UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boško Simeunović

ZAJEM FOTOGRAFIJ IZ SPLETNE STORITVE PICASA WEB ALBUMS

DIPLOMSKO DELO NA
VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 00410/2008
Datum: 15.10.2008

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

Kandidat: **BOŠKO SIMEUNOVIĆ**

Naslov: **ZAJEM FOTOGRAFIJ IZ SPLETNE STORITVE PICASA WEB ALBUMS**
ACQUISITION OF PHOTOGRAPHS FROM PICASA WEB ALBUMS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Sodobne spletne storitve omogočajo javno objavo različnih vrst podatkov številnim uporabnikom spleta. Primer take storitve so različni spletni albumi z ogromnimi količinami fotografij, ki so lahko bogat vir za različne raziskave. Številne fotografije vsebujejo tudi podatke o lokaciji posnetka, kar omogoča še dodatne možnosti uporabe.

Napišite javansko aplikacijo za zajem fotografij iz spletne storitve Picasa Web Albums. Aplikacija naj iz spletnega albuma zajame podatke o vseh fotografijah, ki so bile posnete na določenem delu zemljine površine, ter jih shrani v podatkovno bazo za nadaljnjo obdelavo. Pri izvedbi aplikacije uporabite ustrezen pristop ter poskrbite za njeno učinkovito delovanje. Aplikacijo na koncu preverite tudi v praksi.

Mentor:

viš. dr. pred. Igor Rožanc

Dekan:

prof. dr. Franc Solina

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Boško Simeunović

ZAJEM FOTOGRAFIJ IZ SPLETNE STORITVE PICASA WEB ALBUMS

DIPLOMSKO DELO NA
VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: dr. Igor Rožanc

Ljubljana, 2009

original izdana tema diplomskega dela

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a SIMEUNOVIĆ BOŠKO ,

z vpisno številko 63040336 ,

sem avtor/-ica diplomskega dela z naslovom:

Zajem fotografij iz spletne storitve Picasa Web Albums

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  viš. pred. dr. Igor Rožanc

  in somentorstvom (naziv, ime in priimek)

  _____

- 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 _____ Podpis avtorja/-ice: _____

Zahvala

V prvi vrsti gre zahvala družini, predvsem staršem, ki so mi stali ob strani in me v vsem spodbujali ne le v času študija, ampak že od malih nog. Rad bi se zahvalil dr. Rožancu za strokovno pomoč pri vsebinskem in tehničnem delu diplomske naloge. Podjetju Kliki in njenem kolektivu se zahvaljujem za sodelovanje pri diplomski nalogi, kot tudi za vso izkazano pomoč. Na koncu bi se zahvalil še vsem prijateljem, sošolcem in bližnjim, ki so v času študija kakorkoli prispevali k izdelku, ki ga vidite pred sabo.

Kazalo

Seznam uporabljenih kratic in simbolov

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
DAO	Data Access Object
EXIF	Exchangeable Image File Format
HTTP	Hypertext Transfer Protocol
ICE	Internet Communications Engine
IDE	Integrated Development Environment
Java EE	Java Enterprise Edition
Java ME	Java Micro Edition
Java SE	Java Standard Edition
JRE	Java Runtime Environment
MVC	Model View Controller
OS	Operating System
PHP	Hypertext Preprocessor
PWA	Picasa Web Albums
RFC	Request For Comments
RSS	Really Simple Syndication
Slice	Specification Language for ICE
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol / Internet Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language
XP	Extreme Programming

Povzetek

V pričujoči diplomski nalogi je opisan postopek pridobivanja fotografij iz storitve Picasa Web Albums.

Rezultat diplomske naloge je delujoča aplikacija, ki je napisana v programskem jeziku Java. Aplikacija se s storitvijo Picasa Web Albums poveže s pomočjo ustreznih knjižnic, ki med drugim omogočajo prenos podatkov o fotografijah. Vsi za nas zanimivi podatki se shranijo v ustrezno podatkovno bazo, poleg tega pa se ustvari tudi varnostna kopija v obliki XML dokumentov.

S pomočjo izdelane aplikacije lahko zajamemo fotografije iz poljubnega območja Zemljine površine. Območje definirata dve zemljepisni dolžini in dve zemljepisni širini. Zajete so samo geooznačene fotografije.

Uporabniku je na voljo enostaven grafični vmesnik za uporabo aplikacije. Podobne rešitve za enak ali podoben problem nismo opazili. Aplikacijo je vseeno možno še izboljšati.

Ključne besede:

- zajem fotografij
- spletne storitve za objavo slik
- Picasa Web Albums
- geolokacijske storitve

Abstract

This thesis demonstrates the possibility of capturing photos from the Picas Web Albums service.

The result of the thesis is a working application developed in the Java programming language. The application utilizes special libraries to connect to the Picasa Web Albums service, and with these libraries we are able to extract photo data and thus save it to an appropriate database. A backup of this data is also created in XML format.

The application enables us to capture geotagged photos belonging to a custom area of the earth's surface. The area is defined with two latitude and two longitude values.

A simple graphical interface is provided to assist with the use of this application. To date we have not found any similar solutions to this type of problem, and although the application performs well in its current state, it is possible to develop further improvements and functionality to it.

Keywords:

- photo capture
- web services for photo sharing
- Picasa Web Albums
- geolocalized services

1. Uvod

Živimo v obdobju, ko ima internet veliko vlogo v našem življenju. Veliko storitev obstaja, ki nas dnevno zasipajo z raznoraznimi podatki in informacijami ali pa nam omogočajo vsakodnevno deljenje privatnih trenutkov z našimi najbližjimi. Ena od storitev, ki jih internet ponuja, je prenos fotografij na splet. Tukaj mislimo tako na spletne foto galerije kot tudi socialne mreže. Fotografije lahko obogatimo z raznimi podatki kot npr. kje je bila fotografija posneta. To imenujemo geooznačevanje.

Geooznačevanje (angl. geotagging) je proces dodajanja geografskih podatkov v nek medij. Običajno so ti podatki sestavljeni iz zemljepisne dolžine in širine, vsebujejo pa lahko tudi nadmorsko višino in ime kraja. Z geooznačevanjem je iskanje lokacijsko specifičnih informacij mnogo lažje. Uporabnik lahko recimo poišče fotografije nekega kraja tako, da v za to primeren iskalnik vpiše njegovo zemljepisno dolžino in širino.

Danes obstaja veliko storitev, ki za svoje delovanje uporabljajo geografske podatke. Ena najbolj znanih je Google Maps, ki zna prikazati zemljevid določenega kraja. Ostale spletne storitve, ki kakorkoli uporabljajo lokacijske informacije, so še EveryTrail, Tripracker, SmugMug, Geopedia, itd. Naštetih je bilo le nekaj izmed mnogih. Storitve, katerih dejavnost ima opravka s fotografijami, dobijo potrebne podatke iz posebnega formata EXIF fotografije, ali pa jih uporabniki ročno vnesejo. EXIF (ang. Exchangeable Image File Format) je dejansko specifikacija za fotografije, ki so posnete z digitalnimi fotoaparati. Podatki, ki jih vsebuje EXIF, so lahko:

- ime proizvajalca fotoaparata,
- model fotoaparata,
- čas posnetka,
- način kompresije,
- resolucija fotografije,
- zemljepisna dolžina in širina lokacije, kjer je bila fotografija posneta in
- drugi.

Tabela 1 nam kaže podatke, ki jih lahko vsebuje EXIF:

Filename	IMG_0018.JPG
Filesize	624744 bytes
EXIF ComponentsConfiguration	YCbCr
EXIF DateTimeOriginal	2008:05:25 22:09:13
EXIF ExifImageLength	1136
EXIF ExifImageWidth	852
GPS Latitude	N 33d 52m 31.6589s
GPS Longitude	W 116d 18m 5.8306s
...	...

Tabela 1: Primer vsebine EXIF-a

Nekatere spletne storitve je možno povezati z drugimi storitvami ali pa omogočiti lažjo uporabo spletnih storitev z namizno aplikacijo. Določeni organizaciji lahko veliko število podatkov (npr. o fotografijah) koristi, saj lahko opravijo marsikatero raziskavo. Običajno se take organizacije ne ukvarjajo z razvojem programskih rešitev. Za diplomsko nalogo smo se odločili izdelati aplikacijo, ki bo zmožna zajeti fotografije iz spletne storitve – recimo Picasa Web Albums, ki so označene z geografsko dolžino in širino. Pri preverjanju nismo našli nobene obstoječe programske rešitve, ki bi ponujala kaj takega.

V pričujoči nalogi bomo predstavili ta problem širše in izdelali rešitev za primer storitve Picasa Web Albums. Predstavili bomo postopek izbiranja metodologije, uporabo izbrane metodologije in druge možne rešitve problema.

