UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boštjan Lotrič

APLIKACIJE ZA SOCIALNA OMREŽJA

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2011

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01730/2011
Datum: 15.03.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **BOŠTJAN LOTRIČ**

Naslov: **APLIKACIJE ZA SOCIALNA OMREŽJA**
APPLICATIONS FOR SOCIAL NETWORKS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Z razmahom socialnih omrežij v zadnjem desetletju je uporaba zbranih podatkov v omrežjih postala zanimiva za raznolike aplikacije oglaševanja, socialnega povezavanja, izboljšanja uporabniške izkušnje itd. Za dostop do teh podatkov in izdelavo gostujočih aplikacij v omrežjih ponujajo gostitelji socialnih omrežij tudi aplikacijske programerske vmesnike (API).

Kandidat naj v diplomskem delu predstavi postopke razvoja omenjenih aplikacij. V svojem delu naj preuči in primerja aplikacijske programerske vmesnike dveh razširjenih socialnih omrežij in naj izdelavo aplikacije prikaže na lastnem primeru aplikacije.

Mentor:

doc. dr. Zoran Bosnić

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a Boštjan Lotrič,

z vpisno številko 63050169,

sem avtor diplomskega dela z naslovom:

Aplikacije za socialna omrežja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Zorana Bosnića,
- 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 v zbirki "Dela FRI".

V Ljubljani, dne 8. 9. 2011 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju doc. dr. Zoranu Bosniću za vso pomoč, nasvete in predloge pri izdelavi diplomske naloge.

Kazalo

Seznam uporabljenih kratic in simbolov

Ajax - *Asynchronous JavaScript and XML*
API - *Application Programming Interface*
CSS - *Cascading Style Sheets*
DOM - *Document Object Model*
HTML - *HyperText Markup Language*
HTTP - *Hypertext Transfer Protocol*
JSON - *JavaScript Object Notation*
MVC - *Model–View–Controller*
PHP - *PHP: Hypertext Preprocessor*
SDK - *Software Development Kit*
XHTML - *Extensible Hypertext Markup Language*
XML - *Extensible Markup Language*

Seznam prevedenih izrazov

Application programming interface - *Vmesnik za programiranje*
Access token - *Žeton za dostop*
Canvas - *Platno*
Document object model - *Objektni model dokumenta*
Event listener - *Detektor dogodka*
Permission - *Dovoljenje*
Social plugin - *Socialni vtičnik*
Timeline - *Seznam sporočil*
Token - *Žeton*
Tweet - *Sporočilo*

Povzetek

V diplomskem delu so predstavljeni vmesniki za programiranje socialnih omrežij Facebook in Twitter in razvoj spletne aplikacije, ki deluje z omrežjem Facebook. Pri vsakem od omrežij so opisani tudi socialni vtičniki, ki predstavljajo najenostavnejšo uporabo socialnih podatkov v aplikacijah. Za razvoj spletne aplikacije sta bili izbrani spletni tehnologiji JavaScript in PHP, ki v povezavi z označevalnim jezikom HTML in slogovno predlogo CSS tvorita ogrodje spletne aplikacije. Opisane so funkcionalnosti in struktura spletne aplikacije, ki deluje v vseh (v času pisanja) širše uporabljenih brskalnikih in omogoča ustvarjanje ter oddajo glasu pri spletnih anketah. Predstavljena sta dva najpogostejša primera uporabe aplikacije. V zaključku so predstavljene sklepne ugotovitve in ideje za nadaljnje delo, ki vsebujejo možne razširitve spletne aplikacije.

Ključne besede:

socialno omrežje, spletna aplikacija, socialni podatki, vmesnik za programiranje

Abstract

The thesis contains the review of Facebook and Twitter's application programming interfaces and describes the development of a web application for the Facebook social network. Social plugins, which represent the simplest use of social data in web applications, are described for both social networks. JavaScript and PHP have been selected, along with the markup language HTML and styling language CSS, as enabling technologies for the web application. Features and the structure of the web application, which is developed for all (at the time of writing) popular web browsers and supports creating and voting for web polls, are described. Two of the most common use cases are presented in detail. Final observations and ideas for expanding the web application are explained in the conclusion.

Key words:

social network, web application, social data, application programming interface

Poglavje 1

Uvod

Aplikacija za socialno omrežje je aplikacija, ki pri svojem delovanju uporablja in upravlja s podatki o uporabnikih tega socialnega omrežja. V zadnjih nekaj letih je število tovrstnih aplikacij močno naraslo, predvsem zaradi izredno hitrega naraščanja števila registriranih uporabnikov socialnih omrežjih. Z večanjem števila uporabnikov se veča tudi interes podjetij, ki socialna omrežja vidijo kot nov trg za svoje izdelke ali storitve. Ne glede na to, ali je aplikacija poslovne narave ali ne, pa je vsem aplikacijam skupno, da pri svojem delovanju izkoriščajo že izgrajeno socialno omrežje, ki omogoča dostop do podatkov o svojih uporabnikih.

Ti podatki vsebujejo uporabnikove osebne podatke, njegova zanimanja, aktivnosti, povezave z drugimi uporabniki omrežja itd. Aplikacija dostopa do teh podatkov preko vmesnika za programiranje, ki ga ponuja omrežje, za kar seveda potrebuje uporabnikovo dovoljenje. Socialnih omrežij je veliko in so si med sabo različna tako po velikosti kot tudi po vrstah interakcij, ki so na voljo uporabnikom. V diplomskem delu bomo natančneje predstavili socialni omrežji Facebook in Twitter ter izdelavo spletne aplikacije za omrežje Facebook.

Obe omenjeni omrežji in njuni vmesniki za programiranje delujejo kot spletne aplikacije in se poslužujejo aktualnih tehnologij za spletno programiranje. Te tehnologije smo uporabili pri izdelavi spletne aplikacije in jih bomo podrobneje predstavili v naslednjem poglavju.

V tretjem in četrtem poglavju so natančneje opisani vmesniki za programiranje zgoraj omenjenih socialnih omrežij, socialni vtičniki, ki jih ti dve omrežji ponujata, primeri uporabe, pripadajoča dokumentacija itd. Pri pregledu smo se, predvsem zaradi njegove popularnosti, v veliki meri osredotočili na omrežje Facebook.

V petem poglavju je podrobno opisana spletna aplikacija, ki smo jo razvili

za omrežje Facebook. Opis aplikacije se deli na tri dele: predstavitev funkcionalnosti aplikacije, uporabniški vmesnik aplikacije in predstavitev strežniškega dela aplikacije. Na koncu poglavja sta prikazana tudi dva najpogostejša primera uporabe.

V zadnjem poglavju diplomskega dela smo zapisali opažanja o vmesnikih za programiranje socialnih omrežij in ideje za razširitev funkcionalnosti spletne aplikacije.

Poglavje 2

Pregled uporabljenih tehnologij in protokolov

V nadaljevanju so opisane vse tehnologije, pristopi in protokoli, ki smo jih uporabili pri izdelavi aplikacije za socialno omrežje Facebook.

2.1 Uporabljene tehnologije pri razvoju grafičnega uporabniškega vmesnika

Ker je aplikacija dostopna iz spletnega brskalnika (in v osnovi deluje kot spletna stran), je zelo pomembno izbrati prave tehnologije za gradnjo spletnega vmesnika. S tem želimo zagotoviti čim večjo oziroma popolno funkcionalnost spletne aplikacije. Seznam brskalnikov, ki podpirajo spletno aplikacijo:

- Mozilla Firefox,
- Google Chrome,
- Internet Explorer 7, 8 in 9,
- Opera,
- Safari.

Za samo postavitev elementov uporabniškega vmesnika smo uporabili jezik HTML, ki je v tesni povezavi z jezikom CSS, ki definira njihov izgled. Manipulacijo teh elementov nam omogoča skriptni jezik JavaScript. Pisanje programske kode jezika JavaScript sta nam olajšali knjižnici jQuery in jQuery

UI. Ker smo želeli izboljšati uporabniško izkušnjo, smo se poslužili metode asinhronega JavaScripta in XML (*angl. asynchronous JavaScript and XML*, Ajax), ki omogoča komunikacijo uporabniškega vmesnika z jedrom aplikacije na strežniku, ne da bi se stran osvežila.

2.1.1 HTML, CSS

Za postavitev elementov grafičnega uporabniškega vmesnika smo uporabili jezik HTML, natančneje XHTML™ 1.0, ki je razširitev jezika HTML 4.0. HTML 4.0 je standardni generalizirani označevalni jezik, ki je v skladu z mednarodnim standardom ISO 8879, ki je trenutno najbolj razširjen standard za objavljanje spletnih vsebin na svetovnem spletu [4].

Dokumenti XHTML so v skladu s standardom XML [4], kar pomeni, da je objektni model dokumenta (*angl. Document object model*, DOM) mogoče pregledovati, urejati in preverjati na povsem enak način kot pri dokumentih XML. To nam bistveno olajša spreminjanje (tudi dodajanje in odstranjevanje) elementov uporabniškega vmesnika med samim izvajanjem aplikacije.

```
<html>
  <head>
    <title>Naslov dokumenta</title>
  </head>
  <body>
    Vsebina dokumenta
  </body>
</html>
```

Slika 2.1: Primer dokumenta HTML.

Izgled vsakega od elementov grafičnega uporabniškega vmesnika smo definirali z uporabo slogovnega jezika CSS (*angl. Cascading Style Sheets*). V dokumentih vrste CSS 2.1 določimo izgled elementa tako, da mu posamezne slogovne predloge dodamo s pomočjo posebnih izbirnih ukazov, s katerimi lahko izberemo enega ali več elementov v dokumentu [5].

2.1.2 JavaScript

Kot smo že omenili, smo za manipulacijo elementov v objektnem modelu dokumenta (dokumenta XHTML) uporabili skriptni jezik JavaScript, ki se v celoti izvaja v spletnem brskalniku uporabnika.

Jezik JavaScript je dogodkovno usmerjen (*angl. event-based*), kar pomeni, da lahko definiramo, kakšen naj bo odziv ob določeni akciji uporabnika nad določenim elementom. Akcija lahko pomeni klikanje na gumb, premik miške, pritisk tipke na tipkovnici itd.

Poleg tega nam jezik omogoča tudi asinhrono komunikacijo med brskalnikom in strežnikom, kar pripomore k boljši uporabniški izkušnji, saj lahko na ta način osvežimo le del spletne strani. To bomo podrobneje opisali v poglavju o metodi Ajax.

2.1.3 Knjižnici jQuery in jQuery UI

Za lažje in učinkovitejše pisanje programske kode JavaScript smo si pomagali s knjižnico jQuery [1], ki nam močno olajša delo na naslednje štiri načine:

1) Iskanje in izbiranje elementov v objektnem modelu dokumenta je s pomočjo knjižnice jQuery izredno enostavno, saj nam omogoča uporabo enakih izbirnih ukazov kot pri jeziku CSS. Na ta način je zelo preprosto izbrati npr. vsa prazna vnosna polja in nanje postaviti detektorje dogodkov (*angl. event listeners*).

2) Knjižnica nam ponuja zbirko ukazov za neposredno spreminjanje atributov elementov, kar zelo poenostavi manipulacijo objektnega modela dokumenta. S temi ukazi lahko na primer brez posebnega napora spremenimo določeno lastnost (barva podlage, velikost besedila itd.) izbrani množici elementov.

3) jQuery nam omogoča enostavnejšo uporabo asinhrone komunikacije s strežnikom, kar nam omogoča hkratno izvajanje več algoritmov na strežniku. Na ta način lahko prihranimo nekaj časa.

4) Zadnja velika prednost knjižnice jQuery pa je v tem, da je njeno delovanje preizkušeno v vseh aktualnih brskalnikih, kljub temu da imajo brskalniki različne interpreterje jezika JavaScript. To pomeni, da se ob uporabi določenega ukaza lahko zanesemo, da bo njegovo delovanje brezhibno v že omenjenih brskalnikih.

Knjižnica jQuery UI je dodatek h knjižnici jQuery, ki nam ponuja vtičnike (*angl. plugins*) za gradnjo uporabniškega vmesnika [1]. Tako lahko na zelo preprost način prikažemo modalna okna za komunikacijo z uporabnikom, generiramo zavihke, uporabljamo princip *povleci in spusti* (*angl. drag and drop*) itd. Tako kot jQuery tudi knjižnica jQuery UI sama poskrbi, da ti vtičniki pravilno delujejo v različnih brskalnikih.

2.2 Uporabljene tehnologije za izvajanje aplikacije

Aplikacija za socialno omrežje je spletna aplikacija. Izvajanje spletne aplikacije poteka tako, da spletni brskalnik od strežnika zahteva izvajanje skript PHP (*angl. PHP: Hypertext Preprocessor*), ki so shranjene na strežniku. Izvajanje skript se lahko kliče asinhrono s pomočjo kode JavaScript ali preko privzetega GET ali POST protokola. Strežnik nato brskalniku posreduje rezultate izvajanja skript, ki jih uporabnik lahko vidi preko uporabniškega vmesnika.

2.2.1 PHP

PHP je večnamenski odprtokodni skriptni programski jezik, ki večinoma teče na spletnih strežnikih. Sprva je bil namenjen dinamičnemu generiranju kode HTML (od tukaj pomen kratice PHP: Hypertext Preprocessor), kasneje pa se je njegova uporaba močno razširila.

Interpreter jezika PHP lahko deluje kot program v ukazni vrstici ali kot del spletnega strežnika, ki vsakič, ko prejme zahtevek, zažene nov primerek procesa PHP. Ta proces nato z izvajanjem programske kode pripravi kodo HTML, ki se kot rezultat pošlje spletnemu brskalniku, ki je zahtevek poslal. Celotno izvajanje skript PHP se dogaja na spletnem strežniku, zato odjemalec nikoli ne vidi kode PHP. Vidi le rezultat, ki ga ta koda vrne. Interpreter jezika PHP izvaja le kodo, ki je znotraj posebnih mej (*angl. delimiter*), kar nam omogoča, da lahko kodo PHP pišemo znotraj kode HTML. Na sliki 2.2 je prikazan primer kode PHP.

```
<?php
  $opis_vremena = "prijetno";
  echo "Vreme je danes " . $opis_vremena;
?>
```

Slika 2.2: Primer kode v jeziku PHP.

