Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tanja Gomilar

Integracija delno strukturiranih podatkov iz spletnih virov problemske domene turizma

diplomsko delo

univerzitetni strokovni študijski program prve stopnje računalništvo in informatika

Mentor: doc. dr. Dejan Lavbič

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00058/2012
Datum: 05.11.2012

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

Kandidat: **TANJA GOMILAR**

Naslov: **INTEGRACIJA DELNO STRUKTURIRANIH PODATKOV IZ SPLETNIH VIROV PROBLEMSKE DOMENE TURIZMA**

INTEGRATION OF SEMI-STRUCTURED WEB DATA IN TOURISM

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Na svetovnem spletu najdemo veliko količino podatkov s širšega področja turizma. Uporabniki lahko s pomočjo različnih spletnih strani in portalov poiščejo različne informacije o turističnih destinacijah, mnenju ostalih uporabnikov, iščejo ugodne turistične aranžmaje ipd. Ključna težava, s katero se srečujejo, je delno samodejna integracija podatkov. Spletne strani so namreč pripravljene za človeške uporabnike, ki morajo vsebino interpretirati sami. V okviru diplomske naloge naj kandidat realizira inteligentnega pomočnika, ki iz različnih delno strukturiranih spletnih virov pridobiva informacije in pomaga uporabniku pri sprejemanju odločitev ob planiranju izleta. Pri virih podatkov se osredotočite na slovenske turistične atrakcije, vhodne podatke pomočnika pa naj predstavljajo še drugi viri, kot so koledar uporabnika (zasedenost), lokacija uporabnika (GPS) in družabna omrežja.

Mentor:

doc. dr. Dejan Lavbič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Tanja Gomilar, z vpisno številko **63070100**, sem avtorica diplomskega dela z naslovom:

Integracija delno strukturiranih podatkov iz spletnih virov problemske domene turizma

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Dejana Lavbiča,
- 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 5. novembra 2012 Podpis avtorice:

Zahvala družini, Roku, sošolcem in mentorju za izkazano pomoč, zaupanje in znanje!

Kazalo

Slike

SLIKE

Tabele

Seznam kratic in simbolov

- API (Application Programming Interface): programski vmesnik, ki predstavlja niz rutin, protokolov in orodij za gradnjo programskih aplikacij in medsebojno komunikacijo različnih programskih oprem.

- HTML (Hyper Text Markup Language): označevalni jezik za določitev strukture dokumentov, ki se prenašajo po spletu in pregledujejo s spletnimi brskalniki.

- GPS (Global Positioning System): sistem globalnega določanja lege, satelitski navigacijski sistem, ki se uporablja za določanje točne lege in časa kjerkoli na Zemlji ali v zemeljski tirnici.

- HTTP (Hypertext Transfer Protocol): aplikacijski protokol za porazdeljevanje in sodelovanje med informacijskimi sistemi in hipermediji[1]. Je temelj posredovanja podatkov za svetovni splet.

- JavaScript: objektni skriptni programski jezik, ki omogoča spletnim programerjem ustvarjanje interaktivnih spletnih strani.

- JSON (JavaScript Object zapis): format za izmenjavo podatkov.

- URI (Uniform Resource Identifier): niz znakov, ki se uporabljajo za identifikacijo imena ali vira.

[1]Razširitev hiperteksta, ki poleg besedilnih elementov podpira povezovanje grafičnih, zvočnih in video elementov

- URL (Uniform Resource Locator): niz znakov, ki predstavlja sklicevanje na internetni vir. Predstavlja naslov spletne strani v svetovnem spletu.
- GET zahteva: omogoča pridobitev določenega vira na svetovnem spletu.
- POST zahteva: omogoča pridobitev določenega vira na svetovnem spletu, kjer telo zahtevka vsebuje podatke, ki jih želimo sporočiti strežniku.
- REST (REpresentational State Transfer): slog arhitekture programske opreme, ki omogoča dosleden pristop k spletnim zahtevam in spreminjanjem podatkov.
- DOM (Document Object Model): model, ki podpira vmesnik med dokumenti HTML, XHTML, XML in programi. Dokumenti so z modelom DOM predstavljeni z drevesno strukturo.
- Twitter: spletno družabno omrežje in „microblogging“storitev, ki omogoča svojim uporabnikom pošiljanje in branje tekstovnih sporočil dolžine največ 140 znakov, znanih po izrazu „tweets“.
- Facebook: brezplačna spletna storitev, ki omogoča družabno omrežje preko spletne strani in drugih aplikacij.
- SDK (Software Development Kit): paket za razvoj programske opreme.
- CSS3 (Cascading Style Sheets): kaskadne stilske podloge, predstavljene v obliki preprostega slogovnega jezika, ki skrbi za prezentacijo spletnih strani.
- XSS (Cross-site scripting): vrsta računalniške varnostne ranljivosti tipična pri spletnih aplikacijah.

Povzetek

Namen diplomskega dela je realizacija in opis inteligentnega pomočnika za problemsko domeno turizma, ki temelji na heterogenih podatkih iz spletnih virov s področja turizma in uporabniku ponuja izbiro zanj najbolj ugodne turistične destinacije za izlet. Pomočnik uporabniku omogoča izvedbo kratkoročnega planiranja izleta po Sloveniji.

Za realizacijo inteligentnega pomočnika je bilo ključnega pomena luščenje in obdelava podatkov o uporabnikovi lokaciji in času, ki ga ima na voljo, obdelava spletnih virov ter implementacija različnih programskih vmesnikov (API), ki omogočajo pridobivanje in obdelavo uporabnikovih podatkov.

Programska logika pomočnika je realizirana v programskem jeziku Java. Ta vsebuje potrebne knjižnice za luščenje podatkov iz spletnih virov in programske vmesnike, ki omogočajo pridobivanje uporabnikove trenutne lokacije, vpogled v uporabnikov Google koledar, pretvorbo geografskih (GPS) koordinat v naslove, izračun poti in časa za dosego turističnih destinacij, vpogled v vremensko stanje turističnih destinacij ter interakcijo z družabnim omrežjem Twitter.

Realiziran pomočnik je osnova ideje o razvoju programa oziroma aplikacije, ki uporabniku omogoča na kratek rok izbrati lokacijo izleta v Sloveniji. Opisane so tudi ideje o nadgradnji in razširitvi pomočnika ter njihove slabosti in prednosti.

Ključne besede: turizem, programski vmesnik, heterogeni podatki, luščenje podatkov, Java

Abstract

The aim of this thesis is the realization and description of an intelligent assistant for the problem domain of tourism, which is based on the heterogeneous online data sources in the field of tourism and it enables the user to have the choice of the most favourable tourist destination for the trip. The assistant allows the user to carry out short-term planning trip in Slovenia.

For the realisation of the assistant the scaling and processing of data at the user's location and the available time for the processing of online sources and implementation of various programming interfaces (APIs) that enable the acquisition of user data and the processing thereof were critical.

Program Assistant logic is realized in the Java programming language. It contains the necessary libraries for scaling data from online sources and application programming interfaces which allow obtaining user's current location, access to the user's Google calendar, convert geographic (GPS) coordinates to addresses, route calculation and time to reach tourist destinations as well as insight into the state of the weather tourist destinations.

The realized assistant is the base of an idea of developing a program or applications which allow the user to select the location trip in Slovenia in a short period. The ideas about upgrading and extension assistant and their advantages and disadvantages have also been described.

Keywords: tourism, application programming interface, heterogeneous data, data scaling, Java

Poglavje 1

Uvod

Tempo življenja, ki nam ga narekuje današnji življenjski slog, je naravnan k hitrim odločitvam in dobremu koriščenju časa. Zato je za sodobnega človeka pomembno, da lahko svoj prosti čas izkoristi učinkovito in da za to ne porabi preveč časa in energije. S pomočjo sodobne tehnologije si tako lahko posameznik organizira urnik in uporabi za njegove potrebe narejene programe spletne strani in aplikacije, ki omogočajo hitro in dobro organizacijo aktivnosti in časa.

Ideja o inteligentnem pomočniku, ki uporabniku omogoči planiranje izleta po Sloveniji, na kratek rok izhaja ravno iz omenjenih domnev in dejstev. Ko se uporabnik odloči, da bo svojih nadaljnjih nekaj ur porabil za izlet po Sloveniji, lahko za planiranje le-tega uporabi inteligentnega pomočnika. Inteligentni pomočnik tako uporabniku ponudi izbiro časa za ogled destinacije, ga po vpogledu v uporabnikov Google koledar seznani, če ima za ogled destinacije dovolj časa oziroma v koledarju nima že vnesenih opravkov, izpiše vremenske podatke o izbrani destinaciji, izriše zemljevid poti od trenutne lokacije do izbrane turistične destinacije in mu ponudi možnost interakcije s socialnim omrežjem Twitter.

Sama realizacija inteligentnega pomočnika, ki obsega okvire te diplomske naloge, je bila narejena kot program, napisan v programskem jeziku Java, ki obsega implementacijo in simulacijo logike ter ideje inteligentnega

pomočnika. Pri izdelavi diplomske naloge sem se osredotočila na realizacijo programske logike, z grafičnim uporabniškim vmesnikom se nisem ukvarjala.

Za realizacijo inteligentnega pomočnika je bilo treba najprej izbrati primerne vire podatkov in tehnologij. Ti so podrobneje opisani v poglavju 2, ki natančneje opisuje izbrane tehnologije, ki so bile potrebne pri sami izdelavi programa. Te tehnologije so po večini programski vmesniki. Ti omogočajo dostop do uporabnikovega Google računa in drugih Googlovih storitev in so opisani v poglavju 2.3. Delovanje programskega vmesnika za dostop do Twitter uporabnikovega računa je opisan v poglavju 2.5. Pridobivanje in obdelava podatkov o vremenskih razmerah za določeno turistično destinacijo sta opisana v poglavju 2.4. Opisan je tudi izbor turističnih destinacij, ki je predstavljen v poglavju 2.1, in tehnologija luščenja podatkov, ki je predstavljena v razdelku 2.1.1.