2. Predstavitev uporabljenih metodologij, tehnologij in orodij

2.1. Metodologije

2.1.1. Kaj je metodologija

Metodologija zajema vse, kar redno počnemo, da bi dosegli želen rezultat, torej izdelek ali storitev, ki je cilj našega dela. V primeru razvoja programske opreme to ne pomeni zgolj postopkov, ki so neposredno povezani z razvojem, temveč zajema tudi podporne postopke, načine komunikacije med sodelujočimi, pravila odločanja in podobno. V tem oziru lahko tehnologijo opredelimo kot množico dogovorov, s katerimi se organizacija strinja.

Metodologija je prežeta z idejami ter načeli organizacije in njenih članov, kar še posebej poudari njeno sociološko komponento. Ne more nastati neodvisno od ljudi oz. organizacije. Kljub temu, da so nekatere metodologije formalno določene, si jih posamezna podjetja prilagajajo po svoje, da ustrezajo njihovemu načinu dela ter njihovi domeni. Z uporabo metodologije si uporabniki pridobivajo izkušnje, s čimer se bogati tudi metodologija sama [1].

Poznamo različne metodologije, in vsaka ima svoje prednosti in slabosti. Določena metodologija je primerna za izbrano situacijo, kjer druga metodologija ne bi bila.

2.1.2. Vrste metodologij

Če delimo metodologije glede na njihovo utežitev, jih lahko delimo na *spredaj* in *zadaj utežene* metodologije [13]. Spredaj utežene metodologije so tiste, kjer damo velik poudarek na analizo in načrtovanje kot na kodiranje in testiranje. Pri zadaj uteženih počnemo ravno obratno – večji poudarek damo na kodiranje in testiranje in manj na analizo in načrtovanje. Spredaj uteženo metodologijo imenujemo tudi *težka metodologija*. Drugo ime za zadaj utežene metodologijo je *lahka metodologija* ali *agilna metodologija*. Pojavljajo se še uravnotežene metodologije, ki dajejo poudarek obem aktivnostim.

Načela agilnih metodologij:

- posamezniki in njihova komunikacija so pomembnejši kot sam proces in orodja,
- delujoča programska oprema je pomembnejša kot popolna dokumentacija,
- vključevanje (sodelovanje) uporabnika je pomembnejše kot pogajanje na osnovi pogodb in
- upoštevanje sprememb je pomembnejše kot sledenje planu.

Upoštevanje teh načel ne pomeni, da lahko zanemarimo orodja, dokumentacijo, pogodbe ali plane, ki so tudi pomembni elementi metodologij. Pomeni le, da je pomembneje poskrbeti za posameznike, sodelovanje, delujočo programsko opremo, vključevanje uporabnika ter odzivanje na spremembe, če pride do trenutka, ko je potrebno postaviti stvari na tehtnico.

Agilne metodologije so primerne za razvoj:

- manjših in bolj enostavnih sistemov, z majhnimi in izkušenimi razvojnimi skupinami. Pomanjkanje načrta namreč otežuje delitev dela med člane projekte skupine, zato tak

pristop ni primeren za velike razvojne skupine, hkrati pa podrobnosti načrta nadomešča znanje in izkušenost članov projektne skupine.

- sistemov s slabo določenimi zahtevami, ki se bodo najverjetneje še spreminjale. Kadar zahtev ni mogoče dobro definirati vnaprej, je možen pristop uporaba zadaj utežene metodologije.

- nekritičnih sistemov. Posledica manj podrobne analize in načrtovanja bo tudi manj stabilna arhitektura, ki ni primerna za kritične sisteme. Hkrati je take sisteme pozneje težje nadgrajevati in vzdrževati.

- sistemov, ki uporabljajo tehnologijo, s katero nismo dobro seznanjeni. V primeru, da ne poznamo tehnologije (npr. pogost primer v zadnjem času je razvoj aplikacij za splet), lahko z uporabo zadaj utežene metodologije tehnologijo preizkusimo in spoznamo.

Težke metodologije dajejo poudarek postopkom analize in načrtovanja, ki jih v veliki meri izvedejo vnaprej. Rezultat izvajanja teh postopkov so do podrobnosti opredeljene zahteve za sistem, izdelani natančni načrti in podobno. Delno lahko te postopke izvajajo tudi pozneje in sproti iterativno dopolnjujejo izdelke analize in načrtovanja. Kljub temu se nastali načrt pozneje bistveno ne spreminja. Izdelava sistema večinoma pomeni le »rutinsko« kodiranje metod, ki so z načrtom podrobno opredeljene. Praviloma je potrebnega tudi manj testiranja, saj je zaradi dobrega načrta manj napak v sistemu.

Težke metodologije so primerne za razvoj:

- sistemov s stabilnimi zahtevami. Glede na to, da je poudarek na analizi in načrtovanju, ki ju večinoma izvedemo vnaprej, morajo biti zahteve dobro specificirane in stabilne. Pozneje so večje spremembe v načrtu drage, pogosto pa tudi rušijo dobro arhitekturo prvotno načrtovanega sistema.

- kritičnih sistemov. Pri kritičnih sistemih (npr. življenjsko kritični sistemi) sta dobra analiza in načrtovanje ključnega pomena. Pri tem moramo predvideti tudi čimveč možnih alternativnih scenarijev dogajanja (alternativni dogodki, napake), ki jih zajamemo že v načrtu.

- obsežnih in kompleksnih sistemov, z velikimi razvojnimi skupinami. Kadar imamo opravka z veliko razvojno skupino, so potrebni bolj podrobni načrti, ki omogočajo usklajevanje in razdeljevanje dela med člani.

2.1.3. Metodologija ekstremno programiranje - XP

Metodologija ekstremno programiranje (XP) [3] je eno od najbolj znanih metodologij, ki uporabljajo agilni pristop. Predpostavlja štiri ključna področja dela, vsako področje določa specifične postopke, ki so po vsebini in načinu izvajanja bistveno drugačni od postopkov pri klasičnih metodologijah planiranja in razvoja [12]:

- planiranje:
    - zahteve so kratke, predstavljene so lahko na način kot je UML (grafična notacija za risanje diagramov programskih zamisli)
    - skupinsko znanje je koncept, po katerem so vsi udeleženci na projektu

seznanjeni s specifikacijami in planiranjem oddaje kode

- dnevni sestanki razvojne skupine se uporabljajo za prenos znanja in obveščanje o morebitnih problemih in napakah

- načrtovanje
  - razvoj programske opreme na osnovi določenih vzorcev programiranja kot so to MVC[1], DAO[2], itd.
  - popravljanje kode (ang. refactoring) je najmočnejši koncept agilne metodologije, možnost, da se na osnovi znanih vzorcev popravlja in izboljšuje programska koda

- kodiranje
  - programiranje (ang. Pair Programming) je najbolj nasprotujoč koncept v metodologiji; čas, izkušnje in meritve bodo pokazale dejanske učinke skupinskega programiranja
  - posamezno dodajanje kode (ang. Integrate One by One) pomeni postopno dopolnjevanje programske kode
  - vsak programer pozna kodo drugih programerjev in je sposoben nadaljevati delo drugega programerja
  - koda se hitro oddaja, optimizacija se izvaja na koncu projekta
  - 40-urni delovni čas, saj so samo spočiti programerji sposobni dosegati rezultate

2.2. Programski jezik Java

Java [6] je objektno usmerjen programski jezik, ki je neodvisen od platforme. Zasnova je zamišljena tako, da prevajalnik prevede izvorno kodo v vmesno kodo (ang. bytecode), ki je sestavljena iz množice platformno neodvisnih ukazov. Ti ukazi tečejo na t.i. virtualnem stroju Java, oz. jih interpretira Java Runtime Environment. Tak način delovanja pomeni za razvijalca programske opreme, da aplikacijo napiše samo enkrat, le–ta pa se lahko izvaja na več platformah. Razvijalci pišejo računalniške ukaze v obliki ukazov, ki uporabljajo angleške besede. Ta lastnost uvrsti Javo med visokonivojske jezike, saj se da programe enostavno brati in pisati. Java je varen programski jezik, saj

