono (v ozadju) preko strežnika, ki lahko pošlje podatke v brskalnik, brez da bi vplival na izgled in obnašanje trenutne strani. Podatki se izmenjujejo v XML (angl. *Extensible Markup Language*) ali JSON (angl. *JavaScript Object Notation*) formatu (MDN, 2017).

### 3.7 SQLite in SQL

SQLite je odprtokodna knjižnica in hkrati samozadnostni pogon (angl. *engine*) SQL relacijske podatkovne baze, ki ne potrebuje ločenega strežniškega procesa, saj neposredno bere in zapisuje informacije v običajno datoteko, kjer so shranjeni tudi ostali metapodatki podatkovne baze. Bazira na povpraševalnem jeziku SQL (angl. *Structured Query Language*), ki služi shranjevanju, manipulaciji, pridobivanju in nasploh delu na podatkih v podatkovni bazi (Sqlite, 2017; W3schools, 2017). Ne potrebujem celostnih rešitev, ki jih implementirajo strežniški sistemi, kot so MySQL, Oracle in PostgreSQL, zato za moje potrebe zadošča že SQLite podatkovna baza, ki jo ogrodje Django uporablja že samodejno.

### 3.8 Facebook API

Facebook API (angl. *Application Programming Interface*) je platforma za izgradnjo aplikacij, ki je na voljo uporabnikom socialnega omrežja Facebook. API omogoča aplikacijam, da lahko uporabljajo informacije Facebook profilov in njihove socialne povezave za objavljanje aktivnosti na zbiralnik novic (angl. *news feed*) in Facebook page (»*fan page*«) z uporabnikovo vednostjo in njegovim strinjanjem. Aplikacije se lahko s pomočjo dostopa (preko API-ja) do profilov, prijateljev, »*fan pagov*«, skupin, fotografij in dogodkov, obogatijo s socialnim kontekstom. Facebook trenutno ponuja tri vrste programskih vmesnikov: Graph API, Atlas API in Marketing API. Atlas API je zgrajen na temeljih Graph API-ja in omogoča programatično ustvarjanje in upravljanje oglaševalskih kampanj, generacijo oglaševalskih poročil, dostop do rezultatov kampanj ipd. Marketing API ponuja dostop do oglaševalskih funkcionalnosti, ki so sicer dostopne preko Facebook Ads platforme, vendar omogoča večjo stopnjo prilagoditve in mero kompleksnosti. Za nas najbolj aktualen je Graph API, tj. jedro platforme, preko katerega lahko pridobimo (beremo) in vnašamo (pišemo) podatke v Facebookovo bazo podatkov. Je nižji HTTP baziran API, preko katerega programatično poizvedujemo o podatkih, objavljamo nove vsebine in fotografije, opravljamo z oglasi in opravljamo ostale funkcije, ki so implementirane v sami aplikaciji. Deluje po arhitekturi REST (angl. *Representational State Transfer*), ki omogoča hitro povezavo med napravami preko HTTP protokola, s pomočjo katerega izvaja vse štiri CRUD (angl. *Create/Read/Update/Delete*) operacije; ustvari, beri, posodobi, odstrani. Kot odgovor poizvedb dobimo povratne podatke v JSON (angl. *JavaScript Object Notation*) formatu (developers.facebook, 2017).

# 4. IZDELAVA SPLETNE APLIKACIJE

V nadaljevanju bom postopoma predstavil celotni proces razvoja spletne aplikacije, pri čemer se bom osredotočil na najpomembnejše osnove, reševanje pogostih problemov, implementacijo najboljših praks in odločitve, ki sem jih moral med delom sprejeti.

## 4.1 Priprava delovnega okolja

Vse se začne s programski jezikom Python, ki mora v okolju Windows pod spremenljivkami okolja (angl. *Environment Variables*) imeti pravilno definirano pot, da lahko preko Windows konzole prikličemo Python interpretor. Z namestitvijo Pythona sem poleg mnogih že vključenih

standardnih knjižnic namestil tudi lažjo SQLite podatkovno bazo, ki sem jo uporabil v razvoju. Ta se v času migracije na produkcijski strežnik po navadi zamenja z naprednejšo PostgreSQL ali MySQL podatkovno bazo. S Pythonom sem dobil tudi orodje za inštalacijo Python paketov Pip, preko katerega sem namestil številne manjše pakete in knjižnice (glej sliko 4.1), ki so mi olajšale delo. Zaradi praktičnosti in urejenosti sem za kreacijo izoliranega okolja uporabil paket virtualenv, ki mi je omogočil, da sem ločil lokalno nameščene Python pakete od paketov, potrebnih za projekt diplomske naloge. Na ta način ni prišlo do konfliktov med verzijami, laže pa je tudi nadaljnje opravljanje in nadziranje samega projekta. Preko Pipa, tj. preko terminala, sem namestil tudi ogrodje Django:

```
pip install Django
```

Na enak način sem namestil še ostale pakete in knjižnice, ki so razvidni na spodnji sliki (glej sliko 4.1). Njihov namen uporabe bom pojasnil v nadaljevanju.

Slika 4.1: Seznam paketov nameščenih z orodjem Pip

```
defusedxml==0.4.1
Django==1.10.3
django-allauth==0.28.0
django-countries==4.0
django-embed-video==1.1.0
django-filter==0.15.3
Markdown==2.6.7
oauthlib==2.0.0
python3-openid==3.0.10
requests==2.11.1
requests-oauthlib==0.7.0
django-el-pagination==3.0.1
django-autocomplete-light==3.2.1
celery==3.1.25
```

Z namestitvijo Djanga sem z naslednjim ukazom ustvaril začetno arhitekturo projekta:

```
django-admin startproject rest_api
```

Ta ukaz mi je postavil sledečo podatkovno strukturo – rest_api ime mojega projekta:

```
rest_api/
    manage.py
    rest_api/
        __init__.py
        settings.py
```

```
        urls.py
        wsgi.py
```

Posamične datoteke so sledeče:

- `manage.py` je orodje za interakcijo s projektom in izvajanje specifičnih Django nalog.
- `__init__.py` je prazna datoteka, ki pove Pythonu, naj obravnava mapo rest_api kot modul.
- `settings.py` je datoteka, kjer so shranjene nastavitve in konfiguracije projekta.
- `urls.py` je datoteka, kjer je definirana URL shema.
- `uwsgi.py` je datoteka, s katero se poganja projekt kot WSGI aplikacija (Mele 2015, 4).

Pri ogrodju Django je pomembno razločevanje med pojmoma projekt in aplikacija. Zgoraj ustvarjeni projekt predstavlja začetno namestitev z nastavitvami, medtem ko aplikacija predstavlja skupek modelov (angl. *model*), pogledov (angl. *view*), šablon (angl. *template*) ter URL-jev. Aplikacija komunicira z ogrodjem, da zagotovi specifične funkcionalnosti in je lahko ponovno uporabljena v več projektih. Na projekt lahko gledamo kot na spletno stran, ki vsebuje več aplikacij, kot so blogi, wikiji in forumi, ki jih lahko uporabimo v drugem projektu (Mele 2015, 4).

Z naslednjim ukazom ustvarimo osnovno strukturo aplikacije:

```
python manage.py startapp accounts
```

