SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Proširenje sustava Ferko

Bruno Rahle

Voditelj: *Mr. sc. Marko Čupić*

Zagreb, travanj, 2011.

1. Sadržaj

2. Uvod

Revolucija je počela. Ali ne revolucija koja će uzrokovati promjene u strukturama vlasti. Ne revolucija koja će uzrokovati krvoproliće. Revolucija koja je nastavak industrijske revolucije. Informacijska (znanstveno-tehnološka) revolucija. Ona donosi svijetu nešto sasvim novo - informacije putuju sa jednog kraja svijeta na drugi doslovno brzinom svjetlosti. Zbog toga se često čuje da je danas Zemlja manja no ikad.

Otkako je 1492. Kolumbo otkrio Ameriku, svijet se zauvijek promijenio. Nekad nepojmljivo bespuće postalo je konačno. Magellan je potom oplovio svijet i dokazao da je on okrugao. Time je započeo proces globalizacije. Za ono što je Magellanu (točnije, osamnaestorici preživjelih mornara) brodom trebalo gotovo 3 godine, a Jules Verne pisao o fantastičnom pothvatu u 80 dana pred samo 137 godina, danas avion napravi u nešto više od 30 sati. Sateliti to naprave za svega 90 minuta, a svjetlost više od 7 puta u samo jednoj sekundi.

Informacije se danas prenose svjetlosnim kablovima, tj. putuju brzinom svjetlost. Brže od toga ne može, čak niti u teoriji. Internet je medij gdje se te informacije nalaze i razmjenjuju. Procvatom širokopojasnog (*broadband*) pristupa Internetu i razvojem računala, a u zadnje vrijeme i pametnih telefona (*smartphone*-a), informacije su nam doslovce dostupne na dohvat ruke. I, što je možda i najvažnije, taj se pristup ne plaća suhim zlatom već je vrlo pristupačan običnom čovjeku. Danas možemo govoriti o povezivanju praktički cijelog svijeta u jednu veliku obitelj.

Informacije se s interneta najčešće preuzimaju s web stranice, dok web portalom nazivamo stranicu koja objedinjuje informacije iz većeg broja izvora. Takve stranice (portali) mogu biti izrazito korisne ako su pregledne i jednostavne. Primjeri svjetski poznatih web portala su Google, Yahoo! i MSN, dok su u nas popularni tportal, Index i net.

Upravo iz potrebe za jednostavnim pristupom informacija o fakultetskim obvezama i aktivnostima, razvijen je portal, ili preciznije, sustav, Ferko. Osim toga, on omogućava i izradu, provedbu i analizu provjera znanja, komunikaciju između studenata i nastavnika te stvaranje rasporeda nastave.

U današnje vrijeme, međuljudski se odnosi sve više oslanjaju na internet kao sredstvo komunikacije. Drugim riječima, ljudi često komuniciraju kada su za računalom. Stoga raste i potreba za dijeljenjem informacija. Na početku je to bilo samo preko e-maila, da bi se nakon popularizacije Internet Relay Chata ‘90-ih godina prošlog stoljeća, počeli pojavljivati i drugi protokoli, poput ICQ-a (*I Seek You*), XMPP-a (*Jabber*, *Extensible Messaging and Presence Protocol*), MSNP (*Microsoft Notification Protocol*)...

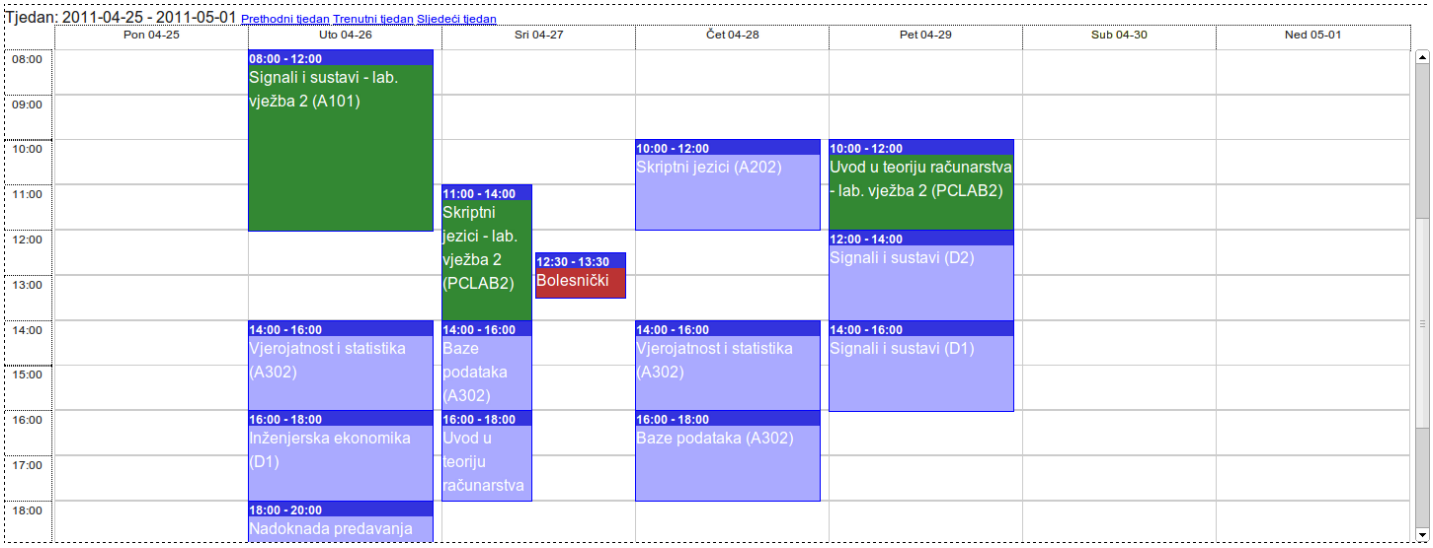
Međutim, nisu samo komunikacijski protokoli nicali kao gljive poslije kiše. Svaki od gore navedenih portala ima svoj vlastiti sustav za prijavu. Takav trend rješavanja riješenog problema prati internet praktički od njegovih početaka. Iako je to nekad korisno i pojave se bolja rješenja, u konačnici je to nepotrebno i štetno za razvoj. Posljednjih nekoliko godina počela se primjećivati konvergencija toga mnoštva tehnologija u jedno. Tako se pojavljuju OpenID i OAuth, centralizirani sustavi za prijavu koji postaju jedinstveni identifikatori korisnika. Na polju komunikacije, Facebook Messages je primjer sustava koji objedinjuje različite protokole - SMS, Instant Messaging, e-mail, sustav privatnih poruka…

Ideja ovog seminara je modernizirati sustav Ferko dodavanjem mogućnosti za dijeljenje kalendara s prijateljima te omogućavanje prijave na sustav koristeći neki (već postojeći) OpenID korisnički račun.