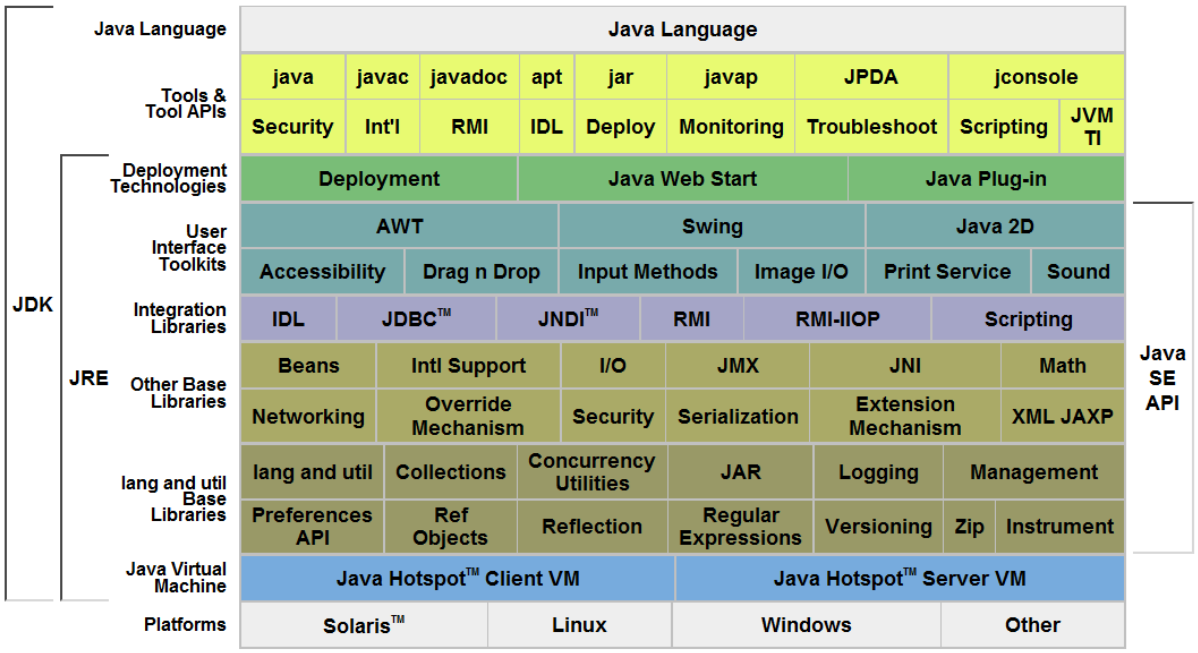
- ni aritmetike s kazalci,
- pretvorbo med različnimi tipi preverja prevajalnik,
- samodejno upravlja s pomnilnikom in
- vsebuje pakete za delo z mrežo, datotekami, nizi, grafiko...

[1] Model – View – Controller → Model je objekt, ki hrani informacije, nad katerim izvajamo operacije v aplikaciji. View je prezentacija modela uporabniku aplikacije. Controller je koda, odgovorna za reakcijo na uporabniški vnos ali interakcijo [7].
[2] Data Access Object → Objekt, ki ponuja abstrakten vmesnik do podatkovne baze.

Razvijalcem so na voljo 3 različne izdaje, glede na to, kakšen produkt razvijajo [8]:

- Java tehnologija v majhnih in mobilnih napravah (Java Micro Edition),
- Java tehnologija v osebnih računalnikih (Java Standard Edition) in
- Java tehnologija v srednjem in velikem poslovnem okolju (Java Enterprise Edition).

Izdaja Java Micro Edition (Java ME) je namenjena razvoju aplikacij za naprave z omejenimi viri, kot so npr. mobilne naprave.



Slika 1. Struktura javanske platforme

Izdaja Java Standard Edition (Java SE) je namenjena razvoju varnih, prenosnih in visoko zmogljivih aplikacij za najširši krog namiznih računalnikov (računalniki z nameščenim Apple Macintosh, Linux, Microsoft Windows ali Sun Solaris operacijskim sistemom).

Izdaja Java Enterprise Edition (Java EE) je primerna za razvoj aplikacij z večnivojsko arhitekturo.

Končnim uporabnikom je namenjen JRE paket (Java Runtime Environment), ki služi izvajanju vmesne kode, oziroma aplikacij, ki jih je razvil razvijalec.

Slika 1 prikazuje strukturo javanske platforme.

2.3. Middleware

2.3.1. Kaj je middleware

Middleware bi najlažje opisali kot programsko opremo, ki omogoča aplikaciji ali komponentam aplikacije povezovanje in izmenjavo podatkov z drugimi aplikacijami oz.

njihovimi komponentami [2]. Middleware na ta način služi povezovanju aplikacij, ki tečejo na različnih platformah. Skriva heterogenost, veča nivo abstrakcije in nudi splošne skupne storitve, omogoča lokacijsko transparentnost, zanesljivost, skalabilnost[3] in neodvisnost od mrežne arhitekture.

Za programerja in uporabnika uporaba middleware-a pomeni, da jima ni potrebno poznati internega procesiranja za doseganje prej omenjenih ciljev, dodatna prednost za razvijalca pa je uporaba visokonivojskega programskega vmesnika API (ang. Application Program Interface).

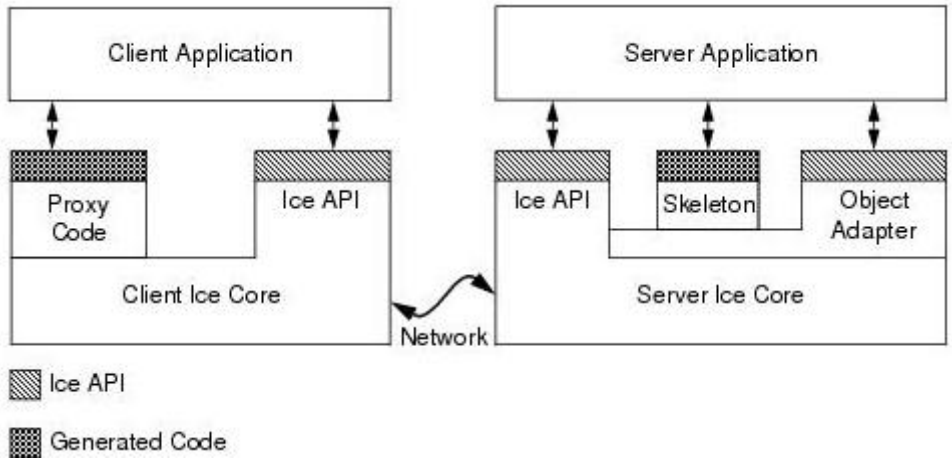
2.3.2. Middleware ICE

ICE (ang. Internet Communications Engine) middleware je sodoben objektno orientiran middleware [14], ki podpira programske jezike C++, .NET, Java, Python, Ruby in PHP. Sestavljen je iz naslednjih paketov:

- Slice (ang. Specification Language for Ice) je jezik, ki določa specifikacije in lastnosti ICE middleware-a, poleg tega določa tudi razmerje med odjemalcem in strežnikom.
- Slice prevajalniki prevedejo Slice specifikacije v različne programske jezike. Na ta način lahko odjemalci in strežniki komunicirajo med seboj, čeprav so napisani v različnih programskih jezikih.
- Ice jedro skrbi za vso komunikacijo, ponuja fleksibilno množico niti (ang. thread pool) za večnitne strežnike ter visoko skalabilnost.
- IceUtil paket vsebuje koristne funkcije.
- IceBox je aplikacijski strežnik, namenjen ICE aplikacijam.

ICE middleware nam tako omogoča pisanje strežniških aplikacij v enem programskem jeziku in na eni platformi, medtem, ko je lahko odjemalska aplikacija izdelana v drugem programskem jeziku in na drugi platformi. To dosežemo tako, da napišemo Slice definicijo (ang. Slice definition) v ICE programskem jeziku. Slice definicijo nato prevedemo v nam priljubljeni programski jezik s pomočjo ICE orodij.

[3] Enostavno spreminjanje sistema tako, da bo dovoljeval povečevanje števila uporabnikov, virov in računalnikov.



Slika 2. Odjemalec - strežnik Ice struktura

Ukaz, ki nam omogoča pretvorbo Slice datotek v (npr.) Java izvorno kodo se imenuje `slice2java`.

```
$ slice2java --output-dir generated Printer.ice
```

`output-dir` je stikalo, kateremu podamo ime imenika (v primeru je to `generated`), kamor naj shrani zgenerirane datoteke. `Printer.ice` je ime slice datoteke, ki vsebuje definicijo, potrebno za pretvorbo v Java izvorno kodo.

Slika 2 prikazuje način komunikacije med strežnikom in odjemalcem, ki za komunikacijo uporabljata ICE middleware.

2.4. SQLite

SQLite [11] je programska knjižnica, ki implementira

- samovsebujoč (ang. Self-contained),
- brezstrežniški (ang. Serverless) in
- transakcijski (ang. Transactional)

SQL pogon podatkovne baze (ang. SQL Database Engine), ki ga ni potrebno konfigurirati (ang. Zero – configuration). SQLite se od ostalih SQL pogonov razlikuje predvsem v tem, da je njegov primarni cilj biti enostaven:

- za uporabo,
- za upravljanje (administracijo),
- za vgradnjo v večji program ter

- za prilagajanje.

Ostale prednosti, ki jih SQLite poseduje, so še, da je majhen, hiter in zanesljiv. SQLite ni potrebno namestiti pred uporabo (ni namestitvenega postopka).