S tem ukazom sem ustvaril dve aplikaciji `posts` in `accounts`. Osnovna struktura aplikacije `accounts` izgleda sledeče:

```
accounts/
    __init__.py
    admin.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

Posamične datoteke so sledeče:

- `admin.py` je datoteka, kjer lahko registriraš svoje modele, da bodo vključeni v Djangovem administrativnem sistemu za upravljanje vsebin – ta je izbiren.
- `migrations` je mapa, ki vsebuje aplikacijske migracije podatkovne baze. Migracije omogočajo Djangu, slediti spremembam modelov in jih sinhronizira s podatkovno bazo.
- `models.py` je datoteka, kjer definiramo modele aplikacije. Model je edini dokončen vir informacij o naših podatkih. Vsebuje bistvena polja (angl. *fields*) in obnašanje podatkov, ki jih hranimo. Po navadi eden model predstavlja eno tabelo v podatkovni bazi.
- `tests.py` je datoteka, kjer lahko pišemo teste za preverjanje delovanje aplikacij.
- `views.py` je datoteka, kjer obstaja logika aplikacije. View prejema HTTP zahteve, jih sprocesira in vrne odziv (Mele 2015, 8).

### 4.2 Avtentikacija in avtorizacija s pomočjo družbenega omrežja Facebook

Upravljanje z identitetami (angl. *Identity Management*) je sistem, ki v računalništvu opisuje upravljanje posameznih uporabnikov, preverjanje njihove pristnosti, dovoljenj in privilegijev znotraj sistema ali zunaj njegovih meja, pri čemer je cilj sistema čim bolj avtomatizirati procese, zagotavljanje boljše varnosti, zmanjšanje ponavljajočih se opravil, poenostaviti upravljanje in sledljivost identitet ter posledično izboljšati storitev nasploh (s poenostavljenim načinom prijave, z možnostjo ponastavitev gesel ipd.) (Leber 2014, 2–3).

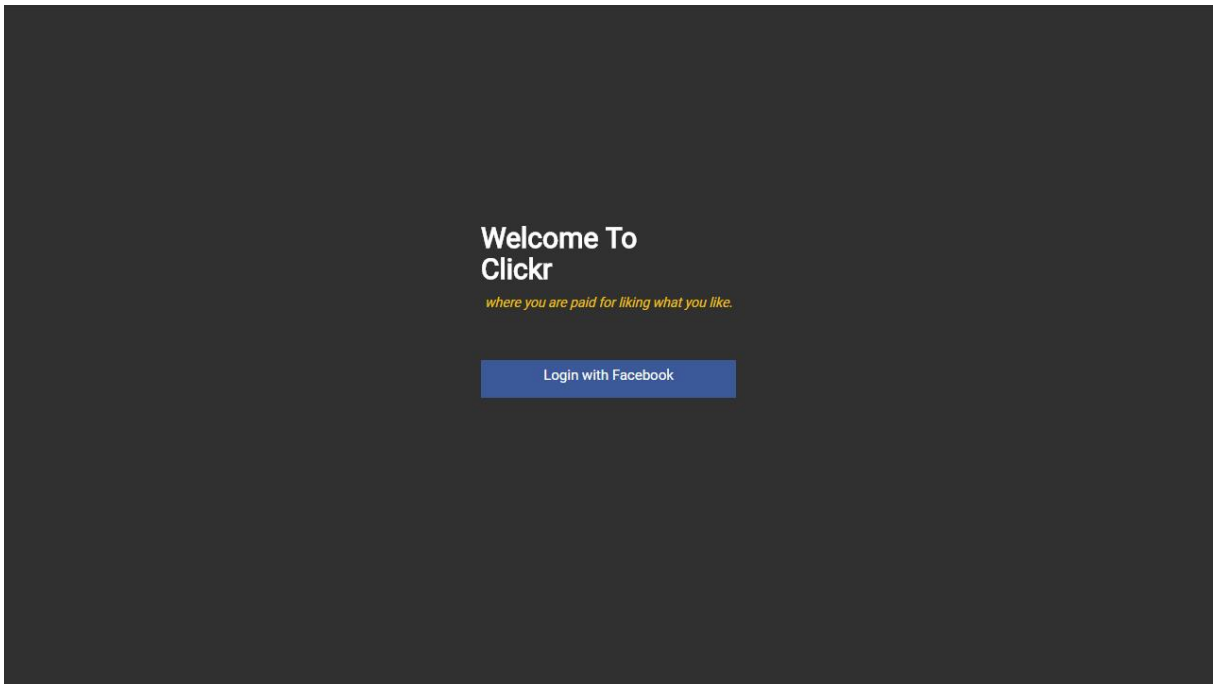
Za uporabo moje aplikacije obvezen Facebook uporabniški račun, zato sem način prijave omogočil samo preko zunanjega ponudnika upravljanja z identitetami, tj. Facebooka. Postopek registracije oziroma prijave poteka tako, da se uporabnik odjemalec prijavi v družbeno omrežje Facebook in pooblasti mojo aplikacijo, da ima pravice za dostop do njegovih privatnih elementov, kot so informacije o osebnih podatkih, prijateljih, albumih ipd. Nabor teh podatkov je seveda odvisen od pravic, ki jih navedemo pri vključitvi v družbeno omrežje in strinjanja uporabnika, ki jih odobri. Vse skupaj se dogaja preko odprtega standarda za avtentikacijo in avtorizacijo OAuth (angl. *Open Authorization*).

Tradicionalni model avtentikacije poteka med odjemalcem in strežnikom, ko uporabnik uporabi svoje poverilnice (uporabniško ime in geslo) za dostop do resursov na strešniku. OAuth temu modelu doda še vlogo lastnika resursov. OAuth odjemalcu (aplikaciji) zagotavlja metodo za dostop do resursov na strežniku (Facebooka) v imenu lastnika resursov (uporabnika), brez

vnosa uporabniškega imena in gesla. Odjemalec torej deluje v imenu lastnika resursov, ki poda zahtevo po resursih, OAuth pa strežniku omogoča, da avtorizira lastnika resursov, kot tudi identiteto odjemalca. Ko odjemalec prejme pravice od lastnika resursov in je pravilno identificiran s strani strežnika, ta prejme žeton (angl. token) s časovno omejeno veljavnostjo, s katerim lahko nato dostopa do podatkov, ki so v lasti lastnika resursov (Eržen ,2012).

Ker v programiranju obstaja zlato pravilo »neizumljanja tople vode« sem si pri celotnem procesu avtentikacije pomagal z manjšo Django aplikacijo django-allauth, ki omogoča združitev računov več družbenih omrežij z že vgrajenim Django-avtentikacijskem sistemom, omogoča lažjo konfiguracijo Facebook Appa z našo spletno aplikacijo, saj se potrebovani App Secret ključ in Client ID vneseta kar v avtomatsko ustvarjeno tabelo v bazi, samodejno pa posodablja tudi dovolilne žetone za nemoteno komunikacijo s Facebook API-jem.

Slika 4.2: Avtentikacijska stran aplikacije

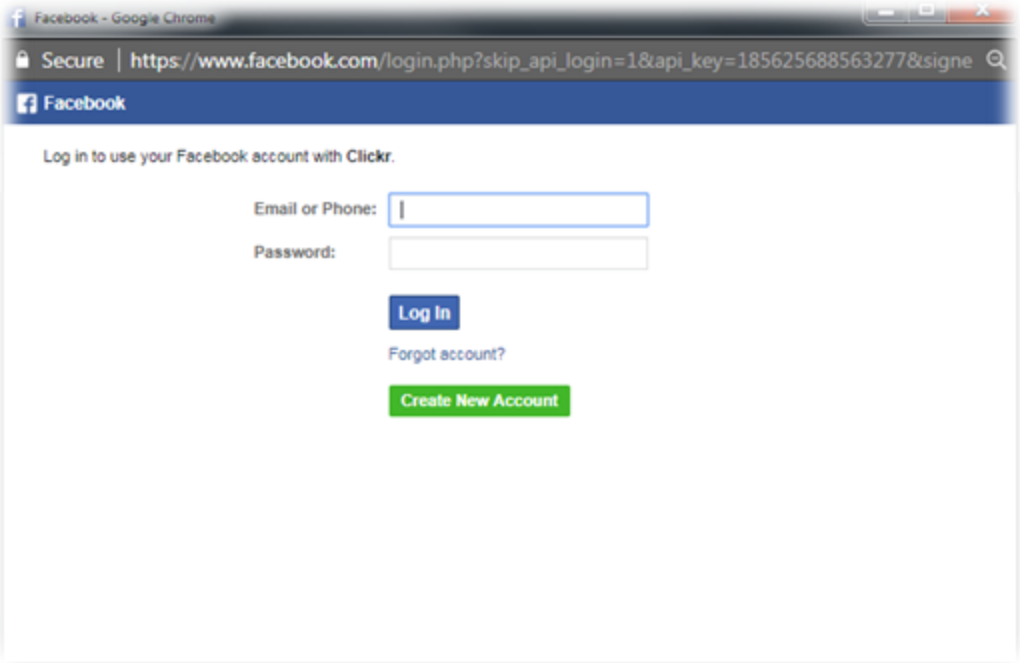



S slike 4.2 lahko vidimo, da je površje spletne aplikacije zelo preprosto, ponazorjeno s čim manj motečimi elementi. Kratek stavek predstavlja bistvo, cilj in namen same aplikacije, na sredini pa je gumb za hkrati vpis in registracijo. To pomeni, da če je uporabnik že registriran, potem ga gumb preusmeri na domačo stran (angl. *Home Page*), v primeru prvega obiska, pa se izvede sledeči proces:

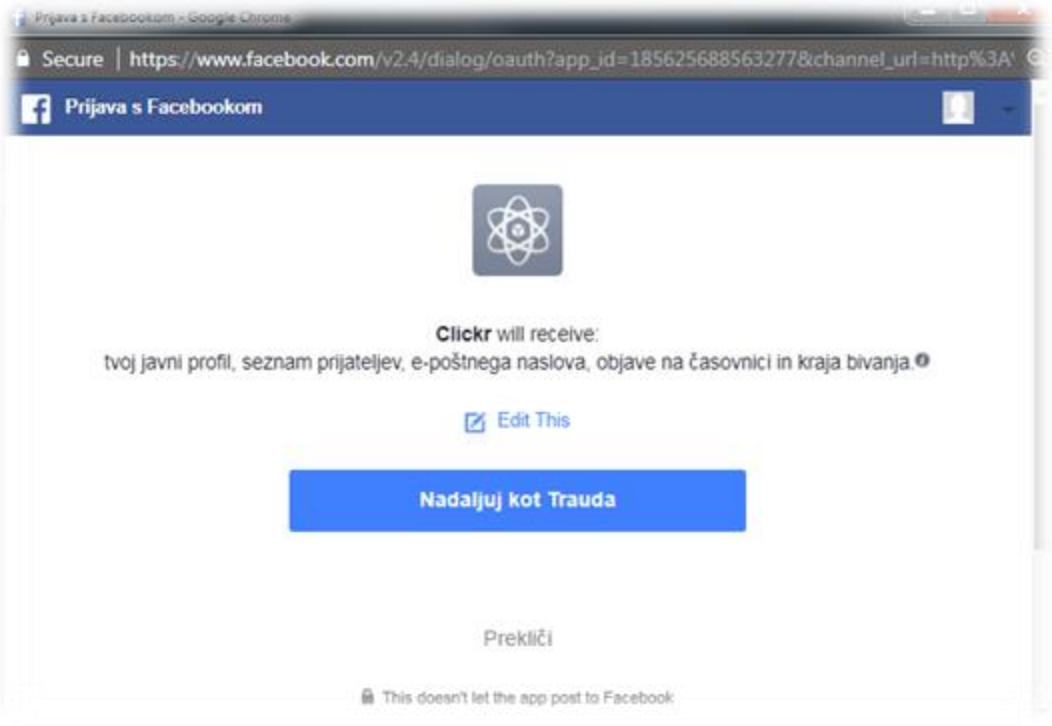
1. Uporabniku se pojavi obrazec za prijavo na družbeno omrežje Facebook (glej sliko 4.3).

2. Ko se pravilno identificira, ga Facebook opozori, da bo s strinjanjem omogočil aplikaciji dostop do nekaterih njegovih privatnih elementov. V mojem primeru do javno dostopnih podatkov profila, seznama njegovih prijateljev, e-poštnega naslova, kraja bivanja ter pravice do objave vsebine na časovnico (glej sliko 4.4).
3. V zadnjem koraku bo primoran izbrati še vlogo, ki jo bo igral med uporabo aplikacije. Registrira se lahko kot »Booster«, uporabnik, ki promovira svojo vsebino ali kot »Clicker«, tisti, ki deli vsebine na svojo Facebook časovnico (glej sliko 4.5), pri čemer je pomembno, da ima uporabnik lahko samo eno vlogo, ki jo ne more spremeniti.

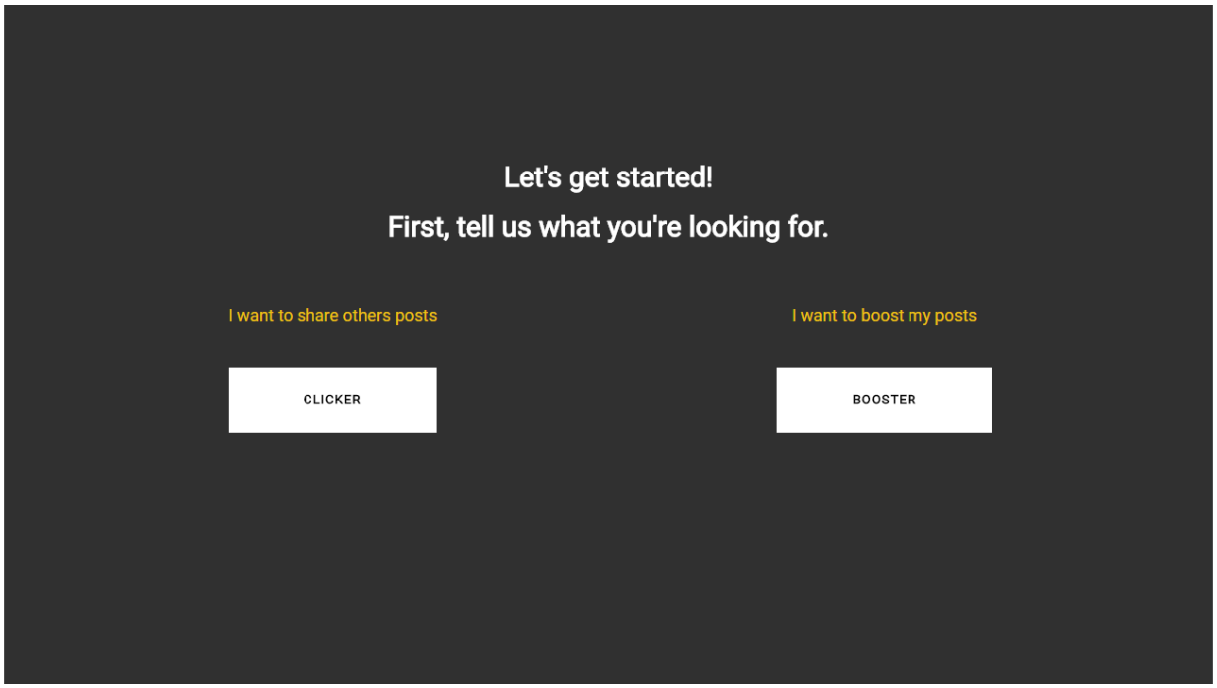
Slika 4.3: Prijava preko zunanjega ponudnika upravljanja z identitetami – Facebooka



Slika 4.4: Prijava preko zunanjega ponudnika upravljanja z identitetami – Facebooka




Slika 4.5: Uporabnikova izbira vloge



### 4.3 Model uporabnika

Projekt sem ločil na dve Django aplikaciji. Ena izmed njiju je aplikacija `accounts`, kjer sem v datoteki `models.py` definiral model (tabelo v podatkovni bazi) svojega uporabnika. Model `MyUser` izlgeda sledeče (glej kodo 4.1):

Koda 4.1: Model uporabnika

