SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 000

Praćenje aktivnosti korisnika usluge računalne provjere pravopisa

Stanko Krtalić Rusendić

Zagreb, lipanj 2016.

SADRŽAJ

1. Uvod **1**

2. Metode za praćenje korisnika **2**

2.1. Metode praćenja pomoću HTTP protokola 2

2.1.1. Kolačići (eng. cookies) 2

2.1.2. ETag i Last-Modified zaglavlja 3

2.1.3. Certifikati vezani uz izvorište (eng. Origin Bound Certificates) 3

2.1.4. Globalna IP adresa klijenta 3

2.1.5. Lokalna IP adresa klijenta 4

2.2. Metode praćenja pomoću Internetskog preglednika 4

2.2.1. Lokalno pohranjeni podaci (eng. Local Storage) 4

2.2.2. Podaci pohranjeni u svrhu brzog pristupa (eng. Cached objects) 4

2.2.3. Pomak takta sistemskog sata 5

2.2.4. Informacije o komponentama računala 5

3. Analiza i opis arhitekture idejnog rješenja **6**

3.1. Komponente sustava 6

3.1.1. Internetska aplikacija 6

3.1.2. Klijentska biblioteka 7

3.1.3. Poslužiteljska biblioteka 7

3.2. Dijagram toka 8

3.3. ER dijagram 9

4. Opis programskog rješenja **10**

4.1. Internetska aplikacija 10

4.2. Poslužiteljska biblioteka 11

4.3. Klijentska biblioteka 11

5. Instalacija i pokretanje **12**
5.1. UNIX 12
5.1.1. Ruby 12
5.1.2. PostgreSQL 13
5.1.3. Aplikacija 13
5.2. MS-DOS 14
5.2.1. Ruby 14
5.2.2. PostgreSQL 14
5.2.3. Aplikacija 14

6. Ideje za daljnji razvoj projekta **16**
6.1. Prijava korisnika 16
6.2. Migracija podataka u sustav za obradu velike količine podataka 16
6.3. Prikupljanje više podataka 17
6.4. Praćenje između uređaja 17

7. Zaključak **18**

Literatura **19**

1. Uvod

Internetske aplikacije dnevno posjećuju na tisuće korisnika koji koriste usluge aplikacije na različite načine. Obrasci ponašanja korisnika su bitni. Pomoću tih podataka se može lakše upravljati i preciznije unaprijediti pojedine usluge aplikacije.

Glavni problem je identifikacija korisnika. Iako je moguće dozvoliti korisniku da stvori račun u Internetskoj aplikaciji, te ga tako identificirati, većina korisnika se ne odluči stvoriti račun niti se prijavi u postojeći račun ako im tim postupkom aplikacija ne nudi značajno proširenu funkcionalnost. Posljedica takvog ponašanja korisnika je gubitak podataka o njihovim uzorcima korištenja aplikacije.

Stoga je cilj ovog završnog rada analiza izvedivosti i arhitekture, te implementacija sustav koji će omogućiti praćenje obrazaca ponašanja korisnika sustava za računalnu provjeru pravopisa koji nisu prijavljeni ili nemaju račun u Internetskoj aplikaciji.

U sklopu ovog završnog rada će se analizirati i objasniti dostupne metode praćenja korisnika. Nakon toga slijedi opis i analiza arhitekture sustava. Potom slijedi detaljan opis programskog rješenja i tehnologija korištenih u izradi rješenja, te upute za instalaciju i pokretanje. Na posljetku će biti opisane ideje za daljnje unapređenje programskog rješenja.

2. Metode za praćenje korisnika

Kako bi pratili, tj. identificirali, korisnika potrebno je prikupiti podatke pomoću kojih možemo jednoznačno, ili s visokom sigurnošću, ustanoviti o kojem se korisniku radi drj (2015). Navedene metode pojedinačno mogu biti dosta neprecizne, međutim sve zajedno nam daju visoku statističku sigurnost da ćemo točno identificirati korisnika. Janc i Zalewski (2014)

2.1. Metode praćenja pomoću HTTP protokola

Internetske aplikacije koriste HTTP protokol kako bi primile zahtjev od korisnika i kako bi mu prenijele generirani odgovor na zahtjev. Mehanizme koje protokol implementira u svrhu optimizaciju upita, te mehanizme za perzistenciju stanja između klijenta i poslužitelja je moguće iskoristiti u svrhu identifikacije korisnika.

