SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

LABORATORIJ PROFILA

PROJEKT IZ PREDMETA RASPODIJELJENI SUSTAVI

Tema: Prisutnost na društvenim mrežama

Tehnička dokumentacija

Josip Ćavar, Nikola Martinec, Petar Mrazović, Vjekoslav Ranogajec, Nikola Martinec

# *Mentor: Doc. dr. sc. Mario Kušek*



Zagreb, Prosinac, 2011.

**1. Uvod**

U sklopu laboratorija profila *Programsko inženjerstvo i informacijski sustavi* praktično obrađujemo temu „Prisutnost na društvenim mrežama“. Glavni cilj ovog projekta je razviti web-aplikaciju koja će provjeravati prisutnost korisnikovih prijatelja na različitim društvenim mrežama kao što su npr. Facebook, Google+, Twitter, LinkedIn ili Gowalla. Također, vrlo važno je da se mi, studenti, koji se bavimo ovim projektom, upoznamo s osnovnim principima razvoja web-aplikacija, te problemima koji on donosi.

Ova tehnička dokumentacija donosi detaljan opis programskog rješenja koji je rezultat našeg rada na projektu. Budući da smo razvili Facebookovu web-aplikaciju koja koristi usluge nekoliko različitih društvenih mreža, u nastavku donosimo opis implementacije i odgovarajuće UML dijagrame podijeljene po društvenim mrežama koje smo uključili u aplikaciju.

Facebookova aplikacija koja je rezultat našeg rada omogućava korisniku jednostavan uvid u prisutnost i statuse njegovih prijatelja na drugim društvenim mrežama. Svaki student koji je bio član našeg razvojnog tima bavio se jednom društvenom mrežom. Za svaku pojedinu društvenu mrežu bilo je potrebno detaljno proučiti njene karakteristike, dostupna sučelja za programiranje aplikacija, tj. API-je (eng. *Application Programming Interface*) i odgovarajuću dokumentaciju.

Programsko rješenje ostvareno je korištenjem ASP .NET radnog okvira u razvojnom okruženju Visual Studio 2010. ASP .NET je Microsoftovo okruženje u okviru .NET platforme koje se može koristit za razvoj web-mjesta, web-aplikacija i XML web-usluga. Budući da se kao i ostali dijelovi .NET platforme temelji na zajedničkom CLR-u (eng. *Common Language Runtime*), njegov kod se može pisat u bilo kojem .NET jeziku. Budući da svi članovi tima imaju najviše iskustva u programiranju u programskom jeziku C#, upravo smo njega odabrali za jezik implementacije.



**Slika 1.1.** ASP .NET okruženje

**2. Arhitektura programskog rješenja**

tu bi trebalo ukratko o rješenju, struktura aplikacije, arhitektura itd.. pa onda dijagram klasa i deployment dijagram

**3. Facebook društvena mreža**

tu ide za fejs

**4. Google+ društvena mreža**

tu ide za g+

**5. Twitter društvena mreža**

2006. godine Jack Dorsey osnovao je Twitter, društvenu internetsku mrežu namijenjenu razmjeni kratkih poruka koje su prema imenu mreže popularno nazvane *tweetovima*. Ova društvena mreža danas broji preko 200 milijuna korisnika, te pruža pristup korisničkim podacima na različite načine, a jedan od njih je i razvoj aplikacija korištenjem nekih od dostupnih sučelja za programiranje aplikacija, tj. API-ja (eng. *Application Programming Interface*). Danas je dostupan niz API-ja za razne programske jezike i radne okvire, pa tako i za .NET Framework 3.5 i 4.0 što nam omogućava razvoj Twitterovih aplikacija u programskom jeziku C#. Neki od njih su *Twitterizer* i *TweetSharp*. U nastavku ćemo opisati usluge koje pruža *TweetSharp* API i način na koji je pomoću njih ostvarena provjera prisutnosti korisnikovih prijatelja.