```
class MyUser(AbstractUser):

    roles = (
        ('UNDEFINED', 'Undefined'),
        ('CLICKER', 'Clicker'),
        ('BOOSTER', 'Booster'),
    )

    gender = (
        ('UNKNOWN', 'Unknown'),
        ('MALE', 'Male'),
        ('FEMALE', 'Female'),
    )

    role = models.CharField(max_length=255, choices=roles, default='UNDEFINED')
    gender = models.CharField(max_length=255, choices=gender, default='UNKNOWN')
```

```
    country = CountryField(blank=True, null=True)
    credit = models.FloatField(default=0.0)

    @property
    def is_undefined(self):
        return self.role == 'UNDEFINED'

    @property
    def is_clicker(self):
        return self.role == 'CLICKER'

    @property
    def is_booster(self):
        return self.role == 'BOOSTER'
```

Razred ali model `MyUser` je uporabljen za izdelavo objektov, tj. uporabnikov. Sestoji iz Djangovega nadrazreda `AbstractUser`, od katerega podeduje komponente in metode. `AbstractUser` že vsebuje komponente, kot so uporabniško ime, geslo, ime, priimek, e-poštni naslov, datum in čas registracije ter zadnjega vpisa, dovolilnice ipd., z modelom `MyUser` pa smo mu dodali še vlogo, spol, državo in dobroimetje. Prej omenjena Djangova knjižnica django-allauth je samodejno ustvarila model `SocialAccount`, ki je v relaciji »ena proti ena«, s tujim ključem povezan z mojim modelom `MyUser`. V modelu `SocialAccount` so shranjeni podatki pridobljeni preko Facebookovega API-ja ob registraciji; to so uid oziroma identifikacijska številka uporabnika moje aplikacije, datum in čas registracije ter zadnjega vpisa, ter polje (angl. *field*) `extra_data`, v katerem so v JSON formatu navedeni spol, ime, priimek, e-poštni naslov, seznam prijateljev, ki tudi uporabljajo to aplikacijo, število vseh Facebook prijateljev, podatki o lokaciji ipd. Django knjižnica django-allauth ustvari tudi model SocialToken, ki je relacijsko »ena proti ena«, s tujim ključen povezan z modelom `SocialAccount`. V njem je za vsakega uporabnika shranjen žeton, ki ga je potrebno dodati prošnji na Facebook API vsakič, ko želimo v imenu uporabnika brati, spremeniti ali zapisati kaj novega.

Koda 4.2: Dodatni metodi na modelu uporabnika

```
@receiver(pre_save, sender=MyUser)
def update_username_from_email(sender, instance, **kwargs):
    instance.username = instance.email


@receiver(user_signed_up)
def set_fb_uid(sender, **kwargs):
    user = kwargs.pop('user')
    extra_data = user.socialaccount_set.filter(provider='facebook')[0].extra_data
    try:
        country = extra_data['location']['location']['country']
    except KeyError:
```

```
        country = ""
    gender = extra_data['gender']