Večina SQL pogonov je implementiranih kot ločen strežniški proces. Programi, ki želijo dostopati do takih pogonov, morajo zato uporabiti neko medprocesno komunikacijo (kot je npr. TCP/IP). Pri SQLite se vse branje in pisanje vrši direktno na disk. Tak način dostopa ima svoje prednosti kot tudi slabosti. Glavna prednost je seveda ta, da ni potrebno zagnati nobenega namestitvenega ali konfiguracijskega postopka. Po drugi strani pa baza, ki uporablja strežnik, ponuja večjo varnost pred hrošči (ang. bugs) v odjemalčevi aplikaciji.

Datotečni format SQLite ni vezan na samo eno specifično platformo. Podatkovna baza, ki je pravzaprav datoteka napisana na enem računalniku, se lahko enostavno prenese na drug računalnik z drugo arhitekturo/platformo. Vsi računalniki uporabijo enak datotečni format. Poleg tega je datotečni format združljiv z nižjimi verzijami, kar pomeni, da lahko novejše verzije SQLite berejo in zapisujejo tudi v stare podatkovne baze SQLite.

SQLite je transakcijska podatkovna baza. V transakcijski podatkovni bazi se vse spremembe in poizvedbe pojavljajo kot atomske, konsistentne, izolirane in trajne (ang. Acid, Consistent, Isolated, Durable – ACID). To velja tudi, če je transakcija prekinjena zaradi napake v programu, operacijskem sistemu ali pa če računalnik izgubi električni tok. Vsaka transakcija se izvrši v celoti ali pa se sploh ne izvrši.

2.5. XML

XML [15] je tričrkovna okrajšava za angleški izraz Extensible Markup Language, ki pomeni razširljiv označevalni jezik. Je preprost jezik, ki nam omogoča format za opisovanje strukturiranih podatkov ali arhitekturo za prenos podatkov in njihovo izmenjavo med več omrežji. XML je razdeljen na 3 dele:

- podatkovni – vanj shranimo podatke v neki obliki z željenimi etiketami (ang. tag),
- deklarativni – skrbi za to, da lahko pri dodajanju novih podatkov vidimo, kaj kakšna etiketa predstavlja ter
- predstavitveni – z njim oblikujemo izpis podatkov.

XML je pravilen, če je ustrezno oblikovan (izpolnjuje vsa sintaksna pravila) in veljaven. Je generično ogrodje za hrambo katerekoli dolžine teksta ali podatkov, katerih struktura je lahko prikazana tudi v obliki drevesa. Sintaksna pravila določajo, da:

- morajo vsi XML elementi imeti končno etiketo,
- so etikete občutljive na velike/male črke,
- morajo XML elementi biti pravilno gnezdeni ter
- mora XML dokument imeti en element, ki bo korenski (ang. root element).

Sintaktičnih pravil je veliko več kot jih je predstavljeno, vendar za predstavitev XML jezika bralcu ni potrebno poznati vseh. Primer pravilno oblikovanega XML dokumenta, ki izpolnjuje vsa zgornja našteta pravila, je recimo:

```
<knjiga><b><i>Sneguljčica in sedem palčkov</b></i></knjiga>
```

Veljaven XML dokument izpolnjuje določena semantična pravila. Ta pravila lahko definira uporabnik, ali pa so definirana v XML shemi. Če dokument vsebuje nedefiniran element, je celoten dokument neveljaven.

2.6. Razvojno orodje Eclipse

Razvoj programov oz. aplikacij je mnogo lažji, če uporabimo razvojno orodje. Razvoj bi bil načeloma mogoč že z enostavno aplikacijo kot je **beležka** v Windows OS ali **vi** urejevalnik v Unix OS, vendar nam razvojna orodja nudijo nekatere prednosti:

- avtomatiziranje ponavljajočih opravil,
- prevajanje kode,
- označevanje določenih segmentov kode za lažjo orientacijo in navigacijo po kodi,
- dokončanje kode ob pisanju (ang. code completion),
- formatiranje kode,
- prikaz pomoči in definicije metod in razredov,
- razhroščevanje kode (ang. debugging),
- preverjanje pravilnosti kode že med pisanjem,
- integriranje z repozitoriji za izvorno kodo (ang. source code repositories),
- integriranje s spletnimi servisi (kot je recimo Apache Tomcat),
- integriranje s prevajalnimi orodji (npr. Apache Ant) ter
- refaktorizacija kode.

Nekateri urejevalniki omogočajo dodajanje vtičnikov (ang. plugin), da lahko zunanji razvijalci razširijo funkcionalnost orodja.

Eclipse je [4] odprtokodna skupnost. Njeni projekti so usmerjeni v razvoj razširljivih razvojnih orodij. Razvijalec, ki se odloči uporabiti Eclipse razvojno orodje, ima na voljo različne možnosti:

- Enterprise Development – razvoj aplikacij za srednja in velika podjetja,
- Embedded + Device Development – razvoj aplikacij za majhne naprave in mobilne aparate,
- Eclipse IDE for Java Developers – razvoj aplikacij v Java programskem jeziku,
- Eclipse IDE for Java EE Developers – razvoj aplikacij v Java programskem jeziku za

srednja in velika podjetja,

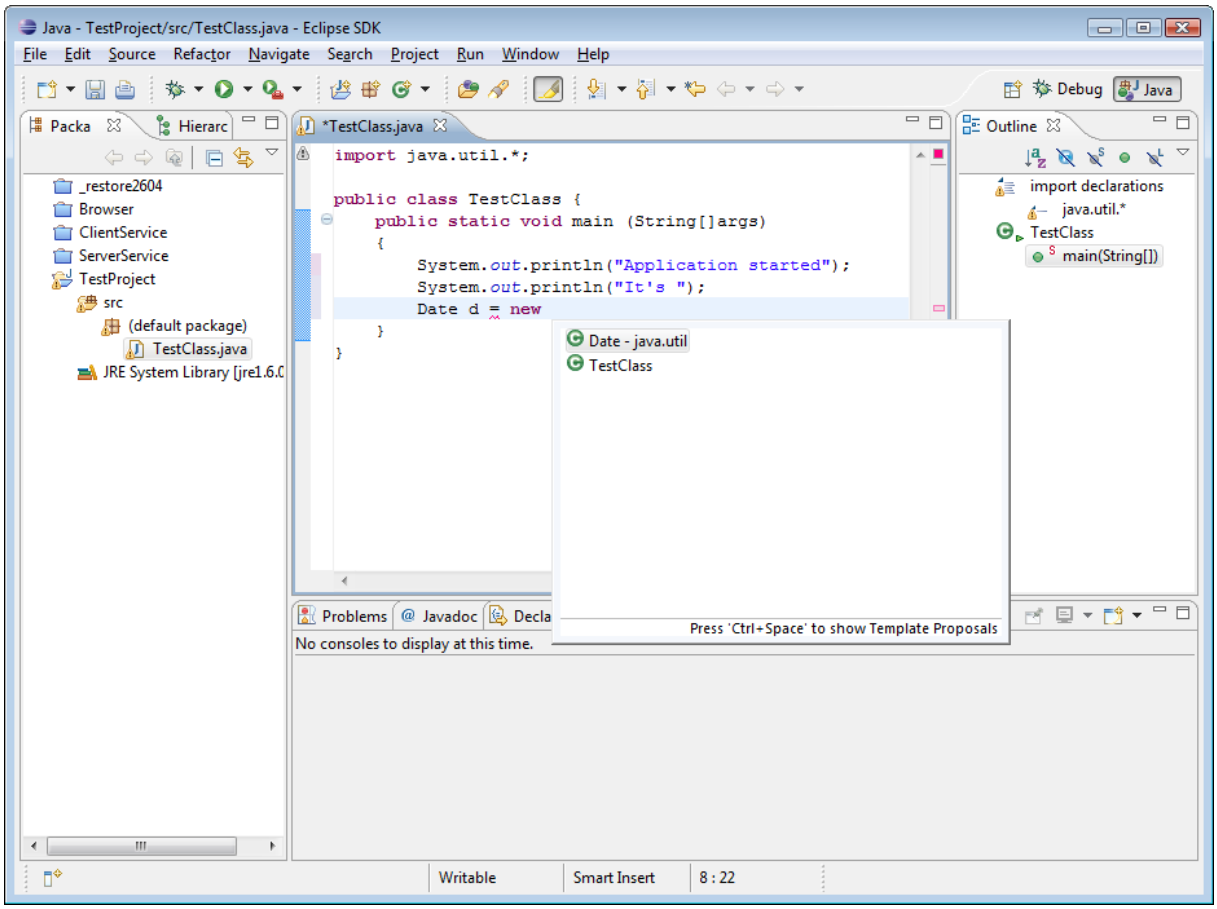
– Eclipse IDE for C/C++ Developers – razvoj aplikacij v programskih jezikih C in C++ ter

– Eclipse for RCP/Plug-in Developers – razvoj vtičnikov za razvojno orodje Eclipse.