2.2.2 MySQL

MySQL je sistem za upravljanje z relacijskimi podatkovnimi bazami, ki je v uporabi predvsem v kontekstu spletnih strani in spletnih aplikacij. Sistem MySQL je podprt na vseh večjih platformah in se ponaša z veliko dobrimi

lastnostmi. Ena izmed teh lastnosti nam na primer omogoča uporabo različnih pogonov za shranjevanje podatkov (vsaki tabeli v podatkovni bazi lahko določimo drug pogon), s čimer lahko optimiziramo delovanje aplikacije.

MySQL je osrednja komponenta priljubljenega nabora programske opreme za izdelavo spletnih aplikacij *LAMP*: Linux, Apache, MySQL, Perl/PHP/Python. Na sliki 2.3 je prikazan primer poizvedbe MySQL.

```
SELECT ime, priimek
FROM studenti
WHERE leto_vpisa > 2005
ORDER BY datum_rojstva ASC;
```

Slika 2.3: Primer poizvedbe MySQL.

2.3 Povezava uporabniškega vmesnika in strežniškega dela

V tem poglavju bomo podrobnjene opisali metodo Ajax za asinhrono komunikacijo s strežnikom ter odprtokodni standard JSON (JavaScript Object Notation) za izmenjavo podatkov.

2.3.1 Metoda Ajax (Asynchronous JavaScript and XML)

Za povezavo med uporabniškim vmesnikom in strežnikom smo uporabili protokol HTTP. Brskalnik strežniku običajno pošlje zahtevek tipa HTTP POST ali HTTP GET, strežnik pa brskalniku odgovori z ustrezno reprezentacijo zahtevka v obliki dokumenta HTML. Pri izgradnji spletne strani, predvsem pa spletne aplikacije, stremimo k temu, da se vsebina uporabniškega vmesnika v celoti osveži čim manjkrat. Namesto tega osvežimo le tisti del uporabniškega vmesnika, ki se dejansko spremeni. Na ta način prihranimo čas in vire. Ker protokol HTTP sam po sebi tega ne omogoča, smo se poslužili uporabe metode Ajax (Asynchronous JavaScript and XML), ki je pravzaprav skupek več metod. Omogoča nam pošiljanje asinhronih (v ozadju) zahtevkov za izvajanje skript na strežniku in prikaz njihovih rezultatov, poleg tega pa omogoča tudi hkratno izvajanje več skript.

Ajax je, kot smo že omenili, nabor več metod oziroma tehnologij. Mednje spadata tudi jezika HTML in CSS, ki skrbita za prikaz rezultatov v uporabniškem vmesniku. Skriptni jezik JavaScript poskrbi za nadzor nad objektnim

modelom dokumenta ter za interpretacijo akcij uporabnika. Jezik JavaScript v povezavi z objektom XMLHttpRequest poskrbi za komunikacijo ter izmenjavo podatkov med brskalnikom in strežnikom. Kljub imenu nam kot format za izmenjavo podatkov ni treba uporabiti standarda XML. Namesto tega se večkrat uporablja standard JSON ali (redkeje) kar dokument HTML ali navadno besedilo.

2.3.2 Standard JSON za prenos podatkov

JSON je preprost odprt standard za izmenjavo podatkov v tekstovni obliki. Zasnovan je bil za izmenjavo podatkov v berljivi obliki, izpeljan pa je iz skriptnega jezika JavaScript, kjer je uporabljen za predstavitev enostavnejših podatkovnih struktur in asociativnih polj, ki jim rečemo tudi objekti. Kljub temu da je izpeljan iz jezika JavaScript, pa je od njega neodvisen in se zato lahko uporablja za izmenjavo podatkov s poljubnim programskim jezikom. Na sliki 2.4 lahko vidimo preprosto podatkovno strukturo, predstavljeno s standardom JSON.

```
{
  "ime": "Bostjan",
  "priimek": "Lotric",
  "starost": 24,
  "telefon":
  [
    {
      "tip": "domaci",
      "stevilka": "04-123456"
    },
    {
      "tip": "mobilni",
      "stevilka": "031-555-555"
         }
  ]
}
```

Slika 2.4: Primer podatkovne strukture po standardu JSON.

2.4 Protokol OAuth

Obe socialni omrežji, ki ju bomo v tej diplomski nalogi obravnavali, za avtorizacijo uporabljata protokol OAuth (Open Authorization), zato se nam zdi pomembno, da ga podrobneje predstavimo že v tem poglavju.

OAuth je odprt standard za avtorizacijo, ki odjemalcu omogoča dostop do uporabnikovih zasebnih virov (npr. slik, video datotek, seznamov s kontakti itd.), shranjenih na določeni spletni strani, ne da bi uporabnik moral odjemalcu posredovati svoje podatke za dostop (tipično sta to uporabniško ime in geslo). Namesto podatkov za dostop uporabnik odjemalcu dodeli žeton (*angl. token*), ki odjemalcu dovoljuje dostop do določenega vira na določeni spletni strani za določen čas.

Protokol OAuth v postopku avtorizacije definira tri vloge:

1) Odjemalec je oseba, storitev ali aplikacija, ki od strežnika zahteva dostop do določenega vira. Odjemalec ni nujno lastnik vira, vendar pa za dostop do vira od lastnika potrebuje dovoljenje.

2) Strežnik, na katerem so viri shranjeni. Strežnik za vsak zahtevek za določen vir preveri, ali ima odjemalec pravice za dostop do tega vira.

3) Lastnik vira (uporabnik) odjemalcu lahko dovoli dostop do določenih virov za določen čas.

V klasičnem modelu avtorizacije odjemalec-strežnik mora uporabnik odjemalcu predati svoje podatke za dostop, kar pomeni potencialno nevarnost za uporabnika. Odjemalec lahko namreč s temi podatki dostopa do celotnega profila uporabnika. Pri protokolu OAuth pa uporabnik strežniku naroči, naj v uporabnikovem imenu odjemalcu izda posebno dovoljenje za dostop do njegovih virov. OAuth uporablja tri vrste dovoljenj za dostop:

1) Dovoljenje za dostop odjemalca se uporablja za avtentikacijo odjemalca in služi strežniku ali uporabniku, da dobita več informacij o odjemalcu ali pa odjemalcu dodelita posebne pravice.

2) Žeton za dostop se uporablja namesto uporabnikovih podatkov za dostop in ga odjemalcu, na podlagi dovoljenja lastnika vira, izda strežnik. Na ta način odjemalec nikoli ne izve uporabniškega imena in gesla lastnika vira. Žeton je predstavljen kot niz znakov, ki ga strežnik posreduje odjemalcu kot odgovor na zahtevek HTTP GET, s katerim je odjemalec žeton zahteval. Struktura žetona za dostop je lahko poljubna (definira jo ponudnik storitve), vendar pa mora biti odjemalcu nepoznana. Odjemalec lahko žeton hrani na različne načine (podatkovna baza, piškotek, podatkovne strukture itd.). Kot smo že omenili, je žeton ponavadi izdan le za določene vire ter je časovno omejen, zato mora odjemalec ob prenehanju veljavnosti žetona zahtevati novega. Lastnik

vira lahko odjemalcu kadarkoli odvzame žeton in s tem seveda tudi pravice za dostop do virov, ki jih je ta žeton omogočal.

3) Začasno dovoljenje za dostop se izda odjemalcu za identifikacijo zahtevka za avtorizacijo. Odjemalec torej v transakciji, s pomočjo katere pridobi žeton za dostop do uporabnikovih virov, uporablja začasno dovoljenje za dostop v povezavi s svojo skrivnostjo.

Delovanje protokola si podrobneje oglejmo na primeru. Recimo, da ima lastnik virov na neki spletni strani (ki igra vlogo strežnika) shranjene svoje slike. Odjemalec naj bo druga spletna stran, ki ponuja retuširanje slik. Odjemalec omogoča tudi, da uporabnik uvozi slike s prve strani. Ta postopek uvoza slik bo naš primer, na katerem bomo podrobneje predstavili postopek izdaje žetona za dostop.

Odjemalec od strežnika najprej zahteva začasno dovoljenje za dostop tako, da strežniku posreduje svojo skrivnost, ki služi za identifikacijo odjemalca. Strežnik začasno dovoljenje izda le v primeru, če je odjemalec uspešno identificiran. Uporabnik mora sedaj odjemalcu dovoliti dostop do virov, kar se ne zgodi na odjemalčevi spletni strani, temveč na spletni strani, kjer so shranjeni viri. Ko uporabnik potrdi odjemalčev zahtevek, odjemalec zamenja začasno dovoljenje za dostop za žeton za dostop (preko strežnika), ki ga lahko potem uporablja za dostop do uporabnikovih virov. Delovanje protokola je prikazano na sliki 2.5.

Slika 2.5: Potek delovanja protokola OAuth.

Poglavje 3

Vmesnik za programiranje Facebook

V tem poglavju bomo podrobneje predstavili socialno omrežje Facebook, njegov vmesnik za programiranje (Facebook API), različne vrste aplikacij, dotaknili se bomo dokumentacije za razvijalce ter opisali socialne vtičnike in protokol Open Graph.

3.1 Socialno omrežje Facebook

Socialno omrežje Facebook deluje od februarja leta 2004 in je v lasti podjetja Facebook. Omrežje je ustanovil Mark Zuckerberg s pomočjo svojih prijateljev. V začetku je bil Facebook na voljo le študentom univerze Harvard, vendar se je kmalu razširil po celem svetu in postal dostopen vsem, ki imajo dostop do interneta.

Uporabnik se mora za dostop do omrežja registrirati, nato pa lahko ustvari svoj profil, označi druge uporabnike kot prijatelje, si izmenjuje sporočila, objavlja spletne povezave, slike, zgodbe itd. Uporabniki se lahko združujejo v skupine s skupnim interesom, organizirajo dogodke in podobno.

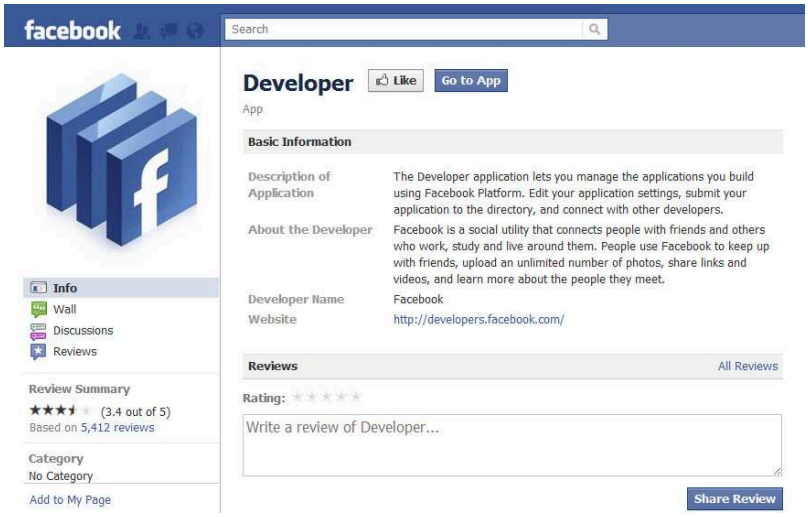
Trenutno ima Facebook približno 750 milijonov aktivnih uporabnikov, od katerih se jih polovica v omrežje prijavi vsak dan. Vsak mesec uporabniki objavijo več kot 30 milijard vsebin (spletnih povezav, novic, slik, video datotek itd.) in namestijo več kot 600 milijonov aplikacij.

Facebookov glavni vir dohodka je oglaševanje. Podjetja lahko objavijo oglase, ki jih vidi le določena skupina uporabnikov (ciljno oglaševanje), na primer samo moški, stari od 25 do 35 let, ki jih zanimajo avtomobili. V

letu 2010 je Facebook na ta način ustvaril približno dve milijardi dolarjev prihodkov.

3.2 Splošno o aplikacijah

Za razvoj aplikacije za omrežje Facebook sta potrebna dva pogoja. Prvi pogoj je, da razvijalec potrebuje svoj uporabniški račun, drugi potrebni pogoj pa je namestitev posebne aplikacije, ki jo je za potrebe razvijalcev pripravil Facebook. Govorimo o aplikaciji z imenom Developer (*slov. Razvijalec*), ki je namenjena upravljanju lastnih aplikacij [2]. S pomočjo te aplikacije lahko ustvarimo lastno aplikacijo, spreminjamo nastavitve obstoječih aplikacij, jih brišemo, objavimo v register itd. Vsaka aplikacija ima, podobno kot uporabniki, na omrežju svoj profil. Na sliki 3.1 je prikazan profil aplikacije Developer.



Slika 3.1: Profilna stran aplikacije Developer.

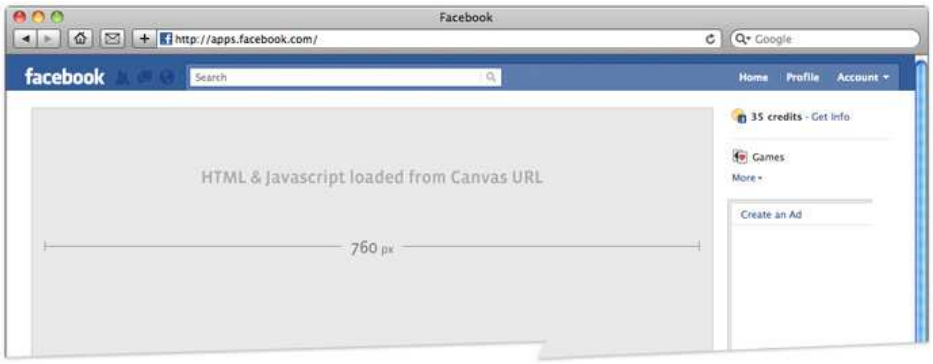
Ob registraciji se aplikaciji dodelita dva pomembna atributa. To sta identifikacijska številka aplikacije (*angl. application ID*) ter skrivnost aplikacije (*angl. application secret*). Identifikacijska številka je unikaten niz številk in se nikoli ne spremeni. Uporablja se ob raznih zahtevkih za identifikacijo aplikacije. Skrivnost aplikacije je naključen niz črk in številk, ki se uporablja v kombinaciji z identifikacijsko številko pri avtorizaciji aplikacije. Skrivnost aplikacije lahko razvijalec ponastavi s pomočjo aplikacije Developer. Identifikacijska številka aplikacije je javnega značaja in je vidna vsem, medtem ko je

skrivnost aplikacije, kot pove že ime samo, skrivna in je zato ne smemo razkriti nikomur.