3. Kalendar

3.1 Postojeći sustav

Kalendar je podsustav koji je Ferko učinio popularnim među studentima. Osim što se na samom portalu nudi pregledan pogled na raspored obaveza (predavanja, laboratorijske vježbe, ispiti...), moguće je i preuzeti kalendar u iCal formatu. Ova druga mogućnost je zanimljiva jer to omogućava uvoz (*import*) kalendara u program ili internetsku uslugu. Upravo je ta mogućnost uvoza obveza ono što omogućava korištenje kalendara na pametnim telefonima koji su praktični kada je, npr., potrebno na brzinu u hodu pogledati u kojoj se dvorani održava predavanje. Osim toga, usluge poput Googleovog kalendara su korisne jer nude lijep i jednostavan prikaz obaveza iz više izvora te zbog toga ovise o njihovom uvozu.



Slika 1. Prikaz kalendara na sustavu Ferko

Ovaj dio sustava napisan je tako da je podržavao mnoge stvari i bez posebnih intervencija u kodu. Najveći problem izazivao je protokol za preuzimanje za informacija s poslužitelja. Ugrubo, on je radio ovako:

1. korisnik A pošalje GET zahtjev na određeni URL s vlastitim tajnim ključem (privateKey)
2. poslužitelj identificira korisnika A preko njegovog tajnog ključa
3. poslužitelj vraća podatke korisnika A

Taj pristup ima nekoliko problema. Budući da se sva komunikacija odvija koristeći samo jedan jedinstven tajni ključ, ako je on kompromitiran, moguće je pristupiti svim podacima. Također, zbog toga nije bilo moguće sigurno dijeljenje svojeg kalendara s kolegama. Prvi problem je nešto složeniji i dosta ga je teško sigurno i robusno riješiti. Kako postoji potreba za identifikacijom korisnika bez da je on prijavljen u sustav, neki identifikator mora biti nekako poslan. Zbog ograničenja nekih sustava, to se rješava kako je opisano. Ono što sustav Ferko nudi je mogućnost ponovnog generiranja tajnog ključa. Drugi je problem, nasuprot prvome, rješiv. U idućem je odlomku objašnjeno kako je to ostvareno.

3.2 Preinake postojećeg sustava

Problem sigurnog dijeljenja vlastitog kalendara sa kolegama rješava slijedeći protokol:

1. korisnik A pošalje GET zahtjev na određeni URL s vlastitim tajnim ključem i opcionalno s javnim ključem (publicKey) korisnika B
2. poslužitelj identificira korisnika A preko njegovog tajnog ključa
3. ako javni ključ nije prisutan, vraćaju se podaci korisnika A, kao što se i prije radilo
4. poslužitelj identificira korisnika B preko njegovog javnog ključa
5. ako je korisniku A korisnik B izričito dopustio pristup podacima, vrate se podaci korisnika B

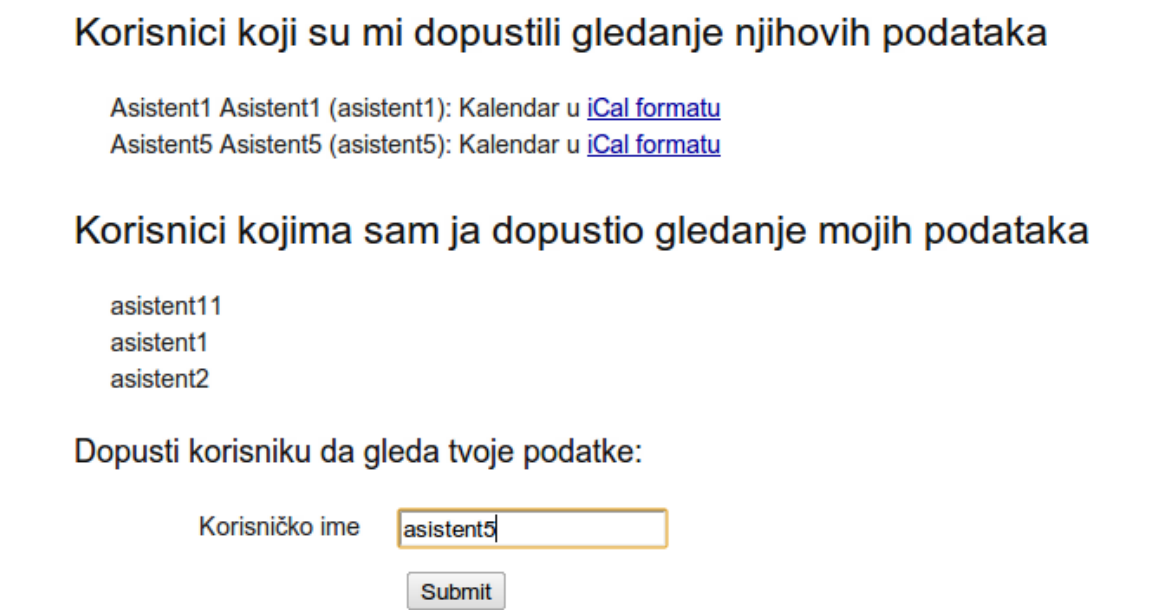
Iako niti ovaj način komunikacije s poslužiteljem nije savršen, on uspješno rješava drugi problem. Naime, da bi pročitao podatke korisnika B, korisnik A više ne treba pristup tajnom ključu korisnika B. Time je osigurano da korisnik A nema neograničen pristup svim informacijama korisnika B. Osim toga, korisnik B sada može kontrolirati što će i kako podijeliti s kojim korisnikom. To je izvedeno preko novog podsustava, podsustava za prijatelje. Dobra strana ovog sustava je i ta što je *backwards* kompatibilan, odnosno već postojeći linkovi neće prestati raditi.