Eclipse kot razvojno orodje ponuja vse prednosti, ki so naštete na začetku tega poglavja. Pri reševanju našega problema smo uporabili izdajo Eclipse IDE for Java Developers. To je tudi osnovna izdaja, ki je zadoščala našim potrebam.

Slika 3 nam kaže uporabniški vmesnik razvojnega orodja Eclipse. Izbran je Java pogled, ki je namenjen hitrejšemu pisanju kode. Namesto tega lahko izberemo Debug pogled, ki je namenjen razhroščevanju programske kode. Obstajajo tudi drugi pogledi dela, vendar sta ta dva najbolj uporabljena. Primeren pogled lahko izberemo v desnem kotu zgoraj.

Na desni strani slike lahko vidimo razrede, ki so uporabljeni v aktivnem projektu, ter metode, ki pripadajo razredu. Razrede in metode lahko sortiramo po abecedi ali po vrstnem redu, kot se nahajajo v izvorni kodi. Na levi strani so prikazani obstoječi projekti. Urejevalni del oz. del, kjer pišemo programsko kodo, vidimo na sredini okna. Prav tako je vidno podokno za dokončanje kode. Na spodnji strani Eclipse urejevalnika se izpisuje stanje izvajanja, razhroščevanja, napake, kot tudi opravila (ang. To – Do task).

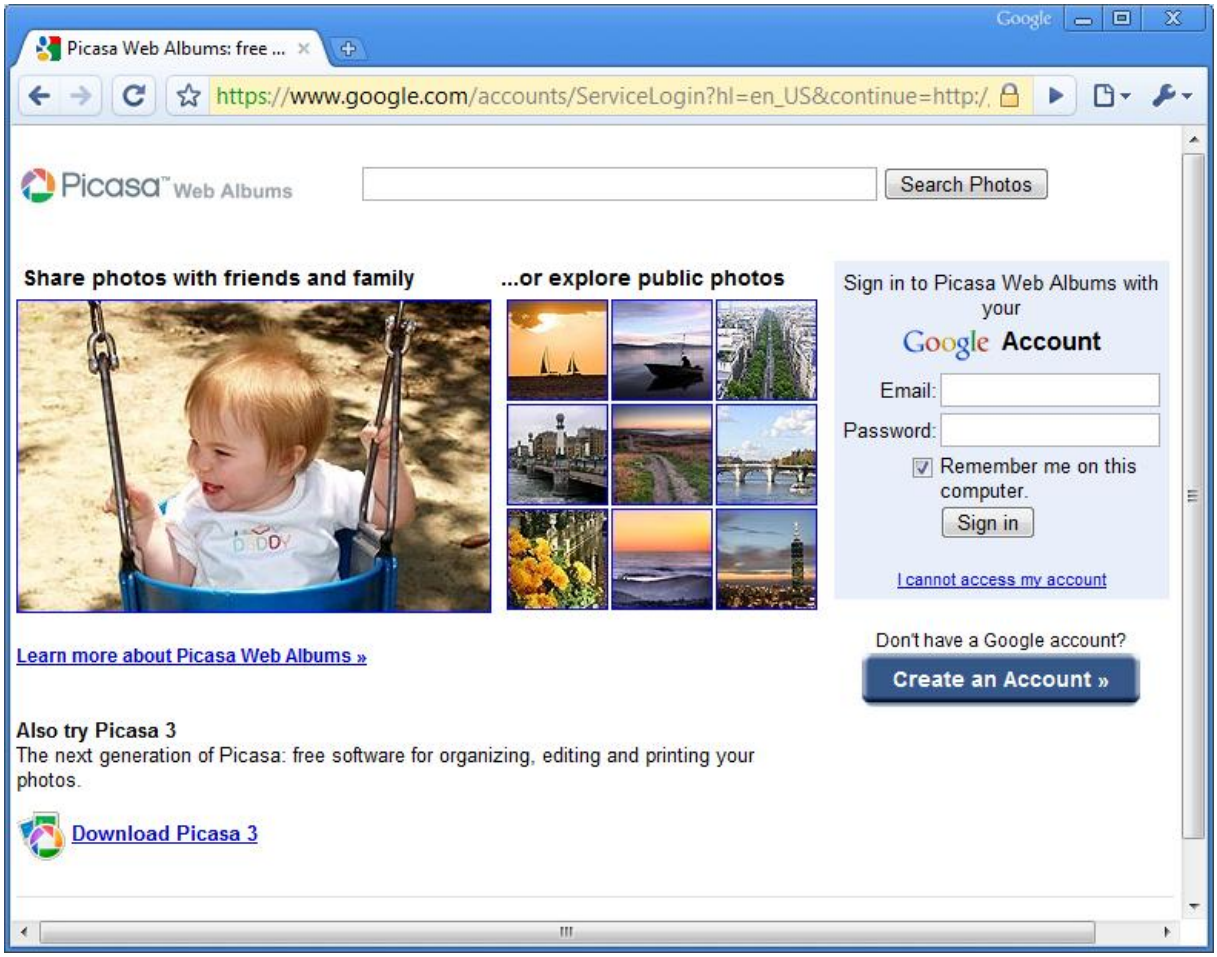


Slika 3. Grafični vmesnik razvojnega orodja Eclipse

2.7. Predstavitev Picasa Web Albums – PWA

Picasa Web Albums (oz. spletni albumi Picasa) [9] je spletna storitev, ki omogoča uporabnikom prenos svojih fotografij na splet. Javnosti je bila najprej predstavljena aplikacija Picasa, ki je omogočala organizacijo in urejanje digitalnih slik na domačem računalniku. Aplikacijo je razvilo podjetje Idealab, leta 2004 pa jo je kupil Google in ponudil uporabnikom brezplačno. V letu 2006 je bila predstavljena Picasa Web Albums. PWA se od aplikacije Picasa razlikuje v tem, da ne omogoča organizacije in urejanja slik. Služi le temu, da uporabniki pokažejo slike javnosti ali pa PWA uporabijo kot privatno galerijo, ki posledično dobi funkcijo varnostne kopije.

Prenos fotografij iz domačega računalnika na PWA je možno izvesti preko spletnega brskalnika, še lažji prenos pa je mogoč z namizno aplikacijo Picasa. Uporabniki morajo za prenos fotografij biti registrirani, prenešene fotografije pa lahko prikažejo bodisi javnosti bodisi samo določenim osebam. Ob prenosu fotografij na splet lahko uporabnik določi, kje so bile fotografije posnete, ni pa nujno. PWA lahko določi kraj fotografije na podlagi EXIF podatkov (če le–ti vsebujejo potreben podatek). Uporabnikom PWA je bilo v času pisanja diplomske naloge na voljo 1 GB spletnega prostora za slike, večjo količino pa je bilo možno dokupiti. Na sliki spodaj je prikazana prijavna stran za vstop v PWA.



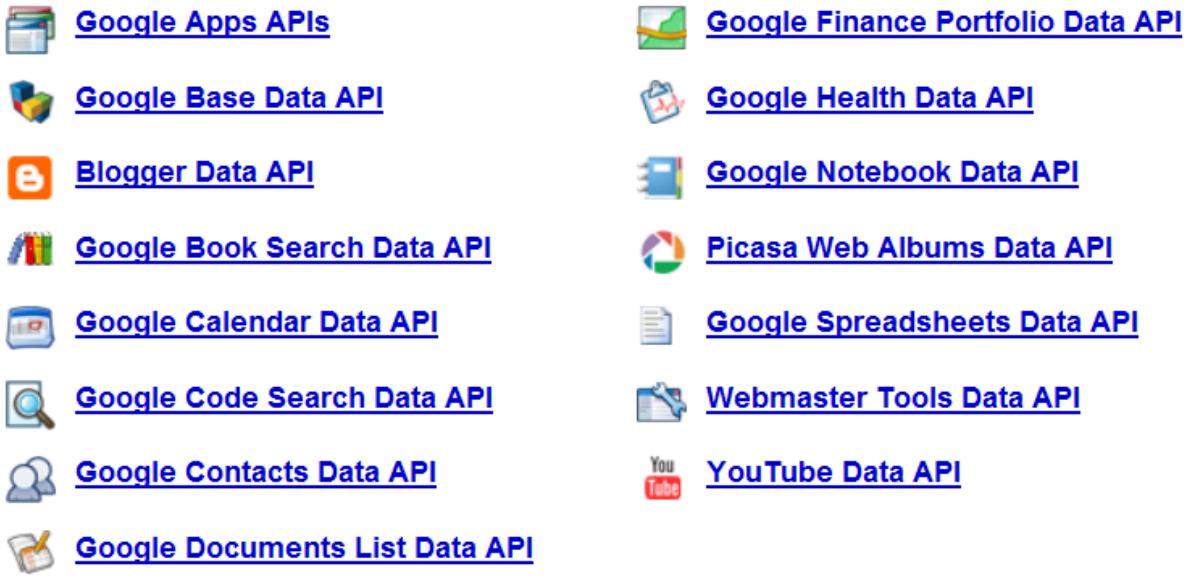
Slika 4. Prijavna stran Picase Web Albums

2.8. Google Data API

Podjetje Google ponuja veliko spletnih storitev končnim uporabnikom. Njegove najbolj znane storitve so Youtube, Gmail, Apps, PWA itd. Uporabniki (oz. razvijalci), ki želijo povezati storitve podjetja Google s svojimi, imajo na voljo Google Data API. Slika 5 kaže Google Data API komponente, ki omogočajo povezavo z večino storitev podjetja Google.