Med osnovnimi nastavitvami aplikacije najdemo različne podatke, kot so ime, opis, jezik, kategorija, ikona aplikacije itd. Kot obvezen podatek moramo navesti kontaktni elektronski naslov in spletni naslov, kjer se nahaja dokument o politiki zasebnosti (*angl. privacy policy*). Facebook navaja dokaj stroga pravila za razvijalce, ki naj bi uporabnike varovali pred škodljivimi ali neprimernimi vsebinami. V primeru kršitve teh pravil Facebook aplikacijo izbriše. Izbrisanih aplikacij ni več mogoče obnoviti, razvijalec pa seveda lahko ustvari novo (identično) aplikacijo.

Vsebina aplikacije je na omrežju Facebook lahko prikazana na dva načina:

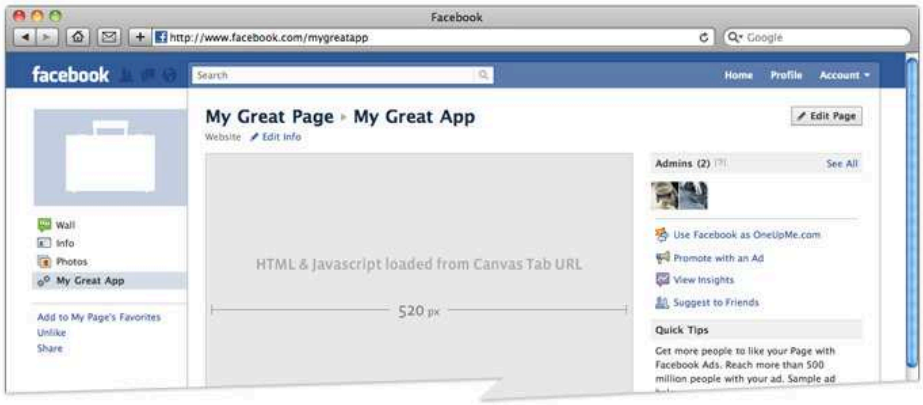
a) Platno (angl. Canvas Page) – platno je stran znotraj omrežja Facebook, na kateri se prikaže naša aplikacija. V nastavitvah aplikacije je potrebno podati le spletni naslov, s katerega zahteva vsebino v obliki dokumenta HTML v povezavi s slogovnim jezikom CSS in skriptnim jezikom JavaScript. Vsebina aplikacije se prikaže znotraj elementa `<iframe>`, ki je po širini omejen na 760 slikovnih točk (*angl. pixels*).



Slika 3.2: Prikaz izgleda uporabniškega vmesnika platna.

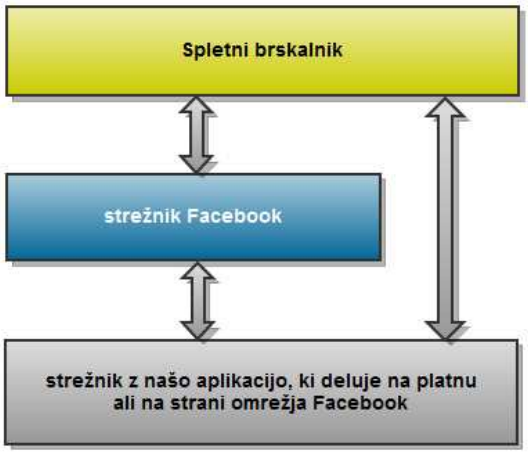
b) Zavihek na strani Facebook (angl. Page Tab) – stran Facebook je v tem odstavku mišljena kot stran, ki jo lahko v okviru omrežja Facebook ustvari vsak uporabnik. Stran služi kot javni profil za organizacije, podjetja, društva, slavne osebe, produkte, stvari itd. Administratorji lahko na svojo stran namestijo poljubne aplikacije, ki se na strani pojavijo kot eden izmed zavihkov. Vsebina aplikacije se zopet prikaže znotraj elementa `<iframe>`, ki pa je zaradi uporabniškega vmesnika strani (prikazan na sliki 3.3) Facebook še ožji od platna (520 slikovnih točk).

V obeh primerih se celotna koda aplikacije izvaja na našem strežniku, ne



Slika 3.3: Prikaz izgleda uporabniškega vmesnika zavihka.

pa na strežnikih omrežja Facebook. Prav tako nam vsebine aplikacije ni treba prikazati v okviru Facebooka. V tem primeru bi aplikacija delovala kot spletna aplikacija, spletna stran ali mobilna aplikacija. Na sliki 3.4 so prikazane komponente, ki so potrebne za delovanje aplikacije, in interakcije med njimi.



Slika 3.4: Arhitektura sistema aplikacije, ki deluje v omrežju Facebook.

Vsem oblikam aplikacij za omrežje Facebook je skupno, da na takšen ali drugačen način uporabljajo vmesnik za programiranje omrežja. Razlogov za uporabo vmesnika je več, med najpogostejšimi pa najdemo izboljšanje uporabniške izkušnje, promocijo in hitrejše širjenje aplikacije ali pa preprosto kot vir zaslužka.

V nadaljevanju bomo predstavili nekaj glavnih konceptov pri izdelavi aplikacije za socialno omrežje Facebook, predvsem se bomo posvetili vmesniku za programiranje z imenom Graph (Graph API), socialnim vtičnikom in protokolu Open Graph.

3.3 Vmesnik za programiranje Graph

Graph API je dobil ime po socialnem grafu (*angl. social graph*), ki predstavlja jedro omrežja Facebook. V grafu so zajeti vsi objekti (uporabniki, slike, dogodki, strani itd.), ki se pojavljajo na omrežju, in povezave med njimi (prijateljstva, skupen interes, lastništvo virov itd.).

3.3.1 Objekt v grafu

Vsak objekt v grafu ima unikatno identifikacijsko številko. S pomočjo te številke lahko dostopamo do lastnosti objekta. Nekatere lastnosti so javno dostopne, ostale pa so zasebne.

Če želimo dostopati do lastnosti objekta, pošljemo zahtevek tipa HTTP GET na spletni naslov `https://graph.facebook.com/ID`, kjer oznaka ID predstavlja identifikacijsko številko objekta. Graph API nam kot odgovor vedno vrne objekt JSON (primer odgovora na sliki 3.5). Primeri objektov: *uporabnik, stran, dogodek, skupina, aplikacija, sporočilo, slika, album, slika profila, video, opomba, itd.*

```
{
   "id": "1303509052",
   "name": "Bo\u0161tjan Lotri\u010d",
   "first_name": "Bo\u0161tjan",
   "last_name": "Lotri\u010d",
   "link": "http://www.facebook.com/lotko",
   "username": "lotko",
   "gender": "male",
   "locale": "en_US"
}
```

Slika 3.5: Primer odgovora Graph API na zahtevek za lastnosti objekta.

3.3.2 Povezave objekta

Za vsak objekt lahko pogledamo njegove povezave (*angl. connections*) z drugimi objekti. Vsak tip objekta ima drugačne povezave z objekti. Uporabnik je lahko na primer prijatelj z drugimi uporabniki, medtem ko slika (kot objekt) ne more biti. Povezave objekta spet zahtevamo s pošiljanjem zahtevka na spletni naslov `https://graph.facebook.com/ID/POVEZAVA`, kjer oznaka ID predstavlja identifikacijsko številko objekta, oznaka POVEZAVA pa predstavlja vrsto povezave za objekt. V primeru branja povezav nam Facebook namesto objekta JSON vrne polje objektov JSON. Druga razlika v primerjavi z zahtevkom za lastnosti objekta je, da je pri zahtevku za povezave vedno potrebno podati žeton za dostop (*angl. access token*), ki deluje kot pooblastilo, da smemo dostopati do želenih podatkov. O žetonu za dostop smo že govorili v poglavju 2.4. Celoten zahtevek HTTP GET (v obliki spletnega naslova) bi torej izgledal tako: `https://graph.facebook.com/ID/POVEZAVA?access_token=ZETON`, kjer oznaka ZETON predstavlja vrednost žetona za dostop. V primeru, da v zahtevku ne bi podali parametra `access_token`, bi nam Graph API vrnil sporočilo z napako, ki je prikazano na sliki 3.6.

```
{
   "error": {
      "type": "OAuthException",
      "message": "An access token is required
                 to request this resource."
   }
}
```

Slika 3.6: Odgovor Graph API na zahtevek brez žetona za dostop.

Žeton za dostop je potrebno podati tudi pri branju zasebnih lastnosti objekta. Običajno aplikacija ne potrebuje vseh zasebnih lastnosti, temveč le določen nabor teh lastnosti. Za vsako zasebno lastnost pa potrebujemo uporabnikovo dovoljenje (*angl. permission*).

3.3.3 Dovoljenja za dostop

Dovoljenje se običajno nanaša na določen vir, do katerega lahko dostopamo, ali na akcijo, ki jo lahko v imenu uporabnika izvedemo. Facebook je dovoljenja razdelil v štiri skupine:

1) Osnovna dovoljenja zajemajo dovoljenja za lastnosti, ki so zasebne, a ne spadajo v nobeno drugo skupino.

2) Dovoljena za podatke uporabnika zajemajo dovoljenja za vse osebne podatke uporabnika, kot tudi vse njegove povezave (njegove dogodke, slike, strani itd.).

3) Dovoljena za podatke prijateljev zajemajo vsa zasebna dovoljenja, ki jih najdemo v drugi skupini, za branje podatkov uporabnikovih prijateljev.

4) Razširjena dovoljenja zajemajo dovoljenja za posebne podatke uporabnika in za vse akcije, ki jih lahko aplikacija izvaja v imenu uporabnika. Sem sodijo na primer dovoljenje za objavo na uporabnikov zid, branje uporabnikovih novic, objava slik in video datotek itd.

Takšna razdelitev dovoljenj nam omogoča, da kot razvijalec zahtevamo dovoljenja le za tiste podatke, ki jih resnično potrebujemo za delovanje naše aplikacije. Za aplikacijo ni potrebno, da vsa dovoljenja zahteva že prvič, temveč lahko med izvajanjem aplikacije zahtevamo dodatna dovoljenja. Če se na primer akcija pisanja na uporabnikov zid v naši aplikaciji pojavi zelo redko oziroma se pojavi pozno v toku dogodkov, je smiselno, da dovoljenje za to akcijo zahtevamo šele takrat, ko do te akcije pride, in ne na začetku izvajanja aplikacije.

3.3.4 Uporaba filtrov

Pri branju lastnosti in povezav objekta nam Graph API omogoča uporabo filtrov, ki nam olajšajo razvoj in pohitrijo delovanje aplikacije. Nekaj primerov filtrov, njihove uporabe in prednosti, ki jih prinašajo:

a) Ožji nabor lastnosti – če zahtevku za branje lastnosti dodamo parameter `fields`, katerega vrednost so imena lastnosti, ločena z vejico, nam bo Graph API v odgovoru posredoval le tiste lastnosti, ki smo jih zahtevali. Primer takega zahtevka je sledeč: `https://graph.facebook.com/ID/?fields=LASTNOST1,LASTNOST2`, kjer je oznaka ID identifikacijska številka objekta, oznaki LASTNOST1 in LASTNOST2 pa sta imeni lastnosti. Prednost tega filtra je v hitrosti, saj zahtevamo le tiste lastnosti, ki jih dejansko potrebujemo in se na ta način izognemu prenosu nepotrebnih podatkov.

b) Zahtevek za lastnosti več objektov – v enem zahtevku lahko beremo lastnosti več objektov hkrati. To nam omogoča parameter `ids`, ki ga uporabimo namesto identifikacijske številke objekta. Vrednost parametra `ids` je seznam identifikacijskih številk objektov, ločenih z vejico. Primer zahtevka: `https://graph.facebook.com/?ids=ID1,ID2`, kjer sta oznaki ID1 ter ID2 identifikacijski številki objektov, katerih lastnosti želimo prebrati. Prednost

tega filtra je zopet hitrost, saj namesto več zahtevkov pošljemo samo enega in s tem prihranimo čas, ki bi drugače nastal zaradi dodatne režije (*angl. overhead*).

c) Uporaba identifikatorja `me` – Velikokrat želimo dostopati do lastnosti ali povezav trenutnega uporabnika aplikacije. Namesto uporabe njegove identifikacijske številke lahko uporabimo poseben identifikator `me`, ki ga Graph API prevede kot identifikacijsko številko trenutnega uporabnika. Ker identifikator `me` sam po sebi ne nosi nobene informacije o uporabniku, je ob uporabi tega identifikatorja potrebno podati tudi žeton za dostop, ki je odjemalcu izdan s pooblastiom uporabnika, zato nosi vso potrebno informacijo o uporabniku, ki jo Graph API potrebuje. Uporaba identifikatorja `me` ne predstavlja nobene zmogljivostne prednosti, poenostavi pa pisanje kode. Primer zahtevka za branje lastnosti trenutnega uporabnika: `https://graph.facebook.com/me?access_token=ZETON`, kjer oznaka ZETON predstavlja vrednost žetona za dostop.

d) Slike objekta – Pogosto moramo v uporabniškem vmesniku aplikacije prikazati sliko nekega objekta. Graph API nam za vsak objekt omogoča enostaven prikaz slike, če ta seveda obstaja. To storimo z uporabo posebnega identifikatorja `picture`, ki ga v zahtevku (v obliki spletne povezave) postavimo za identifikacijsko številko objekta: `https://graph.facebook.com/ID/picture`, kjer oznaka ID predstavlja identifikacijsko številko objekta. Poudariti moramo, da v tem primeru Graph API ne vrne objekta JSON, temveč sliko.

e) Ostranjevanje (angl. Paging) – Pri branju povezav objekta lahko omejimo njihovo število z uporabo parameta `limit`, ki poda največje število elementov, ki jih Graph API vrne. Pri ostranjevanju se parameter `limit` skoraj vedno uporablja v povezavi s parametrom `offset`, ki pove, koliko začetnih elementov naj Graph API spusti. Če na primer želimo prebrati podatke o desetih prijateljih trenutnega uporabnika, pri čemer jih prvih deset spustimo, bi zahtevek izgledal tako: `https://graph.facebook.com/me/friends?limit=10&offset=10&access_token=ZETON`, kjer oznaka ZETON predstavlja vrednost žetona za dostop.