3.3 Podsustav za prijatelje

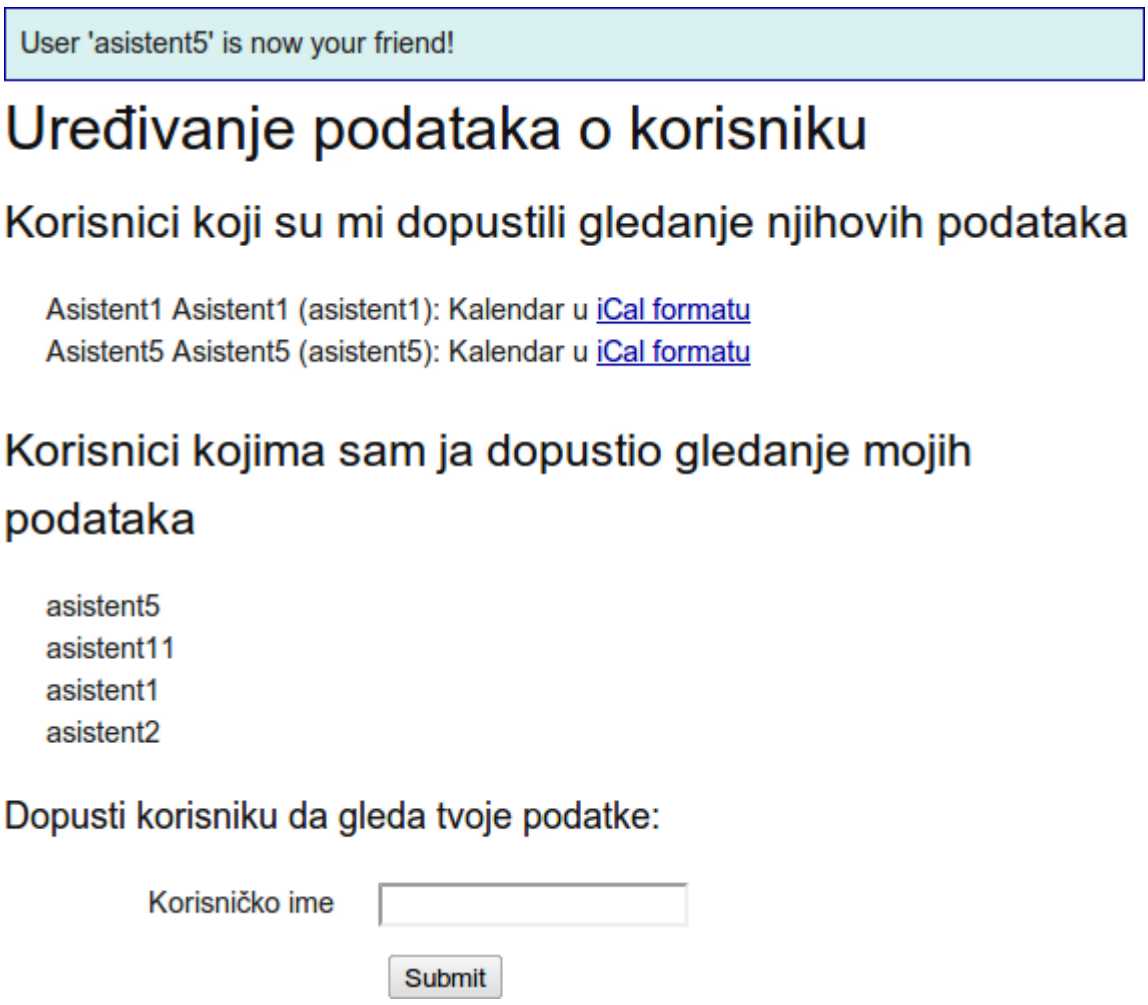
Definirat ćemo prijatelje korisnika A kao skup korisnika koji imaju pravo pristupa njegovim informacijama. Kako bi se to omogućilo, bilo je potrebno dizajnirati sustav dodavanja prijatelja. On je izveden na slijedeći način:

1. korisnik A upisuje korisničko ime korisnika B kojeg želi dodati kao prijatelja
2. korisnik B trenutačno dobije pristup podacima javnim podacima korisnika A
3. korisnik A ne dobije pristup podacima korisnika B

Ovakav sustav kontrole pristupa podataka ima svojstvo da relacije nisu simetrične, što nije nužno izričito dobro ili loše. Moguće je da će korisnike zbuniti i izazvati njihovo nezadovoljstvo to što oni svoje informacije dijele nekim kolegama koji iste ne dijele s njima. Zbog toga je izrazito važno što je ova funkcionalnost tzv. *opt-in* (korisnik sam odlučuje hoće li je koristiti ili ne i kome će je dopustiti) te što postoji mogućnost uskraćivanja pristupa informacijama (micanje korisnika s liste prijatelja). Mogućnost zloporabe ovog sustava u svrhu identificiranja drugih korisnika svakako je smanjena i time što korisnik A kada doda korisnika B kao svojeg prijatelja i dalje vidi samo njegovo korisničko ime koje je upisao a ne i puno ime i prezime, čime je spriječena mogućnost otkrivana punog imena preko korisničkog.