    if gender == 'male':
        user.gender = 'MALE'
    elif gender == 'female':
        user.gender = 'FEMALE'
    if country:
        country_code = countries.by_name(country)
        user.country = country_code
    user.save()
```

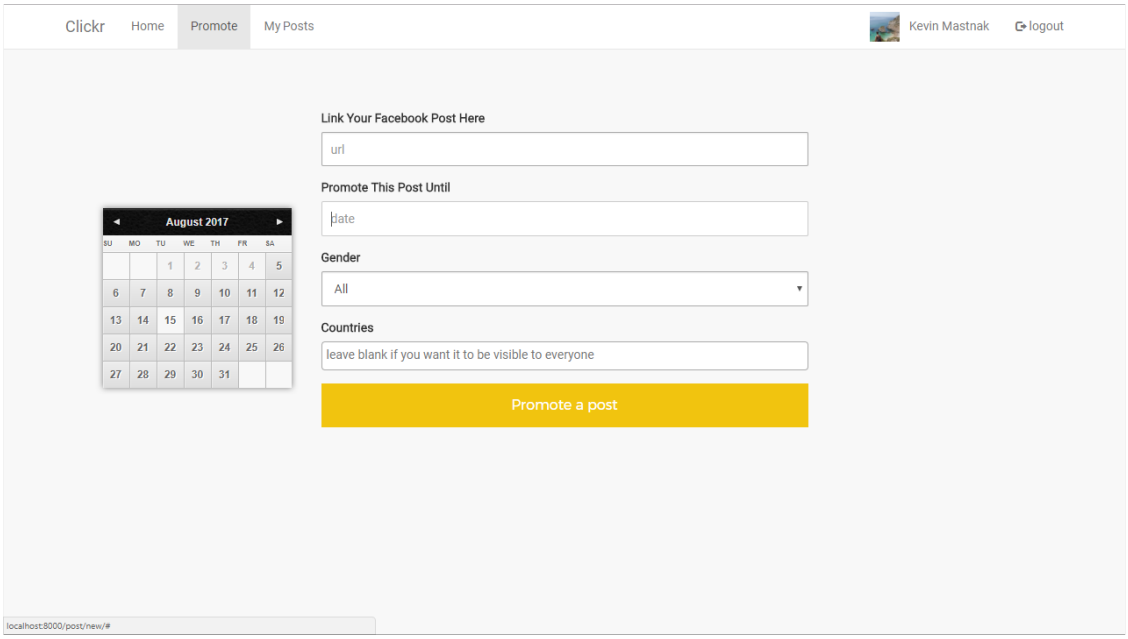
Modelu `MyUser` sem dodelil še dodatni metodi, ki se sprožita ob registraciji uporabnika (glej kodo 4.2). Prva metoda enostavno nastavi uporabnikov e-poštni naslov za uporabniško ime, druga pa iz tabele `SocialAccount` pridobi `extra_data` iz katerega izlušči uporabnikov spol ter njegovo državo bivanja, nakar dobljeno pripiše uporabniku. Res je, da imam sedaj v tabelah `MyUser` in `SocialAccount` podvojene podatke, vendar sem ocenil, da je zaradi preglednosti in krajših poizvedb to boljše. V nasprotnem primeru bi moral vsakokrat iz tabele `MyUser`, preko primarnega ključa dostopati do tabele `SocialAccount` in njenega polja `extra_field`, kjer bi vedno znova razčlenjeval podatke v JSON obliki, da bi izluščil željenega.

### 4.4 Promocija vsebine

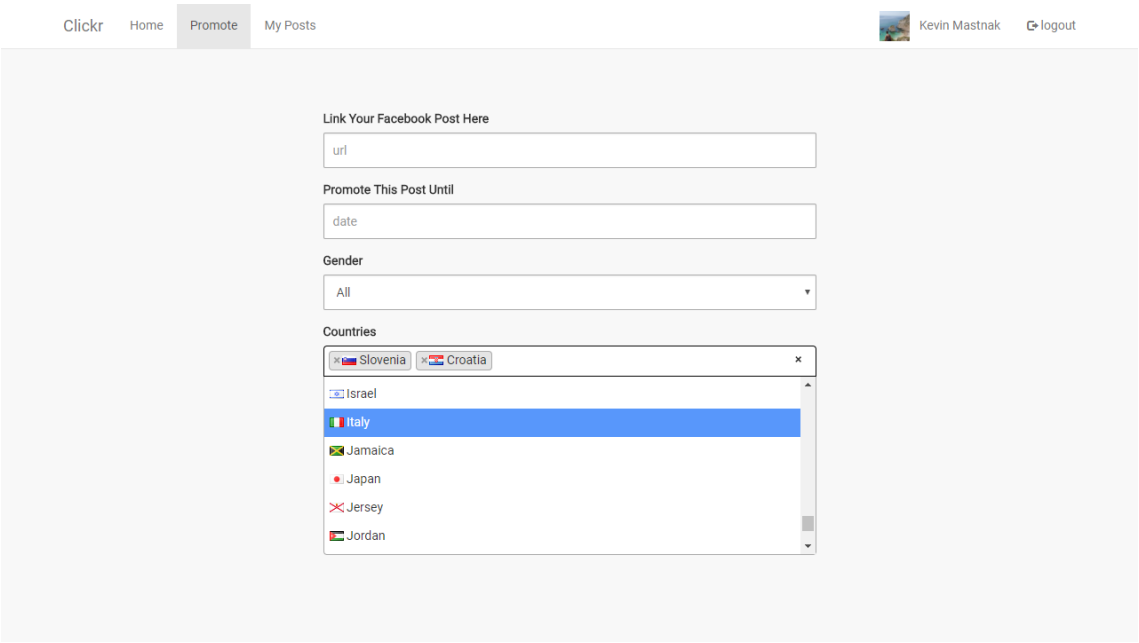
Uporabnik, ki se je v aplikacijo včlanil kot »Booster« ima na voljo promoviranje svojih vsebin iz Facebooka. To naredi preko spodnjega obrazca (glej sliko 4.6). V polje »url« kopira URL povezavo svoje objave iz Facebooka. Zavedam se, da bi lahko objavo ustvaril v sami aplikaciji, vendar sem se zaradi izgleda in vseh funkcionalnosti (»smeškov« in nalepk, »taganja« prijateljev, nastavitve vidnosti objave, pripisovanje lokacije, deljenje videov, slik, albumov ipd.), ki jih lahko vključiš v svoje Facebook objave, odločil, da ne bom reproduciral Facebook forme za objavo vsebine, ampak bom prisilil uporabnika, da dejanje izvede na Facebooku, kjer bo imel večji nadzor nad izgledom objave same, aplikaciji pa posreduje samo URL povezavo ustvarjene objave. Ta odločitev ne pripomore ravno k izboljšanju uporabniške izkušnje, vendar bodo imeli uporabniki s tem večji nadzor nad objavo. V obrazcu se izbere tudi datum, do katerega naj bi bila objava aktivna oziroma vidna ostalim uporabnikom (med tem obdobjem se lahko tudi ročno deaktivira in ponovno aktivira). Za kreacijo koledarja sem uporabil jQuery UI

knjižnico Datepicker, za izgled katerega sem uporabil razne CSS stile. Uporabnik ima na voljo določiti ciljno publiko glede na spol. Postavljanje ciljne starostne skupine sem tukaj opustil, saj bi za zbiranje natančne starosti uporabnikov dolgoval Facebooku dodatno obrazložitev, zakaj bi to potreboval, poleg tega pa je cilj same aplikacije deljenje vsebin Facebook prijateljem, ki so lahko različne starosti. Zadnje polje obrazca predstavlja možen nabor držav, iz katerih uporabniki bodo lahko objavo videli (glej sliko 4.7). Za prikaz in ostale funkcionalnosti (izbor več držav, iskalnik, sprotno AJAX nalaganje držav ob koleščkanju miške ipd.) polja sem uporabil Django knjižnico django-autocomplete-light, ki temelji na jQuery knjižnici Select2. Seznam držav, shranjenih v tabelo in sličice njihovih zastav sem prevzel iz knjižnice django-countries.

Slika 4.6: Obrazec za promocijo vsebine – prikaz koledarja



Slika 4.7: Obrazec za promocijo vsebine. Prikaz koledarja.




S klikom na gumb »Promote a post« se izvede validacija obrazca in njegovih polj. Obrazec javi napako, če nista izpolnjena polja `url` in `date`, ob njuni potrditvi pa se dodatno podrobno preverja še `url` (glej kodo 4.3). Za to sem uporabil Pythonov paket `urllib` in njegov modul `parse`, ki definira standardiziran vmesnik za razčlenitev URL (angl. *Uniform Resource Locator*) nizov na komponente. S funkcijo `urlparse` sem torej iz obrazca dobljeni URL razčlenil na naslovljeno shemo, telekomunikacijski omrežni naslov (angl. *Network Location*), pot ter na poizvedbo, z namenom lažje verifikacije. Pod pogoji, da URL naslov prihajal iz zavarovane različice HTTPS, ki uporablja SSL (angl. *Secure Sockets Layer*) in TLS (angl. *Transport Layer Security*), da šifrira in s tem zaščiti promet pred vmesnimi opazovalci, da prihaja iz omrežnega naslova »www.facebook.com« ter vsebuje nekatere ključne besede, hkrati pa ima pozitivno dobroimetje, da lahko objavo promovira, potem je obrazec potrjen, uporabnika pa uspešno naslovi na domačo stran. V nasprotnem primeru mu nad obrazcem izpiše opozorilno sporočilo.

Koda 4.3: Validacija obrazca in URL polja

```
def clean(self):
    url = self.cleaned data.get('url')
    if url:
        parsed = urllib.parse.urlparse(url)
        netloc = parsed.netloc
         scheme = parsed.scheme
        if (scheme == "https") and (netloc == "www.facebook.com") and ("story fbid"
and "id") in parsed.query or "videos" in parsed.path or "photo" in parsed.path or
"posts" in parsed.path:
            if self.user.credit > 0:
```

```
                return self.cleaned_data
            else:
                raise forms.ValidationError("You don't have enough credit to
promote your post!")
        else:
            raise forms.ValidationError("This is not the right Facebook link!")
```

### 4.5 Model objave

Po uspešno zaključenem obrazcu se podatki shranijo v model objave - `Post` (glej kodo 4.4), ki je definiran v Django aplikaciji `posts`, v datoteki `models.py`.

Koda 4.4: Model objave in njene komponente

```
POST_STATUS = (
    (0, 'Active'),
    (1, 'Deactivated')
)

GENDER_TARGETED = (
    (0, 'All'),
    (1, 'Male'),
    (2, 'Female')
)

class Post(models.Model):
    url = models.CharField(max_length=2000)
    fb_post_id = models.IntegerField(blank=True, null=True)
    fb_user_id = models.IntegerField(blank=True, null=True)
    user_post_id = models.CharField(max_length=255, blank=True, null=True)
    post_starts = models.DateTimeField(default=datetime.datetime.now)
```

```
    post_ends = models.DateField(blank=True, null=True)
    post_status = models.IntegerField(default=0, choices=POST_STATUS)
    message = models.CharField(max_length=4000, blank=True, null=True)
    name = models.CharField(max_length=255, blank=True, null=True)
    description = models.CharField(max_length=255, blank=True, null=True)
    caption = models.CharField(max_length=255, blank=True, null=True)
    picture_url = models.CharField(max_length=2000, blank=True, null=True)
    video_url = EmbedVideoField(blank=True, null=True)
    video_fb = models.URLField(max_length=500, blank=True, null=True)
    link_url = models.CharField(max_length=2000, blank=True, null=True)
    gender_targeted = models.IntegerField(default=0, choices=GENDER_TARGETED)
    countries_targeted = models.ManyToManyField(Country, blank=True,
related_name="posts_countries")
    likes = models.ManyToManyField(MyUser, related_name="likes", blank=True,
through='LikesExtra')
    shares = models.ManyToManyField(MyUser, related_name="shares", blank=True,
through='SharesExtra')
    shared_with_friends = models.PositiveIntegerField(default=0)
    my_user = models.ForeignKey(MyUser)

    def __str__(self):
        return str(self.id)

    @property
    def total_likes(self):
        return self.likes.count()

    @property
    def total_shares(self):
        return self.shares.count()
```

Model `Post` je v relaciji »ena proti mnogo«, s tujim ključem povezan z modelom `MyUser`, kar pomeni, da je lahko vsak uporabnik lastnik mnogo objav, vsaka objava pa lahko ima samo enega avtorja - uporabnika. Model `Post` vsebuje polja, kot so URL Facebook objave, identifikacijsko številko Facebook objave in njegovega uporabnika ter njuna skupaj s podčrtajem združena identifikacijska številka »user_post_id«, datum in čas nastanka objave na moji aplikaciji, datum zaključitve promocije, trenutni status aktivnosti objave ter skupno število Facebook prijateljev, katerim je bil deljen. Z relacijo »mnogo proti mnogo« je objava povezana z modeloma `LikesExtra` in `SharesExtra`, kar pomeni, da lahko več uporabnikov všečka in deli več različnih objav. V podatkovni bazi v praksi to pomeni dve dodatni tabeli za všečke in deljenja, kateri vsaka vsebujeta polja: id uporabnika, ki je objavo všečkal/delil, id deljene/všečkane objave ter datum in timestamp aktivnosti (všečkanja/deljenja). Objava je z relacijo »mnogo proti mnogo« povezana tudi z modelom Country, v katerem so shranjene vse države sveta in njihove geografske kode, kar pomeni še eno tabelo, ki vsebuje podatke identifikacijskih številk objav in držav. Model `Post` definirajo še ostale komponente, ki jih pridobim preko Facebook API-ja: sporočilo, ime, naslov in opis objave, njen Facebook URL ter URL videa, če je bil ta naložen preko Facebooka ali preko tujega portala za deljenje videov, kot sta Vimeo in Youtube. Za prikaz Youtube in Vimeo videov na sprednjem vizualnem delu

spletne strani sem uporabil knjižnico django-embed-video in njeno polje v modelu `EmbedVideoField`, kamor shranjujem URL povezave iz teh dveh portalov.

Pravilno izpolnjeni obrazec za promocijo objav torej preko modela `Post` ustvari objekt objave z zgoraj predstavljenimi komponentami oziroma lastnostmi. Klic na Facebook API se opravi med shranjevanjem. To sem dosegel z modifikacijo metode `save`, ki se pokliče ob shranjevanju objekta. Ker imajo različne Facebook objave drugačno URL strukturo sem vsak tip URL-ja ročno razčlenil ter iz njega izluščil id uporabnika in objave. Nato sem te številki združil s podčrtajem, da sem dobil unikatni id, s katerim skupaj z dovolilnim žetonom dostopam do podatkov objave preko Facebook API-ja. Za pošiljanje zahtev na Facebook API sem uporabil Python knjižnico `requests` in njeno metodo `get` (glej kodo 4.5).

Koda 4.5: Pošiljanje zahtev na Facebook API s Python knjižnico requests