Google Data API [5] omogoča razvijalcem enostavnejše razvijanje aplikacij, ki so zmožne interakcije z Google storitvami. Ponuja enostaven standarden protokol za pisanje in branje podatkov na spletu. Uporablja enega od dveh formatov, ki temeljita na XML-ju: Atom ali RSS. Google Data API vsebuje sistem za objavo virov (ang. feed), ki je sestavljen iz protokola Atom za objavo in dodatkov za delo s poizvedbami.

Če želimo od storitve, ki ponuja Data API, dobiti informacijo, moramo poslati HTTP GET zahtevo. Storitev vrne rezultat v obliki RSS ali Atom feed-a. Podatke je možno posodobiti s HTTP PUT zahtevo. Data API ne ustvari ali zbriše vira, ampak to stori storitev sama.



Slika 5. Google Data API komponente

3. Razvoj aplikacije za zajem fotografij iz storitve PWA

3.1. Zajem zahtev

3.1.1. Uvod

Spletne fotogalerije omogočajo končnim uporabnikom prenos svojih fotografij na splet. Od tega imajo korist vsi uporabniki, ki uporabljajo fotogalerijo, gotovo pa tudi podjetje, ki jim je vse to omogočilo (lastnik storitve). Poleg teh dveh akterjev lahko v igro vstopi tudi tretji igralec. Uporabniku je možno ponuditi izdelek, ki bo uporabo spletne fotogalerije dodatno olajšal. Tretji akter se lahko pojavi tudi v drugačni vlogi. Javno objavljene fotografije lahko analizira in ugotovi marsikatero pravilo za določeno obdobje ali kraj.

Po krajšem raziskovanju nismo našli nobene obstoječe rešitve, ki bi omogočala zajem fotografij iz spletnih fotogalerij. Zato smo se odločili izdelati aplikacijo, ki bo to sposobna oz. bo znala zbrati osnovne podatke o fotografijah, ki se nahajajo v fotogaleriji. Aplikacija bo delovala na primeru storitve Picasa Web Albums. V podjetju, ki se ukvarja z razvojem geolokacijskih storitev, sem v času študija opravljal študentsko delo. Tam sem opazil, da je problem kot tak zanimiv tudi za njih.

3.1.2. Naloge aplikacije

Glavna funkcionalnost, ki jo naloga zahteva, je izdelava aplikacije za zajem fotografij iz storitve PWA. Zajete fotografije je potrebno shraniti v ustrezno podatkovno bazo. Pri tem ne smemo zajeti samih fotografij, ampak le URL povezave do teh. Nad zajetimi fotografijami lahko opravimo marsikatero raziskavo, analizo ali pa jih kar vključimo v katero od obstoječih storitev.

Celoten sklop zahtev, ki so bile predstavljene:

- aplikacija mora omogočati zajem fotografij iz Zemljine površine,
- podatkovna baza mora za vsako fotografijo vsebovati geografsko dolžino in širino, kjer je bila posneta[4],
- podatkovna baza mora vsebovati atribut, po katerem se lahko določi popularnost fotografije,
- podatkovno bazo je možno posodobiti kadarkoli z novimi fotografijami ali iz nje odstraniti tiste, ki se na spletni strani ne nahajajo več (izbrisane fotografije),
- poleg osnovnih podatkov o fotografiji zajamemo še ostale podatke, ki bi bili lahko koristni,
- intuitiven grafični vmesnik,
- izvajanje aplikacije naj bo čim bolj hitro ter
- izdelava primerne varnostne kopije.

[4] Fotografije, ki so objavljene na PWA so lahko z ali brez podatka o geografski lokaciji. Ker je za nalogo ta podatek pomemben, bomo zajeli le fotografije, ki ta podatek vsebujejo – geooznačene fotografije.

3.1.3. Predpogoji

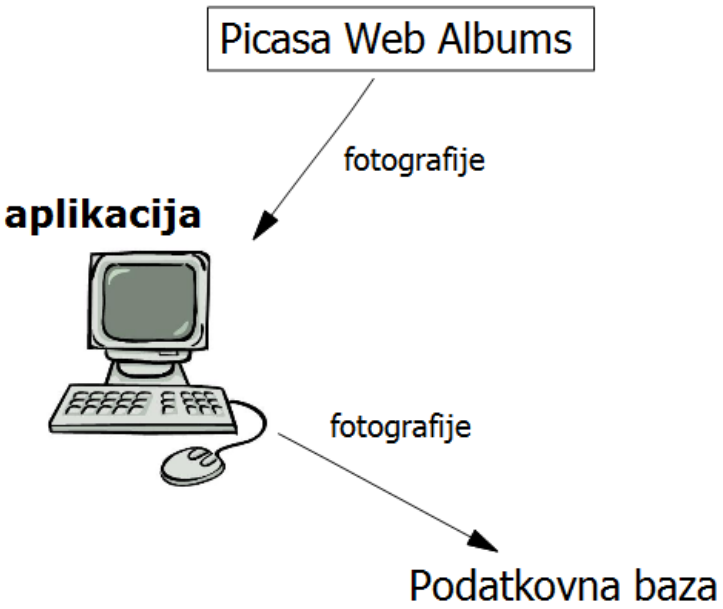
Za uporabo aplikacije za zajem fotografij iz storitve PWA so predpogoji naslednji:

- solidno zmogljiv računalnik,
- brez internetne povezave je aplikacija neuporabna ter
- za uporabo aplikacije je potrebno osnovno poznavanje geografskih pojmov in osnovne uporabe računalnika.

3.2. Razvoj v skladu z agilno metodologijo XP

Pri izbiri primerne metodologije za rešitev našega problema se je kot najbolj primerna metodologija izkazala metodologija ekstremnega programiranja. Naloga zahteva delujočo rešitev, kjer sama dokumentacija ni zahtevana. Glede na to, da je za diplomsko nalogo predviden čas izdelave 3 mesece, smo bili metodologijo XP deloma tudi prisiljeni izbrati zaradi kratkega časa in omejenih sredstev. Ostale metodologije zahtevajo več osebja ali dolgoročnejše planiranje izvedbe celotnega projekta.

Tako smo pri izdelavi diplomske naloge pazili, da nismo prekoračili 40 urnega tedenskega delovnika, držali smo se standardov kodiranja in testirali delovanje po vsaki končani iteraciji. Testiranje je bilo izvedeno s funkcionalnimi testi (ročno). Slika 6 prikazuje model osnovnega delovanja aplikacije, ustvarjenega pred samim začetkom reševanja problema.



Slika 6. Enostaven prikaz sistema za zajem fotografij iz storitve PWA

Naloga zahteva, da mora aplikacija omogočati zajem fotografij iz storitve PWA, kot tudi posodobitev že obstoječih fotografij. Če delamo po XP metodologiji, moramo najprej razviti funkcionalnosti, ki so najbolj pomembne za naročnika (oz. za nalogo). Samoumevno je, da je v našem primeru potrebno najprej razviti osnovni zajem. Brez predhodno zajetih fotografij

posodobitve sploh ne moremo izvesti. Šele, ko bomo končali z implementacijo osnovnega zajema fotografij, bomo lahko nadaljevali z možnostjo posodabljanja. Pri tem bomo naleteli tudi na težave, opisane v kasnejših poglavjih. Ker sta si ti dve funkcionalnosti (osnovni zajem in posodobitev) podobni, smo se izognili podvajanju kode in implementacijo osnovnega zajema spremenili toliko, da je le–ta dobila možnost posodabljanja. S tem smo upoštevali še eno od načel ekstremnega programiranja. Naslednje načelo, ki je bilo očitno uporabljeno, je zagotavljanje pogostih izdaj. Pri tem se morajo obstoječe funkcionalnosti aplikacije po vsakem dopolnjevanju nove pravilno izvesti. Kot že omenjeno je bilo to poskrbljeno s funkcionalnimi testi. Tako je

- prva izdaja omogočala zajem fotografij s prednastavljenim območjem (ni ga bilo možno spreminjati),
- druga izdaja omogočala poleg zajema, še hranjenje zajetih podatkov v podatkovni bazi,
- tretja izdaja omogočala posodabljanje in
- četrta izdaja omogočala vnos vrednosti za specifično področje (po meri).