Slika 2. Dodavanje prijatelja



Slika 3. Potvrda da je prijatelj dodan

3.4 Promjene u kodu

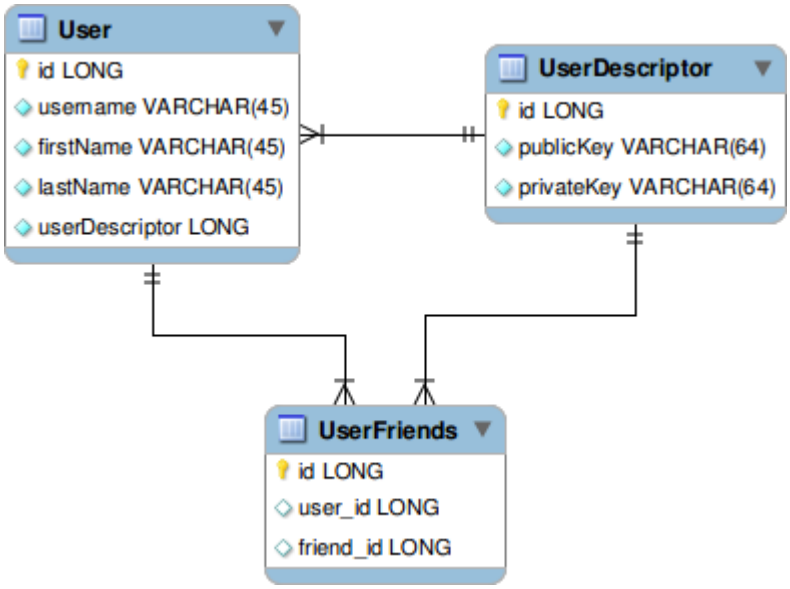
Za ostvarivanje gore navedenih promjena, bilo je potrebno napraviti nekoliko promjena u kodu. Kako je Ferko rađen prema MVC (*model-view-controller*; model-pogled-kontroler) arhitekturi, koristeći Struts2 programsku platformu (*framework*), tako ćemo i promatrati tri cjeline koje su izmijenjene.

3.4.1 Promjene u modelu

U Ferku postoje dva modela koji pamte podatke o korisniku - `User` i `UserDescriptor`. U modelu `User` pamte se najvažniji podaci o korisniku koji se stalno koriste - korisničko ime, ime, prezime... U modelu `UserDescriptor` pamte se manje bitne stvari koje su potrebne određenim komponentama - kodirana lozinka koja služi za prijavu, email, tajni ključ...

Javni je ključ, stoga, dodan modelu `UserDescriptor`. Tehnologija na kojoj se Ferkov perzistencijski sloj zasniva je Hibernate, pa je za dodavanje tog ključa bilo dovoljno u razredu `UserDescriptor` dodati `String publicKey` te napisati getter i setter za njega s potrebnim anotacijama kako bi Hibernate znao kakav stupac u tablici očekuje.

Nešto veći problem za ostvariti bila je lista prijatelja. U model `UserDescriptor` dodan je `Set<User> friends` koji je predstavlja skup prijatelja nekog korisnika. Hibernate taj podatak ostvaruje kao novu tablicu (`UserFriends`).



Slika 4. Prikaz modela koji su dodani

3.4.2 Promjene u pogledu

Pogled u MVC arhitekturi služi samo za prikaz podataka. U ovom je slučaju trebalo izmijeniti samo stranicu s korisničkim postavkama. Ona se nalazi u `UserEditForm.jsp` datoteci.

3.4.3 Promjene u kontroleru

Kontroler je odgovoran da poslužitelj zna što treba napraviti kada mu dođe neki zahtjev. U ovom je slučaju bilo potrebno napravit nekoliko izmjena: jednu koja je omogućila dodavanje prijatelja, jednu koja je promijenila način izrade kalendara i jednu koja je dodala potrebne atribute u razred koji čuva podatke o korisniku.

Razred koji se bavi rješavanjem zahtjeva za promjenu podataka o korisniku je `UserAction`. U njega su dodane funkcije `addFriend` i `removeFriend` koja se pozivaju kada korisnik odluči dodati/maknuti prijatelje. Te dvije funkcije pozivaju `UserActionService` koji onda obavlja posao spremanja podataka u perzistencijski sloj.

Za generiranje kalendara, bilo je potrebno urediti dva razreda, `ICalUser` (koji obrađuje zahtjev) i `ICalService` (koji generira kalendar). U prvi je trebalo dodati podršku za novi parametar (javni ključ), a u drugi novu funkciju, `getCalendarForFriend`, koja iz primljenog javnog i tajnog ključa određuje smije li korisnik vidjeti kalendar i potom generira traženi kalendar.

Kako bi `UserEditForm` mogao znati prijatelje korisnika, u razred `UserBean`, koji čuva podatke koji se prikazuju na stranici za izmjene postavki, dodane su liste `friends` i `friendOf`. Prva čuva popis prijatelja traženog korisnika, a druga čuva popis korisnika kojima je traženi korisnik prijatelj.

4. OpenID i OAuth - sustavi za ovjeravanje

Ovjera služi kako bi se dobio podatak o korisniku koji koristi sustav. Bitno nam je razlikovati dva bliska, ali ipak različita pojma - prijava i ovjera. Ovjera je potvrda točnosti nekog podatka, a u ovom kontekstu to se odnosi na identitet korisnika. Prijaviti se može na neki sustav koji onda poznaje identitet korisnika. je Za Ferko je prijava bitna jer sustav mora znati koje podatke treba prikazati kojem korisniku. Takav je slučaj s većinom ozbiljnijih portala koji te podatke onda mogu iskoristiti za prikaz specifičnih podataka, kupovinu putem Interneta, korisničko dodavanje novog sadržaja…

Prijava se preko Interneta najčešće implementira na slijedeći način:

1. korisnik na poslužitelju prijavi (registrira) korisničko ime i s njime poveže lozinku te još uz to upiše neke podatke o sebi (najčešće su to ime, prezime i e-mail adresa)
2. za prijavu na sustav, potrebno je znati i korisničko ime i lozinku upravo za taj sustav

Takav način prijave ima nekoliko problema. Broj stranica koje prosječan korisnik Interneta redovito posjećuje je relativno malen; mnogo manji od broja onih koje će povremeno posjetiti. Ukoliko je potrebno obaviti prijavu za korištenje stranice iz druge kategorije, postoje četiri glavna scenarija:

- korisnik odluči da ga stranica pretjerano ne zanima, pa odustane od registracije i ne koristi stranicu. Ovaj pristup je loš za vlasnika stranice jer time gubi posjetitelje.
- korisnik odluči da ga stranica zanima te se registrira. Pritom koristi korisničko ime i lozinku specifično samo za tu stranicu. Ovaj pristup ima problem da će se korisnik kasnije teže prisjetiti podataka za prijavu, pa će biti prisiljen otvarati novi korisnički račun.
- u ovom scenariju korisnik također odluči da ga zanima stranica te prilikom registracije upisuje korisničko ime i lozinku koju koristi i na drugim servisima. Ovaj pristup ima manu da korisnik predaje svoje pristupne podatke nekome tko možda nema siguran sustav, tj. nekome od koga maliciozna osoba može ukrasti te podatke.
- prema posljednjem učestalom scenariju, korisnik odabire ovjeru preko treće strane. Ovu opciju mora omogućiti razvijatelj (*developper*) stranice (koja se još naziva i *relaying party*). Korisnik se tada preko korisničkog računa nekog davatelja takve usluge (*identity provider*), kojeg je razvijatelj stranice prethodno odobrio, prijavljuje na sustav. U ovom je slučaju problem što se kompromitiranjem jednog sustava ugrožavaju svi ostali. Međutim, davatelji usluge su najčešće kompanije kojima se "vjeruje" - Google, Facebook, Yahoo i njima slične - tako da je vjerojatnost krađe lozinke s nekog od tih servisa minimalna. (To, međutim, ne znači da neka maliciozna osoba ne može ukrasti lozinku na drugi način, nego samo da ju je teško ukrasti od davatelja servisa.)

Osim četiri gore navedena scenarija, mogući su i hibridni pristupi između drugog i trećeg (korištenje više različitih parova korisničkih računa i lozinki). U posljednje se vrijeme sve više sustava počinje nuditi četvrti način verifikacije korisnika. Za njega danas u svijetu postoje dva

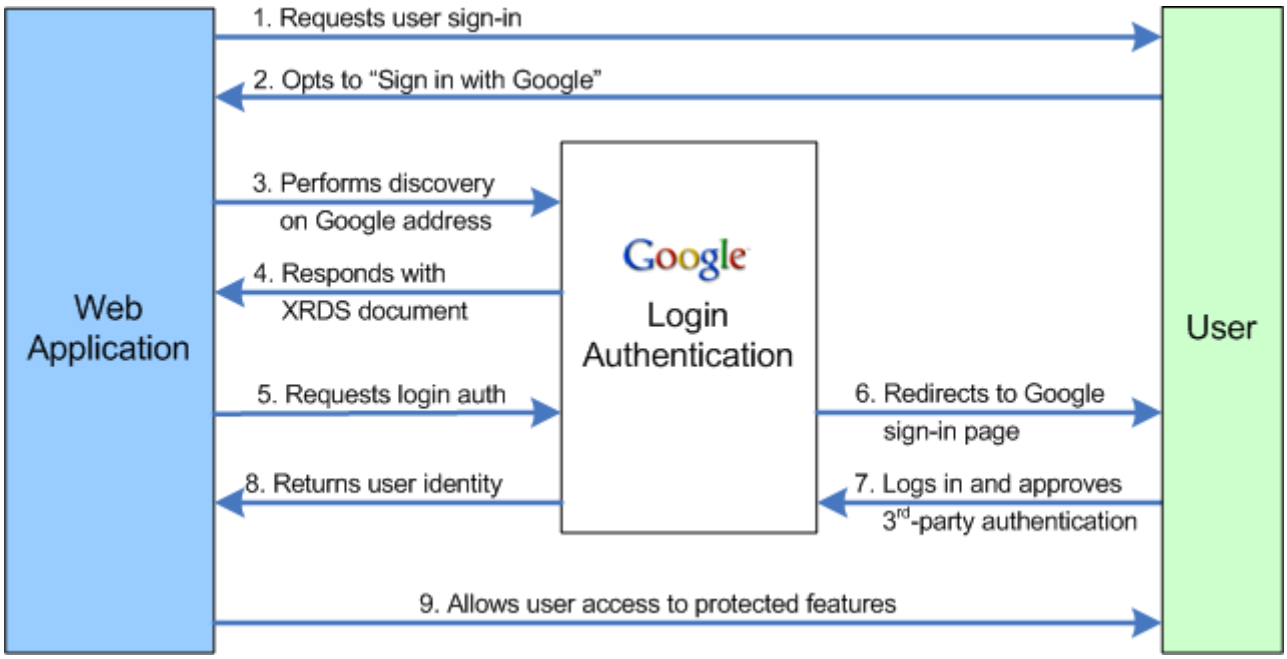
najčešće korištena protokola - OpenID i OAuth. U nastavku je objašnjeno kako oba od njih funkcioniraju te zašto su nastali.

4.1 OpenID

Slika 5. Logo OpenID-a

OpenID je otvoreni standard osmišljen da bi se usluga ovjere mogla izvesti decentralizirano (bez središnjeg autoriteta). Standard je tako napisan da omogućava ovjeru i preko klasičnih metoda (korisničkih imena i lozinki) i preko modernih poput biometrijskih podataka i/ili pametnih kartica (*smart cards*). Ovaj standard implementirale su razne velike tvrtke poput AOL-a, Googlea, IBM-a, VeriSigna i Yahoo!-a. Kako protokol funkcionira i kako se koristi, bit će objašnjeno na primjeru Googlea.

4.1.1 Opis korištenja OpenID standarda



Slika 6. Primjer komunikacije sa Googleovim OpenID servisom-a

Što se događa u kojem koraku?

1. Poslužitelj traži od korisnika da se prijavi na sustav.
2. Korisnik odlučuje koristiti Google način prijave.
3. Poslužitelj šalje zahtjev za „otkrivanje" Googleovom serveru, kako bi dobio informacije o načinu prijave na Google te krajnjoj točci (*endpoint*) za prijavu.
4. Google poslužitelju vrati tražene informacije.