2.1.1. Kolačići (eng. cookies)

Kolačići su mehanizam perzistencije stanja. Kolačići su tekstualne datoteke koje klijent i poslužitelje razmjenjuju pri svakom upitu. U njih se zapisuju podaci koje klijent ili poslužitelj žele perzistirati u slučaju greške ili za potrebe domenske logike. Group (1999)

Ovaj mehanizam se može iskoristiti u svrhu stvaranja 'kolačića za praćenje'.

Kolačić za praćenje (eng. tracking cookie) služi kao spremište za podatak prema kojem možemo identificirati korisnika. (npr. identifikacijski broj ili slijed znamenki)

Problem s kolačićem za praćenje je njegova transparentnost (korisnik lako može otkriti njegovo postojanje), dostupnost alata za uklanjanje kolačića, te direktiva 2009/136/EC Europske unije Eur (2009) koja nalježe da aplikacije moraju jasno i transparentno obavijestiti korisnika koristi li kolačiće i u koje svrhe ih koriste.

2.1.2. ETag i Last-Modified zaglavlja

ETag i Last-Modified zaglavlja su mehanizmi optimizacije upita. ETag i Last-Modified zaglavlja služe za dojavljivanje promjene sadržaja. Prvotno ih postavlja poslužitelj kada šalje klijentu odgovor, a klijent ih šalje poslužitelju pri svakom sljedećem upitu na istu putanju. Poslužitelj može iz poslanih zaglavlja ustanoviti je li se sadržaj koji se servira promijenio od trenutka posljednjeg klijentovog upita i ovisno o tome odgovoriti s novim sadržajem ili s praznim odgovorom. Ovaj mehanizam značajno ubrzava rad poslužitelja.

Prema standardu ETag može biti proizvoljan slijed znakova, dok Last-Modified treba predstavljati datum. Međutim format datuma za Last-Modified nije standardiziran, te je efektivno isto proizvoljan slijed od 32 znaka.

Ovi mehanizmi se ponašaju slično kao kolačići, te se mogu iskoristiti u svrhu pohranjivanja identifikatora. Korisniku nije dostupan alat s kojim bi mogao lagano otkloniti ova zaglavlja, niti može lako otkriti u koju svrhu se koriste zaglavlja. Stoga je ova metoda identifikacije efikasnija od kolačića.

2.1.3. Certifikati vezani uz izvorište (eng. Origin Bound Certificates)

poznatiji kao identifikatori kanala (eng. channelID), služe za perzistenciju podataka između klijenta i poslužitelja na strani poslužitelja.

Ponašanjem su identični kolačićima. Predstavljaju idealni način za pohranjivanje identifikatora jer korisnik ne posjeduje mehanizam s kojim bi izbrisao podatke.

2.1.4. Globalna IP adresa klijenta

Može se koristiti kao identifikator, međutim jako je nepouzdana. Naj rasprostranjeniji oblik IP adresa su IPv4 adrese koje se koriste od sredine 1983. godine. Problem s IPv4 standardom je maksimalni mogući broj adresa. Kako postoji više uređaja koji pristupaju internetu nego slobodnih IPv4 adresa, potrebno je reciklirati adrese. Tj. kada neki uređaj ne koristi njemu dodijeljenu IP adresu ona se dodjeljuje drugom uređaju.

Stoga je ovakav mehanizam za identifikaciju korisnika izrazito nepouzdan. Implementacijom IPv6 standarda bi bilo moguće svakom korisniku dodijeliti unikatnu IP adresu. Međutim postojeća mrežna infrastruktura još nije spremna za potpuni prelazak na novi standard.

2.1.5. Lokalna IP adresa klijenta

Kao jedno od rješenja problema nedostatka IPv4 adresa se koriste prevoditelji mrežnih adresa (eng. Network Address Translator) koji omogućuju da više korisnika iz iste mreže pristupa Internetu pod istom IP adresom. program (1981)

Lokalne mreže nisu podložne čestim promjenama. Stoga se korisnikova IP adresa unutar lokalne mreže rijetko mijenja, te se ta informacija može koristiti u svrhu identifikacije korisnika. Međutim puno korisnika ima sličnu konfiguraciju mreže te je teško jednoznačno identificirati nekog korisnika.