```
self.user_post_id = "_".join([str(self.fb_user_id), str(self.fb_post_id)])

access_token = SocialToken.objects.get(account__user=self.user,
account__provider='facebook')

r = requests.get('https://graph.facebook.com/{0}?fields=message,created_time,name,
description,caption,attachments,source&access_token={1}'.format(self.user_post_id,
access_token))

rjson = r.json()
```

Odgovor na poslano zahtevo dobim v JSON formatu v katerem so lahko sledeči ključi, ki jih, če obstajajo in jih potrebujem, pripišem objektu: »message«, »name«, »description«, »caption«, »source«, »attachments«, »link«, »title«.

Istočasno sem za vsako skreirano objavo aktiviral asinhrono funkcijo, ki bo na določen datum izteka objave, to deaktivirala (glej kodo 4.6). To sem dosegel preko odprto-kodnega sistema Celery natančneje Python knjižnice celery, ki je namenjen asinhronemu distribuiranemu izvajanju opravil (angl. *tasks*). To pomeni, da lahko izvajanje kode preložim na kasnejši čas. Glavna sestavina sistema je razporejevalnik opravil (angl. *message broker*); v mojem primeru RabbitMQ, ki prejema zahteve in jih razporeja med izvajalce opravil (angl. *worker*). Ti izvedejo zahtevano opravilo in vrnejo rezultat ali pa ga shranijo za morebiten kasnejši prevzem (Jakop 2012, 18).

Koda 4.6: Asinhrona funkcija za deaktivacijo objave

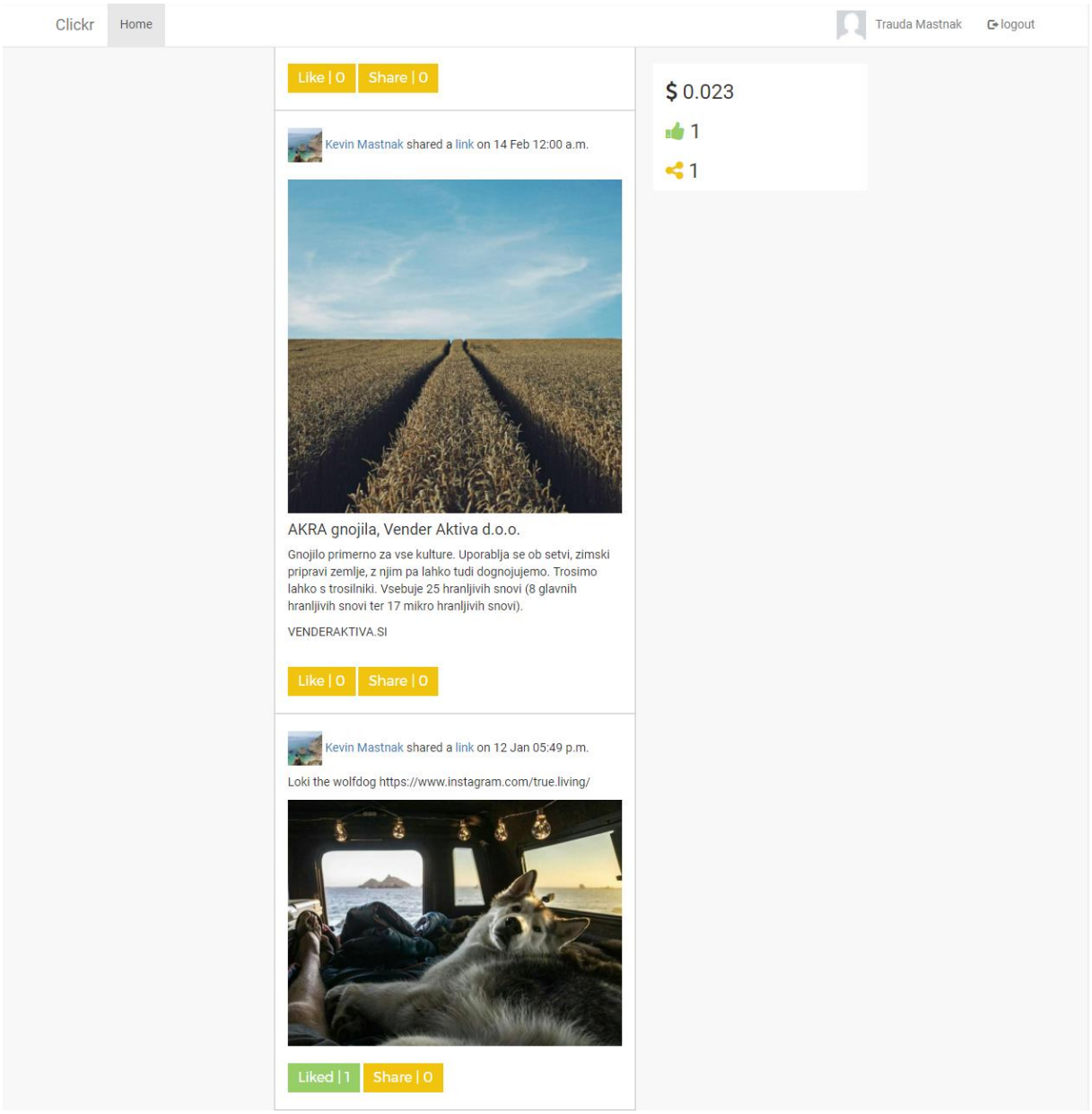
```
@celery_app.task
def deactivate_post(post_id):
    from posts.models import Post
```

```
    post_object = Post.objects.get(id=post_id)
    post_object.post_status = 1
    post_object.save()

post_starts = self.post_starts
post_ends = self.post_ends
date_range = post_ends - post_starts.date()
final_date = post_starts + date_range
deactivate_post.apply_async(args=[self.id], eta=final_date)
```

### 4.6 Domača stran

Domača stran uporabnika »Boosterja«, tistega, ki promovira svoje vsebine izgleda enako kot domača stran »Clickerja« - uporabnika, ki deli vsebino (glej sliko 4.8), vendar ima prvi omenjeni skrita gumba za všečkanje in deljenje, prav tako pa je njuna prikazana statistika na desni strani prikazana malce drugače. Clickerju prikazana statistika beleži trenutno dobroimetje na računu, število všečkov, ki jih je dodelil ter število objav, ki jih je delil na svoj Facebook zid. Boosterju je prav tako prikazano trenutno stanje na računu, poleg tega pa mu je prikazano še število objav, ki jih promovira, ter trenutno število vseh všečkov in deljenj vseh teh objav.

Slika 4.8: Domača stran uporabnika »Clicker« (tisti, ki deli vsebino)




Objave se na domači strani uporabnika pojavijo razporejene po datumu (najnovejše zgoraj) in samo, če uporabnik ustreza vsem pogojem, ki so bili dodeljeni objavi ob njeni kreaciji. Uporabnik bo objavo videl in mu bo dodeljena pravica deljenja in všečkanja samo v primeru, da ta prihaja iz ene izmed držav, ki so bile navedene pri objavi (oziroma objavo vidijo vsi, če države niso bile dodeljene), spada v izbrano kategorijo spola, ter če je objava seveda aktivna (glej kodo 4.7).

Koda 4.7: Razred domače strani in metoda za prikaz relevantnih objav

```
class HomeView(LoginRequiredMixin, AjaxListView):
    template_name = "home.html"
    page_template = "single_post.html"
```

```
    model = Post

    def get_queryset(self):
        my_country = self.request.user.country.name
        my_gender = self.request.user.gender
        if my_gender == "UNKNOWN":
            gender = 0
        elif my_gender == "MALE":
            gender = 1
        elif my_gender == "FEMALE":
            gender = 2
        return Post.objects.filter(post_status=0). \
            filter(Q(countries_targeted__name__contains=my_country) |
Q(countries_targeted__isnull=True) & Q(gender_targeted=0) |
Q(gender_targeted=gender))
```

Ker lahko število objav s časom in številom uporabnikov hitro naraste sem potreboval kar se da učinkovit način za njihov prikaz. Odločil sem se za sprotno AJAX nalaganje objav ob koleščkanju miške in sicer; po vsakih petih preletenih objavah se naloži novih pet. Za slednje sem uporabil Djangovo aplikacijo django-el-pagination (*Django Endless Pagination*) ter njeno funkcionalnost, tj. Twitter-style Pagination. Ker AJAX zahteva ne sme vrniti celotne spletne šablone (angl. *template*) ampak le del dodane oziroma posodobljene strani, sem spletno šablono razdelil na dva dela, tj. osnovno šablono in posamezno objavo. Spletna šablona za posamezno objavo pridobi podatke za pet objav. Za vsako objavo ustvari nov blok v katerem v glavi se nahaja slika uporabnika, ki je objavo promoviral, čas in datum njene kreacije ter njeno prvotno Facebook URL povezavo. V telesu bloka se nahaja vsa vsebina objave ter njena fotografija ali video, če ta obstajata. V nogi bloka pa sta gumba za *všečkanje* in deljenje objave, ki vsebujeta tudi trenutno stanje *všečkov* in *deljenj* (glej kodo 4.8).

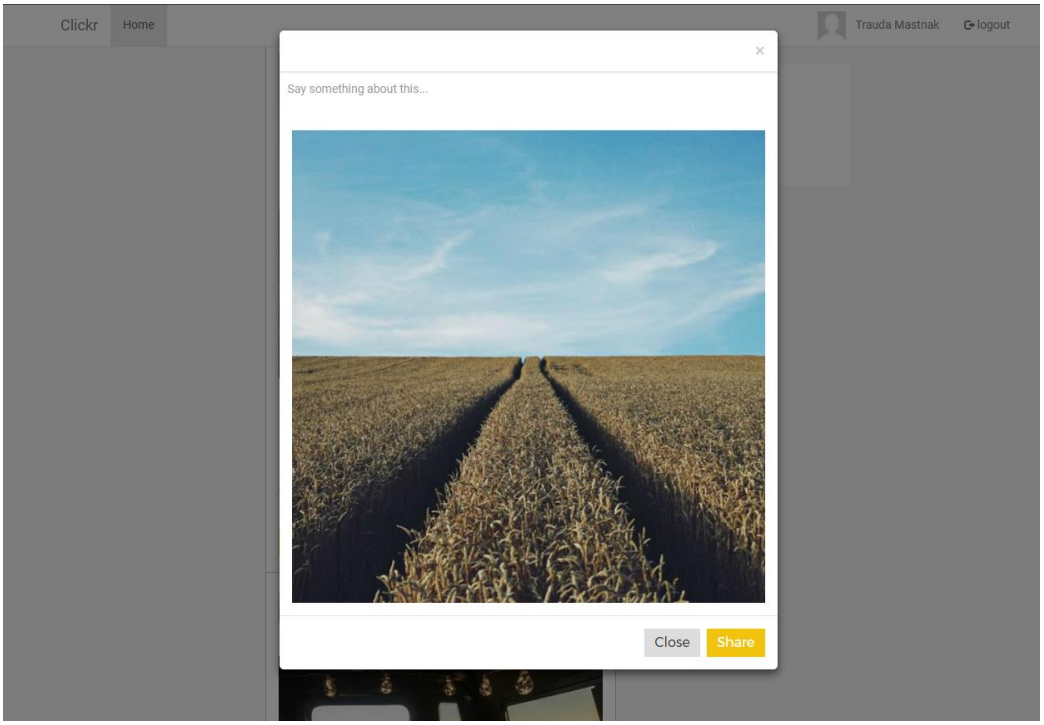
Koda 4.8: Spletna šablona posamezne objave