5. Poslužitelj onda pripremi zahtjev koristeći verziju protokola koju je dobio od servera u prošlom koraku.
6. Poslužitelj preusmjeri korisnika na krajnju točku za prijavu. Prilikom preusmjeravanja, zahtjev generiran u 5. koraku također mora biti prenesen na Googleov poslužitelj. Google prikaže korisniku stranicu za prijavu.
7. Korisnik upisuje svoje podatke za prijavu na Googleov sustav te dopušta stranici (trećoj strani) da pristupi njegovim podacima. Ako korisnik to odabere, idući put kada je već prijavljen u svoj Google korisnički račun, ovaj korak se može preskočiti.
8. Google vraća tražene podatke o korisniku natrag poslužitelju. Poslužitelj tu konačno može identificirati korisnika.
9. Poslužitelj prihvaća ili odbija korisnikov zahtjev za prijavu na sustav.

Ovaj način prijave moguće je izvesti samo jednim klikom miša, ukoliko je korisnik već prijavljen na Google. Tako je to izvedeno na Ferku, a o tome više kasnije.

4.2 OAuth

Slika 7. Logo OAuth-a

Drugi otvoreni standard koji se koristi je Open Authorization (poznatiji kao OAuth) standard. Taj standard je nastao zbog potrebe za dijeljenjem podataka poput slika, videa ili liste prijatelja sa jedne stranice na drugu. Može se koristiti u sprezi s OpenID-em, ali to nije nužno. Verzija 1.0 standarda je napuštena te je verzija 2.0 trenutačno u razvoju i neće podržavati 1.0 standard.

Najpoznatiji opis što je OAuth sigurno je njegova usporedba sa sluginim ključem (*valet key*). Slugin je ključ, naime, rezervni ključ za automobil koji omogućava da se automobil vozi svega nekoliko minuta te nakon toga više prestaje raditi. On se dobije uz luksuzne automobile kako bi vlasnici kada dolaze na neki događaj mogli posluzi ostaviti ključ da im parkiraju vozila bez straha da će ih netko ukrasti. OAuth ima upravu tu svrhu na Internetu - omogućava pristup nekom podatku koji nije javan na neko određeno vrijeme.

Najveći korisnik OAuth protokola je Facebook (koji koristi vlastitu implementaciju 2.0 standarda) koji cijeli svoj Facebook Graph API temelji na njemu. Odnedavno je i Google dodao podršku za OAuth 2.0. Kako je protokol u razvoju, a zbog ograničenosti vremena za pisanje seminara, autor je odlučio da neće implementirati ovaj protokol za pristup informacijama u sklopu ovog rada, iako je to potencijalno bolji način za rješavanje gore opisanog problema dijeljenja podataka u kalendaru.

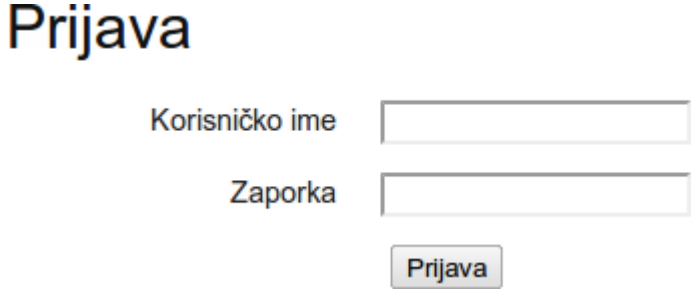
5. Korištenje sustava za ovjeru na Ferku

5.1 Postojeći sustav

Ferko ima specifične korisnike - studente i fakultetsko osoblje jer nije otvoren za širu javnost. Svaki korisnik dobije podatke za prijavu na sustav kada se upiše na fakultet odnosno kada se zaposli na njemu. Zbog toga ne postoji sučelje za registraciju, međutim to ne miče potrebu za prijavom na sustav. U tu svrhu studenti moraju upisati svoje korisničko ime i lozinku. Korisničko ime je nespretno utoliko što se sastoji od inicijala osobe i dijela JMBAG-a te ga je stoga moguće pogrešno napisati. Također, interakcija sa sustavom odvija se većinom koristeći samo miš, pa je i prebacivanje ruku s miša na tipkovnicu te odmah potom nazad, nespretno. Osim toga, Ferko iz sigurnosnih razloga nema opciju "zapamti me na ovom računalu" (za čuvanje podataka o prijavljenoj osobi) pa je stoga taj proces prijave potrebno obavljati svaki puta kada se želite prijaviti na sustav. Zato bi protokol koji, poput OpenID-a, omogućava prijavu samo jednim klikom miša, olakšao korištenje Ferka. To je bila glavna motivacija za implementiranje takvog načina prijave.

Dosadašnji koraci prijave na sustav bili su:

1. Korisnik dolazi na Ferkovu stranicu.
2. Korisnik upisuje korisničko ime i lozinku te šalje POST zahtjev poslužitelju.
3. Poslužitelj provjerava ispravnost upisanih podataka.
4. Ako su podaci ispravni, poslužitelj preusmjeri korisnika na naslovnu stranicu, u suprotnome ga preusmjeri na natrag stranicu za prijavu.



Slika 8. Dijalog za prijavu

5.2 Preinake postojećeg sustava

Kao što je već rečeno, najveći problem u sustavu prijave je upisivanje korisničkog imena i lozinke svaki puta kad se treba prijaviti na sustav. Taj posao može se delegirati na nekog davatelja OpenID usluge (u daljnjem tekstu davatelj). Mnogobrojni korisnici cijelo su vrijeme svoga rada na računalu prijavljeni na njih, a tipični primjeri su Gmail (Googleov e-mail) i/ili Facebook.