3.3.5 Akcije nad objekti

S pomočjo funkcionalnosti Graph API lahko v okviru aplikacije izvajamo tri vrste akcij nad objekti. Radi bi poudarili, da za vsako izmed teh akcij potrebujemo veljaven žeton za dostop.

1) Ustvarjanje objektov – S pomočjo Graph API lahko ustvarimo le določene vrste objektov in sicer tiste, ki so vezani na neki drug objekt (višje stop-

nje), ki je prav tako v lasti uporabnika. Tako na primer ne moremo ustvariti novega uporabnika, ker uporabnik ni vezan na noben objekt višje stopnje (ki bi bil v lasti uporabnika). Lahko pa ustvarimo nov album slik, saj je album vezan na uporabnika. Pri ustvarjanju objekta namesto zahtevka tipa HTTP GET pošljemo zahtevek tipa HTTP POST, v katerem podamo identifikacijsko številko objekta, na katerega bo vezan novo ustvarjeni objekt, ter tip objekta, ki ga želimo ustvariti. Poleg identifikacijske številke in tipa tega objekta podamo tudi tiste parametre, ki jih Graph API potrebuje za izvedbo te akcije. Točne specifikacije si razvijalec lahko prebere v dokumentaciji za Graph API.

2) Ustvarjanje povezav med objekti – Ustvarjanje povezav med objekti je po tehnični plati zelo podobno kot ustvarjanje objektov samih. Razlika je v tem, da povezavo vedno ustvarimo med uporabnikom in obstoječim objektom. V zahtevku HTTP POST, ki ga pošljemo, torej podamo identifikacijsko številko želenega objekta ter tip povezave, ki jo želimo vzpostaviti. Dodatni parametri v tem primeru niso potrebni.

3) Brisanje objektov – Graph API nam omogoča tudi brisanje objektov, ki so v lasti uporabnika. Izbrišemo lahko tudi povezave med uporabnikom in temi objekti. Za brisanje objekta moramo poslati zahtevek tipa HTTP DELETE ali pa zahtevek tipa HTTP POST, ki vsebuje dodaten parameter `method` z vrednostjo `delete`. Poleg tega moramo v zahtevku podati tudi identifikacijsko številko objekta ter tip povezave, če brišemo povezavo med objektom in uporabnikom.

Za generiranje zahtevkov tipa HTTP POST Facebook priporoča uporabo knjižnice *libcurl*, ki med mnogimi različnimi protokoli podpira tudi delo s protokolom HTTP. Knjižnica je prenosljiva in zato deluje na različnih platformah. Običajno se uporablja skupaj z orodjem za delo z ukazno vrstico *cURL*.

V naslednjem podpoglavju bomo na kratko predstavili dokumentacijo objektov Graph API [6].

3.4 Dokumentacija za objekte v Graph API

Dokumentacijo za objekte bomo predstavili na primeru dokumentacije za objekt *slike (angl. photo)*. Dokumentacija je razdeljena na naslednje razdelke:

- začetni razdelek,
- lastnosti objekta,
- povezave objekta,

- razdelek z opisom akcij za posamezno povezavo,
- uporabne povezave in primeri,
- komentarji razvijalcev.

V začetnem razdelku je napisano, katera dovoljenja potrebujemo za branje lastnosti določene različice objekta. Pri sliki imamo na primer pet različic: *javna slika; slika uporabnika; slika, na kateri je bil uporabnik označen; slika uporabnikovega prijatelja ter slika, na kateri so bili označeni uporabnikovi prijatelji.* Za vsako od teh različic potrebujemo drugačno dovoljenje.

V naslednjem razdelku so opisane lastnosti objekta. Prikazane so v tabeli (slika 3.7) s štirimi stolpci: *'ime'*, *'opis'*, *'dovoljenja'* in *'vrača'*. V stolpcu *'dovoljenja'* so zapisana različna dovoljenja, ki jih potrebujemo za branje te specifične lastnosti, glede na različico objekta. Stolpec *'vrača'* nam pove, v kakšnem formatu bo Graph API vrnil to lastnost.

Fields

The `Photo` object has the following fields.

Name	Description	Permissions	Returns
`id`	The photo ID	generic `access_token` or `user_photos` or `friend_photos` or `user_photo_video_tags` or `friends_photo_video_tags`	`string`
`from`	The profile (user or page) that posted this photo	generic `access_token` or `user_photos` or `friend_photos` or `user_photo_video_tags` or `friends_photo_video_tags`	object containing `id` and `name` fields

Slika 3.7: Primer tabele v dokumentaciji objekta Graph API.

V tretjem razdelku so naštete povezave, ki jih ima lahko objekt. Naštete so samo povezave s podrejenimi objekti. V primeru slike imamo tako povezavo s komentarji, nimamo pa povezave z albumi. Povezave so predstavljene v enaki tabeli kot lastnosti objekta.

V četrtem razdelku so opisane akcije, ki jih lahko izvajamo nad povezavami objekta. Tako lahko na primer nad povezavo slike s komentarji izvajamo akcijo `create`, torej lahko ustvarimo nov komentar za to sliko. Dokumentacija nam pove, kateri parametri so za določeno akcijo na voljo, kakšnega tipa so in ali so obvezni ali ne. Poleg tega nam dokumentacija pove še, kakšen odgovor bomo prejeli v primeru uspešno izvedene akcije.

Predzadnji razdelek se ne pojavi pri vseh objektih, kjer pa se, vsebuje spletne povezave do uporabnih primerov ali navodil.

Zadnji razdelek je seznam komentarjev razvijalcev.

Na tem mestu bi radi opozorili na nekaj pomankljivosti celotne dokumentacije za Facebook. Celotna dokumentacija je pretežno opisna, torej predstavljena s pomočjo velike količine besedila, kar za razvijalca v začetku izgleda kot prednost, vendar se kmalu opazi pomanjkanje bolj tehničnih definicij funkcionalnosti. Določene funkcionalnosti so slabo razložene, opazili pa smo celo nekaj funkcionalnosti, ki sploh niso obrazložene. Dokumentacija objektov, ki smo jo predstavili v tem podpoglavju, je eden boljših primerov dokumentacije Facebooka, a tudi to dokumentacijo bi lahko izboljšali (na primer z več primeri uporabe).

Druga slabost dokumentacije je slaba vzdrževanost. Velikokrat namreč naletimo na primere uporabe, ki ob testiranju v aplikaciji ne delujejo. Med besedilom najdemo spletne povezave, ki nas ob kliku preusmerijo na stran, ki ne obstaja. Te pomankljivosti so eden izmed razlogov, ki pripomorejo k slabšemu razvoju aplikacij za socialno omrežje Facebook ter nezadovoljstvu programerjev, ki ga je občutiti v tej skupnosti.

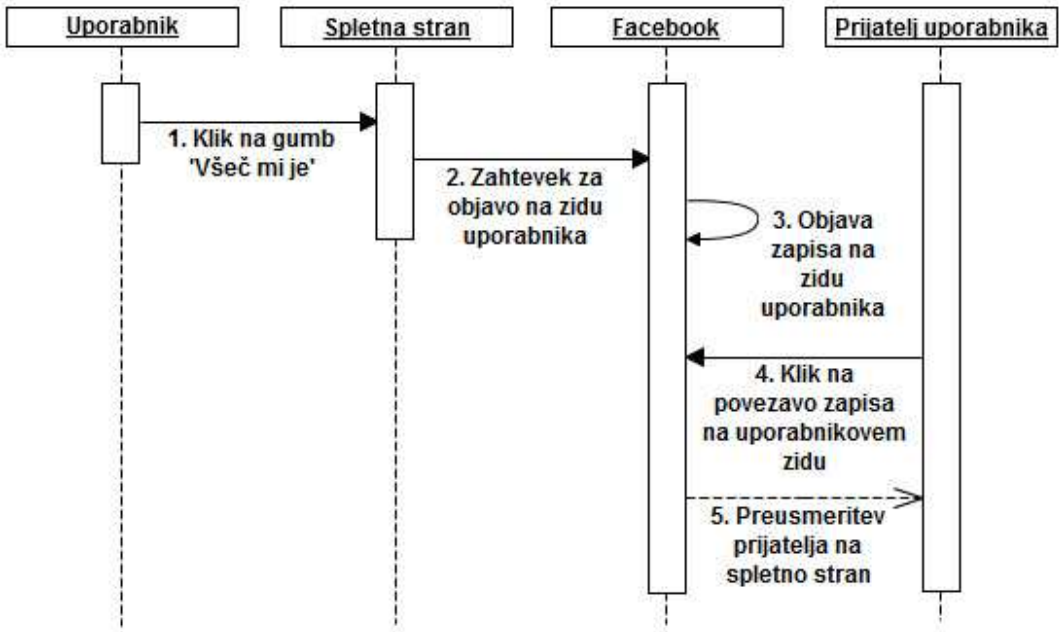
3.5 Socialni vtičniki

V tem poglavju bomo podrobneje predstavili socialne vtičnike za omrežje Facebook. Pogledali si bomo, kaj socialni vtičniki sploh so, kako se jih uporablja, ter kakšen je njihov pomen.

Socialni vtičniki so predpripravljeni elementi uporabniškega vmesnika, ki jih lahko enostavno vgradimo. So najenostavnejši način uporabe funkcionalnosti, ki jih ponuja socialno omrežje Facebook, saj lahko vtičnike uporablja tudi nekdo, ki sicer ne zna razvijati aplikacij. Facebook na straneh dokumentacije namreč ponuja obrazce za generiranje kode vtičnikov. V obrazec preprosto vnesemo želene nastavitve, ki pripadajo določenemu vtičniku, kot odgovor pa dobimo kodo HTML, ki jo nato vstavimo v svoj dokument HTML. Vizualne lastnosti vtičnikov so v celoti določene s strani Facebooka, razvijalec lahko vpliva le na postavitev, velikost in jezikovno različico.

Glavni namen socialnih vtičnikov je širjenje prepoznavnosti oziroma prisotnosti določene vsebine. Vsak vtičnik ima parameter *spletna povezava*, ki predstavlja spletno povezavo tiste vsebine, ki bi jo radi promovirali. Recimo, da imamo na spletni strani predstavljen nek izdelek. Če sedaj določenemu spletnemu vtičniku, na primer *gumbu 'Všeč mi je'*, za parameter *spletna po-*

vezava nastavimo spletno povezavo tega izdelka, se bo ob kliku obiskovalca spletne strani na ta gumb na zidu obiskovalca pojavil zapis, da je obiskovalcu všeč naš izdelek. Ta zapis vidijo tudi obiskovalčevi prijatelji, ki so ob kliku na spletno povezavo, ki je vsebovana v zapisu, preusmerjeni nazaj na našo spletno stran. Na ta način smo na svojo spletno stran s pomočjo socialnega vtičnika pripeljali novega obiskovalca, ki je v nekaterih primerih lahko tudi potencialna stranka. Na sliki 3.8 je prikazan postopek delovanja *gumba 'Všeč mi je'*.



Slika 3.8: Postopek delovanja *gumba 'Všeč mi je'*.

Za implementacijo vtičnikov obstajata dve metodi. Prva metoda je s pomočjo uporabe označevalnega jezika XFBML (primer na sliki 3.9), ki ga je razvil Facebook. Ta metoda za svoje delovanje potrebuje uporabo paketa za razvoj programske opreme JavaScript, o katerem bomo več povedali kasneje. Pri uporabi druge metode nam Facebook pripravi element HTML `<iframe>`, ki je popolnoma samostojen in ga preprosto vstavimo v naš dokument HTML.

Posebnost med vtičniki je vtičnik za registracijo, ki se od ostalih razlikuje po namenu uporabe. Uporablja se ga namreč, kot namiguje že njegovo ime, pri procesu registracije uporabniškega računa uporabnikov. Z uporabo tega vtičnika lahko uporabniku poenostavimo izpolnjevanje obrazca za registracijo. Vtičnik omogoča avtomatski vnos določenih podatkov v obrazec. V nastavitvah definiramo polja, ki jih želimo imeti v obrazcu (ta polja so popolnoma

```
<fb:like href="www.example.com"
  send="false"
  width="450"
  show_faces="false"
  font="">
</fb:like>
```

Slika 3.9: Primer kode označevalnega jezika XFBML.

poljubna), vtičnik pa bo določena polja avtomatsko zapolnil s podatki uporabnika, ki jih pridobi preko vmesnika za programiranje Facebook. Vtičnik bo polja zapolnil samo v primeru, da je uporabnik prijavljen v socialno omrežje Facebook. V nasprotnem primeru polja ostanejo prazna. Tako lahko obrazec uporabljajo tudi uporabniki, ki nimajo uporabniškega računa na omrežju Facebook, zato razvijalcu ni potrebno narediti ločenega procesa registracije za te uporabnike. Primeri polj, ki jih ta vtičnik lahko izpolni: *ime, rojstni datum, elektronski naslov, spol itd.* Vtičnik nam omogoča tudi uporabo testa *Captcha*, ki služi za preprečevanje avtomatiziranega izpolnjevanja obrazca.

Socialni vtičniki predstavljajo najpreprostejšo obliko uporabe vmesnika za programiranje Facebook, kar se odraža tudi pri njihovi razširjenosti. Čeprav je bila uporaba vtičnikov mogoča šele v aprilu leta 2010, jih je do tega trenutka integriralo že več kot dva milijona spletnih strani [7].