Programsko okolje inteligentnega pomočnika je predstavljeno v poglavju 3.1. Postopek razvoja inteligentnega pomočnika je podrobneje opisan v poglavju 3.2. Samo delovanje inteligentnega pomočnika z opisanimi fazami delovanja (poglavje 3.3.1) in konkretnim primerom uporabe programa (poglavje 3.3.2) pa v poglavju 3.3. V poglavju 3.4 je obrazložena uporaba tehnologij predstavljenih v poglavju 2. V poglavju 3.5 pa so navedene omejitve obravnavanih programskih vmesnikov in težave, ki so jih le-te povzročale.

Konkretneje so predstavljene tudi nadaljnje nadgradnje ideje in programa. Te so predstavljene v poglavju 4. V diplomski nalogi je v poglavju 4.1 raziskana in obrazložena tudi možnost nadgradnje oz. transformacije programa v mobilno aplikacijo, saj so mobilne aplikacije v času pisanja te diplomske naloge zelo razširjene in med uporabniki zaželene. Če je pomočnik realiziran v obliki mobilne aplikacije, lahko uporabnik z uporabo mobilnega telefona požene aplikacijo, ta pa mu svetuje turistično destinacijo, do katere bo porabil najmanj časa in ga opozori, če ima na svojem koledarju v časovnem intervalu, ki ga potrebuje za dosego in ogled destinaceije, že vnesene opravke. Prav tako pomočnik uporabnika obvesti o trenutnih vremenskih razmerah destinacije in mu svetuje primerno oblačilo. Pomočnik prikaže tudi zemljevid

in omogoči uporabniku povezavo z družbenim omrežjem Twitter.

V zadnjem delu diplomske naloge so predstavljene sklepne ugotovitve, ki so nastale ob raziskovanju in predstavljenih področij.

Poglavje 2

Raziskovalna področja in uporabljene tehnologije

2.1 Luščenje podatkov iz delno strukturiranih spletnih virov

Ker je bil namen obdelave problematika turizma na območju Slovenije, sem posledično preiskala slovenske spletne vire, ki omogočajo pregled najzanimivejših turističnih destinacij po Sloveniji, ki pri omenjenem inteligentnem pomočniku služijo kot baza turističnih destinacij. Odločila sem se, da podatke o turističnih destinacijah izluščim z uradnega slovenskega turističnega informacijskega portala[1], kjer se promovira znamka Slovenije I FEEL SloveNIA.

Ker je uporabljena tehnika luščenja podatkov dinamična, ni zaželeno, da je teh veliko, saj upočasnjujejo delovanje programa. Tako sem se odločila, da izberem le nekaj najbolj prepoznavnih turističnih destinacij po Slovaniji. Želene destinacije sem pridobila z omejene strani v zavihku Destinacije, Vredno ogleda![2], kjer je bilo, v času pisanja te diplomske naloge, zbranih devetinpetdeset destinacij.

2.1.1 jsoup

Podatke o želenih destinacijah sem izluščila s pomočjo prosto dostopne knjižnice jsoup[3]. jsoup je Java knjižnica za delo s HTML dokumenti. jsoup deluje po WHATWG[1] HTML5 specifikaciji in omogoča razčlenitev HTML dokumentov v DOM strukturo, kot to počnejo sodobni brskalniki. Knjižnica je torej omogočila luščenje potrebnih podatkov na osnovi HTML zapisa spletne strani.

Jsoup knjižnica sicer omogoča strganje in razčlenjevanje HTML dokumenta iz URL datoteke ali niza, iskanje in pridobivanje podatkov z uporabo DOM prečkanja ali z uporabo CSS selektorjev, manipulacijo HTML elementov, atributov in besedila, čiščenje HTML-jev za preprečevanje XSS napadov.

2.1.2 Podatki o destinaciji

Poleg izluščenih imen turističnih destinacij sem za nadaljnjo izvedbo potrebovala še podatek o naslovu destinacije oziroma njenih geografskih koordinatah. Te sem ravno tako izluščila z uporabo jsoup knjižnice. Podatke o geografskih koordinatah sem uporabila pri izračunu razdalje od trenutne lokacije uporabnika do posamezne turistične destinacije. Uporabno bi bilo izluščiti tudi natančnejše podatke o posamezni turistični destinaciji, kot so na primer opis, naslov, kontakt in naslov spletne strani turistične destinacie, vendar je bila urejenost teh podatkov v času pisanja diplomske naloge pomanjkljiva in neskladna, zato je bilo luščenje teh podatkov neuporabno.

2.2 OAuth

OAuth[4] je odprt standard, protokol, ki omogoča avtentifikacijo oziroma preverjanje uporabnikove prisotnosti in izdajo dovoljenj, ki omogočajo deljenje uporabnikovih zasebnih virov. V primeru omenjenega inteligentnega pomočnika so to uporabnikovi viri podatkov o trenutni lokaciji, koledarju in

[1] http://www.whatwg.org/

Twitter računu. Podatke, ki so shranjeni na določenih spletnih virih, tako lahko nek drug vir (npr. spletna stran, aplikacija, program) pridobi, ne da bi uporabnik moral deliti svoje poverilnice[2], ki običajno oskrbujejo uporabniško ime in geslo. Namesto tega se pridobijo tako imenovani žetoni[3]. Vsak žeton omogoča dostop do določene spletne strani (npr. uporabnikov Twitter profil) za specifične vire (npr. le branje Twitter sporočil) in za določeno obdobje (npr. za obdobje enega tedna). OAuth tako omogoča uporabniku odobritev dostopa do svojih podatkov, ki so shranjeni pri določenih storitvah na neki tuji strani.

2.2.1 OAuth 2.0

OAuth 2.0 je naslednja različica razvoja protokola OAuth in ni združljiva s predhodno različico OAuth 1.0. OAuth 2.0 je osredotočen na odjemalca in razvijalca ter zagotavlja preprost način uporabe. Zagotavlja posebne tokove za izdajo dovoljenj za spletne aplikacije, namizne aplikacije, mobilne telefone in t.i. „living room“naprave, ki so v precejšnjem porastu.

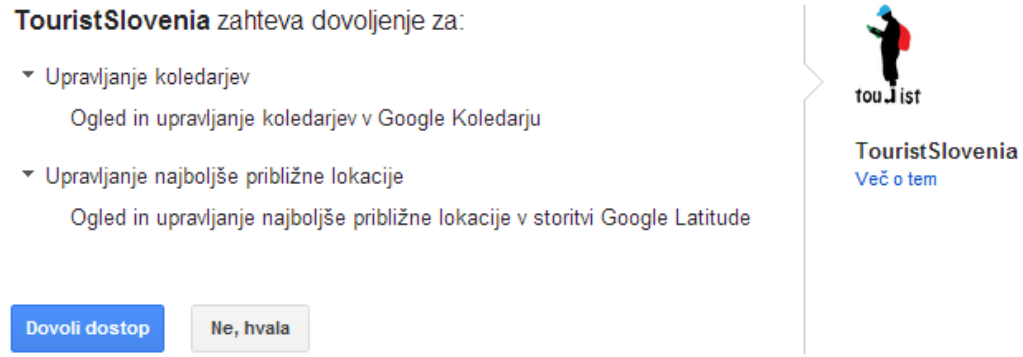
Od leta 2011 uporablja za podporo programskih vmesnikov OAuth 2.0 tudi Google.

Slika 2.1 prikazuje potrebno avtentifikacijo za dostop do Googlovih storitev, kjer je uporabljen OAuth protokol.

2.3 Googlovi programski vmesniki API

Ker so Googlove storitve po večini brezplačne in imajo dobro napisano dokumentacijo, jih je bilo smiselno uporabiti pri realizaciji inteligentnega pomočnika. Omembe vredno pa je tudi dejstvo, da so Googlove storitve med uporabniki spletnih tehnologij zelo pogosto uporabljene in omogočajo programerju širok nabor knjižnic in drugih tehnologij za uporabo programskih vmesnikov.

[2]angl. credentials
[3]angl. tokens



Slika 2.1: Pridobitev uporabnikovega dovoljenja za poseg aplikacije do zasebnih virov

Hkrati pa so te tehnologije omogočile kar nekaj rešitev za zastavljeno idejo o inteligentnem pomočniku

Za pridobitev uporabnikove lokacije sem uporabila Google Latitude API[5], za izračun razdalje, potrebnega časa za vožnjo do turistične destinacije, pretvorbo geografskih koordinat in prikaz zemljevida pa Google Maps API[6]. Vpogled v uporabnikov Google koledar sem realizirala z Google Calendar API[7]

Da razvijalec lahko koristi vse možnosti, ki jih ponujajo Googlovi programski vmesniki, mora svoj projekt registrirati v t.i. Googlovi API konzoli[4]. Še prej pa mora razvijalec odpreti svoj Google račun. Z registracijo projekta v Googlovi konzoli razvijalec prejme unikaten API ključ[5], uporabnikov ID[6], skrivne kode uporabnika[7] in preusmeritveni URI[8], le-te nadalje lahko uporabi pri izdaji dovoljenj in avtentifikaciji ter manipulaciji s programskimi vmesniki.

Obstajata dva načina prek katerih se lahko razvijalci sklicujejo na Goo-

[4]https://code.google.com/apis/console/
[5]angl. API key
[6]angl. client ID
[7]angl. consumer secret
[8]angl. redirect URI

glove programske vmesnike. Razvijalec to lahko naredi neposredno z uporabo REST sloga ali pa z uporabo knjižnic. Razvijalec tako lahko sam presodi, katera opcija je primernejša za izdelavo njegovega projekta.

2.3.1 Google Latitude API

Ta Googlov programski vmesnik omogoča določanje uporabnikove trenutne lokacije. S pomočjo sistema GPS za globalno določanje lege tako vmesnik vrne podatke o trenutnih geografskih koordinatah, torej zemljepisna širino in dolžino. Uporaba Google Latitude API je pri inteligentnem pomočniku realizirana s pomočjo HTTP GET zahtevka, ki kot odgovor vrne JSON strukturo, ki vsebuje omenjene podatke.

Pridobitev uporabnikovih podatkov je mogoča le z uporabnikovo potrditvijo preverjanja pristnosti in dovoljenja aplikaciji, da dostopa do njegovega Google Latitude profila. Za uporabo Google Latitude programskega vmesnika je torej potrebna predhodna uporaba OAuth standarda oziroma avtentifikacije s pravilno nastavljenimi parametri (API ključ, uporabnikov ID, skrivna koda uporabnika, preusmeritveni URI).