2.2. Metode praćenja pomoću Internetskog preglednika

Korisnici koriste Internetske preglednike (eng. 'browser') kako bi ostvarili interakciju s aplikacijom, te kako bi na jednostavan i vizualno razumljiv način konzumirali HTML datoteke koje im ona šalje. Moderni preglednici implementiraju razne tehnologije i mehanizme pomoću kojih omogućuju Internetskoj aplikaciji da servira interaktivni sadržaj i optimira zahtjeve za vanjskim resursima. Slično kao i kod mehanizama HTTP protokola, ove mehanizme je moguće iskoristiti za pohranjivanje identifikatora. Group (2013)

2.2.1. Lokalno pohranjeni podaci (eng. Local Storage)

. Moderni preglednici dozvoljavaju korisniku da pomoću JavaScript, Flash i Silverlight skripti korisniku ponudi interaktivni sadržaj koji se perzistira.

Slično kao i kod kolačića u HTTP protokolu, moguće je pohraniti podatke na korisnikovo računalo te u njih pohraniti identifikator. Iako je ovakav način pohrane podataka manje transparentan od kolačića, korisnicima su lako dostupni alati za brisanje lokalno pohranjenih podataka.

2.2.2. Podaci pohranjeni u svrhu brzog pristupa (eng. Cached objects)

. Moderni preglednici pohranjuju podatke poput slika i CSS datoteka lokalno kako bi korisniku mogli brže prikazati Internetsku stranicu. Pomoću stenografskih metoda je moguće stvoriti sliku koja u sebi sadrži identifikator i predati ju pregledniku da ju pohrani.

2.2.3. Pomak takta sistemskog sata

Svako računalo ima oscilacijski kristal koji generira takt jezgre procesora i sistemskog sata. Svi kristali imaju pomak od njihove rezonantne frekvencije koji je unikatan za taj kristal.

Pomak frekvencije je moguće izmjeriti kroz preglednik i time jedinstveno identificirati korisnikovo računalo.

2.2.4. Informacije o komponentama računala

Korisnika je moguće identificirati prema popisu komponenta njegovog računala.

Ovakav način identifikacije korisnika je izrazito nepouzdan jer danas postoji ogroman broj sličnih ili identičnih računalnih konfiguracija.

3. Analiza i opis arhitekture idejnog rješenja

Programsko rješenje se mora integrirati s uslugom računalne provjere pravopisa. Kako nam je arhitektura i izvedba usluge za provjeru pravopisa nepoznata nalaže se arhitektura servisa.

Odnosno, programsko rješenje će biti Internetska aplikacija koju će druge aplikacije koristiti kao servis za identifikaciju korisnika.

3.1. Komponente sustava

Sustav se sastoji od tri komponente. Programske biblioteke koja se izvodi u pregledniku korisnika (u daljnjem tekstu klijentska biblioteka), biblioteke koja se izvodi na poslužitelju (u daljnjem tekstu poslužiteljska biblioteka), te Internetske aplikacije.

3.1.1. Internetska aplikacija

Internetska aplikacija je središnja komponenta sustava. Klijentska i poslužiteljska biblioteka razgovaraju s Internetskom aplikacijom preko HTTP-a kako bi joj dojavile podatke o korisniku prema kojima će ona pronaći ili generirati identifikator, te uputiti biblioteku koje podatke da pohrani ili servira korisniku.

Aplikacija treba implementirati programsko sučelje preko kojeg može razgovarati s bibliotekama.

Ustroj baze podataka je izrazito bitan. Kako se radi o potencijalno velikom setu podataka koje će aplikacija morati pretraživati bi svi primarni ključevi trebali biti tipa UUID. Te je izrazito bitno dobro postavljanje indeksa na kolumne tablica. Optimalna implementacija baze podataka bi bila PostgreSQL baza zbog brzine i mogućnosti pohranjivanja i pretraživanja indeksiranih no-SQL podataka što omogućuje pohranjivanje kompleksnih seta podataka.

Aplikacija za svakog korisnika stvara profil. Korisnikov profil se asocira uz njegov 'otisak prsta' koji predstavljaju podaci o CPU, GPU i kristalu računala koje posjećuje servis. Profil se također asocira uz oznake (eng. tags) koji predstavljaju oznake HTTP zaglavlja i putanju na koju je to zaglavlje bilo poslano. Konačno, profil se asocira i uz ip adresu s koje je upit napravljen.

3.1.2. Klijentska biblioteka

Klijentska aplikacija prikuplja podatke o pregledniku i računalu korisnika, zapisuje ih u format koji Internetska aplikacija može interpretirati i šalje na obradu.

Kada dobije od Internetske aplikacije odgovor koji sadrži identifikator, zapiše ga u sve dostupne medije za pohranu u klijentovom pregledniku, te dojavljuje usluzi za provjeru pravopisa identifikator korisnika i putanju kojoj je pristupio.