3.6 Protokol Open Graph

Protokol Open Graph se uporablja v povezavi s socialnimi vtičniki. Omogoča integracijo spletne strani v socialni graf Facebooka. Z uporabo elementov, ki jih definira protokol Open Graph, dosežemo, da Facebook našo stran vidi kot objekt v svojem grafu. V primeru, da uporabnik klikne na gumb *'Všeč mi je'*, ki se nahaja na tej strani, se med uporabnikom in našo stranjo (objektom v grafu, ki predstavlja našo stran) ustvari povezava. Trenutno je protokol razvit do te mere, da omogoča uporabo svojih elementov le na straneh, ki predstavljajo stvari iz resničnega sveta (osebnost, produkt, kraj itd.). V elementih Open Graph podamo podatke, ki definirajo, kako bo Facebook prikazal objekt, ki predstavlja našo stran. To je še posebej uporabno v primeru, ko uporabnik želi povabiti svoje prijatelje k uporabi naše spletne strani oziroma spletne aplikacije. Brez uporabe protokola Open Graph bo prijatelj dobil generično sporočilo s povezavo na našo stran, ob uporabi protokola pa bo prejel lepo

oblikovano povabilo s sliko in spremnim besedilom, ki smo ga določili mi.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:og="http://ogp.me/ns#"
      xmlns:fb="http://www.facebook.com/2008/fbml">
  <head>
    <title>The Rock (1996)</title>
    <meta property="og:title" content="The Rock"/>
    <meta property="og:type" content="movie"/>
    <meta property="og:url"
      content="http://www.imdb.com/title/tt0117500/"/>
    <meta property="og:image"
      content="http://ia.media-imdb.com/rock.jpg"/>
    <meta property="og:site_name" content="IMDb"/>
    <meta property="og:description"
          content="A group of U.S. Marines ..."/>
  </head>
  ...
</html>
```

Slika 3.10: Primer uporabe protokola Open Graph.

Na sliki 3.10 je prikazan primer uporabe protokola Open Graph. Pogoj za delovanje protokola je uporaba posebnih parametrov, ki jih vstavimo v element `<html>`. Same podatke, ki jih Facebook kasneje uporabi za prikaz objekta, ki predstavlja našo stran, podamo v elementih `<meta>`. Tako na primer lahko določimo prikazno sliko objekta z uporabo elementa: `<meta property='og:image' content='http://ia.media-imdb.com/rock.jpg'/>`.

Podatkov, ki jih lahko podamo s pomočjo elementov `<meta>`, je veliko, obvezno pa moramo podati štiri podatke: `og:title, og:type, og:image` ter `og:url`. Pri tem `og:url` še dodatno izstopa, saj se uporablja namesto identifikacijske številke objekta v socialnem grafu Facebook.

Za preverjanje pravilnosti uporabe elementov Open Graph je Facebook pripravil orodje *Facebook URL Linter*, ki nam za določen spletni naslov pove, kateri elementi Open Graph so uporabljeni in ali so uporabljeni pravilno.

3.7 Paketi za razvoj programske opreme

Facebook razvijalcem ponuja več paketov za razvoj programske opreme (*angl. Software development kit*, SDK), med katerimi je najpomembnejši oz. najbolj uporaben paket za skriptni jezik JavaScript, ki ga bomo tudi natančneje opisali.

3.7.1 JavaScript SDK

Paket za razvoj programske opreme za skriptni jezik JavaScript omogoča razvijalcu dostop do vseh funkcionalnosti vmesnika Graph API, poleg tega pa paket omogoča tudi uporabo elementov jezika XFBML. Paket sestavljajo funkcije, ki na tak ali drugačen način uporabljajo vmesnik za programiranje Facebook. Ker gre za uporabo jezika JavaScript, se vsi zahtevki pošljejo iz brskalnika odjemalca. Kot smo že omenili v poglavju 2.1.2, je JavaScript dogodkovno usmerjen jezik, kar nam omogoča, da ob izvedbi neke akcije uporabnika kličemo tej akciji ustrezno funkcijo. Tako lahko na primer ob kliku uporabnika na gumb za prijavo v spletno stran kličemo funkcijo `login()`, ki zgenerira ter prikaže modalno okno, ki vsebuje obrazec za prijavo v omrežje Facebook. JavaScript SDK je še posebej uporaben pri razvoju spletnih aplikacij, saj dogodkovno usmerjeno izvajanje približa uporabniško izkušnjo tisti, ki smo je navajeni iz namiznih aplikacij.

Za inicializacijo paketa so potrebni trije koraki:

1) V dokumentu HTML moramo z uporabo elementa `<script>` naložiti skripto JavaScript, ki predstavlja paket JavaScript SDK. To skripto najdemo na strežniku Facebook in je na voljo v več jezikovnih različicah.

2) V objektni model dokumenta moramo vstaviti prazen element `<div>`, ki ima nastavljen parameter `id` z vrednostjo `"fb-root"`.

3) Poklicati moramo funkcijo `init()`, v kateri kot parameter posredujemo identifikacijsko številko aplikacije.

Na sliki 3.11 vidimo primer kode za inicializacijo paketa za razvoj programske opreme JavaScript z običajnimi nastavitvami. Spremenljivka `FB`, ki jo vidimo na sliki, predstavlja objekt, ki ga ustvari skripta `all.js`. Oznaka ID_APLIKACIJE na sliki predstavlja identifikacijsko številko aplikacije.

Identifikacijsko številko aplikacije najdemo med podatki o aplikaciji, ki jih vidimo z uporabo aplikacije *Developer*. V primeru generirane kode socialnih vtičnikov Facebook v ozadju avtomatsko registrira aplikacijo, njeno identifikacijsko številko pa nam posreduje v generirani kodi.

Izmed funkcij paketa sta najbolj pomembni oziroma najbolj uporabljeni dve funkciji: `FB.api()` in `FB.ui()`.

```
<div id="fb-root"></div>
<script src="http://connect.facebook.net/sl_SI/all.js">
</script>
<script>
  FB.init({
    appId  : 'ID_APLIKACIJE',
    status : true, // preveri status uporabnika
    cookie : true, // omogoči piškotke
    xfbml  : true  // omogoči procesiranje jezika XFBML
  });
</script>
```

Slika 3.11: Primer kode za inicializacijo JavaScript SDK.

a) FB.api() je namenjena pošiljanju zahtevkov vmesniku Graph API. S to funkcijo lahko pošiljamo vse zahtevke in izvajamo vse akcije, opisane v razdelku 3.3. Primer uporabe funkcije je prikazan na sliki 3.12.

```
FB.api('/platform/posts', { limit: 3 }, function(response) {

  // Za vsako izmed treh sporočil izpiši vsebino sporočila
  // oziroma naslov priponke, če ne gre za besedilno sporočilo.
  for (var i=0, l=response.length; i<l; i++) {
    var post = response[i];
    if (post.message) {
      alert('Message: ' + post.message);
    } else if (post.attachment && post.attachment.name) {
      alert('Attachment: ' + post.attachment.name);
    }
  }
});
```

Slika 3.12: Primer uporabe funkcije FB.api().

b) FB.ui() je funkcija, ki se uporablja za generiranje in prikaz modalnih oken, ki jih Facebook imenuje dialogi (*angl. dialogs*). Dialoge prikažemo v primeru, če želimo izvesti eno izmed naslednjih akcij:

- objava zgodbe na uporabnikov zid,
- uporabniku ponudimo, ali želi drugega uporabnika dodati za prijatelja,

- uporabniku ponudimo možnost, da avtorizira našo aplikacijo,
- uporabniku prikažemo obrazec za plačevanje (produkta ali storitve),
- uporabniku prikažemo obrazec za pošiljanje vabil za uporabo naše aplikacije,
- uporabniku prikažemo obrazec za objavo spletne povezave.

Kot smo že omenili v poglavju 3.5, se paket za razvoj programske opreme JavaScript potrebuje pri uporabi označevalnega jezika XFBML. Paket po inicializiciji pregleda objektni model dokumenta in v njem poišče vse elemente XFBML s pripadajočimi parametri, nato pa pošlje zahtevek HTTP GET vmesniku za programiranje Facebook, ki ob pravilnem zahtevku vrne gradnik uporabniškega vmesnika, ki je reprezentacija elementa XFBML v obliki kode HTML. Za delovanje te funkcionalnosti pa je pri inicializaciji paketa JavaScript SDK potrebno podati parameter `xfbml` z vrednostjo `true`.

3.7.2 Ostali paketi za razvoj programske opreme

Poleg paketa za razvoj programske opreme za skripni jezik JavaScript nam Facebook ponuja še tri pakete:

- PHP SDK,
- iOS SDK,
- Android SDK.

Vsi trije paketi sicer omogočajo enake funkcionalnosti kot paket za jezik JavaScript, vendar se razlikujejo v načinu izvedbe pošiljanja zahtevkov, saj je okolje, v katerem se paketi izvajajo, drugačno. Za razliko od paketa JavaScript SDK je potrebno datoteke teh paketov namestiti na strežnik (v primeru paketa PHP SDK), kjer se izvajajo skripte PHP, oziroma morajo biti prisotne med datotekami mobilne aplikacije (v primeru paketov iOS SDK ali Android SDK).

Poglavje 4

Twitter API

4.1 Socialno omrežje Twitter

Socialno omrežje Twitter je leta 2006 ustanovil Jack Dorsey in je v lasti podjetja Twitter. Omrežje ponuja uporabnikom branje in pošiljanje sporočil, omejenih na dolžino 140 znakov. Uporabniki lahko berejo sporočila drugih uporabnikov, nanje odgovorijo s svojim sporočilom ali pa jih posredujejo drugim uporabnikom. Twitter je na voljo vsem uporabnikom svetovnega spleta, ki pa se morajo pred uporabo registrirati.

V omrežju je trenutno registriranih okoli 200 milijonov uporabnikov, ki ustvarijo več kot 200 milijonov sporočil dnevno. Število uporabnikov se dnevno poveča za skoraj pol milijona.

Twitter svoje prihodke ustvarja s pomočjo oglaševanja, ki so ga v omrežju omogočili šele sredi leta 2010. Kljub temu je Twitter v preostali polovici leta ustvaril 45 milijonov dolarjev prihodka.

4.2 Splošno o aplikacijah

Aplikacije za omrežje Twitter pri svojem delovanju uporabljajo tri glavne objekte socialnega omrežja:

- objekt sporočila, imenovanega tudi Tweet,
- objekt uporabnika,
- sezname sporočil (*angl. timelines*).

Sporočilo (*angl. tweet*), ki ga lahko ustvari uporabnik, je v socialnem omrežju Twitter omejeno na dolžino 140 znakov, čemur je podjetje namenilo malo več pozornosti, kot bi pričakovali. Problem namreč nastane pri uporabi posebnih znakov, kot so na primer šumniki iz slovenske abecede. Ti so namreč v kodnih tabelah predstavljeni s kodo, ki je dolga več kot en bajt, medtem ko so znaki iz angleške abecede predstavljeni s kodo, ki je dolga samo en bajt. Iz tega razloga Twitter za vse podatke uporablja kodno tabelo UTF-8, za katero so priredili štetje znakov, da pravilno prepozna posebne znake in jih upošteva kot znak dolžine enega bajta.

Uporabnik je objekt, ki lahko ustvarja sporočila, pošilja zasebna sporočila drugim uporabnikom ali sledi (*angl. follow*) drugim uporabnikom. Sledenje drugim uporabnikom pomeni, da bomo v domačem seznamu sporočil (*angl. home timeline*) videli sporočila uporabnikov, ki jim sledimo. Uporabnike lahko združujemo v skupine, ki jih Twitter imenuje *lists*.

Seznami sporočil so časovno urejeni seznami sporočil, ki imajo skupen vir. Poznamo več vrst seznamov: *domači seznam, seznam uporabnika, seznam sporočil, v katerem je uporabnik omenjen, seznam sporočil, ki ga vrne iskanje.*

Podobno kot omrežje Facebook tudi Twitter ponuja nekaj vtičnikov, ki služijo za izvajanje osnovnih operacij socialnega omrežja. Predvsem sta to operaciji za objavo sporočila ter za sledenje uporabniku. Vtičnik sestavljata koda HTML ter skripta JavaScript, ki jo pripravi omrežje Twitter. Obe komponenti je preprosto integrirati na spletno stran oziroma v spletno aplikacijo. V nasprotju z aplikacijami vtičnikov ni potrebno registrirati. O vtičnikih bomo več povedali v naslednjem podpoglavju.

Aplikacije za socialno omrežje Twitter, ki jih registriramo, imajo za dostop do socialnih podatkov omrežja na voljo tri različne vmesnike za programiranje, ki jih bom podrobneje predstavil v nadaljevanju:

- REST API,
- Search API,
- Streaming API.

4.3 Vtičniki

Socialno omrežje Twitter nam ponuja naslednje vtičnike in orodja, ki jih lahko integriramo na spletni strani oziroma v spletni aplikaciji:

- gumb za sporočanje (*angl. Tweet button*),

- gumb za sledenje uporabniku (*angl. Follow button*),
- nabor akcij Web Intents,
- knjižnico @Anywhere za izboljšanje uporabniške izkušnje.

Gumba za sporočanje in sledenje uporabniku sta najpreprostejši obliki vtičnika, zato ju je najpreprosteje integrirati. Primer integracije gumba za sporočanje je prikazan na sliki 4.1.

```
<script src="http://platform.twitter.com/widgets.js"
  type="text/javascript"></script>

<a href="http://twitter.com/share"
  class="twitter-share-button">Tweet</a>
```

Slika 4.1: Primer integracije gumba za sporočanje.

Iz slike je razvidno, da moramo v dokumentu HTML najprej naložiti skripto `widgets.js`, ki je shranjena na strežniku omrežja Twitter. Ta skripta nato poišče vse elemente `<a>`, ki imajo parameter `class`, z vrednostjo `twitter-share-button`. S tem so izpolnjene vse zahteve za pravilno integracijo tega vtičnika. Tudi če želimo v uporabniškem vmesniku prikazati več vtičnikov, je potrebno skripto `widgets.js` naložiti samo enkrat. Alternativa temu načinu integracije je uporaba elementa `<iframe>`, ki nam ga pripravi Twitter in ga najdemo v dokumentaciji gumba za sporočanje. Integracija gumba za sledenje uporabniku je popolnoma enaka integraciji gumba za sporočanje, le da parametru `href` elementa `<a>` podamo spletni naslov uporabnika, ki mu želimo slediti.

4.3.1 Nabor akcij Web Intents

Web Intents predstavlja akcije, ki jih lahko uporabnik izvede nad določenim sporočilom. Te akcije so naslednje: *odgovori, sporoči naprej, dodaj med priljubljene (angl. reply, retweet, favorite).* Akcijo izvedemo tako, da uporabnika preusmerimo na določen spletni naslov. V primeru akcije *odgovori* izgleda ta spletni naslov tako: `https://twitter.com/intent/tweet?in_reply_to=SPOROCILO_ID`, kjer oznaka SPOROCILO_ID predstavlja identifikacijsko številko sporočila, na katerega bi radi odgovorili. Najlažja implementacija teh akcij je z uporabo že omenjene skripte `widgets.js`. Ko je skripta naložena na strani, moramo v spletno stran integrirati le še elemente `<a>`, ki imajo v parametru

`href` spletni naslov, ki smo ga podali zgoraj. Za vsako akcijo seveda obstaja ločen spletni naslov. Pri tem načinu implementacije bo skripta uporabniku ob kliku na gumb prikazala modalno okno, v katerem bo lahko izvedel akcijo, ki jo določa podani spletni naslov. Ker se akcijo do konca izvede na omrežju Twitter, nam ni treba skrbeti, ali je uporabnik prijavljen v socialno omrežje Twitter ali ne.

4.3.2 Knjižnica @Anywhere

Knjižnico @Anywhere sestavlja nabor funkcionalnosti, ki razvijalcu omogočajo uporabniški vmesnik na svoji spletni strani približati socialnemu omrežju Twitter. Knjižnica je vsebovana v JavaScript skripti `anywhere.js`, ki jo najdemo na omrežju Twitter. Pogoj za uporabo knjižnice je registrirana aplikacija za omrežje Twitter. Pri inicializaciji knjižnice moramo namreč kot parameter podati ključ za uporabo vmesnika za programiranje (*angl. API key*). Knjižnico inicializiramo tako, da v dokument HTML vstavimo element `<script>`, kot je prikazano na sliki 4.2 (oznaka API_KLJUC predstavlja ključ za uporabo vmesnika za programiranje naše aplikacije).

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-type"
      content="text/html; charset=utf-8">
    <title>Inicializacija @Anywhere</title>
    <script src="http://platform.twitter.com/
      anywhere.js?id=API_KLJUC&v=1"
      type="text/javascript"></script>
  </head>
  <body>
    ...
  </body>
</html>
```

Slika 4.2: Inicializacija knjižnice @Anywhere.