```
{% paginate 5 post_list %}
{% for post in post_list %}
    <div class="row">
        <div class="col-lg-3 col-md-3 col-sm-3 col-xs-1"></div>
        <div class="col-lg-6 col-md-6 col-sm-6 col-xs-10">
            <div class="row">
                <div class="col-lg-8 col-md-10 col-sm-12 single-post"
                     {% if forloop.last %}style="border-bottom: 1px solid silver;"{% endif %}>
                    <div class="post-header">
                        <img src="{{ post.my_user.socialaccount_set.all.0.get_avatar_url }}"
                             style="width:40px; height:40px;">
                        <a target="_blank"
                           href="{{ post.my_user.socialaccount_set.all.0.extra_data.link }}">
                            {{ post.my_user.socialaccount_set.all.0.extra_data.name }}</a>
                        shared a <a target="_blank" href="{{ post.url }}">
                        link</a> on
```

```
                        {{ post.post_starts|date:"j M h:i a" }}
                    </div>

                    {% if post.message %}
                        <div class="post-message" style="margin: 10px 0px;">
                            {{ post.message }}
                        </div>
                    {% else %}
                        <br>
                    {% endif %}

                    <div class="post-body">
                        {% if post.video_url %}
                            {% video post.video_url 'small' %}
                        {% elif post.video_fb %}
                            <video src="{{ post.video_fb }}" controls></video>
                        {% elif post.picture_url %}
                            <img src="{{ post.picture_url }}" class="img-responsive">
                        {% endif %}
                        {% if post.name != "Timeline Photos" and post.name != null and "cover
photo" not in post.name %}
                            <h4>{{ post.name }}</h4>
                        {% endif %}
                        {% if post.description %}
                            <p>{{ post.description }}</p>
                        {% endif %}
                        {% if post.caption %}
                            <p>{{ post.caption | upper }}</p>
                        {% endif %}
                        <br>
                    </div>
                    {% if user.is_clicker %}
                        <div>
                            <button id="{{ post.id }}" type="button"
                                    class="btn like-button {% if user in post.likes.all
%}like-button-active{% endif %}">
                                {% if user in post.likes.all %}Liked{% else %}Like{% endif %}
                                | {{ post.total_likes }}</button>
                            <button id="{{ post.id }}" type="button"
                                    class="btn share-button {% if user in post.shares.all
%}share-button-active{% endif %}"
                                    {% if user in post.shares.all %}disabled{% endif %}>
                                {% if user in post.shares.all %}Shared{% else %}Share{% endif
%}
                                | {{ post.total_shares }}</button>
                        </div><br>
                    {% endif %}
                </div>
            </div>
        </div>
    </div>
{% endfor %}
```

S klikom uporabnika na gumb za deljenje objave se mu najprej prikaže novo okno s fotografijo (če jo objava ima), kjer lahko doda svojo sporočilo (glej sliko 4.9).

Slika 4.9: Ponovni prikaz deljenja želene objave




S klikom na gumb »Share« se izvede Javascript funkcija, ki pridobi napisano sporočilo ter že prej definiran id objave, ki ju z AJAX opravilom pošlje na strežnik (glej kodo 4.9).

Koda 4.9: Javascript funkcija z AJAX opravilom sprožena z deljenjem objave

```
$("#share-to-fb").click(function () {
    var csrftoken = getCookie('csrftoken');
    var message = $('input[name="share-message"]').val();
    console.log(message);
    $.ajax({
        url: '{% url 'posts:like_share' %}',
        type: 'POST',
        data: {
            action: "share",
            message: message,
            post_id: post_id,
            csrfmiddlewaretoken: csrftoken
        },
        dataType: 'json',
        context: this,
        success: function (response) {
            var post_id = response.post_id;
            var $button = $('#' + post_id + ".share-button");
            $('#user_total_shares').html(" " + response.user_total_shares);
            $('#user_credit').html(" " + response.user_total_credit);
            $button.addClass("share-button-active");
            $button.prop("disabled", true);
            $button.html("Shared | " + response.shares_count);
        }
    });
});
```

Strežnik AJAX zahtevo prejme in iz nje izlušči parametra id objave in akcijo, ki pomeni všečkanje ali deljenje objave. V primeru deljenja objave se izvede slednja koda (glej kodo 4.10).

Strežnik pridobi še sporočilo delilca, njegov dovolilni žeton, število njegovih prijateljev na Facebooku ter URL povezavo prvotne objave. Strežnik s Python knjižnico `requests` in njeno metodo `post` na Facebook API pošlje zahtevo za deljenje objave na uporabnikov zid. Facebook API vrne odziv v JSON formatu, ki ob uspešnem izidu vsebuje id novo ustvarjene objave. Ta id shranim v novo tabelo, ki je zaenkrat v neuporabi, služi pa morebitnemu nadaljnjemu sledenju stanja deljenih objav. Kakorkoli že, v vmesno tabelo med objavo in uporabnikom se preko modela `SharesExtra` zabeleži uporabnik, ki je delil povezavo, objekt objave same, število Facebook prijateljev, ki jih delilec ima ter čas in datum deljenja. Prav tako se delilcu po formuli število facebook prijateljev deljeno s 1000 prišteje dobroimetje, ki se hkrati odšteje lastniku objave. Na ta način simuliram transakcijo denarja od promotorja k delilcu. Če ima promotor po transakciji na računu znesek, ki je manjši ali enak 0, se mu deaktivirajo vse objave, ki jih lahko aktivira samo, ko na račun naloži več denarja. Funkcija na koncu v sprednji del spletne strani pošlje odziv v JSON formatu, ki ga obdela AJAX »success« funkcija (glej kodo 4.9). Ta posodobi uporabnikovo statistiko in stilsko spremeni in onemogoči gumb za ponovno deljenje. Postopek za *všečkanje* je deloval na podoben način, vendar je Facebook 17. Novembra 2016 opustil podporo za *všečkanje* objav preko API-ja; od takrat lahko to počno samo Facebook strani – Facebook Pages, zatorej se *všečkanje* trenutno izvaja samo znotraj aplikacije. *Všeček* se torej vseeno zapiše v bazo in je prikazan na strani, vendar ni nagrajeno, ni podprto s transakcijami.

Koda 4.10: Funkcija, ki sprejme AJAX zahtevo za deljenje objav.

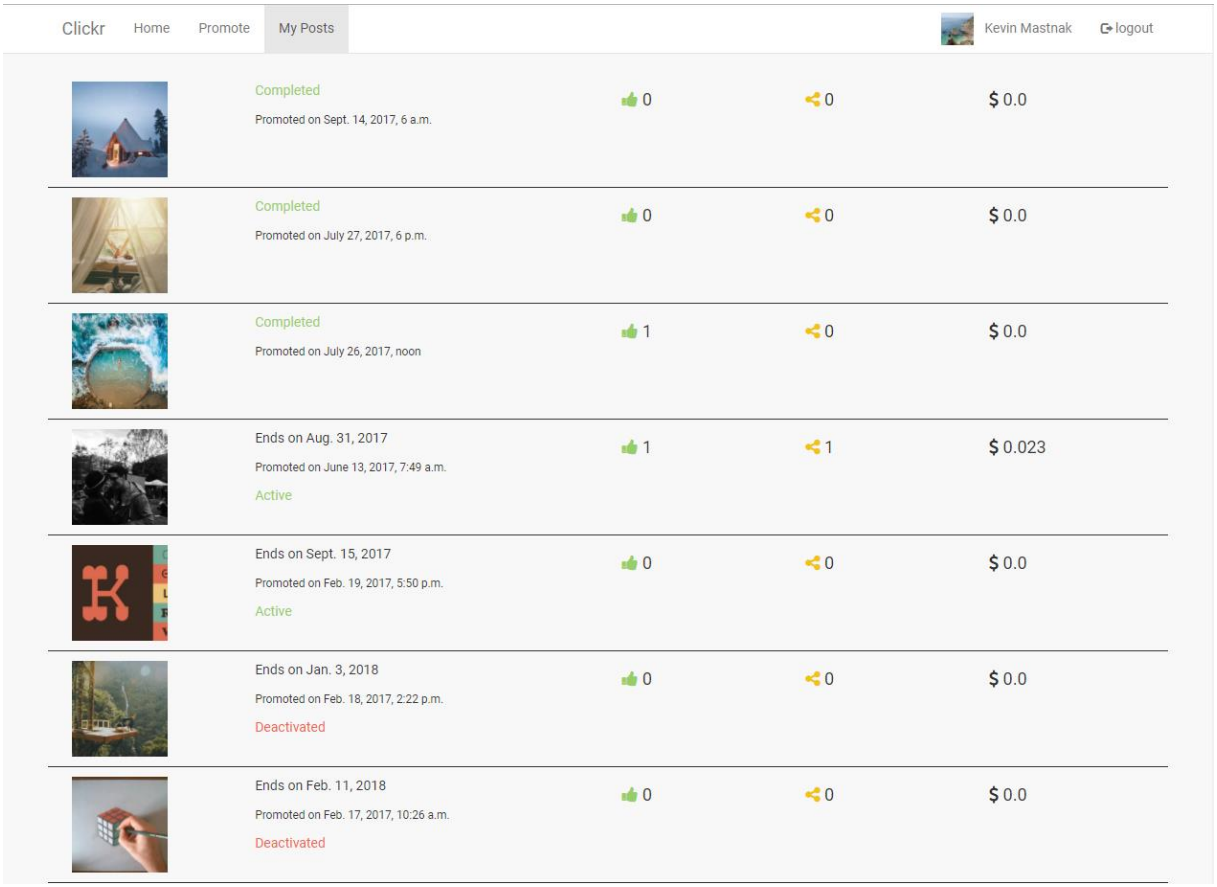
```
def like_share_ajax(request):
    if request.method == 'POST' and request.is_ajax():
        post_id = request.POST.get('post_id', '')
        action = request.POST.get('action', '')
        user = request.user
        post = get_object_or_404(Post, id=post_id)