Kako svi moderni preglednici implementiraju i dozvoljavaju izvršavanje JavaScript skripti na korisnikovom uređaju, implementacija ove biblioteke će biti u programskom jeziku JavaScript.

Problem izvršavanja logike na klijentovom pregledniku je sposobnost korisnika da ugasi mogućnost izvršavanja JavaScript skripti u pregledniku. Ovakva situacija nije česta, korisnik tim činom gubi cjelovitu funkcionalnost usluge za provjeru pravopisa. Međutim domišljat korisnik bi mogao iz izvornog JavaScript koda servisa iščitati putanje i tipove upita koje mora napraviti kako bi servisu predao tekst na ispravljanje. Kao mehanizam detekcije takvog načina korištenja servisa postoji poslužiteljska biblioteka.

3.1.3. Poslužiteljska biblioteka

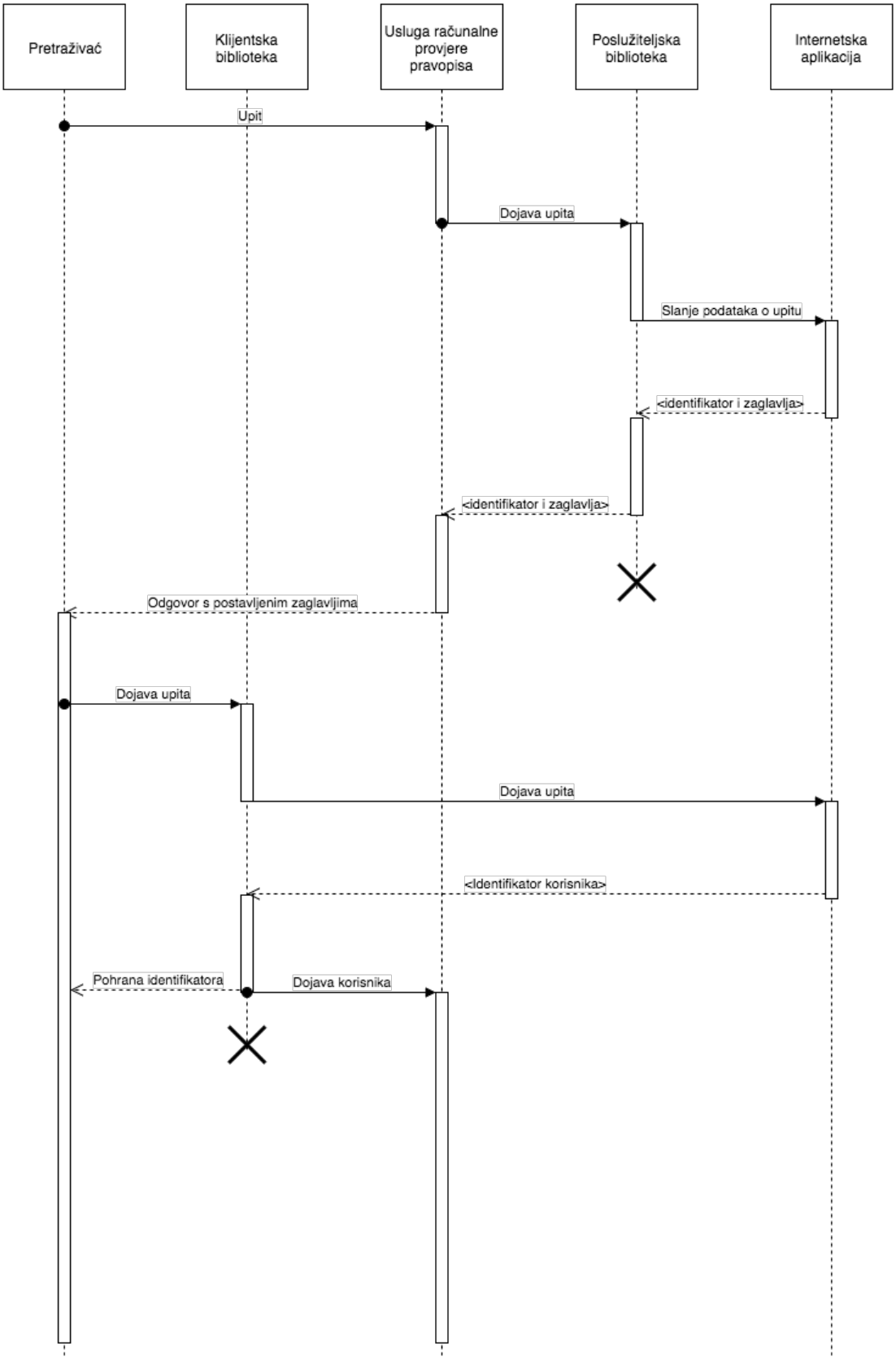
Poslužiteljska biblioteka pri svakom korisnikovom upitu ekstrahira podatke iz zaglavlja HTTP upita i šalje ih na obradu Internetskoj aplikaciji.