Knjižnica ob inicializaciji ustvari globalni objekt `twttr`, preko katerega dostopamo do naslednjih funkcionalnosti:

- samodejno ustvarjanje povezav za uporabniška imena Twitter,

- prikaz plavajočih oken z informacijo o uporabnikih,
- generiranje gumbov za sledenje uporabnikom,
- generiranje obrazca za ustvarjanje sporočila,
- poenostavitev registracije oziroma prijave v omrežje Twitter.

Funkcionalnosti knjižnice @Anywhere omogočajo uporabo izbirnih ukazov slogovnega jezika CSS, ki nam omogočajo, da omejimo področje delovanja funkcij na določen del objektnega modela dokumenta, s čimer pohitrimo njihovo delovanje. Primer uporabe funkcije za generiranje obrazca za ustvarjanje sporočila je prikazan na sliki 4.3.

```
<div id="tbox"></div>
<script type="text/javascript">
  twttr.anywhere(function (T) {

    /* Funkcija tweetBox() je poklicana nad elementom, ki ima
     * v parametru div vrednost 'tbox'. Funkcija iz tega elementa
     * generira obrazec za ustvarjanje sporočila višine 100 in
     * širine 400 slikovnih točk ter prednastavljeno vsebino.
     */
    T("#tbox").tweetBox({
      height: 100,
      width: 400,
      defaultContent: "<PREDPRIPRAVLJENA VSEBINA SPOROČILA>"
    });

  });

</script>
```

Slika 4.3: Primer uporabe knjižnice @Anywhere.

4.4 Vmesniki za programiranje

Omrežje Twitter razvijalcu ponuja tri namenske vmesnike za programiranje [3], ki smo jih omenili v poglavju 4.2 in jih bomo v tem poglavju na kratko predstavili.

4.4.1 REST API

REST API omogoča razvijalcu dostop do vseh virov omrežja (uporabniki, sporočila, seznami, zasebna sporočila itd.). Razvijalec preprosto pošlje ali zahtevek HTTP GET ali zahtevek HTTP POST (odvisno od tega, ali želi z zahtevkom vir prebrati ali pa ga zbrisati oziroma dodati) na določen spletni naslov. Primer takega spletnega naslova: `http://api.twitter.com/1/TIP_VIRA.FORMAT?DODATNI_PARAMETRI`, kjer oznaka TIP_VIRA predstavlja oznako vira, do katerega želimo dostopati, oznaka FORMAT pa predstavlja oznako formata, v katerem želimo prejeti odgovor vmesnika. Twitter namreč podpira več formatov, med katerimi sta najpogosteje uporabljena *JSON* in *XML*. Oznaka DODATNI_PARAMETRI predstavlja parametre, ki so obvezni ali opcijski, odvisno od vrste vira, ki ga zahtevamo. Primer zahtevka za branje domačega seznama, ki kot odgovor vrne objekt JSON, je prikazan na sliki 4.4.

```
https://api.twitter.com/1/statuses/home_timeline.json
```

Slika 4.4: Primer zahtevka za REST API.

4.4.2 Search API

Search API je namenjen izvajanju iskalnih poizvedb nad sporočili omrežja Twitter. Sporočila, nad katerimi lahko izvajamo poizvedbe, so le del vseh sporočil v omrežju. Gre za sporočila, ki niso starejša od približno enega tedna.

Poizvedbe izvajamo tako, da pošljemo HTTP GET zahtevek na spletni naslov `http://search.twitter.com/search.FORMAT?PARAMETRI`, kjer oznaka FORMAT predstavlja oznako formata, v katerem želimo prejeti odgovor vmesnika, oznaka PARAMETRI pa predstavlja parametre iskanja. Edini parameter, ki je obvezen, je parameter `q`, preko katerega vmesniku podamo iskalni niz. Primer poizvedbe, s katero želimo prebrati vsa sporočila, v katerih je bil omenjen *@twitterapi*, je prikazan na sliki 4.5.

```
http://search.twitter.com/search.json?q=%40twitterapi
```

Slika 4.5: Primer zahtevka za Search API.

4.4.3 Streaming API

Če želimo v spletni aplikaciji oziroma na spletni strani prikazati sporočila v realnem času, torej takoj, ko jih uporabniki objavijo, se lahko poslužimo uporabe

vmesnika Streaming API. Ta nam namreč omogoča branje toka (*angl. stream*) sporočil. Zahtevek še vedno pošljemo v obliki HTTP GET ali HTTP POST zahtevka (odvisno od toka, ki ga želimo brati), vendar pa moramo odgovor vmesnika brati drugače kot na primer pri vmesniku Search API. Večina odjemalcev v modelih odjemalec-strežnik nam odgovor na zahtevek HTTP GET ali HTTP POST vrne takrat, ko se povezava s strežnikom prekine. Pri vmesniku Streaming API pa moramo uporabiti takega odjemalca, ki nam odgovor vrača inkrementalno (postopoma). To pomeni, da nam povezave s strežnikom ni treba nikoli prekiniti oziroma jo prekinemo takrat, ko je ne potrebujemo več.

Streaming API nam ponuja tri različne skupine tokov: *splošni tokovi, tokovi uporabnika* in *tokovi strani*. S splošnimi tokovi lahko dostopamo do vseh javnih sporočil, ki jih lahko tudi filtriramo. Tokovi uporabnika nam ponujajo različne informacije za določenega uporabnika, tokovi strani pa informacije o straneh na socialnem omrežju Twitter.

4.5 Dokumentacija za Twitter

Dokumentacija za vtičnike in vmesnike za programiranje na omrežju Twitter je dobro strukturirana, pregledna in predvsem zelo uporabna. Prva stran dokumentacije služi kot kazalo do vseh funkcionalnosti, ki jih kot razvijalec lahko uporabimo. Dokumentacija posamezne tematike je predvsem opisna s priložnostnimi primeri, vendar pa so vse posamezne funkcije oziroma koncepti definirani na pregleden, tehničen in nedvoumen način, kar je za razvijalca zelo pomembno. Predvsem pa je zaradi dobre strukture in preglednega uporabniškega vmesnika dokumentacije zelo enostavno najti tisto, kar iščemo.

Poglavje 5

Izdelava aplikacije za socialno omrežje Facebook

Pri izdelavi aplikacije smo se za socialno omrežje Facebook odločili predvsem zaradi večjega zanimanja za izdelavo socialnih aplikacij s strani podjetij. Vedno več aplikacij namreč nastaja izključno za potrebe prodaje in promocije storitev ali produktov. K izbiri omrežja pa je tudi doprinesla večja svoboda pri izbiri namena aplikacije, ki jo ponuja omrežje Facebook.

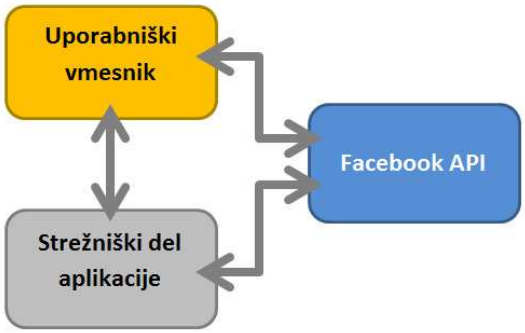
V nadaljevanju bom podrobneje opisal aplikacijo, predstavil njeno delovanje, strukturo ter interakcijo z omrežjem Facebook.

5.1 Podroben opis aplikacije in njenega delovanja

Zgradbo aplikacije najlažje predstavimo s pomočjo sheme (slika 5.1). Uporabniški vmesnik, ki je opisan z jezikom HTML v povezavi s slogovnim jezikom CSS, predstavlja del aplikacije, ki je namenjen interakciji z uporabnikom. Večjo dinamičnost uporabniškega vmesnika ter asinhrono komunikacijo s strežnikom zagotovi uporaba skriptnega jezika JavaScript. Strežniški del aplikacije predstavlja del aplikacije, kjer so shranjene in se izvajajo skripte PHP, poleg tega pa se v tem delu nahaja tudi podatkovna baza MySQL, v kateri so shranjeni podatki aplikacije. Zadnji del aplikacije predstavlja socialno omrežje Facebook, ki služi kot vir podatkov socialnega mreženja, ki jih od omrežja zahtevamo ali z uporabo skriptnega jezika JavaScript ali pa z uporabo orodja *cURL*.

Uporabniški vmesnik je funkcionalno razdeljen na dva dela. Prvi del predstavlja spletna stran, ki je razširjena z več gradniki socialne aplikacije. Drugi

del je v obliki dveh aplikacij, ki sta uporabnikom dosegljivi v okviru omrežja Facebook. Obe aplikaciji uporabljata isti strežniški del aplikacije.



Slika 5.1: Zgradba aplikacije po plasteh.

5.1.1 Spletna stran

Glavni objekt, s katerimi se uporabniki srečujejo tako na spletni strani kot tudi v obeh aplikacijah, je anketa. Anketa je objekt, ki ima naslednje lastnosti:

- vprašanje,
- od 2 do 5 možnih odgovorov,
- sliko ali video datoteko kot dodatek.

Vprašanje in odgovori so poljubno besedilo do največje dolžine 255 znakov. Slike so hranjene na strežniku, kamor jih ob ustvarjanju ankete uporabniki naložijo preko spletnega obrazca. Video datoteke uporabniki podajo v obliki spletnega naslova video datotek na spletni strani YouTube. Glede na to, ali vprašanje spremljata slika ali video, ločimo tri vrste anket:

- besedilna anketa,
- slikovna anketa,
- video anketa.

Spletna stran je namenjena glavni interakciji uporabnika z anketami, medtem ko sta obe aplikaciji namenjeni širjenju baze uporabnikov ter izboljšanju uporabniške izkušnje.

Uporabniki lahko nad anketami izvajajo tudi določene dejavnosti in sicer lahko ustvarijo ankete, za obstoječo anketo (na katero še niso odgovorili) izberejo enega izmed odgovorov in s tem oddajo svoj glas, oddajo komentar za obstoječo anketo, jo priporočijo prijatelju itd.

Na spletni strani lahko delujeta dva različna tipa uporabnikov:

- gost,
- registriran uporabnik.

Gost je vsak uporabnik, ki ali nima ustvarjenega uporabniškega računa na spletni strani ali pa vanj ni prijavljen. Gost lahko izvaja vse akcije razen ustvarjanja ankete. Za pridobitev tega dovoljenja se mora gost prijaviti v svoj uporabniški račun oziroma ga ustvariti, če ga še nima.

Če je uporabnik prijavljen s svojim uporabniškim računom, torej gre za registriranega uporabnika, lahko poleg ustvarjanja anket tudi ureja podatke svojega uporabniškega računa.

Vstopna stran spletne strani, ki jo vidi vsak uporabnik (ne glede na tip), prikazuje sezname anket, razvrščene po različnih kriterijih. Anketa je v teh seznamih prikazana z ikono, ki ponazarja tip ankete (glede na dodatno sliko ali video), vprašanjem, imenom avtorja, datumom izdelave ankete ter številom glasov, ki so bili oddani za to anketo. Ankete so razdeljene v pet skupin:

- *najnovejše ankete* - vse ankete, razvrščene po padajočem datumu,
- *najboljših 20 anket* - 20 anket z največjim številom oddanih glasov,
- *besedilne ankete* - vse besedilne ankete,
- *slikovne ankete* - vse slikovne ankete,
- *video ankete* - vse video ankete.

Ob kliku uporabnika na eno izmed anket v seznamih se prikaže stran, na kateri so podrobnosti ankete. V primeru, da je uporabnik za anketo že oddal svoj glas, vidi na tej strani rezultate ankete v obliki stolpičnega grafa. V nasprotnem primeru pa vidi možne odgovore na vprašanje, izmed katerih lahko enega izbere in na ta način odda svoj glas. Poleg odgovorov na tej strani vidimo tudi sliko ali video, če seveda za posamezno anketo obstajata.

5.1.2 Aplikacija v zavihku strani na Facebooku

Prvo aplikacijo smo razvili za uporabo v zavihku poljubne strani na Facebooku. V zavihku na strani so prikazani seznami anket, kot jih uporabnik vidi na vstopni strani spletne strani. Ob kliku na določeno anketo (slika 5.8 spodaj desno) je uporabnik preusmerjen na stran s podrobnostmi ankete na spletni strani. Na ta način privabimo na spletno stran nove uporabnike, ki bodo mogoče ustvarili svoj uporabniški račun ali pa bodo o spletni strani obvestili svoje prijatelje.

5.1.3 Aplikacija na platnu omrežja Facebook

Drugo aplikacijo smo razvili za uporabo na platnu omrežja Facebook 3.2. Ta aplikacija omogoča povsem enake funkcionalnosti ter deluje na enak način kot spletna stran. Razlika je v tem, da neregistriran uporabnik (gost) ob uporabi te socialne aplikacije ne more ustvariti svojega uporabniškega računa oziroma ne more urejati svojih podatkov, če gre za registriranega uporabnika.