**5.1. Funkcionalni zahtjevi**

Twitter kao društvena mreža ne pruža mogućnost uvida u trenutnu prisutnost korisnika, a upravo je to cilj projekta: razviti aplikaciju koja će pratiti trenutnu prisutnost na društvenim mrežama. Iz tog razloga koristit ćemo se kratkim porukama, tj. *tweetovima*, koji su objavljeni od strane korisnikovih prijatelja. Budući da za svake *tweet* poznajemo vrijeme nastanka može ustanoviti koji je od korisnika nedavno bio prisutan na mreži. Tako će naša aplikacija za sve korisnikove prijatelje ispisivati *tweetove* koje će poredati po vremenu nastanka. Tako će korisnik dobiti jasan uvid u aktivnost prijatelja te kada je ona bila ostvarena. Također omogućit ćemo korisniku postavljanje novog *tweeta* putem naše aplikacije. Ove jednostavne funkcionalne zahtjeve ćemo u nastavku opisati koristeći tzv. slučajeve korištenja, tj. *use caseove*.

Korisnik pregledava aktivnosti prijatelja na Twitteroj društvenoj mreži

1. Korisnik odabire opciju za pregledavanje aktivnosti prijatelja na Twitterovoj društvenoj mreži.

2. Sustav poziva Twitterovu uslugu za korisničku autorizaciju aplikacije.

3. Korisnik autorizira aplikaciju.

4. Sustav vraća popis *tweetova* korisnikovih prijatelja s pridruženim vremenom nastanka.

Ekstenzija:

2a. Korisnik je već autorizirao aplikaciju

1. Sustav dohvaća podatke o obavljenoj autorizaciji.

2. Sustav vraća popis *tweetova* korisnikovih prijatelja s pridruženim vremenom nastanka.

Korisnik postavlja novu kratku poruku (*tweet*)

Preduvjet: Korisnik je uspješno autorizirao aplikaciju.

1. Korisnik odabire opciju za postavljanje novog *tweeta*.

2. Korisnik upisuje tekst *tweeta*.

3. Korisnik pregledava tweet.

4. Korisnik potvrđuje postavljanje *tweeta*.

5. Sustav postavlja korisnikov *tweet* na Twitterovu društvenu mrežu.

Ovi jednostavni *use caseovi* kratko upisuju funkcionalnost koju je potrebno ostvariti programskim rješenjem. U nastavku ćemo vidjeti na koji način je ostvarena pojedina funcionalost korištenjem *TweetSharp* API-ja i ASP .NET platforme.

**5.2. Programsko rješenje**

5.2.1. Autorizacija aplikacije

Da bi preko web-aplikacije pristupili osobnim podacima korisnika Tweeterove društvene mreže, potrebno je prvo autorizirati aplikaciju. Svi programi pisani za rad u okviru različitih operacijskih sustava koji koriste Tweeterove usluge (tzv. third-party aplikacije), moraju se autorizirati koristeći OAuth, otvoreni standard za autorizaciju. OAuth (Open Authorization) je način autorizacije na koji se omogućava pristup third-party aplikacijama podacima Twitterovog korisničkog profila bez prethodnog unošenja korisničkog imena i lozinke. Ovo osigurava veću sigurnost povjerljivim podacima korisnika s obzirom da podatke za prijavljivanje na Twitterov korisnički profila neće moći koristiti ni na jednom drugom mjestu osim na twitter.com. Sve third-party aplikacije koje žele koristiti osobne podatke Twetterovih korisnika moraju se povezati s Twitterovim profilom isključivo koristeći **OAuth** proceduru.