2.3.2 Google Maps API

Google Maps programski vmesnik omogoča razvijalcu širok spekter možnosti za manipulacijo in delo z Googlovimi zemljevidi. V sklopu Google Maps spletnih storitev[9] najdeno tudi Google Directions API[8], Google Geocoding API[9] in Google Static Maps API[10], ki so bili uporabljeni pri omenjenem inteligentnem pomočniku.

Google Directions API

Za realizacijo tega spletnega pomočnika je bila potrebna možnost izračuna časa za dosego razdalje med različnimi lokacijami, konkretneje izračun časa potovanja z avtom od uporabnikove trenutne lokacije, podane v geografskih

[9] angl. Google Maps API Web Services

koordinatah, do izluščenih turističnih destinacij, katerih lokacije so ravno tako podane v obliki geografskih koordinat. To je mogoče realizirati s pomočjo Google Directions programskega vmesnika, ki izračunava smeri med lokacijami. Za pridobitev teh podatkov ni potrebna predhodna avtentifikacija ali uporaba unikatnega ključa. Podatke je mogoče pridobiti z uporabo HTTP GET zahteve. S pomočjo Google Directions uporabniškega vmesnika se lahko določi izvor, cilj in vmesne točke, bodisi v obliki besedilnih nizov (npr. Ljubljana, Slovenija) ali pa z uporabo koordinat zemljepisne širine in dolžine (npr. 46.052,14.4998).

Izračunani rezultati o smereh se lahko vrnejo v obliki XML dokumenta ali pa kot JSON dokument.

Google Geocoding API

Google Geocoding programski vmesnik omogoča geokodiranje in obraten postopek geokodiranja[10]. Geokodiranje je postopek pretvarjanja naslovov (npr. Cesta Borisa Kidriča 13, Zagorje ob Savi, Slovenija) v geografske koordinate kot zemljepisno širino in dolžino (npr. 46.1314485,14.9969802). Postopek, ki je obraten geokodiranju, torej pretvarja geografske koordinate v naslove. Za potrebe tega inteligentnega pomočnika sem potrebovala pretvorbo geografskih koordinat v naslove.

Izračunani rezultati o smereh se lahko vrnejo v obliki XML dokumenta ali pa JSON dokumenta.

Za pretvorbo koordinat ni potrebna nikakršna avtentifikacija niti uporaba Google API ključa.

Google Static Maps API

Google Static Maps API omogoča prikaz Googlvega zemljevida v obliki slike, ne da bi zahteval uporabo JavaScript-a ali kakršno koli dinamično nalaganje strani. Ta Googlova storitev je statična in ustvarja zemljevid na podlagi

[10] angl. Reverse Geocoding

ustreznih URL parametrov, poslanih prek standardne HTTP GET zahteve, ki vrne zemljevid v obliki slike.

Predhodna avtentifikacija ali uporaba Google API ključa pri uporabi Google Static Maps API ni potrebna.

2.3.3 Google Calendar API

Google Calendar API omogoča razvoj aplikacije odjemalca, ki lahko ustvari nove dogodke, ureja ali briše obstoječe dogodke in išče ali prebira dogodke uporabnika Googlovega koledarja.

Ker za potrebe tega inteligentnega pomočnika ni zadoščalo pridobivanje podatkov prek HTTP zahtev po REST standardu, sem uporabila Google Calendar knjižnico za programski jezik Java.

Google Calendar API knjižnica ima veliko metod za manipulacijo s podatki o uporabnikovem koledarju. Pri izdelavi omenjenega inteligentnega pomočnika je bila uporabna poizvedba o uporabnikovem prostem času oziroma informacija o tem, ali je uporabnik v določenem časovnem intervalu zaseden.

Za realizacije te poizvedbe sem uporabila naslednje metode in objekte:

```
//Inicializacija poizvedbe FreeBusyRequest
FreeBusyRequest request = new FreeBusyRequest()
//nastavitev spodnje meje časovnega intervala poizvedbe
request.setTimeMin()
//nastavitev zgornje meje časovnega intervala poizvedbe
request.setTimeMax()
//nastavitev časovnega pasa koledarja
request.setTimeZone()
//nastavi postavke in uporabniško ime
request.setItems(Arrays.asList(new FreeBusyRequestItem()
.setId(userName+"@gmail.com")));
//izvršitev poizvedbe
```

```
FreeBusyResponse busyTimes = service.freebusy()
.query(request).execute();
```

Za poizvedbo o uporabnikovem prostem času je uporabnik moral potrditi avtentifikacijo in dovoliti pametnemu pomočniku, da razpolaga s podatki iz njegovega Google koledarja. Uporaba Google Calendar programskega vmesnika tako zahteva ustrezno implementacijo API ključa, uporabnikovega ID-ja, skrivne kode uporabnika in preusmeritvenega URI-ja v povezavi z OAuth standardom.

2.4 Weather API

Spletna stran in storitev Weather Underground[11] se ukvarja z vremenskimi napovedmi. Za razvijalce imajo omogočen tudi programski vmesnik. Programski vmesnik Weather API omogoča pridobivanje podatkov o vremenskih razmerah za določeno območje.

Zahteve uporabniškega vmesnika so narejene prek HTTP GET zahtev. Rezultat zahtev so podatkovni elementi v obliki JSON ali XML datoteke.

Tako kot večina uporabniških vmesnikov tudi Weather API potrebuje za pravilno delovanje API ključ, ki ga razvijalec prejme, ko na uradni strani opravi registracijo.

2.5 Twitter API

Twitter programski vmesnik[11] omogoča dostop in upravljanje uporabnikovega Twitter računa.

Za uporabo Twitter programskega vmesnika je potrebna registracija programa, aplikacije ali spletne strani na uradni strani Twitter, ki je namenjena razvijalcem. Tako lahko razvijalec prejme ustrezne ključe, žetone in skrivne kode stranke za delo s Twitter uporabniškim vmesnikom oziroma Twitter računi.

[11] http://www.wunderground.com/

Za uporabo Twitter programskega vmesnika je potrebna uporaba OAuth standarda.

2.5.1 Twitter4J

Twitter4J[12] je neuradna Java knjižnica za delo s Twitter programskim vmesnikom. Twitter4J vključuje programsko opremo JSON za razčlenitev JSON datoteke, pridobljene kot odgovor od Twitter programskega vmesnika. Na uradni spletni strani knjižnice je veliko primerov implementacije kode in dobro opisana dokumentacija knjižnice.

Z uporabo te knjižnice sem uporabniku omogočila deljenje podatkov o načrtovanem izletu s Twitter sledilci.

Poglavje 3

Integracija podatkov v problemski domeni turizma

3.1 Programsko okolje

Za realizacijo omenjene problematike turizma in razvoj pametnega pomočnika je bil uporabljen programski jezik Java. Programski jezik Java je med programerji zelo uporabljen in tudi meni poznan. Zanj je napisanih veliko knjižnic, ki omogočajo lažjo, hitrejšo in večkrat učinkovitejšo realizacijo problemov. Dobro za programerja pa je tudi dejstvo, da je na svetovnem spletu dostopnih veliko virov in dokumentacij v povezavi s pisanjem javanskih programov, kar programerju lajša delo. Inteligentni pomočnik je torej realiziran kot program, napisan v programskem jeziku Java.

Pisanje kode, razhroščevanje, povezovanje knjižnic in urejanje programske kode je bilo izvedeno v programskem razvojnem orodju Eclipse, ki je namenjeno razvoju integriranih razvojnih okolij (IDE[1]), katera lahko uporabimo za izdelavo različnih aplikacij in programov.

Ker pa so v poglavju 4 omenjene tudi nadgradnje inteligentnega pomočnika v mobilno aplikacijo, je na tem mestu omembe vredno tudi dejstvo, da so mobilne aplikacije, ki tečejo na mobilnem operacijskem sistemu Andrioid,

[1] angl. Integrated development environment

napisane v času pisanja diplomske naloge, zelo razširjene. Aplikacije, ki so narejene za Android operacijske sisteme, pa so napisane v programskem jeziku Java, zato je možna transformacija inteligentnega pomočnika preprostejša in zahteva malenkostne spremembe pri že napisani logiki programa. Na izbiro programskega jezika Java je posledično vplivalo tudi zavedanje se zgoraj omenjenega dejstva in želja po nadaljnji nadgradnji inteligentnega pomočnika.

3.2 Razvoj ideje pametnega pomočnika

Razvoj inteligentnega pomočnika za problemsko domeno turizma, ki temelji na heterogenih podatkih iz spletnih virov s področja turizma in uporabniku samodejno priporoča zanimive turistične destinacije glede na njegovo trenutno lokacijo in časovno omejenost, je potekal v več fazah. Najprej je bilo treba idejo razdelati in ugotoviti, kateri podatki so za obdelavo in izvedbo potrebni, nato pa na podlagi problematike izbrati primerna programska orodja, ki omogočajo realizacijo pridobivanja heterogenih podatkov iz spletnih virov s področja turizma. Pridobiti in raziskati je bilo treba tudi tehnologije, ki omogočajo ostale funkcionalnosti inteligentnega pomočnika.

Najprej je bilo treba izbrati primerne turistične destinacije po Sloveniji. Na tem mestu se je bilo treba odločiti, kako pridobiti te podatke, in ali jih je smiselno hraniti v obliki fizične podatkovne baze ali pa jih pridobivati dinamično z metodo luščenja podatkov. Ker število turističnih destinacij ni veliko in ker je zmerno hitra internetna povezava na voljo večini potencialnih uporabnikov omenjenega inteligentnega pomočnika, je bilo smiselno izbrati uporabo tehnologije luščenja podatkov, saj le-ta podatke izlušči v doglednem času.

Nato je bilo treba določiti, katere funkcionalnosti in tehnologije bodo poleg luščenja podatkov turističnih destinacij v inteligentnem pomočniku vsebovane ter, kje in kako jih pridobiti in uporabiti. Razvoj inteligentnega pomočnika je zahteval uporabo funkcionalnosti, kot so pridobivanje uporabni-

kove trenutne lokacije, uporabnikovih podatkov iz Google koledarja, izračun razdalje poti med trenutno lokacijo in turističnimi destinacijami, pridobivanje vremenskih razmer za izbrano turistično destinacijo in integracija z družabnim omrežjem Twitter.