Poleg tega je dodana funkcionalnost, ki uporabniku omogoča, da povabi svoje prijatelje s socialnega omrežja Facebook k uporabi aplikacije. V ta namen smo v uporabniškem vmesniku dodali gumb, ki ob kliku nanj kliče funkcijo `FB.ui()` paketa za razvoj programske opreme JavaScript (izsek kode je prikazan na sliki 5.2). Ta funkcija prikaže modalno okno, kjer uporabnik izbere tiste prijatelje, ki jim želi poslati povabilo.

```
FB.ui({
  method: 'apprequests',
  title: 'Povabi prijatelje k sodelovanju na strani
    Aplikacije za socialna omrežja',
  message: 'Pridruži se reševanju zabavnih anket na
    strani Aplikacije za socialna omrežja.',
  filters: ['app_non_users']
});
```

Slika 5.2: Programska koda za povabilo prijateljev.

Dodatni filter, ki smo ga podali v parametrih funkcije, zagotovi, da se med prijatelji, ki jim lahko pošljemo povabilo, prikažejo le tisti, ki še ne uporabljajo te aplikacije.

5.2 Uporabniški vmesnik aplikacije

Grafični uporabniški vmesnik je tisto, kar uporabniki vidijo in uporabljajo. Zgrajen je s pomočjo tehnologij HTML, CSS in JavaScript. Pri tem smo si pomagali s knjižnicama jQuery in jQuery UI.

5.2.1 Spletna stran

Uporabniški vmesnik spletne strani je sestavljen iz uporabniških vmesnikov vseh tipov strani, ki jih uporabnik lahko vidi na spletni strani. Obstajajo naslednji tipi strani: vstopna stran, stran za registracijo in prijavo, stran s podrobnostmi ankete, stran za izdelavo ankete ter stran za urejanje podatkov uporabniškega računa. Vsak tip strani ima na vrhu glavo strani, na dnu pa nogo strani. V nadaljevanju bom podrobneje predstavil uporabniški vmesnik glave strani, stran s seznami anket in stran s podrobnostmi ankete.

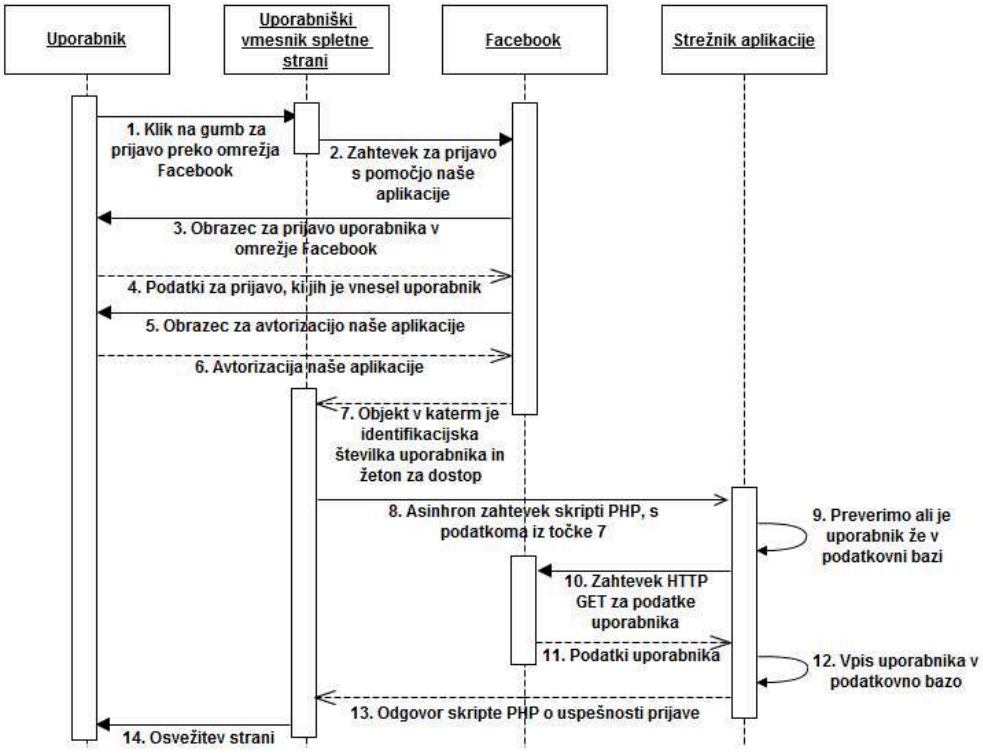
Glava strani

V glavi (slika 5.3) najdemo naslov spletne strani, spletno povezavo do strani na omrežju Facebook ter povezave bodisi za prijavo ali registracijo bodisi za odjavo ali urejanje uporabniškega računa. Poleg povezave za prijavo najdemo še povezavo za prijavo s pomočjo omrežja Facebook. Ta način prijave omogoča uporabniku, da se prijavi v spletno stran oziroma ustvari uporabniški račun s pomočjo obstoječega uporabniškega računa na omrežju Facebook.

Slika 5.3: Glava spletne strani.

Ob kliku na povezavo za prijavo s pomočjo omrežja Facebook se pokliče funkcija `FB.login()` paketa za razvoj programske opreme JavaScript, ki omrežju Facebook pošlje zahtevek za avtorizacijo uporabnika. V primeru, da uporabnik ni prijavljen v omrežje Facebook, se uporabniku prikaže modalno okno z obrazcem za prijavo v omrežje Facebook. Nato mora uporabnik aplikacijo avtorizirati (če tega še ni storil). Tu bi radi poudarili, da mora uporabnik

aplikacijo avtorizirati vsakič, ko aplikacija potrebuje dovoljenje za dostop do določene skupine uporabnikovih podatkov, vendar tega dovoljenja še nima. Če uporabnik aplikacijo avtorizira, torej ji dovoli dostop do svojih podatkov na omrežju Facebook, nam funkcija `FB.login()` kot rezultat vrne objekt, v katerem je zapisana identifikacijska številka uporabnika in žeton za dostop. Ta dva podatka nato s pomočjo asinhrone komunikacije s strežnikom pošljemo skripti PHP, ki preveri, ali uporabniški račun s to identifikacijsko številko že obstaja v podatkovni bazi. Če uporabniški račun že obstaja, uporabnika prijavimo v spletno stran, če pa uporabniški račun še ne obstaja, vmesniku za programiranje Facebook pošljemo zahtevek HTTP GET, v katerem zahtevamo vse podatke uporabnika. Ko te podatke dobimo, ustvarimo uporabniški račun in uporabnika prijavimo v spletno stran. Celoten postopek je prikazan na sliki 5.4, v dodatku B pa najdemo dva izseka programske kode, ki sta uporabljena v postopku.



Slika 5.4: Postopek prijave v spletno stran s pomočjo omrežja Facebook.

Stran s seznami anket

Seznami anket so predstavljeni na eni strani v obliki zavihkov. Te zavihke smo generirali z uporabo vtičnika, ki ga ponuja knjižnica jQuery UI. Ta vtičnik prikaže elemente `<div>` (ki jih določimo mi) v obliki zavihkov, kjer je vedno viden le eden izmed elementov `<div>`. Na sliki 5.5 so prikazani seznami anket v obliki zavihkov ter primer prikaza ankete v seznamih.

Slika 5.5: Prikaz seznamov anket v obliki zavihkov.

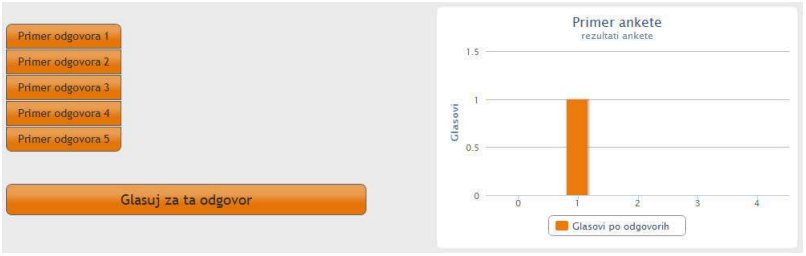
Poseben zavihek je zavihek za izdelavo ankete, ki je viden le v primeru, da je uporabnik prijavljen v spletno stran.

Stran s podrobnostmi ankete

Na tej strani vidimo različne podrobnosti ankete, ki so dopolnjene s tremi socialnimi vtičniki omrežja Facebook. Na vrhu strani je vprašanje ankete, desno od njega pa je prvi izmed treh vtičnikov, in sicer gumb *Všeč mi je* (slika 5.6). Na sliki je poleg gumba *Všeč mi je* tudi gumb *Pošlji (angl. Send)*, ki je namenjen objavi povezave na zid uporabnikovega prijatelja na Facebooku. Na sliki je napis na gumbu v angleškem jeziku, kar je posledica napake (v času pisanja) v delovanju omrežja Facebook.

Slika 5.6: Socialni vtičnik *Všeč mi je*.

Pod naslovom so prikazani, v primeru, da na anketo še nismo odgovorili, vsi možni odgovori na vprašanje (slika 5.7 levo) ali pa stolpični graf, ki prikazuje število glasov za posamezen odgovor (slika 5.7 desno). V primeru, da gre za



Slika 5.7: Prikaz možnih odgovorov ankete (levo) in grafa z rezultati ankete (desno).

slikovno ali video anketo, je slika oziroma video datoteka prikazana desno od možnih odgovorov oziroma grafa.

Na dnu strani sta prikazana preostala dva socialna vtičnika. Prvi vtičnik prikazuje komentarje uporabnikov, ki so jih uporabniki oddali za posamezno anketo. Komentar lahko odda vsak uporabnik, ki je prijavljen v socialno omrežje Facebook. Drugi vtičnik pa v primeru, da je uporabnik prijavljen v omrežje Facebook, prikazuje slike uporabnikovih prijateljev, ki uporabljajo to aplikacijo.

5.2.2 Stran na omrežju Facebook

Omenili smo že, da aplikacija, ki je integrirana na stran na omrežju Facebook, prikazuje sezname anket (slika 5.8). Ti seznami so zopet predstavljeni v obliki zavihkov, ki jih zgenerira vtičnik knjižnice jQuery UI, le da so prirejeni za prikaz na strani omrežja Facebook. Aplikacija se namreč prikaže znotraj elementa `<iframe>`, ki je širok 520 slikovnih točk, kar je približno 60 odstotkov širine spletne strani, zato smo morali zožati vse elemente grafičnega vmesnika.

Slika 5.8: Grafični vmesnik aplikacije na Facebook strani.

5.2.3 Platno na omrežju Facebook

Grafični vmesnik aplikacije, ki je prikazana na platnu omrežja Facebook, je vizualno povsem enak tistemu na spletni strani, vendar smo aplikaciji spremenili funkcionalnost, kot je opisano v razdelku 5.1.3.

5.3 Strežniški del aplikacije

Strežniški del aplikacije je tisti del, do katerega uporabnik eksplicitno nima dostopa. Lahko se (in običajno tudi se) nahaja na drugem računalniku kot uporabnikov brskalnik, s katerim dostopa do grafičnega uporabniškega vmesnika. Strežniški del aplikacije za izvajanje skript uporablja interpreter jezika PHP (opisan v razdelku 2.2.1).

5.3.1 Struktura in delovanje aplikacije na strežniku

Pri izdelavi aplikacije in spletne strani smo se poslužili arhitekture model-pogled-krmilnik (*angl. Model-View-Controller*, MVC), ki smo jo za lastne potrebe poenostavili. V naši arhitekturi tako ni komponente *model*. Arhitektura ločuje logiko aplikacije od uporabniškega vmesnika. To pomeni, da se programsko kodo za obdelavo podatkov loči od kode za prikaz rezultatov teh obdelav. Datotečna struktura je prikazana na sliki 5.9.

```
/classes/
/content/
/controllers/
/includes/addons/
/includes/css/
/includes/img/
/includes/js/
/templates/
/index.php
```

Slika 5.9: Datotečna struktura aplikacije.

Ob vsakem zahtevku se najprej kliče skripta PHP `index.php`, ki poskrbi za inicializacijo vseh potrebnih komponent in za klicanje dodatnih skript, ki so potrebne za obdelavo zahtevka. Vsakemu zahtevku ustreza eden izmed krmilnikov, v katerem je programska koda, ki obdela zahtevek. Vsakemu krmilniku

pripada eden ali več pogledov, ki definirajo, kako se rezultati obdelave zahtevka prikažejo. Znotraj pogledov se vstavijo tudi dodatni elementi, kot so na primer datoteke JavaScript in datoteke CSS.

Vsakemu tipu strani, ki ga vidimo na spletni strani, pripada eden izmed krmilnikov. Pravi krmilnik se razbere iz spletnega naslova zahtevka, v katerem so tudi dodatni parametri, če so potrebni. Primer takega spletnega naslova je: `http://diploma.pollvortex.com/poll/a36fypk6c5`. Prvi del naslova (`http://diploma.pollvortex.com/`) predstavlja ime domene, drugi del (`poll`) nam pove, da zahtevamo stran s podrobnostmi ankete, tretji del (`a36fypk6c5`) pa je unikaten identifikacijski niz, ki enolično določa katera anketa bo prikazana.

5.3.2 Podatkovna baza MySQL

Podatki o anketah in uporabnikih so shranjeni v podatkovni bazi MySQL. Za delo s podatkovno bazo smo pripravili razred `class.database.php`, napisan v jeziku PHP. Primerek tega razreda se ustvari avtomatsko med izvajanjem skripte `index.php`. V konstruktorju tega razreda (dodatek A) se poleg inicializacije spremenljivk ustvari tudi povezava s podatkovno bazo.