Izdaj je dejansko bilo več, naštete so le najpomembnejše izdaje.

Nekaterih načel XP ni bilo mogoče upoštevati. Narava diplomske naloge zahteva, da se ta opravi samostojno. Zaradi tega recimo ni bilo mogoče programirati v parih.

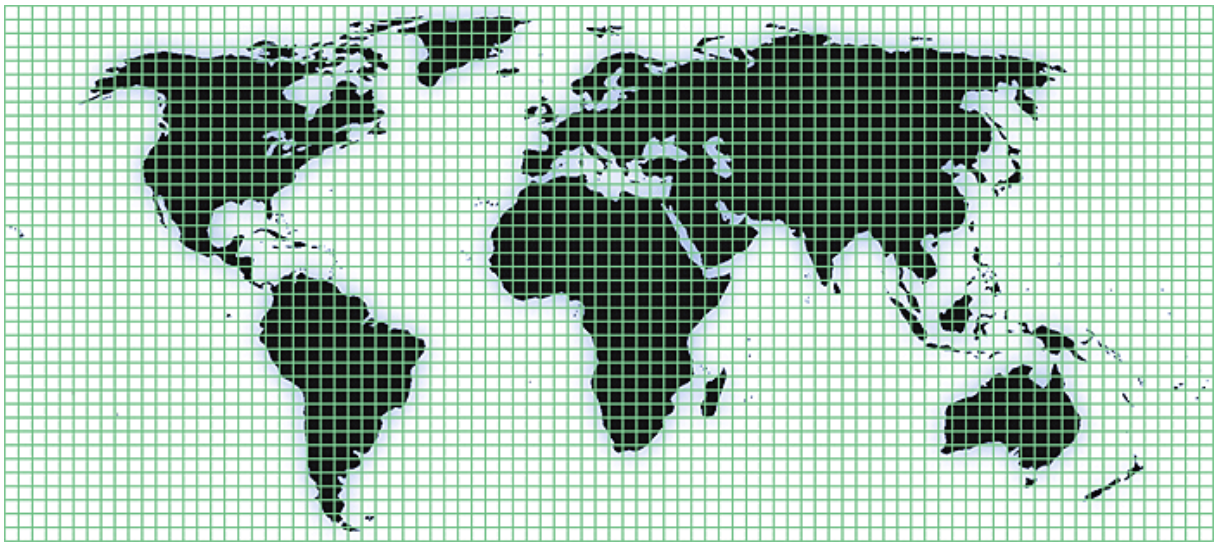
3.3. Izvedba rešitve

3.3.1. Opis

Pregledati je potrebno del Zemljine površine. Gledano s stališča zemljevida je to pravokotnik, ki je omejen z dvema zemljepisnima dolžinama in širinama. Pravokotnik, ki zajema celotno površino Zemlje, ima naslednje vrednosti:

- levo stranico -180° zemljepisne dolžine,
- desno stranico 180° zemljepisne dolžine,
- zgornjo stranico 90° zemljepisne širine ter
- spodnjo stranico -90° zemljepisne širine.

Ko uporabnik zažene aplikacijo, mora vanjo vnesti geografski dolžini in širini območja, ki ga želi preiskati. Preden se pregledovanje začne, se izbrano območje razdeli na enake kvadrate. Vsak kvadrat se pregleda in če je potrebno še dodatno zmanjša, kot je opisano v nadaljevanju. Ko se prvotni kvadrat konča, se obdela naslednji iz seznama še nepregledanih kvadratov. Kvadrati so imeli eno stranico dolžine 0.2°. Taka vrednost je bila izbrana na podlagi testiranj med razvojem aplikacije. Kako bi izgledalo, če bi hoteli zajeti fotografije iz celotne Zemlje, razdeljeno na pregledovalne kvadrate, lahko vidimo na sliki 7.



Slika 7. Zemlja razdeljena na pregledovalne kvadrate

Pri zajemanju slik iz storitve PWA je prav prišel PWA Data API [10]. PWA Data API je del Google Data API in omogoča spletnim stranem in aplikacijam integracijo s PWA. Vsebuje knjižnice (ang. libraries), ki delo z njim olajšajo. Programski jezik, v katerem smo se odločili razviti aplikacijo, je Java, saj je to jezik v katerem smo najbolje tehnično podkovani. Posledično je s tem izbrana Java knjižnica (ang. Java client library). Knjižnica omogoča delo ne samo s PWA, ampak tudi z ostalimi Google storitvami (Youtube, Googlemaps, ...).

3.3.2. Povezava s PWA in zajem slik

Pisanje programske kode za aplikacijo smo začeli z naslednjim stavkom:

```
PicasawebService picacaService = new
  PicasawebService("application name");
```

S tem stavkom ustvarimo objekt, ki se lahko poveže z Google Photo storitvijo (PWA), storitvi PWA se pa predstavi kot `application name`. Sledil je zajem slik, pri katerem je potrebno najprej sestaviti URL feed kot je npr:

```
String feedUrl =
"http://picasaweb.google.com/data/feed/api/all?kind=
  photo&max-results=500&start-index=1&bbox=0.0,0.0,1.0,1.0 ";
```

URL je (lahko) sestavljen iz več parametrov:

- `max-results` – število vrnjenih slik na eno stran.

- `start-index` – indeks slike. S pomočjo tega indeksa se lahko premikamo po straneh.
- `bbox` – področje pregledovanja (pravokotnik) iz katerega želimo slike. Pravokotnik določimo z vrednostmi `west` (levi rob), `south` (spodnji rob), `east` (desni rob), `north` (zgornji rob). Vrednosti morajo biti izražene v geografskih merah – stopinjah.
- `published-max` – zahtevamo lahko slike, ki so na PWA objavljene do določenega datuma.
- `published-min` – zahtevamo lahko slike, ki so na PWA objavljene od določenega datuma naprej.

Datuma morata biti podana v RFC 3339 formatu: `yyyy-MM-dd'T'hh:mm:ss`. Slike smo dobili, če smo izvedli

```
GphotoFeed gPhotoFeed = picasaService.getFeed(new URL(feedUrl),
  GphotoFeed.class);
List<PhotoEntry> returnList =
  gPhotoFeed.getEntries(PhotoEntry.class);
```

`getEntries()` funkcija vrne objekte tipa `PhotoEntry` v seznam `returnList`. `PhotoEntry` je razred, ki nam da večino potrebnih podatkov o fotografiji. Vsebuje metode:

- `getGphotoId()` – edinstven ključ fotografije katerega največja vrednost je lahko 2^{64},
- `getGeoLocation().getLatitude()` – zemljepisna širina,
- `getGeoLocation().getLongitude()` – zemljepisna dolžina,
- `getPublishedTime()` – čas, ko je slika objavljena na PWA ter
- `getHtmlLink().getHref()` – URL naslov slike na PWA.

Podpis metode, ki je sprejela potrebne parametre za pregledovanje določenega območja izgleda tako:

```
public boolean getPublicPhotos(double west, double south, double
  east, double north, Date publishedMax, int depth, long
  threadId, DatabaseManager databaseManager, Date publishedMin)
```

Metoda za argumente sprejme:

- območje iskanja (`west`, `south`, `east`, `north`),
- najmanjši in največji datum objave slik (`publishedMin`, `publishedMax`),
- globino (`depth`) – metoda se je klicala rekurzivno, poleg tega je treba zaradi

določenih težav omejiti iskanje – glej poglavje 4.1.3,

- oznako niti (`threadId`), ki je preiskala območje ter
- objekt preko katerega je možno shraniti slike v Geoserver podatkovno bazo (`databaseManager`).

Če je PWA Data API vrnila več kot 1000 slik iz zahtevanega območja, smo območje iskanja zmanjšali. Območje smo zmanjšali na 4 enake kvadrate in vsakič rekurzivno izvedli metodo `getPublicPhotos()` z novimi vrednostmi (območje iskanja in globina). V določenih primerih nismo smeli nadaljevati z manjšanjem področja (omenjeno v poglavju 4.1.3).

Aplikacija je ob prvem zagonu (ko ni bila shranjena še nobena slika) nastavila `publishedMin` parameter na vrednost `null`. To je pomenilo, da je aplikacija zajela vse slike do datuma, ki ga je predstavljal `publishedMax` parameter. Uporaba je bolj podrobno opisana v poglavju 3.4.