Razvoj inteligentnega pomočnika vsebuje t.i. vhodne kanale, ki omogočajo pridobivanje potrebnih podatkov, ki jih program dobi kot vhodne podatke. Ti so predstavljeni v poglavju 3.2.1. Nadalje inteligentni pomočnik izvede računsko in ostalo logiko nad pridobljenimi podatki. Po uspešni interakciji z uporabnikom in pravilno pridobljenimi vhodnimi kanali program nato uporabniku poda t.i. izhodne kanale, ki so predstavljeni v poglavju 3.2.2.

Diagram na sliki 4.2 ponazarja realizacijo ideje inteligentnega pomočnika z vpeljavo vhodnih in izhodnih kanalov v delovanje inteligentnega pomočnika.

Ko so bile določene vse potrebne tehnologije in metode pridobivanja leteh, jih je bilo treba povezati v sam program. Nato je sledil razvoj programskega dela inteligentnega pomočnika. Še pred pisanjem programskega dela je bilo seveda treba določiti vrsto programa in programskega jezika. Le-ta je v primeru omenjenega inteligentnega pomočnika programski jezik Java.

3.2.1 Vhodni kanali inteligentnega pomočnika

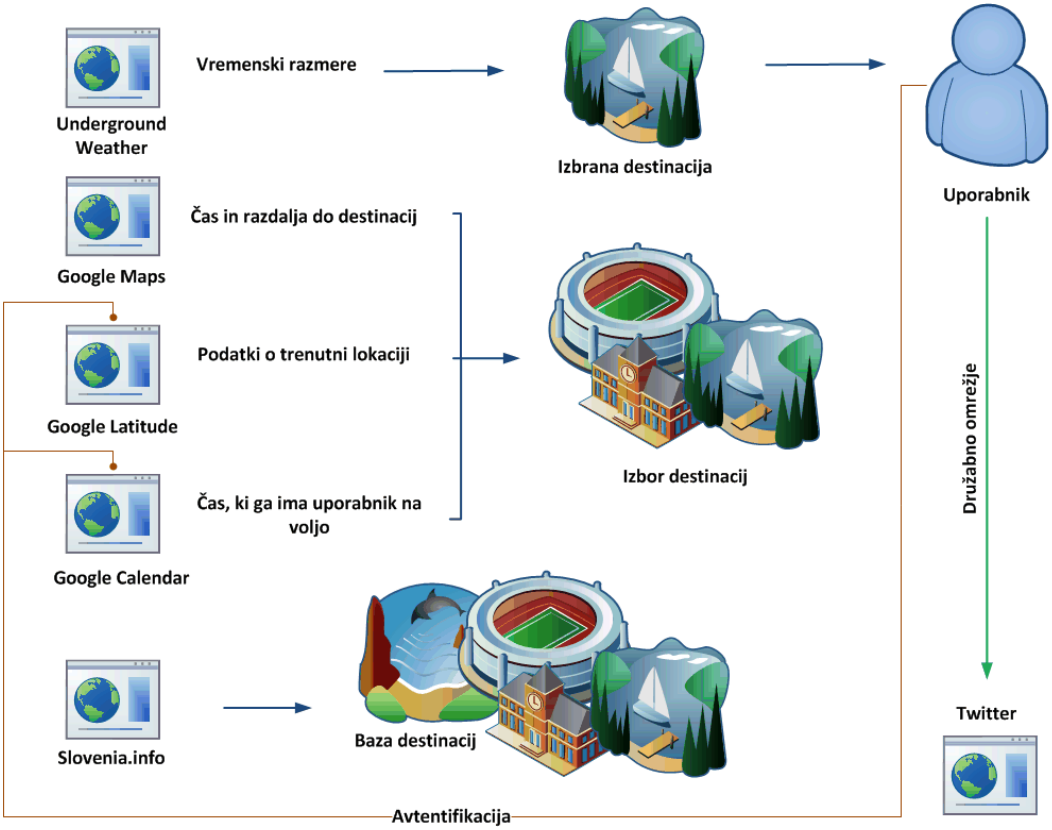
Vhodni kanali so podatki, ki jih program prejme prek različnih tehnologij oziroma virov, kot so uporabniški vmesniki in knjižnice. Le-ti programu priskrbijo podatke, ki so potrebni za nadaljnjo manipulacijo in se pridobivajo ob zagonu in nadaljnjem izvajanju programa.

Za vhodne kanale inteligentni pomočnik uporablja naslednje vire:

- uradni slovenski turistični informacijski portal Slovenia.info,
- trenutno lokacijo uporabnika prepoznano z Google Latitude API programskim vmesnikom, ki pridobi uporabnikove geografske koordinate,
- izračun razdalje poti in potrebnega časa za dosego destinacije s pomočjo programskega vmesnika Google Maps API,

- delovni urnik uporabnika zabeležen na Google Calendar, pridobljen s programskim vmesnikom Google Calendar API,
- trenutne vremenske razmere turističnih destinacij pridobljene, s pomočjo programskega vmesnika Weather Underground Weather API.

Inteligentni pomočnik uporabniku s pomočjo zgoraj navedenih pridobljenih podatkov predlaga zanj najbolj primerno turistično destinacijo.



Slika 3.1: Diagram realizirane ideje

3.2.2 Izhodni kanali inteligentnega pomočnika

Izhodni kanali predstavljajo podatke in možnosti, ki jih uporabnik prejme, ko se program pravilno izvede. Program omogoča interakcijo z različnimi

tehnologijami in viri.

Za izhodne kanale inteligentni pomočnik uporabila naslednje vire:

- izbor turistične destinacije,
- načrt poti s pomočjo Google Maps,
- objavo o dogodku na družbenem omrežju Twitter.

Pomočnik uporabniku pokaže možnosti poti, ki jo načrtuje s pomočjo Googlovih zemljevidov Google Static Maps. Uporabniku ponudi tudi možnost deljenja informacije o izbrani lokaciji in izletu z njegovimi sledilci na Twitter družabnem omrežju.

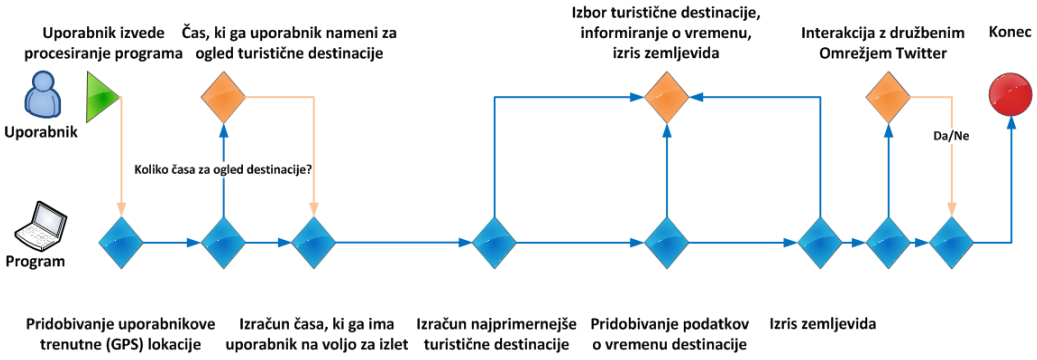
3.3 Delovanje inteligentnega pomočnika

Omenjeni inteligentni pomočnik deluje kot program, ki najprej s spletne strani *http://www.slovenia.info* v zavihku Destinacije, nadaljnji izbor Vredno ogleda! izlušči turistične destinacije, ki služijo kot baza vseh turističnih destinacij, ki jih program nadalje obdeluje. Za vsako destinacijo program izlušči tudi podatke o imenu in točni legi kraja in geografski širini in dolžini. Geografska širina in dolžina služita za določanje točnih razdalj med različnimi lokacijami. Od uporabnika program nadalje pridobi trenutno uporabnikovo lokacijo. Le-to pridobi prek programskega vmesnika Google Latitude API, ki uporablja sistem za globalno določanje lege (GPS). S pomočjo pridobljenih geografskih koordinat turističnih destinacij in uporabnika pomočnik z uporabo tehnologije Google Maps API izračuna razdalje med danimi koordinatami in čas, ki je potreben za dosego teh razdalj, če uporabnik potuje z avtom. Pomočnik nato izbere destinacije, za katere bo uporabnik, če potuje z avtom, porabil najmanj časa. Uporabnika tudi vpraša, koliko časa v urah želi nameniti ogledu turistične destinacije in ta čas upošteva poleg časa, potrebnega za vožnjo z avtom v obe smeri, kot potreben čas za celoten izlet. Program s pomočjo programskega vmesnika Google Clendar preveri, če ima

uporabnik za dan časovni interval prost koledar oziroma na koledarju nima zabeleženih opravkov. Če je uporabnik zaseden, ga pomočnik na to opozori, vendar mu kljub temu predlaga izbrano destinacijo in mu dopušča lastno izbiro odločitve o izletu. Če pa uporabnik na koledarju za dan časovni interval nima zabeleženih opravkov, mu zgolj predlaga, kam naj odpotuje, ne da bi uporabnika obveščal o dogodkih, ki jih ima zabeležene na koledarju.

Za predlagano destinacijo nato program preveri vremenske razmere, ki jih pridobi prek programskega vmesnika Wounderground Weather API. Uporabnika glede na vreme in temperaturo opozori, kako naj bo oblečen in da naj ne pozabi dežnika v primeru padavin. Na koncu program izpiše povezavo na sliko zemljevida, kjer sta kot točki označeni začetna lokacija izleta in končna destinacija ter ponudi uporabniku možnost objave informacije o izletu na družabnem omrežju Twitter.

Diagram zaporedja na sliki 3.2 prikazuje ključne segmente izvajanja programa inteligentnega pomočnika v odnosu do uporabnika.



Slika 3.2: Diagram zaporedja delovanja inteligentnega pomočnika

Da se program zažene in deluje pravilno uporabnik potrebuje:

- vzpostavljeno internetno povezava na računalniku, ki ga uporablja,
- Google račun,
- Googlove storitve Google Calendar in Google Latitude,

- mobilno napravo, ki ima vgrajen sistem za določanje trenutne pozicije (GPS), s katero preko storitve Google Latitude zabeležiti trenutno lokacijo.

Če so zagotovljeni vsi pogoji za pravilno delovanje programa, se program izvede in konča, sicer se lahko ne zažene ali pa predčasno konča.