        if action == "share":
            message = request.POST.get('message', '')
            user_friends =
user.socialaccount_set.all()[0].extra_data['friends']['summary']['total_count']
            link = post.url
            access_token = SocialToken.objects.get(account__user=user,
account__provider='facebook')
            data = {'message': message, 'link': link, 'access_token': access_token}
            r = requests.post('https://graph.facebook.com/me/feed', params=data)
            response = json.loads(r.text)

            try:
                shared_id = response['id']
            except TypeError:
                shared_id = ""
            if shared_id:
                SharedPost(original=post, shared_id=shared_id).save()

            shares_extra = SharesExtra(user=user, post=post,
reach=int(user_friends))
            shares_extra.save()
            post.shared_with_friends += user_friends
            post.save()
            post_owner = post.my_user
            post_owner.credit -= float(user_friends / 1000)
            post_owner.save()
            if post_owner.credit <= 0:
                for post in post_owner.post_set.all():
                    post.post_status = 1
                    post.save()

            user.credit += float(user_friends / 1000)
            user.save()
            user_total_credit = user.credit
            user_total_shares = user.shares.count()
            return JsonResponse(
                {'shares_count': post.shares.count(), 'post_id': post_id,
'user_total_shares': user_total_shares,
                 'user_total_credit': user_total_credit})
```

### 4.7 Dashboard

Dashboard ali armaturna plošča je poslovno orodje, ki je ime pridobilo po analogiji armaturne plošče avtomobila, gre pa za vizualni prikaz pomembnih informacij. Je orodje, sestavljeno iz več orodij za vizualizacijo podatkov, ki po navadi prikazuje indikatorje uspešnosti ali katerekoli

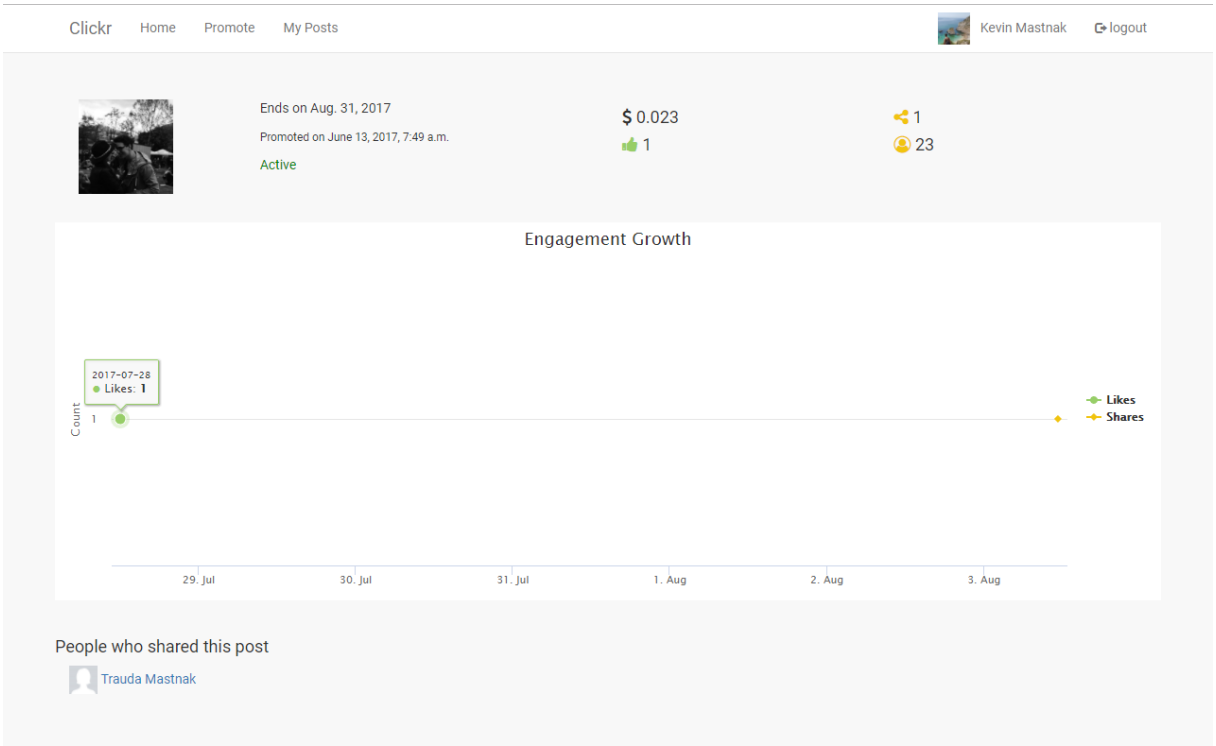
druge za poslovne uporabnike zanimive informacije (Gjerek 2016, 8). V mojem primeru to pomeni prikaz stanja vseh promoviranih objav in njihovo osnovno statistiko; število *všečkov*, *deljenj* in strošek promocije (glej sliko 4.10).

Slika 4.10: Osnovni prikaz statistike objav




S klikom na posamezno objavo je uporabnik preusmerjen na novo stran, kjer ima vpogled nad več informacijami o posamezni objavi (glej sliko 4.11). Uporabniku je na tej strani, poleg osnovne statistike omogočen še grafični prikaz števila *všečkov* in deljenj po datumih, vpogled ima nad uporabniki, ki so njegovo objavo delili, prikaže se mu pa tudi skupno število prijateljev, katerim je bila objava deljena.

Slika 4.11: Podrobnejša statistika posamezne objave




Uporabniku je vstop na stran za prikaz posamezne objave dovoljen seveda samo, če je avtor objave. V nasprotnem primeru ga strežnik preusmeri na domačo stran. Ko uporabnik uspešno vstopi na to stran, se na strežniku izvede metoda, ki iz baze pridobi nekatere podatke in jih pošlje na sprednji, uporabniku vidni del (angl. *frontend*) spletne strani (glej kodo 4.11). Strežnik iz URL povezave pridobi identifikacijsko številko objave, preko katere jo poišče v bazi. Preko objave in njene vmesne tabele `LikesExtra` pridobi čas prvega *všečka* oziroma *deljenja*. Potem preko *»for«* zanke preleti vse *všečke* oziroma *deljenja* za to objavo ter skupaj prišteje dnevne odzive. Dobimo seznam seznamov, vsak pa vsebuje posamezni datum in število odzivov na ta dan za določeno objavo. Seznam je ob pravilno naloženi strani dostopen JavaScript knjižnici Highcharts. Njen namen je vizualni prikaz podatkov na grafu.

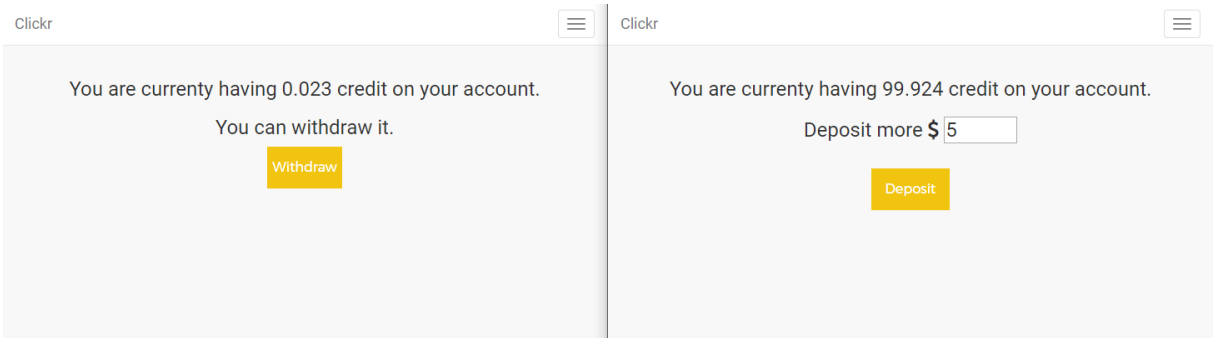
Koda 4.11: Razred DetailPostView in njegove metode, ki se nanašajo na vstop v stran za statistiko posamezne objave.