Internetska aplikacija odgovara biblioteci s identifikatorom korisnika, ako on postoji, i podacima koje treba postaviti u zaglavlje HTTP odgovora.

Poslužiteljska biblioteka može biti implementirana u bilo kojem programskom jeziku.

Poslužiteljska biblioteka služi kao mehanizam za detekciju korisnika koji nisu pokrenuli klijentsku biblioteku, te korisnika koji pokušavaju zaobići transparentne mehanizme identifikacije.

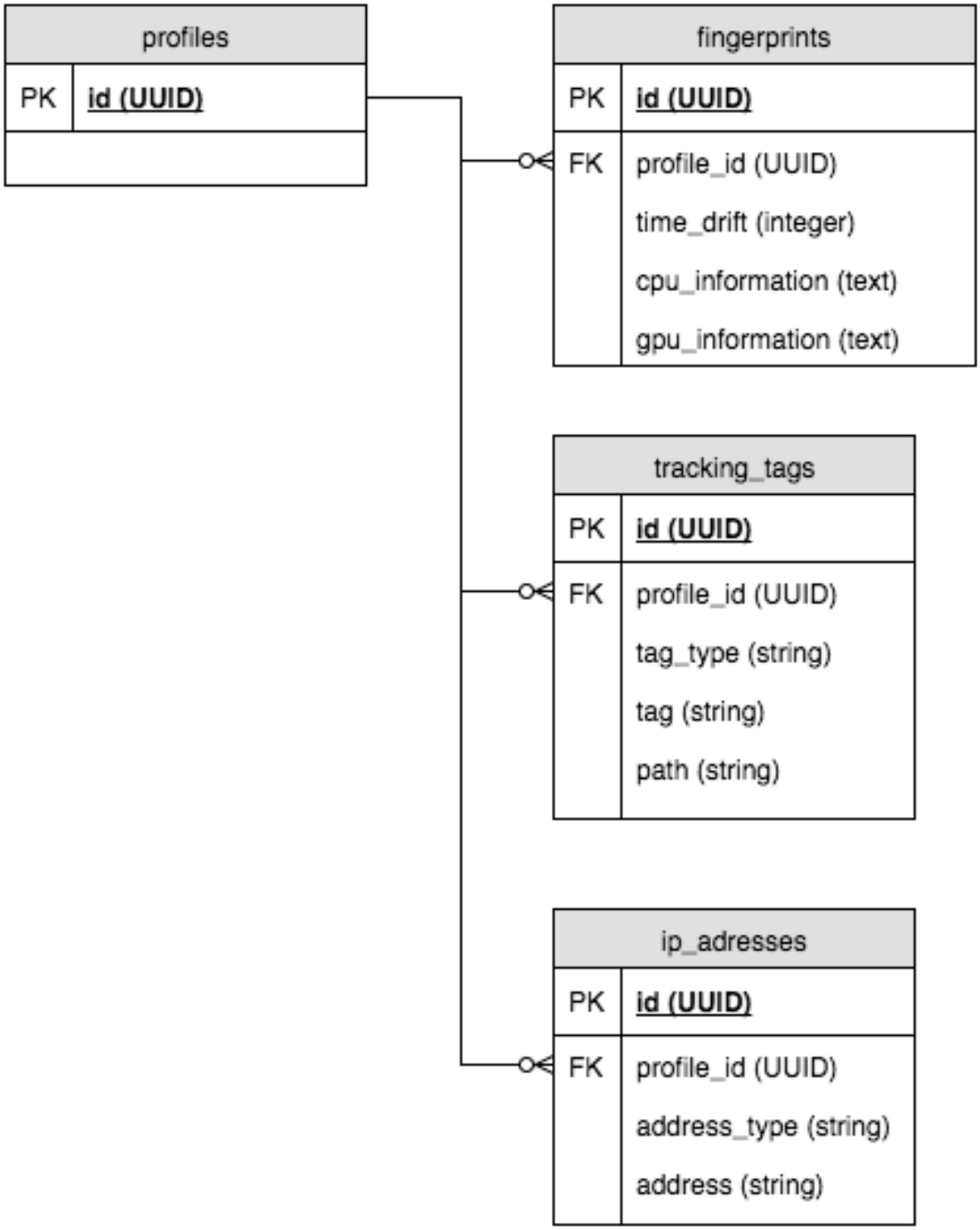
3.2. Dijagram toka



Dijagram toka prikazuje tok izvođenja programa od strane klijentskog preglednika,

servisa za ispravljanje pravopisa, Internetske aplikacije, klijentske i poslužiteljske biblioteke.