3.3.1 Faze delovanja inteligentnega pomočnika

Namen omenjenega inteligentnega pomočnika je omogočanje planiranja izleta na kratek rok. Uporabnik z uporabo inteligentnega pomočnika lahko planira nekaj urni izlet po Sloveniji. Da je izguba časa za uporabnika čim manjša, je izbor turistične destinacije omejen na uporabnikovo trenutno lego in uporabo avtomobila kot prevoznega sredstva. Za boljšo organizacijo časa se inteligentni pomočnik orientira tudi glede na uporabnikov Google koledar in prilagodi izbiro tudi glede na uporabnikove že vnesene opravke.

Če uporabnik zažene program, sproži procesiranje programa. Program tako v prvi fazi po zagonu pridobi uporabnikovo trenutno lokacijo. Nato ponudi uporabniku izbiro časovne omejitve samega ogleda turistične destinacije oziroma količino časa, ki ga uporabnik želi porabiti, ko bo že prispel na turistično destinacijo. Ta čas je mišljen kot čas, ki je potreben npr. za ogled, športno aktivnost, obrok. Program nato preračuna, katera destinacija uporabniku glede na omejitve (trenutna lokacija, zasedenost koledarja, čas za ogled destinacije) najbolj ustreza. Program v naslednjih fazah pridobi podatke o vremenskem stanju izbrane turistične destinacije in podatke o poti do turistične destinacije. Ko program izbere destinacijo in pridobi podatke o vremenu ter pridobi zemljevid poti od trenutne lokacije uporabnika do turistične destinacije, uporabniku prikaže izbor turistične destinacije, podatke o vremenskem stanju in izris zemljevida. Program glede na podatke o vremenskih razmerah izbrane turistične destinacije uporabniku predlaga tudi primerno obleko. Ker so v času pisanja diplomske naloge družabna omrežja zelo popularna, program upošteva tudi uporabnikovo interakcijo z družabnim

omrežjem Twitter. Program tako v zadnji fazi procesiranja uporabniku ponudi možnost objave informacije o izletu na Twitter družabno omrežje. Ko uporabnik ponujeno možnost objave potrdi ali zavrne, se program zaključi.

3.3.2 Konkreten primer uporabe inteligentnega pomočnika

Uporabnik želi nekaj prostih ur izkoristiti za kratek izlet po Sloveniji. V ta namen uporabi inteligentnega pomočnika.

Uporabnik se trenutno nahaja v bližini Celja, natančneje v Žalcu. Program pridobi uporabnikovo trenutno geografsko širino (46.251673) in dolžino (15.167421). Program tako izračuna, da je od trenutne lokacije najkrajša razdalja do Celjskega gradu. Ko uporabnik izvede procesiranje programa, je ura 15:00. Program uporabnika vpraša, koliko ur želi porabiti za ogled in druge aktivnosti, ko se bo nahajal na turistični destinaciji. Uporabnik poda, da bo ogledu turistične destinacije namenil 2 uri. Program izračuna, da uporabnik za dosego destinacije z avtom potrebuje 18 minut. Ker je uporabnik 2 uri namenil ogledu destinacije in traja vožnja z avtom v obe smeri (od trenutne lokacije do turistične destinacije in nazaj) 36 minut, program preveri, če ima uporabnik v nadaljnjih 2 urah in 36 minutah prost Google koledar. Možna sta slednje dva izida glede na podatke v uporabnikovem Google koledarju:

a) uporabnik v danem časovnem intervalu (2 uri in 36 minut) nima zabeleženih dogodkov,

b) uporabnik ima v danem časovnem intervalu (2 uri in 36 minut) že zabeležen dogodek ali več dogodkov.

V primeru a) program uporabniku predlaga izlet na Celjski grad. V primeru b) pa program poleg predlaganega izleta na Celjski grad uporabnika opozori, da ima na koledarju že vnesene dogodke in te dogodke izpiše. Če uporabnik, kljub temu da ima že zavedene opravke, želi izvesti izlet, se program nadaljuje, sicer se konča.

Program nato informira uporabnika o zanj najbolj primerni turistični destinaciji (Celjski grad) in o trajanju izleta (2 uri in 36 minut). Ker program za izbrano turistično destinacijo preveri tudi vremenske pogoje, uporabnika obvesti o vremenu. Glede na dano temperaturo destinacije program uporabniku svetuje oblačila navedena v tabeli 3.1. Če so napovedane padavine, program uporabnika opomni na uporabo dežnika.

°C	dan	tip oblačil
< 0	hladen zimski dan	topla zimska oblačila
0-10	hladen dan	topla oblačila
10-15	zmerno hladen dan	jesensko-zimska oblačila
15-20	zmerno toplo	pomladansko-jesenska oblačila
20-25	toplo	poletna oblačila
$25 <$	vroče	poletna oblačila

Tabela 3.1: Oblačila glede na temperaturo

Poleg informacij o izbrani turistični destinaciji program izpiše tudi povezavo na zemljevid. Na zemljevidu sta označeni začetna lokacija (Žalec), izbrana turistična destinacija (Celjski grad) in izrisana pot med njima.

Program ponudi uporabniku možnost deljenja informacije o izletu s Twitter sledilci. V primeru, da uporabnik izbere možnost objave na njegov Twitter profil, se na njegov Twitter profil objavi: *Grem na izlet v/na Celjski grad, Celje. # trip # Slovenja.*

3.4 Uporabljene tehnologije

3.4.1 jsoup

Javanska knjižnica jsoup, ki je omenjena v poglavju 2.1.1, je omogočila luščenje podatkov iz uradnega slovenskega turističnega informacijskega portala Slovenia.info. Za luščenje podatkov iz spletnega portala bi se lahko

uporabile različne vrste tehnologij, vendar se je glede na dostop in strukturo podatkov omenjenega portala uporaba jsoup knjižnice izkazala za preprostejšo in bolj učinkovito, kot je na primer uporaba tehnologije luščenja podatkov z uporabe regularnih izrazov. Rezultat pridobljenih podatkov z uporabo knjižnice jsoup je zapisan kot JSON datoteka. Ker ima JSON datoteka svojevrstno obliko zapisa, je bilo treba te podatke pridobiti po JSON standardu in jih pravilno izpisati.

Za učinkovito uporabo knjižnice jsoup je bilo treba najprej preveriti strukturo HTML dokumenta uradnega slovenskega turističnega informacijskega portala Slovenia.info. Ugotoviti je bilo treba, kje se želeni podatki v HTML dokumentu nahajajo in nato poiskati primerne metode, ki jih ponuja jsoup knjižnica.

Slika 3.3 predstavlja napisani del kode programa, ki uporablja jsoup knjižnico.

```
public JSONArray destinationLinks(String link) throws JSONException {
    String html = c.get(link);
    JSONObject zadetek;
    JSONArray rezultati = new JSONArray();
    Document doc = Jsoup.parse(html);
    System.out.println("Pridobivanje povezav do znamenitosti.");
    for (Element div : doc.select("div.wpContent")) {
        for (Element a : div.select("a")) {
            zadetek = new JSONObject();
            zadetek.put("opis", a.text());
            zadetek.put("povezava", a.attr("abs:href"));
            rezultati.put(zadetek);
        }
    }
    System.out.println("Pridobljenih " + rezultati.length()
                       + " povezav do znamenitosti.");
    return rezultati;
}
```

Slika 3.3: Pridobivanje podatkov z uporabo knjižnjice jsoup

S slike 3.3 je razvidna uporaba metode *Jsoup.parse()*, ki razčleni niz, ki predstavlja pridobljeno HTML povezavo. Klic metode *doc.select("div. wpContent")* v zanki omogoča sprehod po HTML elementu tipa *div* z imenom

wpContent. Znotraj tega elementa pa je bilo treba poiskati vsebino elementov, ki imajo oznako *a*, natančneje opis in povezava. Opis je pridobljen s klicem metode *a.text()*, ki pridobi tekstovni del, povezava pa je pridobljena prek metode *a.attr("abs:href")*, ki pridobi vsebino atributa *href*.

3.4.2 Google API

Google programski vmesniki so omogočili večji del funkcionalnosti, ki jih je zahtevala realizacija inteligentnega pomočnika. Omogočili so naslednje:

- določanje trenutne lokacije uporabnika (Google Latitude API),
- izračun časa potovanja z avtom od trenutne lokacije do turistične destinacije (Google Directions API),
- pretvarjanje naslovov navedenih z besedami v geografske koordinate (Google Geocoding API),
- prikaz statičnega zemljevida (Google Static Maps API),
- upravljanje uporabnikovega koledarja (Google Calendar API).

Googlovi programski vmesniki so predstavljeni v poglavju 2.3, njihove omejitve pa v poglavju 3.5.1.

Google Latitude API

Pravilna uporaba Google Latitude API-ja je zahtevala uporabo OAuth standarda, ki je omogočila uporabnikovo privoljenje za uporabo podatka o njegovi trenutni lokaciji. Sama poizvedba o uporabnikovi trenutni lokaciji pa je zahtevala uporabo REST standarda, ki ima sledečo formo: *https://www.googleapis.com/latitude/v1/currentLocation*. Ta zahteva vrne parametra geografska dolžina[2] in širina[3] za zgolj zadnjo znano uporabnikovo lokacijo. Ta dva parametra sta podana v datoteki tipa JSON.

[2]angl. longitude
[3]angl. latitude

Google Directions API

Z uporabo REST standarda za Google Directions programskei vmesnik, ki je v primeru inteligentnega pomočnika imel obliko *"http://maps.googleapis.com/maps/api/directions/json?origin="+ latitudeStart + ","+ longitudeStart +& "destination="+ latitudeEnd + ","+ longitudeEnd + "&sensor=false"*, sem izluščila podatek o času vožnje z avtom med določenima lokacijama. Naveden REST standard za Google Directions API je napisan v slogu uporabljene javanske kode. Podatke o lokacijah je pri Google Directions API-ju mogoče podati v geografskih koordinatah ali pa z imenom. V navedenem primeru so podane kot geografska dolžina *longitudeStart, longitudeEnd* in širina *latitudeStart, latitudeEnd*, ločeni z vejico. Parameter *origin* podaja, v formatu REST standarda, začetno lokacijo. Parameter *destination* pa končno lokacijo. Parameter *json* določa, da je odgovor na poizvedbo tipa REST datoteka JSON, parameter *sensor=false* pa, da je delovanje senzorja izklopljeno (uporabno pri napravah, ki uporabljajo GPS). Kot odgovor na poizvedbo je v JSON dokumentu navedenih več parametrov. Za primer delovanja programa inteligentnega pomočnika je bil potreben podatek, ki ga podaja parameter *duration*. Ta podaja čas, ki je potreben za vožnjo z avtom do želene destinacije. Pri Google Directions programskem vmesniku je privzeto nastavljen čas potovanja z avtom. Možne so tudi dodatne formacije Google Directions API REST standarda, ki določajo čas vožnje z avtobusom ali čas potovanja v primeru pešačenja.