Slika 5.2.1. *Open Autorization*

Da bi omogućili autorizaciju naše aplikacije, moramo ju prvo registrirati na Tweeterovoj mreži. Registracijom aplikacije dobivamo tzv. potrošački ključ (eng. *Consumer key*) te potrošačku tajnu (eng. *Consumer secret*), koji trebaju ostati poznati samo nama koji razvijamo aplikaciju. Na temelju tih podataka aplikacija generira tzv. request token pomoću kojeg od Twittera dobivamo URL adresu na kojoj nas čeka pin koji korisnik predaje aplikaciji. Aplikacija zatim na temelju pina traži tzv. access token, tj. token za pristup korisničkim podacima, te se konačno pomoću njega autorizira sama aplikacija. Opisani postupak je detaljno ilustriran na sljedećoj shemi (slika 5.2.2.). *Tweetsharp* API pruža vrlo dobru podršku otvorenoj autorizaciji pa ćemo se nastavku upoznati s klasama i metodama koje koristimo u programskom rješenju da bi ostvarili autorizaciju aplikacije.



**Slika 5.2.2.** Postupak autorizacije aplikacije korištenjem OAuth

Da bi autorizirali aplikaciju korištenjem *TweetSharp* API-ja moramo iskoristiti potrošački ključ i tajnu koje smo dobili prilikom registracije aplikacije. Na temelju tih podataka stvaramo objekt tipa TwitterClientInfo koji nam služi za identifikaciju aplikacije te stvaranje novog objekta tipa TweeterService pomoću kojeg komuniciramo s Tweeterovom društvenom mrežom i manipuliramo podacima korisnika.

TwitterClientInfo twitterClientInfo = new TwitterClientInfo();

twitterClientInfo.ConsumerKey = ConsumerKey;

twitterClientInfo.ConsumerSecret = ConsumerSecret;

TwitterService twitterService = new TwitterService(twitterClientInfo);

Sljedeći korak je stvaranje već prije spomenutog request tokena pomoću kojeg ćemo dohvatit URL adresu za autorizaciju, te ćemo ju otvoriti u web-pregledniku kako bi korisnik mogao unijeti dobiveni pin te tako dobiti access token. Request token i URL adresa za dobivanje pina generiraju se prilikom pritiska na gumb „Get Pin“, a access token generira se pritiskom na gumb „Authorize“ (slika 5.5.3.). URL adresa s pinom otvara se u novom skočnom prozoru.

OAuthToken requestToken = twitterService.GetRequestToken();

Session["TwitterRequestToken"] = requestToken;

authUrl = twitterService.GetAuthorizationUrl(requestToken);

authUrl = Page.ResolveClientUrl(authUrl);

-------------------------------------------------------------------------------

OAuthToken accessToken;

accessToken = twitterService.GetAccessToken((OAuthToken) Session["TwitterRequestToken"], this.TextBoxPin.Text);

Session["TwitterAccessToken"] = accessToken;

OAuthToken requestToken = twitterService.GetRequestToken();

Session["TwitterRequestToken"] = requestToken;

authUrl = twitterService.GetAuthorizationUrl(requestToken);

authUrl = Page.ResolveClientUrl(authUrl);

OAuthToken accessToken;

accessToken = twitterService.GetAccessToken((OAuthToken) Session["TwitterRequestToken"], this.TextBoxPin.Text);

Session["TwitterAccessToken"] = accessToken;

pin.png

auth.png

Access i request token spremamo u tzv. sesijske varijable kako korisnik ne bi trebao više puta autorizirati aplikaciju. Naime, kada klijent prvi puta pristupa web-aplikaciji, generira se jedinstveni SessionID broj i sprema se na korisnikovo računalo u obliku *cookie*-a ili ga dodaje u svaki korisnikov zahtjev. Vezano uz taj ID možemo stvarati sesijske (eng. *session*) varijable koje se čuvaju na serveru.



**Slika 5.2.3.** Autorizacija aplikacije

5.2.2. Pristup podacima s Tweeterovog korisničkog profila

Sljedeći korak je pristupiti korisnikovim podacima kako bi dohvatili njegove prijatelje i njihove *tweetove*. Prije dohvaćanja podataka korištenjem access tokena koji je spremljen u session varijabli potvrdit ćemo autoriziranost aplikacije. Podatke s Tweeterove društvene mreže dohvaćat ćemo jednostavnim pozivima metoda klase TwitterService. Metoda ListTweetsOnFriendsTimeline() dohvaća *tweetove* prijatelja kao objekte tipa TwitterStatus. Svakom tom objektu pridruženo su svojstva: CreatedDate (vrijeme nastanka *tweeta* po kojem se sortiraju svi dohvaćeni *tweetovi*), User (korisnik koji je postavio *tweet*), te Text (tekst *tweeta)*.