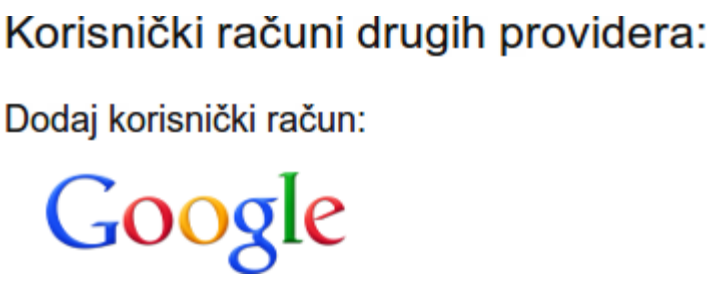
To u praksi znači da će oni jednom upisati svoje korisničke podatke, a nakon toga svi ostali sustavi bi trebali moći detektirati o kojem se korisniku radi međusobnom komunikacijom.

Naravno, o korisniku ovisi želi li to i napraviti jer on treba inicirati tu komunikaciju klikom na ikonu davatelja. Preciznije, prijava sa strane korisnika izgleda ovako:

1. Korisnik dolazi na Ferkovu stranicu.
2. Korisnik odabire davatelja.
3. Ako korisnik nije prijavljen na servisu davatelja, otvara mu se dijalog da se prijavi na njega.
4. Ako korisnik nije dopustio Ferku pristup računu kod davatelja, davatelj otvara novi dijalog s tim pitanjem.
5. Davatelj preusmjeri korisnika na stranicu koju mu je dao poslužitelj te u zahtjev spremi tražene podatke. Najvažniji podatak koji pošalje svakako je identifikator.
6. Poslužitelj provjerava postoji li korisnik koji s tim identifikatorom i davateljem u bazi korisnika.
7. Ako su podaci ispravni, poslužitelj preusmjeri korisnika na naslovnu stranicu, u suprotnome ga preusmjeri na natrag stranicu za prijavu.

Iako se na prvi pogled čini da korisnik ima više posla, to nije nužno tako jer, ako je već prijavljen na servisu davatelja, neće se morati ponovo prijavljivati te će se sustavi sami sporazumjeti. Biblioteka OpenID4Java odrađuje komunikaciju s davateljem, preglednik se brine za otvaranje dijaloga, korisnik obavlja prva dva koraka samostalno, a kako je 7. korak identičan onome u običnom sustavu prijave, potrebno je još samo analizirati 6. korak. U njemu poslužitelj provjerava poznaje li korisnika kojega mu je davatelj predstavio. Da bismo to znali, korisnik mora povezati svoj račun kod davatelja s računom na Ferku. Kako bi to učinio, korisnik se mora prijaviti se na običan način i u postavkama odabrati davatelja i račun (ili račune) s kojim želi povezati svoj Ferko korisnički račun. Tada Ferko može uspješno identificirati korisnika prema identifikatoru koji dobije od davatelja. Da bi se to ostvarilo kako treba, bilo je potrebno osmisliti podsustav za vanjske prijave (*remote authorization*).

Slika 9. Dijalog za prijavu koji podržava i prijavu preko Google korisničkog računa



Slika 10. Povezivanje Ferkovog i Googleovog računa

Implementacija ovog podsustava je vrlo jednostavna ali i dalje relativno moćna. Napravljena su dva nova modela koje Ferko razumije: davatelj usluge i korisnik usluge vanjske prijave. Model davatelja usluge pamti sve potrebne podatke da Ferko zna s njim komunicirati i da ga zna prezentirati korisniku. Zbog toga se dodavanje novog davatelja svodi na dodavanje novog retka u bazu podataka. Korisnik usluge za sada samo povezuje model davetelja usluge s modelom korisnika.

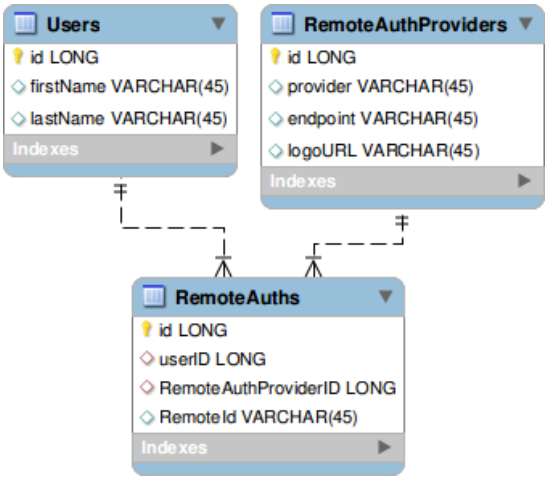
5.3 Promjene u kodu

Kao i kod kalendara, razmatrat ćemo svaku od tri cjeline MVC arhitekture posebno.

5.3.1 Promjene u modelu

Kao što je već rečeno, dodana su dva modela: davatelj usluge (`RemoteAuthProvider`) i korisnik usluge vanjske prijave (`RemoteAuth`). Model `RemoteAuthProvider` pamti osnovne podatke o davatelju - njegovo ime (`String provider`), krajnju točku za komunikaciju (`String endpoint`) te URL na kojem se nalazi logo (`String logoURL`). Na primjer, za Googleov korisnički račun ime je "Google", krajnja točka „https://www.google.com/accounts/o8/id", a URL logoa je adresa Googleovog logoa na Ferkovom poslužitelju („`img/loginLogos/Google.png`").

Model `RemoteAuth` pamti u sebi identifikator (`String remoteID`) korisnika koji dobija od davatelja te, naravno, davatelja (`RemoteAuthProvider provider`) i korisnika (`User user`).



Slika 11. Prikaz modela koji su dodani

5.3.2 Promjene u pogledu

Za ostvarenje prikaza potrebnih promjena, trebalo je modificirati dvije JSP datoteke - `UserEditForm.jsp` i `Login.jsp`. U oba slučaja potrebno je ispisati sve davatelje, pa na to moramo paziti kada prosljeđujemo podatke u njih.

5.3.3 Promjene u kontroleru

Kontroler omogućava komunikaciju poslužitelja i korisnika. On je zadužen za odgovaranje na zahtjeve koje šalje korisnik. Ovdje je bilo potrebno napraviti nekoliko izmjena.