```
class DetailPostView(LoginRequiredMixin, AuthorRequiredMixin, DetailView):
    template_name = "post_detail.html"
    model = Post

    def authors(self):
        return [self.get_object().my_user]

    def get_context_data(self, **kwargs):
        context = super(DetailPostView, self).get_context_data(**kwargs)
        post = get_object_or_404(Post, id=self.kwargs.get('pk'))
        try:
            start_date = int(post.likesextra_set.first().timestamp)
        except AttributeError:
            start_date = time.time()

        last_date = ""
        likes = []
        like_extras = post.likesextra_set.all()
        for like in like_extras:
            if str(like.date) != last_date:
                likes.append([like.timestamp * 1000,
int(like_extras.filter(date=like.date).count())])
                last_date = str(like.date)

        last_date = ""
        shares = []
        share_extras = post.sharesextra_set.all()
        for share in share_extras:
            if str(share.date) != last_date:
                shares.append([share.timestamp *1000,
int(share_extras.filter(date=share.date).count())])
                last_date = str(share.date)

        context['start_date'] = start_date
        context['todays_date'] = datetime.datetime.now().date()
        context['likes'] = likes
        context['shares'] = shares
        return context
```

### 4.8 Transakcije na osebni račun

Spletna aplikacija po vsej verjetnosti ne bo nikoli izšla, zato sem onemogočil operacije s pravim denarjem in namesto tega za njeno pravilno delovanje ustvaril poenostavljeno simuliranje nakazovanja in dvigovanja sredstev na ter z uporabniškega računa. Uporabnik, ki ima vlogo oglaševalca, si lahko torej na ta način naloži neomejeno količino kreditov (glej sliko 4.12, desno), s katerimi promovira svojo vsebino; na drugi strani pa uporabnik, ki to vsebino deli, lahko simulira dvig prisluženih sredstev, ki si jih je z aktivnostjo pridobil (glej sliko 4.12, levo).

Slika 4.12: Simulacija dvigovanja sredstev iz računa (leva stran) in nakazovanje kreditov oglaševalca (desna stran).




# 5 ZAKLJUČEK

Spletno programiranje me zanima že dalj časa, zato sem si za diplomsko nalogo izbral tehnično primeren, hkrati pa vsebinsko dokaj zanimiv projekt, kjer sem lahko uporabil, nadgradil in

razširil skupek znanj, ki se nanašajo na področje spletnega programiranja in programiranja nasploh.

Z razvojem spletne aplikacije sem obstoječim uporabnikom Facebooka ponudil novo marketinško orodje za promocijo vsebin, ki se od ostalih sistemov razlikuje po tem, da obstaja med oglaševalcem in ciljno publiko vmesni člen, tj. uporabnik, ki je prav tako tarča oglaševalca, ta pa zavestno a prikrito deli oglaševalčevo vsebino svojim Facebook prijateljem, za kar mu neposredno plača oglaševalec. Moja aplikacija predstavlja vlogo glavnega oglasnega prostora, kjer se zbirajo uporabniki, njihov Facebook »zid« pa postane ob deljenju vsebin nov oglasni prostor. Pomembno je tudi, da oglaševalec ni nujno podjetje, ampak je lahko katerakoli fizična oseba, ki želi deliti svoje ideje, znanja oziroma tako ali drugače išče pozornost.

Za izdelavo spletne aplikacije sem izbral meni najbližji Python programski jezik in njegovo najpriljubljenejše ogrodje Django. Izkazalo se je, da je za takšne projekte, ki so povezani s podatkovno bazo in delujejo po principu zahteve uporabnika, kateremu sledi odziv strežnika, to dobra izbira. Edina slabost Djanga je podpora za spreminjanje spletne strani v realnem času, kar pomeni, da je za spremembe, ki so se zgodile v podatkovni bazi in morajo biti vidne na strani, potrebno osvežiti brskalnik oziroma uporabnika preusmeriti na novo stran. Slednje sem seveda rešil z asinhronim pristopom AJAX. Ugotovil sem tudi, da za vsak nerešen problem obstaja rešitev na prvi strani zadetkov internetnega iskalnika, kar priča o razširjenosti uporabe ogrodja. Prav tako za vsak problem obstaja mnogo različnih pristopov reševanja tega problema. Jaz sem večje probleme namesto z odkrivanjem lastnih rešitev, reševal s pomočjo implementacije tujih, že preverjenih knjižnic in praks ter si s tem prihranil ogromno časa.

Menim, da lahko potrdim predvideni odgovor na izhodiščno raziskovalno vprašanje, ali je možno v Python programskem jeziku s pomočjo Facebook API-ja izdelati spletno aplikacijo, ki bo oglaševalcem omogočala promocijo, ostalim uporabnikom pa promoviranje njihove vsebine s pomočjo Facebook operacij *všečkanja in deljenja*. Zaradi omejitev Facebook API-ja trenutno ni bilo mogoče implementirati funkcionalnosti *všečkanja*, vendar tega ne moremo pripisati Pythonu. Po zaključni evalvaciji spletne aplikacije lahko opazim nove možnosti za izboljšave in spremembe končnega produkta. Zavedam se tudi stvari, ki bi jih lahko naredil drugače – nove ideje imam o nadgradnji, dodajanju funkcionalnosti itd., a menim, da mi je z izbrano tehnologijo uspelo postaviti uporabniku prijazno spletno aplikacijo, ki služi svojemu namenu.

# 6 LITERATURA


1. Anderson, Paul. 2007. *What is Web 2.0? Ideas, technologies and implications for education*. Bristol: JISC.
2. Boyd, Danah M. In Nicole B. Ellison. 2008. Social Network Sites: Definition, History and Scholarship. *Journal of Computer-Mediated Communication* (13): 210–230.
3. Broz, Roman in Viktorija Sulčič. 2009. Vpliv uporabniške izkušnje na uspešnost e-poslovanja. *Management* 4 (2): 149–168.

4. Channel 4. 2013. *Jaron Lanier on how to make the internet pay*. Dostopno prek: https://www.channel4.com/news/jaron-lanier-on-how-to-make-the-internet-pay (17. julij 2017).
5. Damjan, Jure. 2016. *Plačljivo spletno oglaševanje*. Diplomsko delo. Ljubljana: Fakulteta za računalništvo in informatiko.
6. *Django*. Dostopno prek: https://www.djangoproject.com/ (21. julij 2017).
7. Docs.python. 2017. *General Pyton FAQ*. Dostopno prek: https://docs.python.org/3/faq/general.html (21. julij 2017).
8. Eržen, Miha. 2012. *Odprta avtentikacija*. Diplomsko delo. Ljubljana: Fakulteta za računalništvo in informatiko.
9. Developers.facebook. 2017. *Facebook for developers*. Dostopno prek: https://developers.facebook.com/docs/ (18. julij 2017).
10. Garrett, Jesse James. 2011. *The elements of user experience: user-centered design for the web and beyond*. Berkeley (CA): New Riders.
11. Gjerek, Uroš. 2016. *Razvoj orodja za grafično predstavitev podatkov o tržnem deležu*. Diplomsko delo. Ljubljana: Fakulteta za družbene vede.
12. Hern, Alex. 2015. *Facebook is making more and more money from you. Should you be paid for it?* Dostopno prek: https://www.theguardian.com/technology/2015/sep/25/facebook-money-advertising-revenue-should-you-be-paid (17. julij 2017).
13. Jakop, Matej. 2012. *Sistem za avtomatizacijo upravljanja oglasnega prostora*. Diplomsko delo. Ljubljana: Fakulteta za računalništvo in informatiko.
14. Leber, Dominik. 2014. *Avtentikacija s pomočjo ponudnikov družbenih omrežij*. Diplomsko delo. Ljubljana: Fakulteta za računalništvo in informatiko.
15. Lorenzo-Romero, Carlota, Maria-del-Carmen Alarcon-del-Amo in Miguel-Angel Gomez-Borja. 2011. Do You Have Social Profile? Users And Non-Users Of Social Networking Sites In The Web 2.0. V *Review Of Business Information Systems*, 41–50. Littleton: The Clute Institute.
16. MDN – Mozilla Developer Network. 2017. *Web technology for developers*. Dostopno prek : https://developer.mozilla.org/en-US/docs/Web (21. julij 2017).
17. Mele, Antonio. 2015. *Django By Example*. Birmingham: Packt Publishing.
18. Muntinga, Daniel G., Edith Smit in Marjolein Moorman. 2012. Social Media DNA: How Brand Characteristics Shape COBRAs. V *Advances in Advertising Reserach (Vol.*

*III): Current Insights and Future Trends*, ur. Martin Eisend, Tobias Langner in Shintaro Okazaki, 121–136. Germany: Springer Gabler.

19. Newsroom.fb. 2017. *Company info*. Dostopno prek: https://newsroom.fb.com/company-info (3. julij 2017).
20. O'Reilly, Tim. 2005. *Web 2.0: Compact Definition?* Dostopno prek: http://radar.oreilly.com/2005/10/web-20-compact-definition.html (28. junij 2017).
21. O'Reilly, Tim. 2010. What is Web 2.0? Design patterns and business models for the next generation of software. V *Online Communication and Collaboration*, ur. Helen Donelan, Karen Kear in Magnus Ramage, 225–236. Routledge.
22. Phillips, Sarah. 2007. *A brief history of facebook*. Dostopno prek: https://www.theguardian.com/technology/2007/jul/25/media.newmedia (3. julij 2017).
23. Rauschnabel, Philipp A., Sandra Praxmarer, Bjorn S. Ivens. 2012. Social Media Marketing: How Design Features Influence Interactions with Brand Postings on Facebook. V *Advances in Advertising Reserach (Vol. III): Current Insights and Future Trends*, ur. Martin Eisend, Tobias Langner in Shintaro Okazaki, 153–175. Germany: Springer Gabler.
24. Sqlite. 2017. *About SQLite*. Dostopno prek: https://www.sqlite.org/about.html (23. avgust 2017).
25. Statista – The Statistics Portal. 2017. *Facebook's advertising revenue worldwide from 2009 to 2016 (in million U.S. dollars).* Dostopno prek: https://www.statista.com/statistics/271258/facebooks-advertising-revenue-worldwide (24. julij 2017).
26. Tullis, Tom in Bill Albert. 2013. *Measuring the User Experience. Collecting Analyzing, and Presenting Usability Metrics.* Aaltham (MA): Elsevier.
27. W3schools. 2017. *SQL Tutorial*. Dostopno prek: https://www.w3schools.com/sql/default.asp (24. julij 2017).
28. Young, Anthony. 2014. *Brand media strategy: integrated communications planning in the digital era.* New York: Palgrave Macmillan US.

## PRILOGA A: Povezava na izvorno kodo projekta

https://gitlab.com/kevakm/clicker