protected void refreshFriendTweets()

{

IEnumerable<TwitterStatus> tweets = twitterService.ListTweetsOnFriendsTimeline();

foreach (TwitterStatus tweet in tweets)

{

ListBoxTweets.Items.Add("[" + tweet.CreatedDate + "] --> " + tweet.User.ScreenName + " says: " + tweet.Text + ".");

}

}

5.2.3. Dizajn programskog rješenja

U ovom poglavlju opisan je dizajn programskog rješenja koje ostvaruje pristup korisnikovim podacima s Twitterovog korisničkog profila.

**6. Gowalla društvena mreža**

Gowalla je zamišljena kao lokacijski orijentirana društvena mreža. Glavna ideja Gowalle je podijeliti priče, slike i iskustva o mjestima koje posjećujemo sa drugim ljudima, npr. gradove, muzeje i sl. Upravo je to imao na umu jedan od osnivača (Josh Williams, Scott Raymond) kada je 2007. godine pokrenut projekt pod nazivom Gowalla. Prije svega, ona je prvenstveno mobilna aplikacija jer se nije moguće „prijavljivati“ na raznim lokacija sa standardne web stranice[1].

Nažalost, početkom prosinca 2011. godine službeno je objavljeno da je Gowalla kupljena od strane Facebooka, te se krajem siječnja ova mreža u potpunosti gasi[2].

**6.1. Funkcionalni zahtjevi**

Na Gowalli nije moguće dobiti uvid u status prisutnosti korisnika jer on uopće ne postoji. Takav status nije ni potreban za funkcioniranje Gowalle. Ono što zanima njezine korisnike je da li je netko od njihovih prijatelja bio negdje te to objavio. Zbog toga smo kao zamjenu za statuse uzeli vrijeme zadnjeg objavljivanja „prijavljivanja“ na nekom mjestu. Prema tome korisnik koji koristi našu aplikaciju može lako zaključiti da li se je dogodilo nešto zanimljivo vrijedno odlaska na samu Gowallu. Tipičan *use case*, odnosno slučaj korištenja opisan je u nastavku.

Korisnik pregledava aktivnosti prijatelja na Gowalli

1. Korisnik odabire pregled aktivnosti prijatelja na Gowalli
2. Sustav ga traži korisničko ime i lozinku
3. Dohvaćaju se i dostavljaju podaci o zadnjim aktivnostima korisnikovih prijatelja

**6.2. Programsko rješenje**

Za programsko ostvarivanje zadatka korišten je Gowalla API. API poštuje REST konvenciju te nije ovisan o specifičnom programskom jeziku. Za dohvat npr. korisnikovih podataka potrebno je poslati određen HTTP zahtjev. URL-ovi kojima dohvaćamo različite podatke popisani su u dokumentaciji (API Explorer)[3]. Odgovor koji dobijemo je u JSON formatu – tekstualno baziranom standardu koji je čitljiv ljudima.

Da bismo uopće mogli koristiti API potrebno je dobiti tzv. *API* *key.* Da bi smo ga dobili potrebno je registrirati aplikaciju. Ključ koji dobijemo moramo dodati u zaglavlje HTTP upita kojeg koristimo. Ovjera (eng. *authentication*) korisnika se vrši pomoću osnovne ovjere korisnika (eng. *basic access authentication*) koju podržava većina web preglednika.