3.3. ER dijagram



ER dijagram prikazuje ustroj baze podataka.

4. Opis programskog rješenja

Kao što je i opisano u prethodnom poglavlju. Programsko rješenje se sastoji od tri dijela.

4.1. Internetska aplikacija

Internetska aplikacija implementirana je u programskom jeziku Ruby pomoću Ruby on Rails aplikacijskog okvira (eng. framework). Ruby on Rails (u daljnjem tekstu Rails) nam nudi slojeve modela, pogleda i kontrolera (eng. MVC - Model View Controller), te mnoge biblioteke koje nas štite od sigurnosnih propusta poput injekcije SQL naredbi (eng. SQL injection), među straničnog skriptiranja (eng. Cross Site Scripting), preljeva kolačića (eng. Cookie overflow) i mnogih drugih. Osim što nam nudi visoki sigurnosni standard nam Rails nudi obradu upita kroz posredničke programe (eng. middleware) i bogati ekosustav gotovih biblioteka. Rub (2016)

U Ruby on Rails aplikacijama model reprezentira podatak iz baze podataka. Manipulacijom atributa modela se direktno manipuliraju podaci u bazi podataka. Kroz modele možemo i raditi upite nad bazom i dobivati kolekcije modela koji zadovoljavaju dani SQL upit. Rails se automatski brine o optimizaciji upita.

Kontroleri u Ruby on Rails aplikacijama služe kao posrednički sloj između pogleda i modela. Rails aplikacija ne mora nužno odgovoriti s pogledom. Iako kontroleri služe kao posrednički sloj oni su u potpunosti zaslužni za formiranje odgovora korisniku.

Pogledi su HTML datoteke koje se poslužuje kao odgovor na upit. Rails koristi ugrađeni Ruby kod (eng. Embeded Ruby - ERB) kako bi dinamički izmijenio HTML koji poslužuje klijentima.

Rails aplikacija ima četiri modela. Profil u koji se pohranjuje identifikator korisnika. Zatim 'otiske prstiju' u koji se pohranjuju podaci specifični za korisnikovo računalo. IP adrese u koje se pohranjuju IP adrese s kojima je korisnik napravio zahtjeve na servis. I konačno, oznake za praćenje (eng. tracking tags) gdje se pohranjuju oznake HTTP zaglavlja.

Postoji samo jedan kontroler. Taj kontroler je zaslužan za komunikaciju između aplikacije i biblioteka. Kako kontroler prima i odgovara na zahtjeve s JSON serijaliziranim objektima nema potrebe za pogledima.

Logika za identifikaciju korisnika pomoću predanih podataka se nalazi u servisnim objektima (eng. service objects) koji razgovaraju s objektima za izgradnju SQL upita (eng. query objects). Ovakav način korištenja je proširenje obične MVC arhitekture. Cilj ovakvog proširenja je apstrakcija implementacije poslovne logike i baze podataka. Odnosno, cilj je omogućiti nam da možemo proizvoljno centralizirano mijenjati logiku, bez da moramo mijenjati kod u više datoteka.

4.2. Poslužiteljska biblioteka

Poslužiteljska biblioteka je implementirana kao Ruby datoteka, odnosno Ruby on Rails servisni objekt. Kontroler servisu predaje zaglavlja koja je dobio u zahtjevu, a servis mu odgovara sa zaglavljima koje mora postaviti u odgovoru, te identifikatorom korisnika ako je korisnik uspješno identificiran.

4.3. Klijentska biblioteka

Klijentska biblioteka implementirana je kao JavaScript objekt. Nakon što se doda u HTML koji se servira korisniku, potrebno je konfigurirati putanju Rails aplikacije i putanju na koju se dojavljuje identifikator korisnika.

Biblioteka koristi evercookie biblioteku kako bi perzistirala podatke u klijentovom pregledniku.

5. Instalacija i pokretanje

Za pokretanje programskog rješenja potrebno je računalo s operativnim sustavom koji implementira POSIX ili MS-DOS standard na kojem je instaliran Ruby 2.3.0 virtualni stroj, te pristup PosgreSQL bazi podataka.