Prva izmjena odnosi se na dodavanje koda za komunikaciju s davateljem, a ostvarena je u razredu `UserLoginOpenID`. On uključuje metodu koja traži korisnika koristeći njegov identifikator koji je dobio od davatelja (`checkUser`), metodu koja priprema zahtjev za preusmjerene na davateljev poslužitelj (`getRemoteUrl`) te metodu koja čita identifikator iz podataka koji su dobiveni od poslužitelja (`getVerificationResult`).

Druga izmjena donosi povezuje gore navedeni razred s pogledom, tj. omogućava komunikaciju s korisnikom. Kod koji to radi dodan je u razred `Login`.

Treća izmjena je dodavanje razreda `RemoteAuthService` koji omogućava pristup podacima o davateljima usluge te dodavanje veza između korisnika i davatelja.

Posljednja, četvrta izmjena odnosi se na dodavanje koda u razred `UserEditForm` koji omogućuje korisniku da doda i makne račun kod nekog davatelja usluge.

6. Problemi prilikom implementacije

Tijekom implementacije gore navedenih stvari, nekoliko bih stvari želio izdvojiti kao problematične:

- Nedostatak dokumentacije. Ovaj problem je čest kod velikih projekata koji imaju rokove. U prvi mah se čini da je dokumentacija gubitak vremena koje se može potrošiti na razvijanje drugih funkcionalnosti, međutim kada se nova osoba treba uključiti u razvijanje sustava ili kada se treba vratiti i promijeniti neki kod, tada se vidi zašto je isplativo pisati dokumentaciju. Pomoć mentora u ovom slučaju je bila i više nego korisna.
- Glomaznost projekta. Ferko je ogroman sustav i protiv ovoga se jako teško boriti. Najbolji je način mudro organizirati module koje se koristi u njemu, što je već i učinjeno.
- Nepoznavanje tehnologija. Očekivan problem budući da je autor bio svjestan da nije u potpunosti upoznat s Java tehnologijama koje je Ferko koristi. Uz pomoć menotra i interneta ovaj problem je sveden na najnižu moguću razinu.
- Googleov OpenID servis očekuje da će svaki zahtjev dolaziti s istog URL-a ili će zahtjev sadržavati podatak o području (*realm*). Ukoliko to nije ispunjeno, identifikator korsinika će se razlikovati. Na primjer, ako se pošalje zahtjev bez podatka o području s URL adrese “http://ferko.fer.hr/ferko” vraćeni identifikator neće biti jednak onome koji se vrati za zahtjev poslan s adrese “http://ferko.fer.hr/ferko/Login.action”. To je stvaralo probleme budući da većina ostalih davatelja OpenID usluge ne traže te podatke pa je nepotrebno utrošeno vrijeme za provjeravanje koda koji je točan.

7. Zaključak

Od početka Interneta, broj tehnologija koje se na njemu koriste je ogroman i raste iz dana u dan. Kako postoje tvrdokorni zagovornici gotovo svake od njih, teško je za očekivati da će se u dogledno vrijeme taj broj smanjiti. Štoviše, budući da još nema niti jedne "savršene", možemo očekivati da će broj pokušaja izgradnje nove koja će to biti. A kada govorimo o tehnologijama koje se koriste na Internetu, govorimo o programskim jezicima za poslužitelja, protokolima za komunikaciju na Internetu, programskim platformama...

Prilikom odabira tehnologija, izbor je prilikom izrade ovog seminara bio sužen budući da je Ferko pisan u programskom jeziku Java, a osim toga koristi Struts 2 i Hibernate. Odabir OpenID-a za uslugu ovjere bilo je relativno lako napraviti budući da se ona već koristi na mnogim stranicama. OpenID ubrzano postaje standard koji se koristi svugdje. Glavna (i praktički jedina) odluka bila je ona o načinu rješavanju dijeljenja kalendara. Nakon analize postojećih sustava, odabrana je alternativa sa javnim i tajnim ključem te sustavom prijatelja. Detalji donošenja te odluke, te njene prednosti i mane, objašnjeni su u samom radu.

Zbog svega navedenog, sasvim je jasno da je područje računarstva daleko od svojeg konačnog oblika te da još mnoge stvari nisu potpuno definirane. Očekuju nas uzbudljive godine u kojima će se sasvim sigurno Internet i tehnologije koje se koriste na njemu redovito mijenjati.

8. Literatura

1. Eckel, B. *Thinking in Java*, 3rd Edition. Upper Saddle River, New Yersy, USA: Prentice Hall, 2002.
2. Fitzpatrick, B., Recordon, D., Hardt, D., Hoyt, J. et al. *Open ID Authentication 2.0*
3. Brown, D., Davis, C. M., Stanlick, S. *Struts 2 in Action*. Greenwich, Connecticut, USA: Manning Publications, 2008.
4. Bauer, C., King, G. *Java Persistance with Hibernate*. Greenwich, Connecticut, USA: Manning Publications, 2006.
5. *OpenID Foundation Website*, http://openid.net/ (2. 5. 2010.)
6. *OAuth Community Site*, http://openid.net/ (2. 5. 2010.)
7. *Struts2 - Tutorial*, http://struts.apache.org/2.2.1/docs/tutorials.html (2. 5. 2010.)

9. Sažetak

Ovaj seminarski rad govori o unaprjeđenjima sustava Ferko. Slijedeće su stvari napravljene kako bi se korisnicima olakšalo njegovo korištenje:

Implementiran je sustav sigurnog dijeljenja informacija. Svaki korisnik sada može bilo kojem drugom korisniku jednostavno omogućiti pristupanje svojem kalendaru u iCal obliku.

Korisnik se može prijaviti na Ferko koristeći svoj OpenID korisnički račun. Time se miče potreba da eksplicitno upisuje svoje podatke za prijavu, nego može i jednostavno kliknuti na ikonu davatelja OpenID usluge te se time prijaviti na sustav.

U radu je objašnjeno zašto je dizajn sustava koji je napravljen kvalitetan i koje probleme on rješava. Također su u glavnim crtama opisani protokoli OpenID i OAuth.