Google Geocoding API

Uporaba Google Geocoding API REST standarda je v primeru inteligentnega pomočnika imela obliko *"http://maps.googleapis.com/maps/api/geocode/json?latlng="+ longitude + ","+ latitude + &sensor=false"*. Ta poizvedba je omogočila pretvorbo geografskih koordinat v naslov napisan z besedami. Naveden REST standard za Google Directions API je napisan v slogu uporabljene javanske kode. Parameter *longitude* v navedenem primeru določa geografsko dolžino, *latitude* pa geografsko širino. Parameter *json* določa, da

je odgovor na poizvedbo tipa REST datoteka JSON, parameter *sensor=false* pa, da je delovanje senzorja izklopljeno (pri GPS napravah). Kot odgovor na poizvedbo je v JSON dokumentu navedenih več parametrov. Za primer delovanja omenjenega pomočnika je bil potreben izpis naslova, ki ga podaja parameter *formatted_address*.

Google Static Maps API

Google Static Maps programski vmesnik je omogočil izris zemljevida določenih lokacij. V primeru inteligentnega pomočnika je imel sledečo obliko *http://maps.googleapis.com/maps/api/staticmap?zoom=10&size=1100x1100&maptype= roadmap&markers=color:blue%7Clabel:VI%7C"+start+"&markers=color:green %7Clabel:D%7C"+end+"&sensor=false"*. Ta poizvedba uporablja HTTP zahtevo po REST standardu. Naveden REST standard za Google Directions API je napisan v slogu uporabljene javanske kode. Rezultat zahteve je izris slike zemljevida v brskalniku. Parameter *zoom* določa stopnjo povečave zemljevida, ki je v navedenem primeru 10. Parameter *size* določa velikost slike v slikovnih pikah. Parameter *maptype* določa tip zemljevida, parametra *start* in *end* pa določata podane lokacije. Parameter *markers* je namenjen formaciji oznak na zemljevidu. Znotraj le-tega so podane lastnosti kot sta barva in ime oznake, ločeni z znakom navpična črta[4] (|). V zgornji formaciji REST standarda niz znakov *%7C* predstavlja navpično črto. Niz znakov *%7C* ustreza standardu URL kodiranja, ki je potreben za prenašanje tovrstnih znakov preko interneta.

Omejitve uporabe Google Static Maps so predstavljene v poglavju 3.5.1.

Google Calendar API

Pri manipulaciji podatkov z uporabnikovim Google koledarjem je bila potrebna uporaba Google Calendar programskega vmesnika. Google Calendar programski vmesnik ima sicer omogočene HTTP zahteve z uporabo REST standarda, vendar te za potrebe delovanja inteligentnega pomočnika niso bile

[4]angl. vertical bar

uporabne. Zato sem namesto REST standarda uporabila Google Calendar API programsko knjižnico za programski jezik Java. Uporabljene metode in objekti Google Calendar API knjižnice so predstavljeni v razdelku 2.3.3.

Slika 3.4 prikazuje uporabo metod in objektov Google Calendar API v programu inteligentnega pomočnika.

```
FreeBusyRequest request = new FreeBusyRequest();
request.setTimeMin(DateTime.parseRfc3339(tripStart + GMT1 + ":" + GMT2));
request.setTimeMax(DateTime.parseRfc3339(tripEnd + GMT1 + ":" + GMT2));
request.setTimeZone(GMT1 + GMT2);
request.setItems(Arrays.asList(new FreeBusyRequestItem().setId(userName+"@gmail.com")));

FreeBusyResponse busyTimes = service.freebusy().query(request).execute();
    for (Map.Entry<String, FreeBusyCalendar> busyCalendar : busyTimes.getCalendars().entrySet()){
        System.out.println("Ime koledarja: " + busyCalendar.getKey());
        System.out.println();

        if (busyCalendar.getValue().getBusy().isEmpty()){
            System.out.println("Na koledarju ni opravkov! Svetujem izlet z avtom v "+name+".");
        }

        else{
            for (TimePeriod busy : busyCalendar.getValue().getBusy()) {
                Date realTimeStart = df.parse(busy.getStart().toString());
                Date realTimeEnd = df.parse(busy.getEnd().toString());
                System.out.println("Na koledarju imš zabeležene opravke!");
                System.out.println("Zaseden si od " + realTimeStart+ " do " + realTimeEnd);

            }System.out.println("Sicer svetujem izlet v "+name+".");
        }
    }
```

Slika 3.4: Del kode uporabe javanske knjižnice Google Calendar API

3.4.3 Weather API

Programski vmesnik Weather API je omogočil pridobivanje potrebnih vremenskih podatkov za izbrano turistično destinacijo. Podrobneje je predstavljen v poglavju 2.4, njegove omejite pa so podane v poglavju 3.5.1.

Poizvedba o vremenskih pogojih izbrane turistične destinacije prek Weather Underground programskega vmesnika je bila narejena z uporabo GET zahtevka, ki ima sledeč format *http://api.wunderground.com/api/uporabnikovAPIključ/lastnostiPoizvedbe/nastavitve/q/poizvedba.format*. Format, uporabljen v pro-

gramu, ki ga predstavlja inteligentni pomočnik, pa je imel prilagojen format: *GET = "http://api.wunderground.com/api/uporabnikovAPIključ/conditions/ lang:SL/q/SI/"+parseAddress(address)+".json";*. Parameter *conditions* vrne podatke o trenutni temperaturi, vremenskih razmerah, vlažnosti, vetru, občutku temperature, zračnem tlaku in vidljivosti. Parameter *lang:SL* določa jezik pridobljenih podatkov, ki je v tem primeru slovenščina. Parameter *q* je del GET zahtevka in ga ni mogoče spreminjati ali izpustiti. Parameter *SI* definira geografsko območje poizvedbe, ki je v tem primeru Slovenija. Sledi parameter, ki predstavlja ime želene lokacije, kot je na primer. Ljubljana. V tem primeru je uporabljena metoda *parseAddress(address)*, ki ustrezno po standardu GET zahtevka prilagodi presledke med imenom mesta in nadomesti črke: č, š, ž s črkami: c, s, z. Zamenjava je potrebna, saj Weather programski vmesnik ne podpira šumnikov. Zadnji parameter predstavlja želen izpis formata, ki je v tem primeru tipa JSON.

3.4.4 Twitter API in Twitter4J

Za omogočanje pošiljanja uporabnikovega sporočila prek inteligentnega pomočnika je bilo treba uporabiti za ta namen narejen Twitter programski vmesnik Twitter API, ki je podrobneje predstavljen v poglavju 2.5.

Na sliki 3.5 je prikazan del kode inteligentnega pomočnika, ki omogoča interakcijo s Twitter programskim vmesnikom prek javanske knjižnice Twitter4J. S slike 3.5 je razvidna uporaba OAuth standarda, pridobivanje uporabnikovega dovoljenja in uporaba žetonov, samo pošiljanje sporočila na uporabnikov Twitter profil z uporabo metode *updateStatus()* in izpis statusa, ki ga prikaže program. Uporaba knjižnice Twitter4J je predstavljena v poglavju 2.5. Z uporabo te knjižnice je bilo uporabniku omogočeno deljenje podatkov o načrtovanem izletu s Twitter sledilci.

```
//post trip information on Twitter
public static void postToTwitter(String string) throws TwitterException, IOException{
    // The factory instance is re-useable and thread safe.
    Twitter twitter = new TwitterFactory().getInstance();
    twitter.setOAuthConsumer("Consumer Key", "Consumer Secret");
    RequestToken requestToken = twitter.getOAuthRequestToken();
    AccessToken accessToken = null;
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    while (null == accessToken) {
        System.out.println();
        System.out.println("Za dostop do Twitter računa prekopiraj povezavo v brskalnik:");
        System.out.println(requestToken.getAuthorizationURL());
        System.out.print("Prilepi pridobljeno kodo:");
        String pin = br.readLine();
        try{
            if(pin.length() > 0){
                accessToken = twitter.getOAuthAccessToken(requestToken, pin);
            }else{
                accessToken = twitter.getOAuthAccessToken();
         }
        }catch (TwitterException te) {
            if(401 == te.getStatusCode()){
                System.out.println("Unable to get the access token.");
        }else{
                te.printStackTrace();
            }
      }
    }
    //persist to the accessToken for future reference.
    storeAccessToken(twitter.verifyCredentials().getId(), accessToken);
    Status status = twitter.updateStatus("Grem na izlet v: "+string+". #trip #Slovenia");
    System.out.println("Vaše novo Twitter sporočilo je: " +status.getText()+ ".");
  }
```

Slika 3.5: Del kode inteligentnega pomočnika, ki uporablja Twiter4J

3.5 Težave

Večjih težav pri pisanju programskega dela diplomske naloge ni bilo.

Težave pri programiranju so se pojavljale le pri realizaciji programske kode na način, ki pravilno upošteva omejitve, ki jih zahtevajo posamezni programski vmesniki. Vsi obravnavani programski vmesniki imajo namreč za brezplačno možnost uporabe določene omejitve. Te omejitve so časovne in številčne omejitve dostopov in poizvedb ter omejitve dolžine URL znakov, ki jih izvaja programski uporabniški vmesnik za določeno storitev. Obstoj tovrstnih omejitev je potreben za preprečitev zlorab in vdorov v storitve. Če razvijalec prekorači omejitve, je lahko odgovor na poizvedbo nepravilen (prazen JSON ali XML dokument). Če se prekoračitve pojavijo večkrat,

je lahko dostop do programskega vmesnika blokiran. Spodaj so navedene omejitve posameznih storitev. Navedene omejitve so bile aktualne v času pisanja te diplomske naloge.