5.1. UNIX

5.1.1. Ruby

Dok je aplikaciju moguće upogoniti pomoću sistemske instalacije Ruby virtualnog stroja to nije preporučljivo za produkcijska okruženja jer ne ostavlja mogućnost pokretanja više aplikacija koje koriste različite verzije virtualnog stroja.

Preporučena metoda instalacije virtualnog stroja je pomoću menedžera verzija. Rbenv je jedan od najpopularnijih manedžera verzija za Ruby virtualni stroj. Kako bi ga instalirali potreban nam je Git sustav za verzioniranje.

Na debian distribucijama je potrebno izvršiti naredbu koja će na sustav instalirati Git:

```
apt-get update
sudo apt-get install build-essential
apt-get install git
```

Potom je potrebno preuzeti rbenv pomoću Gita i instalirati ruby-build plugin:

```
git clone https://github.com/rbenv/rbenv.git ~/.rbenv
cd ~/.rbenv && src/configure && make -C src

# za sustave koji koriste ZSH terminal
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.zshrc
```

```
# za Ubuntu Desktop operativne sustave
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc

# za ostale operativne sustave
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bash_profile

~/.rbenv/bin/rbenv init
git clone https://github.com/rbenv/ruby-build.git \
~/.rbenv/plugins/ruby-build
rbenv install 2.3.0
rbenv global 2.3.0
```

5.1.2. PostgreSQL

Za instalaciju PosgreSQL baze podataka potrebno je izvršiti sljedeće naredbe:

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/\
`lsb_release -cs`-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'
wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - |\
sudo apt-key add -
sudo apt-get update
sudo apt-get install postgresql postgresql-contrib
```

5.1.3. Aplikacija

Kako bi se instalirala aplikacija potrebno ju je samo kopirati na računalo. To je moguće napraviti kopiranjem izvornog koda za CD-a ili pomoću Gita te pokrenuti sljedeće naredbe iz direktorija u kojem se nalazi aplikacija:

```
# ako zelite kopirati aplikaciju pomocu gita
cd
cd Desktop
git clone git@github.com:Stankec/fer_final_paper.git
cd fer_final_paper
```

```
gem install bundler
cd web_app
bundle install
```

Konačno možemo pokrenuti aplikaciju i demo aplikaciju koja implementira biblioteke:

```
cd web_app
rails s -p 3000
```

Ovim naredbama smo podignuli dvije web aplikacije. Aplikaciju opisanu u programskom rješenju i demonstracijsku aplikaciju koja implementira klijentsku i poslužiteljsku biblioteku.

Aplikacija koja predstavlja programsko rješenje nalazi se na URL adresi http://localhost:3000

5.2. MS-DOS

5.2.1. Ruby

Instalirati Ruby na Windows operativnim sustavima je najlakše napraviti pomoću instalacijskog čarobnjaka kojeg se može preuzeti s ove poveznice: http://rubyinstaller.org/

5.2.2. PostgreSQL

PostgreSQL bazu podataka je najlakše instalirati kroz instalacijski čarobnjak dostupan na ovoj poveznici https://www.postgresql.org/download/

5.2.3. Aplikacija

Kao i kod instalacije na POSIX operativne sustave potrebno je kopirati izvorni kod aplikacije na računalo, otvoriti Command prompt i pomaknuti se u direktorij aplikacije i potom izvršiti sljedeće naredbe:

```
gem install bundler
cd web_app
bundle install
rails s -p 3000
```

Ovim naredbama smo podignuli dvije web aplikacije. Aplikaciju opisanu u programskom rješenju i demonstracijsku aplikaciju koja implementira klijentsku i poslužiteljsku biblioteku.

Aplikacija koja predstavlja programsko rješenje nalazi se na URL adresi http://localhost:3000

6. Ideje za daljnji razvoj projekta

Programsko rješenje se može dalje unaprjeđivati. U nastavku je nekoliko ideja koje bi mogle značajno unaprijediti programsko rješenje:

6.1. Prijava korisnika

U sustav bi se mogao implementirati sustav za stvaranje korisničkih računa za pojedinačne korisnike i za firme. To bi omogućilo da se sustav monetizira iznajmljivanjem odnosno plaćanjem pristupa.

6.2. Migracija podataka u sustav za obradu velike količine podataka

Za programsko riješenje bi se moglo reći da ima naivnu implementaciju dohvata, pohranjivanja i agregacije podataka. Makar će ovakav sustav zadovoljiti potrebe servisa za programsko ispravljanje pravopisa ne bi ga bilo moguće skalirati na servise koji primaju više upita. Kako bi se otklonio ovaj problem bilo trebalo bi periodički brisati podatke iz PostgreSQL baze i ostaviti samo agregati obrisanih podataka. Obrisane podatke bi bilo korisno migrirati u bazu podataka za obradu velikog broja podataka (eng. Big Data Database) Mon (2015) U takvoj konfiguraciji bi sustav imao dvije baze podataka. PosgreSQL bazu koju bi koristio za identifikaciju korisnika u realnom vremenu i bazu podataka za velki broj podataka koju bi koristio za generiranje statističkih modela. Generirani modeli bi se mogli iskoristiti za treniranje neuralne mreže koja bi mogla prema danim podacima identificirati korisnika.

6.3. Prikupljanje više podataka

U trenutnom programskom rješenju se ne prikupljaju svi mogući podaci od korisnika već samo podaci potrebni za identifikaciju korisnika. Sustav bi se mogao unaprijediti pohranjivanjem dodatnih podataka koji bi se mogli iskoristiti u svrhu analize demografije korisnika.

6.4. Praćenje između uređaja

Praćenje između uređaja je moguće izvesti kombiniranjem tradicionalnog principa prijave u korisnički račun s predstavljenim programskim rješenjem. U trenutku prijave korisnika u sustav bi se Internetskoj aplikaciji dojavio njegov identifikator u sustavu u koji se prijavio. Kako se korisnik prijavljuje na različite uređaje bi se u Internetskoj aplikaciji njegov profil povezao s podacima koje je sustav dobio od istog korisnika s drugih uređaja, te bi mogao identificirati uređaj uz korisnika iako korisnik nije prijavljen u sustav.

7. Zaključak

Kao što se moglo zaključiti kroz dosadašnja poglavlja ovog rada, praćenje aktivnosti korisnika usluge računalne provjere pravopisa je izvedivo i predstavlja neintruzivan oblik agregacije podataka u svrhu unapređenja usluge.

Iako sustav ne garantira da će točno identificirati korisnika. On to čini sa dovoljno velikom statističkom sigurnošću da se može koristiti u produkcijskim okruženjima.

Uz daljnja unaprjeđenja sustava koja su opisana u prethodnom poglavlju bi se povećala pouzdanost sustava i sposobnost sustava da generira relevantne statističke podatke i modele koji bi se mogli iskoristiti u svrhu unapređenja usluge računalne provjere pravopisa.

LITERATURA

11 Ways To Track Your Moves When Using a Web Browser, 2015. URL https://isc.sans.edu/forums/diary/11+Ways+To+Track+Your+Moves+When+Using+a+Web+Browser/19369/.

Directive 2009/136/EC of the European Parliament and of the Council. European Parliament, 2009. URL http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32009L0136.

Network Working Group. *Hypertext Transfer Protocol*. IETF, 1999. URL https://www.ietf.org/rfc/rfc2616.txt.

W3C Working Group. *Hypertext Markup Language*. W3C, 2013. URL https://www.w3.org/TR/html-markup/Overview.html#toc.

Arthur Janc i Michal Zalewski. *Technical analysis of client identification mechanisms*. Google, 2014. URL https://www.chromium.org/Home/chromium-security/client-identification-mechanisms#TOC-Explicitly-assigned-client-side-identifiers.

Big Data Explained. MongoDB Team, 2015. URL https://www.mongodb.com/big-data-explained.

DARPA Internet program. *Internet protocol*. DARPA, 1981. URL https://tools.ietf.org/html/rfc791.

Ruby on Rails Guides. Ruby on Rails core team, 2016. URL http://guides.rubyonrails.org/.

Praćenje aktivnosti korisnika usluge računalne provjere pravopisa

Sažetak

Praćenje aktivnosti korisnika usluge računalne provjere pravopisa.

Procjena izvedivosti, te opis arhitekture i izvedbe sustava za praćenje aktivnosti korisnika na usluzi računalne provjere pravopisa poput Hrvatskog akademskog spelling checkera (Hascheck).

Ključne riječi: praćenje aktivnosti korisnika, identifikacija korisnika, praćenje korisnika, Internetska aplikacija

Tracking the Activity of Online Spellchecker Users

Abstract

Tracking the Activity of Online Spellchecker Users.

Feasibility assessment, architecture and implementation of a system for monitoring user activity on a online spell check like the Croatian academic spelling checker (Hascheck).

Keywords: user activity tracking, user identification, user tracking, web application