Podatki v podatkovni bazi so razdeljeni v štiri tabele:

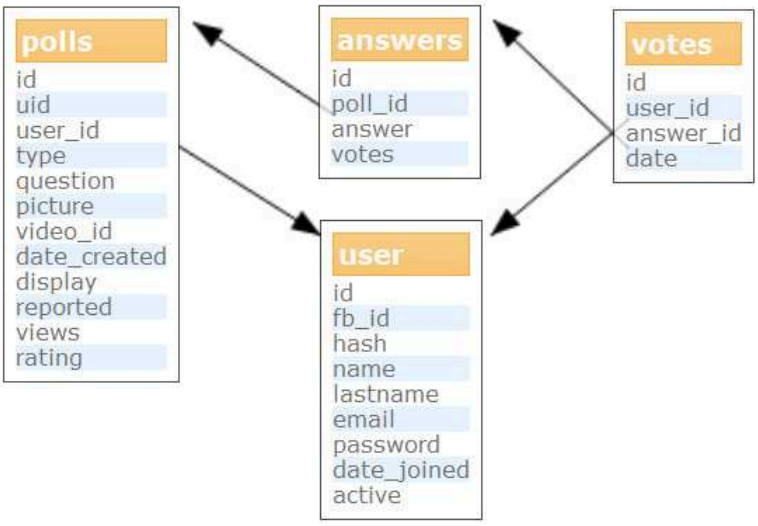
- user,
- polls,
- answers,
- votes.

Na sliki 5.10 je prikazana shema podatkovne baze. Puščice na sliki predstavljajo tuje ključe.

V tabeli `user` so shranjeni vsi podatki o uporabnikih. En atribut je namenjen tudi identifikacijski številki uporabnika na socialnem omrežju Facebook. Pri načrtovanju baze je treba biti pozoren na ta atribut, saj je zaradi velikega števila uporabnikov omrežja Facebook kot tip tega atributa potrebno nastaviti tip `bigint`.

V tabeli `polls` so shranjeni podatki o anketah. Vsaka anketa pripada enemu uporabniku, katerega identifikacijska številka je v tej tabeli predstavljena v obliki tujega ključa `user_id`.

V tabeli `answers` so shranjeni odgovori za ankete, ki so na pravo anketo vezani prek tujega ključa `poll_id`. Poleg samega odgovora se za vsak odgovor beleži tudi skupno število glasov, ki jih je ta odgovor prejel.



Slika 5.10: Shema podatkovne baze.

V tabeli `votes` so zabeleženi vsi glasovi. Vsak glas pripada ali registriranemu uporabniku ali gostu. Podatki te tabele so pomembni za ugotavljanje, ali je registriran uporabnik že glasoval za določeno anketo ali ne. Tu moramo dodati, da za goste tega ne moremo storiti, saj o njih ne vemo pravzaprav ničesar. Zato vsakemu gostu ob glasovanju shranimo piškotek, v katerem so shranjene identifikacijske številke anket, za katere je ta gost že glasoval.

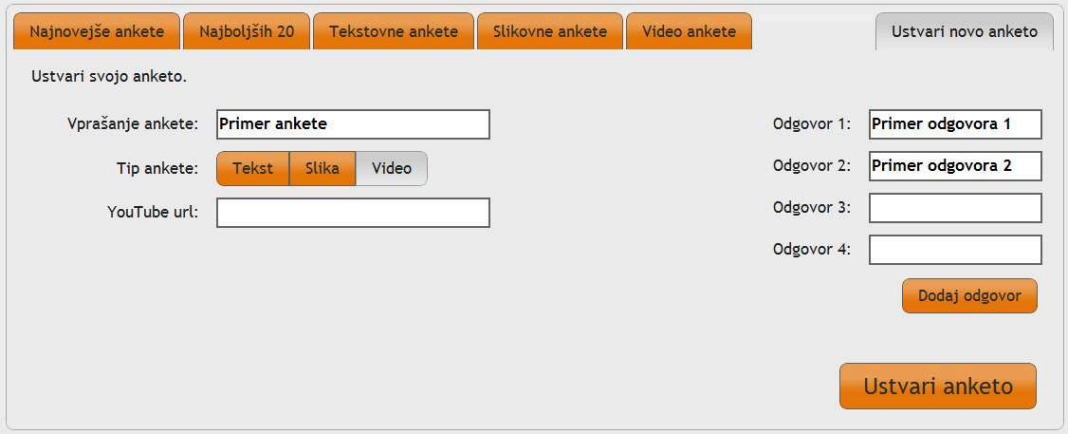
5.4 Primeri uporabe

V tem podpoglavju bomo natančneje opisali potek dveh primerov uporabe. Prvi primer uporabe je izdelava ankete na spletni strani, drugi primer pa je glasovanje za anketo v aplikaciji, ki je prikazana na platnu omrežja Facebook.

5.4.1 Izdelava ankete na spletni strani

Predpostavimo, da uporabnik že ima uporabniški račun in je prijavljen v spletno stran. Uporabnik mora najprej klikniti na zavihek *Ustvari anketo* 5.11 na vstopni strani spletne strani. Uporabniku se prikaže obrazec za izdelavo ankete. Obvezna polja obrazca so vprašanje, prva dva odgovora ter datoteka s sliko oziroma spletni naslov video datoteke na spletni strani YouTube, če gre

za slikovno oziroma video anketo. Uporabnik lahko doda do pet odgovorov. To stori s klikom na gumb *Dodaj odgovor*. Ko uporabnik pravilno izpolni obrazec, mora klikniti na gumb *Ustvari anketo*. Če se ustvarjanje ankete izvede brez napake, je uporabnik preusmerjen na stran s podrobnostmi ankete, ki jo je pravkar ustvaril. Pred preusmeritvijo se uporabniku prikaže modalno okno z obvestilom o uspešni izdelavi ankete. V tem modalnem oknu se nahaja tudi gumb *Objavi na Facebooku*, ki (ob kliku nanj) na zid uporabnika objavi povezavo do ankete, ki jo je pravkar ustvaril.



Slika 5.11: Obrazec za kreiranje ankete na spletni strani.

5.4.2 Glasovanje za anketo v aplikaciji na platnu

Tudi v tem primeru privzemimo, da je uporabnik že prijavljen v svoj uporabniški račun. Uporabnik si s seznamov anket izbere želeno anketo in klikne ali na sliko, ki predstavlja tip ankete, ali na vprašanje ankete. Ob kliku je uporabnik preusmerjen na stran s podrobnostmi ankete, kjer (pod pogojem, da za anketo še ni glasoval) lahko svoj glas dodeli enemu izmed možnih odgovorov (slika 5.12). Uporabnik za odgovor glasuje tako, da klikne na želeni odgovor, nato pa klikne na gumb *Glasuj za ta odgovor*. Če med procesiranjem glasovanja ni prišlo do napake, se uporabniku platno osveži. Ker je uporabnik sedaj že glasoval za to anketo, bo namesto možnih odgovorov videl graf z rezultati ankete.

Slika 5.12: Obrazec za glasovanje pri anketi na platnu.

Poglavje 6

Sklepne ugotovitve in ideje za nadaljnje delo

V diplomskem delu sta predstavljeni socialni omrežji Facebook in Twitter ter vmesniki za programiranje, ki jih ti dve omrežji ponujata. Poleg tega je opisana tudi aplikacija, ki smo jo razvili za socialno omrežje Facebook.

Pregled vmesnika za programiranje omrežja Facebook vsebuje le glavne komponente, ki pa razvijalcu zadoščajo za izdelavo aplikacije za omrežje Facebook. Iz pregleda smo izpustili podrobnejši opis označevalnega jezika FBML, saj ga podjetje Facebook počasi opušča. Prav tako smo izpustili vse komponente za razvoj aplikacij za mobilne telefone, ker smo se osredotočili na izdelavo spletnih aplikacij.

Pri pregledu vmesnika za programiranje omrežja Twitter smo vključili vse glavne kompomente vmesnika, vendar se pri njihovem opisu nismo spuščali v podrobnosti, predvsem zaradi osredotočenja na omrežje Facebook ter njegove dominantnosti na spletu.

Aplikacija omogoča uporabniku ustvariti uporabniški račun, s katerim lahko ustvarja lastne ankete in glasuje za že obstoječe ankete. Uporabnik lahko s pomočjo socialnih vtičnikov odda komentar na anketo, klikne na gumb *Všeč mi je* ali pošlje povezavo do ankete svojemu prijatelju na omrežju Facebook. Spletna aplikacija na strani omrežja Facebook omogoča uporabniku pregled vseh seznamov anket, aplikacija na platnu omrežja Facebook pa omogoča uporabniku skorajda enake funkcionalnosti kot spletna stran. Uporabniki aplikacije na platnu imajo tudi možnost, da k uporabi aplikacije povabijo svoje prijatelje z omrežja Facebook.

Aplikacija uporablja le tisti del uporabnikovih podatkov z omrežja Facebook, ki jih nujno potrebuje za delovanje. Aplikacijo bi lahko razširili z dodat-

nimi funkcionalnostmi, ki bi vključevale druge podatke socialnega mreženja. Tako bi lahko uporabniki aplikacije dodali druge uporabnike kot prijatelje oziroma bi se te povezave generirale avtomatsko z uporabo obstoječih povezav med uporabniki socialnega omrežja Facebook. Prijateljstvo med uporabniki nam odpre cel nabor novih funkcionalnosti, kot so na primer prikaz anket samo za prijatelje, izmenjava sporočil, skupinski pogovori, izmenjava daril itd.

Kot smo že omenili, smo bili zelo nezadovoljni z dokumentacijo vmesnika za programiranje Facebook in smo mnenja, da si podjetje s tako velikim številom uporabnikov tega ne bi smelo privoščiti.

Dodatek A

Konstruktor razreda `class.database.php`

```
/**
 *
 * Class constructor
 */
public function __construct($host = DB_HOST, $user = DB_USER,
  $password = DB_PASSWORD, $database = DB_DATABASE) {

  // inicializacija spremenljivk
  $this->db_host = $host;
  $this->db_user = $user;
  $this->db_password = $password;
  $this->db_database = $database;

  $this->db_last_query = null;
  $this->db_affected_rows = null;
  $this->db_insert_id = null;

  $this->db_table_structures = array();

  // povezava s podatkovno bazo
  $connector = mysql_connect($this->db_host, $this->db_user,
    $this->db_password);

  // v primeru, da je povezava uspešno vzpostavljena
```

```
  if($connector) {
    // izberemo specifično podatkovno bazo
    if($this->db = mysql_select_db($this->db_database,
      $connector)) {

      // izvedemo poizvedbo, ki nastavi uporabo
      // kodne tabele UFT-8
      $set_collation = mysql_query("SET NAMES 'UTF8',
        CHARACTER SET 'UTF8',
        COLLATION_CONNECTION='" . DB_COLLATION . "'");
      if(!$set_collation)
        // če poizvedba ne uspe, javimo napako
        $this->throw_error(mysql_error(), true);
    } else {
      // če želena podatkovna baza ne obstaja,
      // javimo napako
      $this->throw_error("Podatkovna baza s tem
        imenom ne obstaja.");
    }
  } else {
    // če povezava s podatkovno bazo ni uspela,
    // javimo napako
    $this->throw_error(mysql_error(), true);
    return false;
  }

  return true;
}
```

Dodatek B

Izseka programske kode pri uporabi funkcije `FB.login()`

B.1 Klic funkcije `FB.login()` v jeziku JavaScript

```
FB.login(function(response) {

  /* V primeru, da nam funkcija login() vrne pozitiven odgovor,
   * pošljemo skripti ajax.facebook-login.php zahtevek HTTP POST
   * v katerem podamo identifikacijsko številko uporabnika
   * in žeton za dostop, ki nam ju vrne funkcija login().
   * V primer, da tudi skripta ajax.facebook-login.php vrne
   * pozitiven odgovor, stran osvežimo.
   */
  if (response.session) {
    $.ajax({
      url: basepath + "controllers/ajax.facebook-login.php",
      type: "POST",
      data: "uid=" + response.session['uid'] + "&access_token=" +
        response.session['access_token'],
      success: function(data) {
        if(data == "1") {
          location.reload();
        }
      }
    });
  } else {
```

```
    $(".jq_ui_tabs").eq(0).tabs("select", 1);
  }

}, {perms: "email,publish_stream"});
```

Zgoraj prikazana programska koda prikazuje uporabo funkcije `FB.login()` in pošiljanje zahtevka Ajax strežniku. Iz kode je razvidno, da od uporabnika zahtevamo dve posebni dovoljenji in sicer dovoljenje za dostop do njegovega elektronskega naslova ter dovoljenje za objavljanje na njegov zid. Dovoljenje za objavljanje bi lahko sicer zahtevali šele takrat, ko bi to dovoljenje zares potrebovali, vendar za delovanje te aplikacije potrebujemo samo ti dve dovoljenji, zato je smiselno, da ju zahtevamo že zdaj, saj s tem zmanjšamo število zahtevkov za avtorizacijo, ki jih prikažemo uporabniku. Ti zahtevki so za uporabnika moteči in včasih uporabnika tudi odvrnejo od uporabe naše aplikacije.

B.2 Izsek kode iz skripte PHP ajax.facebook-login.php

```
// Zahtevek HTTP GET,
// s katerim zahtevamo podatke uporabnika.
$graph_url = "https://graph.facebook.com/" . $uid .
  "?access_token=" . $access_token;

$ch = curl_init(); // inicializacija
curl_setopt($ch, CURLOPT_URL, $graph_url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

// pošljemo zahtevek HTTP GET in s pomočjo
// funkcije json_decode pretvorimo objekt JSON
// v polje $fb_user.
$fb_user = json_decode(curl_exec($ch), true);

curl_close($ch);
```

Zgoraj je izsek programske kode PHP, ki vmesniku za programiranje Facebook pošlje zahtevek HTTP GET, v katerem zahtevamo podatke uporabnika [2].

Slike

Literatura

[1] B. Bibeault, Y. Katz, *jQuery in Action*, Greenwich, CT: Manning, 2010.

[2] W. Graham, *Facebook API Develpers Guide*, New York City, NY: Apress, 2008.

[3] K. Makice, *Twitter API: Up and Running*, Sebastopol, CA: O'Reilly, 2009, pogl. 4.

[4] S. Pemberton et al.: "XHTML™1.0 The Extensible HyperText Markup Language (Second Edition): A Reformulation of HTML 4 in XML 1.0", *W3C Recommendation (http://www.w3.org/TR/2002/REC-xhtml1-20020801)*, avgust 2002.

[5] B. Bos, H. W. Lie, C. Lilley, I. Jacobs: "Cascading Style Sheets, level 2 (CSS2) Specification" *W3c Recommendation (http://www.w3.org/TR/CSS2/)*, september 2009.

[6] Dokumentacija za Graph API ter pripadajoče objekte. Dostopno na: http://developers.facebook.com/docs/reference/api/

[7] Statistika socialnega omrežja Facebook. Dostopno na: http://www.facebook.com/press/info.php?statistics