3.5.1 Omejitve programskih vmesnikov

Google API

Ker sem v programu za pridobivanje in obdelavo podatkov večinoma uporabljala Googlove programske vmesnike, sem prav tako želela uporabiti Googlov programski vmesnik za delitev informacije o izletu na spletnem družabnem omrežju, in sicer Google+ API. Težava pa je, da Google+ API trenutno omogoča zgolj branje podatkov, ne pa objavo dogodkov na zid uporabnika, saj ne omogoča nikakršnih POST funkcij, zato njegova uporaba ni bila koristna.

Pri uporabi Googlovih programskih vmesnikov je za pravilno delovanje programa treba upoštevati spodaj opisane zahteve.

- **Google Latitude API**

  Zahteva poizvedbe o trenutni lokaciji se izvrši in vrne pravilen odgovor le v primeru, da uporabnik uporablja Google Latitude aplikacijo. Ker je Google Latitude mobilna aplikacija, mora imeti uporabnik aplikacijo nameščeno na svojem pametnem telefonu ali kakšni drugi pametni napravi, ki uporablja sistem za določanje geografskih koordinat, in jo uporabljati. Omejitev uporabe tega programskega vmesnika je 1000000 zahtev na dan. Če razvijalec potrebuje zmogljivosti, ki presegajo to mejo poizvedb, lahko pošlje prošnjo za več poizvedb prek Google API konzole.

- **Google Latitude API**

  Pri uporabi Google Maps API kot brezplačne storitve mora biti končen izdelek (spletna stran, aplikacija, program) brezplačno in javno dostopen končnim uporabnikom.

  Pri uporabi Google Directions API in Google Geocoding API se pri brezplačni različici dnevno lahko izvede 2500 zahtevkov. Poleg tega je treba upoštevati omejitev dolžine URL-ja, ki je omejena na 2048 znakov. Pri uporabi brezplačne storitve Google Static Maps API je slika zemljevida omejena na resolucijo 640 x 640 slikovnih pik in dvakratno razmerje med velikostjo na zemljevidu.

  V tabeli 3.2 so prikazane razlike med brezplačno Google Maps API in plačljivo Google Maps for Business API različico uporabe Googlovih zemljevidov.

Oprema	Maps API	Maps API for Businss
Geocoding Web Service		
Zahtevkov na dan	2500	100 000
Directions Web Service		
Zahtevkov na dan	2500	100 000
Smernih točk na zahtevo	10	23
Static Maps API		
Maksimalna resolucija	640 x 640	2048 x 2048
Razmerje velikosti	2X	4X

Tabela 3.2: Google Maps API in Google Maps API for Businss

- **Google Calendar API**

  Google Calendar API ima mejo 10.000 poizvedb na dan. Če razvijalec potrebuje zmogljivosti, ki presegajo to mejo poizvedb, lahko pošlje prošnjo za več poizvedb prek Google API konzole.

Weather underground Weather API

Za uporabo programskega vmesnika, ki omogoča pridobitev podatkov o vremenskih razmerah določene lokacije, sem najprej želela uporabiti Googlov neuradni uporabniški vmesnik Google Weather API, vendar so ga tekom razvoja programa pametnega pomočnika ukinil. Tako sem se odločila za uporabo Weather Underground Weather API.

Ta uporabniški vmesnik uporablja dnevno in minutno stopnja omejitev števila prošenj, ki jih je mogoče izvršiti prek programskega vmesnika brezplačno. Za brezplačno uporabo je limit na dan 500, na minuto pa 10 poizvedb.

Spletni uporabniški vmesnik (WUI) omogoča funkcijo „dežne kaplje[5]“kot pomoč razvijalcem, ki presežejo limit. „Dežne kaplje“so krediti, ki se uporabljajo ob preseženem limitu uporabe in omogočajo, da se razvijalec izogne blokadi storitve Weather API. Ob registraciji vsak razvijalec prejme 3 „dežne kaplje“. Ob prekoračitvah omejitev se število„dežnih kapel“zmanjšuje. Nove „dežne kaplje“je mogoče pridobiti z implementacijo Weather API na način, da generira ali pa uporabniku omogoča povezavo na spletno stran *http://www.wunderground.com/*. Ko je zabeleženih 2500 povezav, razvijalec prejme eno „dežno kapljo“. Prav tako bo eno nova „dežna kaplja“dodana na razvijalčev račun vsako nedeljo od polnoči po newyorškem lokalnem času.

Twitter API

Omejitev[14] Twitter API REST dostopov je odvisna od uporabe avtorizacije. Če se uporablja avtorizacija z OAuth klici, je dovoljenih 350 prošenj na uro in se merijo v OAuth žetonih, ki se uporabljajo v zahtevi. Sicer je dovoljenih prošenj na uro 150. Omenjene omejitve se uporabljajo za metode, ki zahtevajo informacije s HTTP GET poizvedbo. Na splošno večina API metod, ki uporabljajo HTTP POST poizvedbe, ni omejenih.

Ukrepi, kot so objavljanje posodobitve statusa, pošiljanje neposrednih

[5]angl. raindrops

sporočil, sledenje[6] in ne-sledenje[7] Twitter uporabnikov, niso neposredno omejeni s programskim vmesnikom, temveč so predmet pravične omejitve Twitter uporabe[15].

[6]angl. following
[7]angl. unfollowing

Poglavje 4

Možna nadgradnja

Ideja inteligentnega pomočnika je bila realizirana v namizno aplikacijo, ki zavzema naslednje funkcionalnosti:

- interakcijo z uporabnikom,
- pridobivanje podatkov o geografskih legah turističnih destinacij in uporabnikove lokacije,
- upravljanje z uporabnikovim Google koledarjem,
- izračun razdalj med uporabnikovo lokacijo in turističnimi destinacijami,
- izračun časovnih intervalov in omejitev,
- izbor najprimernejše turistične destinacije,
- pridobivanje vremenskih podatkov o izbrani turistični destinaciji,
- nasvet primerne obleke glede na vremenske pogoje dane turistične destinacije,
- izris zemljevida,
- interakcijo z družabnim omrežjem Twitter.

Razširitve in druge možnosti so bile prav tako upoštevane, vendar določene presegajo okvire te diplomske naloge in zaradi časovne omejenosti niso bile realizirane. V razdelkih 4.1 in 4.2 sta opisani možni nadgradnji ideje in potrebne spremembe.

4.1 Mobilna aplikacija

Vse več je uporabnikov pametnih mobilnih naprav, ki so dandanes že nepogrešljiv del posameznikovega življenjskega stila. Pametni telefoni in druge pametne naprave podpirajo vrsto različnih aplikacij za organizacijo življenjskega sloga, zato bi bilo inteligentnega pomočnika smiselno implementirati v obliko mobilne aplikacije. Inteligentnega pomočnika bi bilo najpreprostejše preoblikovati v t.i. Android mobilno aplikacijo, saj je napisan v programskem jeziku Java, ki se uporablja za pisanje omenjenih aplikacij. Prav tako se za razvoj Android aplikacij zelo pogosto uporablja razvojno okolje Eclipse, ki je bilo uporabljeno tudi pri izdelavi programskega dela te diplomske naloge.

Možnih je več različnih načinov razvoja spletnega pomočnika v obliki mobilne aplikacije in so odvisni od različnih platform in mobilnih operacijskih sistemov, kot so iOS, Android, Windows Phone, BlackBerry, Symbian in drugi. V poglavju 4.1.1 je opisan odprto-kodni mobilni razvojni okvir, imenovan PhoneGap, ki omogoča razvoj mobilnih aplikacij za različne operacijske sisteme. Slika 4.1 prikazuje skico inteligentnega pomočnika v obliki mobilne aplikacije.

Vse Googlove tehnologije, opisane v poglavju 2, bi delovale tudi v obliki spletne aplikacije. Nekatere, kot je Google Maps, bi se lahko smiselno dopolnjevale tudi s inteligentnim pomočnikom. Uporabnik bi lahko tako s pomočjo pametne prenosne naprave uporabljal inteligentnega pomočnika kot aplikacijo, ki bi mu svetovala zanj najbolj ugoden izlet po Sloveniji.

4.1.1 PhoneGap

PhoneGap[16] je odprtokodno razvojno ogrodje, namenjeno razvoju mobilnih aplikacij. Omogoča razvoj in gradnjo aplikacij za mobilne naprave z uporabo spletnih tehnologij JavaScript, HTML5 in CSS3. Z uporabo PhoneGap ogrodja razvijalcu ni treba uporabiti točno določenega programskega jezika za konkreten operacijski sistem, kot je na primer Objective-C, ki se uporablja za razvoj aplikacij na iOS operacijskem sistemu. Razvijalcu tudi

(a) (b)

Slika 4.1: Mobilna aplikacija

ni potrebno nameščati različne SDK-je na različnih računalnikih. PhoneGap omogoča razvoj mobilnih aplikacij za naslednje operacijske sisteme: iOS, Android, webOS, Symbian, BlackBerry, Windows Phone in Bada.

Koda PhoneGap je dodatek k t.i. Apache fundaciji programske opreme (Apache Software Foundation) pod imenom Apache Cordova. Apache fundacija programske opreme omogoča prihodnji razvoj odprtokodnega razvojnega ogrodja PhoneGap. PhoneGap ostaja do nadaljnjega brezplačen in odprtokodni projekt pod licenco Apache.

Aplikacije, narejene s pomočjo tehnologije PhoneGap, so tako imenovani hibridi, kar pomeni, da niso resnično izvorne[1] in niso zgolj spletne aplikacije. Izraz izvorna aplikacija se nanaša na aplikacijo, ki je zasnovana tako, da deluje v določenem računalniškem okolju (v strojnem jeziku in operacijskem sistemu) in je uporabniški program, ki je razvit za uporabo na določeni platformi ali napravi. Izraz se pogosto omenja v kontekstu mobilnega računalništva, saj so bile mobilne aplikacije ustvarjene za delo na določeni platformi naprave. Aplikacije, ki uporabljajo PhoneGap pa niso resnično prave spletne aplikacije, saj so pakirane v aplikacije za distribucijo in imajo dostop do izvirnega programskega vmesnika naprave.

Napisana koda s pomočjo tehnologije PhoneGap se prevede v oblaku. Razvijalec preprosto naloži JavaScript, HTML (HTML5) in CSS (CSS3) kodo prek storitve PhoneGap Build, ki ustvari aplikacijo, ki deluje na različnih mobilnih platformah.