**6.3. Primjer dohvaćanja korisnikovih podataka**

Da bi smo mogli dohvatiti korisnikove podatke prvo se moramo prijaviti na Gowallin sustav. Prilikom svakog upita na Gowallin sustav gradi se objekt tipa *HttpWebRequest* koji je sastavni dio kazala *System.Net.*  U njemu definiramo potrebna zaglavlja , kao što je *API Key* koji smo spomenuli ranije. Metoda prima dva parametra – URL traženog resursa i enumerator *authentication.* Pošto se ova metoda koristi za sve upite samo je kod prijavljivanja korisnika na Gowallin sustav potrebno postaviti vrijednost te varijable na *required.*

public HttpWebRequest createRequest(string url, authentication aut)

{

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

if (aut == authentication.required)

{

request.Credentials = new NetworkCredential(userName, passWord);

request.Headers.Add("Authorization", "Basic" +

Convert.ToBase64String(new

ASCIIEncoding().GetBytes(userName + passWord)));

}

request.ContentType = "application/json";

request.Accept = "application/json";

request.Headers.Add("X-Gowalla-API-Key:" + Gowalla.key);

return request;

}

Sljedeći odsječak koda zapravo poziva gore navedenu metodu sa odgovarajućim parametrima. Nakon što je zahtjev kreiran, odnosno imamo objekt tipa *HttpWebRequest,* možemo zatražiti odgovor od Gowallinog poslužitelja. Kao odgovor dobiti ćemo JSON objekt. Dio odgovora slijedi u nastavku:

"stamps\_count": 0,

"url": "[/users/2898661](http://gowalla.com/users/2898661)",

"bookmarked\_spots\_url": "[/users/2898661/bookmarks](http://gowalla.com/users/2898661/bookmarks)",

**"friends\_activity\_url": "**[**/checkins/recent**](http://gowalla.com/checkins/recent)**",**

"can\_post\_to\_facebook": false,

"\_auto\_connect\_addressbook\_friends": null,

"current\_checkins": [],

**"following\_count": 7,**

"friend\_requests\_count": 0,

"\_is\_friend": true,

"\_visited\_spots\_urls\_url": "[/users/2898661/visited\_spots\_urls](http://gowalla.com/users/2898661/visited_spots_urls)",

"can\_post\_to\_twitter": false,

…

Iz navedenog odgovora je vrlo lako izvući željene podatke – bilo da se radi o jednostavnom podatku, kao što je npr. *following\_count,* odnosno broj prijatelja čiju aktivnost pratimo, ili o kolekciji podataka npr. *friends\_activity\_url* za koju dobijemo URL preko kojeg možemo dohvatiti elemente kolekcije. Konkretno, ovaj kod postavlja korisnikovo ime, prezime, te korisničko ime koje može biti proizvoljno.

Na sličan način se dohvaćaju podaci o korisnikovim prijateljima te spremaju u objekte tipa *Friend* koji sadrže sve potrebne podatke o njima.

private void fetchBasicUserData(authentication aut)

{

HttpWebRequest request = createRequest("http://api.gowalla.com/users/me", aut);

using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())

{

StreamReader reader = new StreamReader(response.GetResponseStream());

string result = reader.ReadToEnd();

String[] data = result.Split(',');

this.firstName = data.First(x =>

x.Contains("first\_name")).Split('"')[3];

this.lastName = data.First(x => x.Contains("last\_name")).Split('"')[3];

this.alias = data.First(x => x.Contains("username")).Split('"')[3];

}

}

**6.4. Tehničke poteškoće vezane uz Gowalla API**

Nažalost, API ne radi ispravno u potpunosti te mnogi URL-ovi dobiveni u odgovoru rezultiraju pogreškom o nepostojanju traženog resursa (*404 Not Found).* Zbog toga ne možemo pristupiti popisu korisnikovih prijatelja preko API-a. Kako bi zaobišli ovaj problem, nakon što smo prijavili korisnika na sustav koristeći API, popis korisnikovih prijatelja učitavamo izravno filtrirajući HTML stranicu na kojoj bi oni bili prikazani kada bi korisnik koristio Gowallu izravno. To može predstavljati problem - ako se nešto promijeni u vezi URL-a koji vodi do te HTML stranice aplikacija više neće ispravno raditi.

**7. LinkedIn društvena mreža**

tu ide za linkedIn