[1] angl. native application

4.2 Zajem Facebook preferenc

Zahtevnejša realizacija pametnega pomočnika bi uporabniku izbrala več turističnih destinacij na podlagi Facebook[17] preferenc. Te preference bi bile vzete iz uporabnikovih všečkov[2], ki so zabeleženi na uporabnikovem profilu družabnega omrežja Facebook.

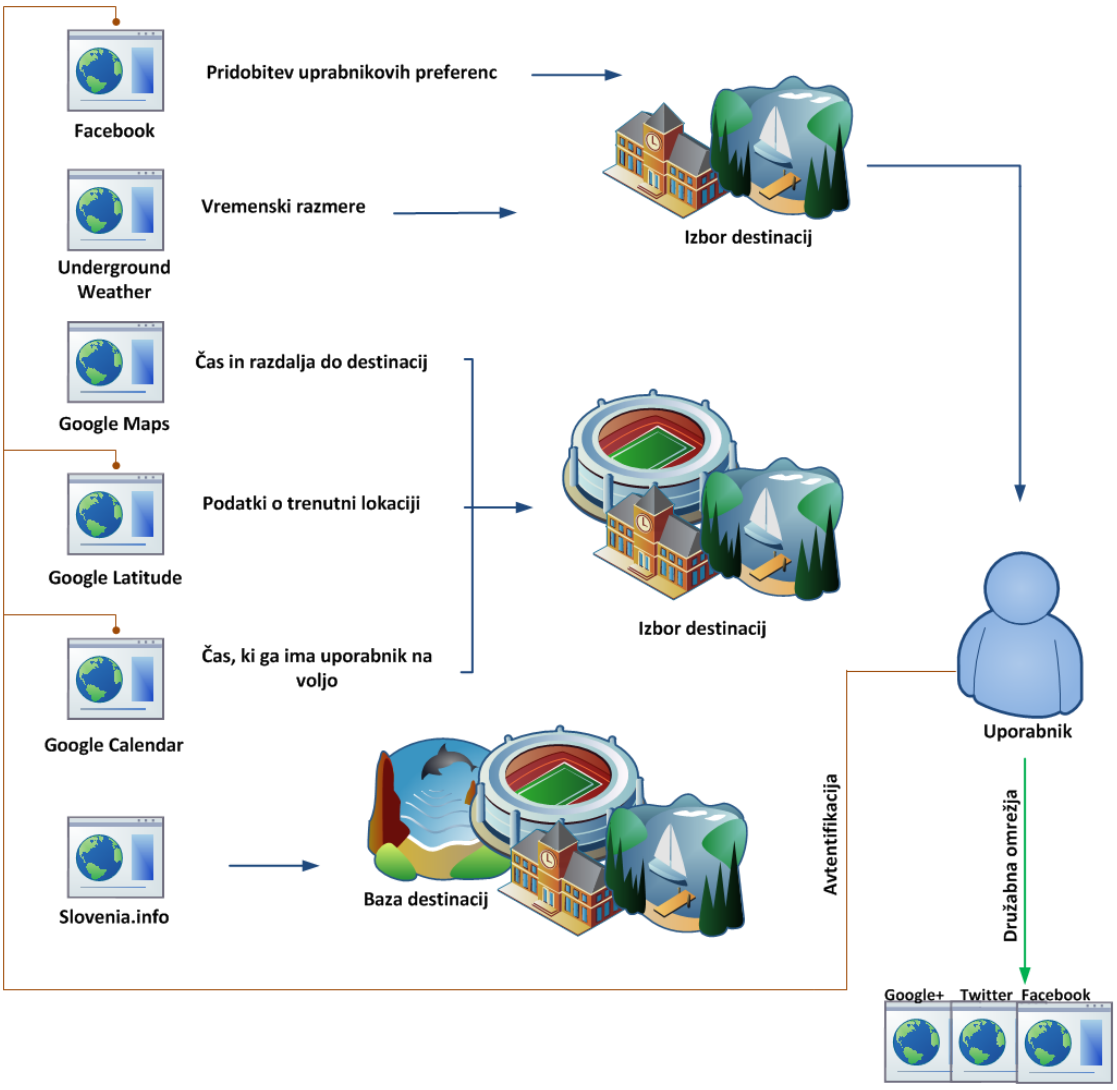
Pomočnik bi moral imeti precej večjo bazo destinacij in zasnovano tako, da bi poleg splošnih podatkov in opisa hranila tudi oznake, ki označujejo lastnosti posamezne turistične destinacije. Na primer oznake za turistično destinacijo Triglavski narodni park bi lahko bile: gore, pohodništvo, narava, šport. Pomočnik bi torej pogledal, katere destinacije ustrezajo večini uporabnikovim všečkom, jih primerjal z oznakami destinacij in na podlagi primerjave izbral tiste, ki se največkrat pojavijo. Na podlagi pogostosti pojavitve bi tako lahko dal uporabniku na izbiro tri ali več turistične destinacije, med katerimi bi uporabnik izbral zanj najbolj primerno. Seveda bi pred tem pametni pomočnik naredil selekcijo lokacij na podlagi razdalje od trenutne lokacije uporabnika. Pomočnik bi ravno tako uporabil vsa že prej navedena Googlova orodja in programski vmesnik za napoved vremenske situacije za izbrane turistične destinacije.

Poleg uporabe Twitter družabnega omrežja za objavo o uporabnikovem izletu bi bilo smiselno uporabniku omogočiti tudi objavo na družabnem omrežju Facebook in drugih družabnih omrežjih.

Sama logika programa bi seveda za tovrstno realizacijo zahtevala poznavanje in implementacijo zahtevnejših primerjalnih, tekstovnih algoritmov. Poleg primerjave slovensko-slovenskih besed bi bilo smiselno realizirati primerjalnik na podlagi angleško-angleških, slovensko-angleških besed, saj veliko slovenskih Facebook uporabnikov uporablja angleške všečke.

Slika 4.2 predstavlja diagram razširjene ideje pametnega pomočnika, ki uporablja zajem Facebook preferenc.

[2]angl. likes



Slika 4.2: Diagram razširjene ideje

Poglavje 5

Sklepne ugotovitve

Namen diplomskega dela je bil razvoj inteligentnega pomočnika v obliki programa, ki bi uporabniku izbral zanj najprimernejšo turistično destinacijo v Sloveniji, ga seznanil z vremenskimi pogoji izbrane destinacije, prikazal lokacijo destinacije v obliki zemljevida in mu omogočil interakcijo o dogodku na spletnem omrežju. Program je bil uspešno končan.

Ko je bila ideja o inteligentnem pomočniku zastavljena, sem začela z razvijanjem programa, izborom primernih in potrebnih javanskih knjižnic in programskih vmesnikov ter druge literature. Tekom razvoja programa sem se seznanila z uporabo različnih programskih vmesnikov in knjižnic, potrebno avtentikacijo za uporabo le-teh in različnimi, meni še ne poznanimi, slovenskimi turističnimi destinacijami. Opraviti je bilo treba registracije za dostop do uporabe želenih programskih vmesnikov, pregledati dokumentacijo in se seznaniti z omejitvami uporabe le-teh.

Ideja spletnega pomočnika se je realizirala v namizno aplikacijo, ki vsebuje vse tehnologije predstavljene v poglavju 2 (Raziskovalna področja) in uporabljene tehnologije in funkcionalnosti, ki so opisane in razložene v poglavju 3 (Integracija podatkov v problemski domeni turizma). Razširitev ideje na mobilno ali spletno aplikacijo bi dodala inteligentnemu pomočniku dodatno uporabnost in boljšo vizualno podobo, zato je želja, da se v prihodnje to tudi realizira. Za določanje trenutne lokacije uporabnika bi bilo

bolje uporabiti foursquare API[18] kot pa Google Latitude API. Foursquare je namreč družabno omrežje, ki bazira na določanju lokacije uporabnika. Uporaba Foursquare storitve je med uporabniki mobilnih naprav veliko bolj popularna kot pa uporaba Google Latitude. Za izračun razdalje in načrt poti med različnimi lokacijami bi bilo morda bolj smiselno uporabiti programski vmesnik ViaMichelin API[19], saj omogoča tudi pregled podatkov o ceni potovanja, ki je za uporabnika pri načrtu izleta koristna.

Literatura

[1] Uradni slovenski informacijski portal:
http://www.slovenia.info

[2] Izvzete turistične destinacije in opisi le-teh so dostopni na:
http://www.slovenia.info/si/Vredno-ogleda!.htm?_ctg_slo_sights=0&lng=1

[3] Uradna spletna stran knjižnice jsoup:
http://jsoup.org/

[4] Uradna spletan stran OAuth protokola:
http://oauth.net/

[5] Google Latitude API. Dokumentacija dostopna na:
https://developers.google.com/latitude/

[6] Google Maps API. Dokumentacija dostopna na:
https://developers.google.com/maps/

[7] Google Calendar API. Dokumentacija dostopna na:
https://developers.google.com/google-apps/calendar/

[8] Google Directions API. Dokumentacija dostopna na:
https://developers.google.com/maps/documentation/directions/

[9] Google Geocoding API. Dokumentacija dostopna na:
https://developers.google.com/maps/documentation/geocoding/

[10] Static Maps API. Dokumentacija dostopna na:
https://developers.google.com/maps/documentation/directions/

[11] Twitter API. Dokumentacija dostopna na:
https://dev.twitter.com/

[12] Twitter4J API. Dokumentacija dostopna na:
http://twitter4j.org/en/index.html

[13] Weather Underground, Weather API. Dokumentacija dostopna na:
http://www.wunderground.com/weather/api

[14] Stopnje Twitter API omejitev. Dokumentacija dostopna na:
https://dev.twitter.com/docs/rate-limiting

[15] Splošno o Twitter omejitvah. Dokumentacija dostopna na:
https://support.twitter.com/articles/15364-about-twitter-limits-update-api-dm-and-following

[16] Uradna stran PhoneGap. Dokumentacija dostopna na:
http://phonegap.com/about

[17] Socialno omrežje Facebook:
http://www.facebook.com

[18] foursquare API. Dokumentacija dostopna na:
https://developer.foursquare.com/

[19] ViaMichelin API. Dokumentacija dostopna na:
http://dev.viamichelin.com/