3.3.2.1. Razred SimplePhotoEntry

Razred `PhotoEntry`, ki se nahaja v Google Data knjižnicah, vsebuje ogromno podatkov, celo veliko preveč za zahteve naloge. Zato je aplikaciji pri malo daljšem testiranju zmanjkalo pomnilnika. Za rešitev tega problema smo izdelali razred `SimplePhotoEntry`, ki je vseboval le pomembne podatke za nalogo. Posledično se je to poznalo tudi na hitrejšem delovanju aplikacije.

3.3.2.2.Določanje pomembnosti fotografije

Fotografiji oz. zapisu smo morali nekako določiti prioriteto. Prioriteto lahko drugače imenujemo tudi pomembnost, še bolj razširjen izraz pa je popularnost. Kako zanimiva ali popularna je neka fotografija bi najlažje določili s številom ogledov ali pa s povprečno oceno, ki jo dajo uporabniki PWA. PWA API na žalost tega ne ponuja. To je tudi razumljivo, saj PWA ni storitev kot npr. Flickr[5] ali kakšna druga podobna storitev. PWA je namenjen uporabnikom, ki bi radi pokazali fotografije svojim prijateljem, ter ostalim, poleg tega pa bi mogoče radi imeli še varnostno kopijo svojih fotografij na spletu.

Za določanje popularnosti smo uporabili število komentarjev na fotografijo. Razred `PhotoEntry` ni ponujal nobene metode ali atributa, s katerim bi ugotovili število komentarjev. Zato smo opravili novo poizvedbo na PWA API, kjer smo kot odgovor dobili vir (ang. feed) tipa `PhotoFeed`, ki zna podati število komentarjev za sliko.

```
URL url = new URL(pe.getLink(Link.Rel.FEED, null).getHref());
PhotoFeed f = new PicasawebService("Get comments").getFeed(url,
  PhotoFeed.class);
int numberOfComments = f.getCommentCount().intValue();
```

Objekt `pe` je tipa `PhotoEntry`, ki se nahaja v seznamu slik (prva poizvedba na PWA API). Na ta način smo dobili število komentarjev le za eno fotografijo. S to zahtevo se je čas

[5] Spletna storitev za objavo fotografij.

izvajanja aplikacije zelo povečal, saj je potrebno opraviti novo poizvedbo za vsako fotografijo posebej.

3.3.3. Zapis slik v podatkovno bazo - Geoserver

Naloga zahteva, da se morajo fotografije shraniti v primerno podatkovno bazo, poleg tega pa smo želeli našo rešitev implementirati tudi v pravem produkcijskem okolju. Priložnost se nam je ponudila v podjetju, ki se ukvarja z geolokacijskimi storitvami. Tam se je uporabljala specializirana podatkovna baza z imenom Geoserver. Podatkovno bazo je odlikovala hitrost v okolju, kjer se je uporabljala. Tako smo se odločili uporabiti kar podatkovno bazo Geoserver.

Do omenjene podatkovne baze je bilo možno dostopati preko middleware ICE-a. ICE middleware je bolj podrobno opisan v poglavju 2.3.2. Slice definicija funkcije, ki doda fotografijo v bazo, je sledeča:

```
void addEntry(long entryId, int entryType, double latitude,
  double longitude, double priority, long createTime, long
  eventTime, string textData);
```

- `entryId` – ključ pod katerim bo zapis shranjen, največja vrednost je lahko 2^{32}
- `entryType` – oznaka/plast, pod katero bo zapis shranjen
- `latitude` – zemljepisna širina
- `longitude` – zemljepisna dolžina
- `priority` – prioriteta ali pomembnost zapisa
- `createTime` – čas nastanka fotografije
- `eventTime` – čas dogodka
- `textData` – poljuben tekst, ki lahko služi kot opis zapisa, najdaljša dolžina je 2040 znakov. V to polje smo podali URL fotografije.

Geoserver podatkovna baza se je izvajala na enakem računalniku kot je računalnik, na katerem smo razvijali in testirali samo aplikacijo. Potrebno je omeniti, da se je na računalniku izvajala testna podatkovna baza (in ne produkcijska), s katero smo lahko eksperimentirali. Če smo želeli dostopati do podatkovne baze, smo morali uporabiti Java razrede, ki smo jih dobili s prevajanjem Slice definicij (postopek omenjen v poglavju 2.3.2). Sledila je povezava z Geoserver podatkovno bazo:

```
Ice.Communicator ic = Ice.Util.initialize();
Ice.ObjectPrx base = ic.stringToProxy("GeoPointsOfInterest:
  tcp -h "+host+" -p "+port);
poi = GeoIce.PointsOfInterestPrxHelper.checkedCast(base);
```

in ko je bilo potrebno, še zapis v podatkovno bazo:

```
poi.addEntry(geosrvId, layer, latitude, longitude, priority,
  createTime, eventTime, jsonString);
```

Za operacije in komunikacijo s podatkovno bazo (dodajanje in brisanje fotografij, poizvedbe, itd.) smo napisali razred `DatabaseManager`, ki je delo ustrezno olajšal.

3.3.4. Varnostna kopija

Varnostno kopijo smo se odločili izdelati v obliki XML dokumentov. Za tak format smo se odločili zato, ker je ena najprimernejših oblik za izmenjavo informacij med različnimi vrstami računalnikov, aplikacij, itd. V XML dokument smo naenkrat shranili podatke o 1000-ih fotografijah. Aplikacija je hranila podatke o fotografijah v pomnilniku in jih zapisala le takrat, ko je prišla do vrednosti 1000. Na ta način smo zmanjšali dostop do diska in malce pridobili na hitrosti. Pisanje v XML dokument je zgledalo takole:

```
/* zacetek xml dokumenta */
private SAXTransformerFactory tf = (SAXTransformerFactory)
  SAXTransformerFactory.newInstance();

private TransformerHandler hd = tf.newTransformerHandler();
Transformer serializer = hd.getTransformer();
serializer.setOutputProperty(OutputKeys.ENCODING,"ISO-8859-1");
serializer.setOutputProperty(OutputKeys.INDENT,"yes");
hd.setResult(streamResult); //stream do datoteke
hd.startDocument();
AttributesImpl atts = new AttributesImpl();
hd.startElement("","","PICASAPHOTOS",atts);
```

Zgornji del programske kode je procedura, ki jo vedno izvršimo, ko ustvarjamo XML dokument. Sledi dodajanje elementa, ki se lahko izvrši večkrat – v našem primeru 1000x. Objekt `epe` je tipa `SimplePhotoEntry`, ki je opisan v poglavju 3.3.2.

```
/*   dodajanje elementa  */
atts.clear();
atts.addAttribute("","","PICASAID","CDATA", epe.getPicasaId());
hd.startElement("","","PHOTO",atts);
hd.characters(id.toCharArray(), 0, id.length());
hd.endElement("", "", "PHOTO");
```

Ko smo v XML dokument zapisali vse potrebne podatke, smo dokument še zaključili z naslednjima stavkoma:

```
/* konec xml dokumenta */
hd.endElement("","","PICASAPHOTOS");
hd.endDocument();
```

3.3.5. Posodobitev podatkovne baze

Posodobitev podatkovne baze z novimi fotografijami je potekala na podoben način kot prvotni zajem slik. Uporabila se je metoda `getPublicPhotos()`, ki je za `publishedMin` parameter dobila vrednost, ki smo jo v prvotnem zajemu določili pri parametru `publishedMax`. To vrednost mora seveda nastaviti uporabnik, ki uporablja aplikacijo. Na ta način smo dobili le novo objavljene slike na PWA od datuma `publishedMin`, ne pa tudi izbrisanih ali kako drugače spremenjenih obstoječih slik. Zakaj ne, je razloženo v poglavju 4.1.4.

3.4. Uporaba aplikacije

Kot je že bilo omenjeno, mora imeti uporabnik aplikacije osnovno znanje iz tovrstnega področja. Prav tako ni bilo zahteve po zahtevnem in lepem grafičnem vmesniku. Od uporabnika se torej pričakuje, da bo v aplikacijo vnesel podatke o območju, ki ga želi pregledati in da ima osnovno znanje o uporabi računalnika/aplikacij.

Uporabniku smo omogočili vnosna polja, kjer lahko vnese zahodno, južno, vzhodno in severno koordinato območja, ki ga želi pregledati. Na voljo sta še dve polji:

- `Minimum date` – fotografije dodane na PWA pred tem datumom, ne bodo zajete in
- `Maximum date` – fotografije dodane na PWA po tem datumu, ne bodo zajete

Ti dve polji sta na voljo iz več razlogov. Lahko se jih uporabi, če želimo omejiti iskanje, ali ga uporabljamo za posodobitev podatkovne baze. Uporabnik se lahko odloči, da bi rad najnovejše slike ali le slike iz določenega obdobja (npr. fotografije dodane okoli novega leta). V drugem primeru lahko uporabnik posodobi podatkovno bazo z najnovejšimi slikami, ki so bile dodane na PWA od zadnjega zajema tako, da

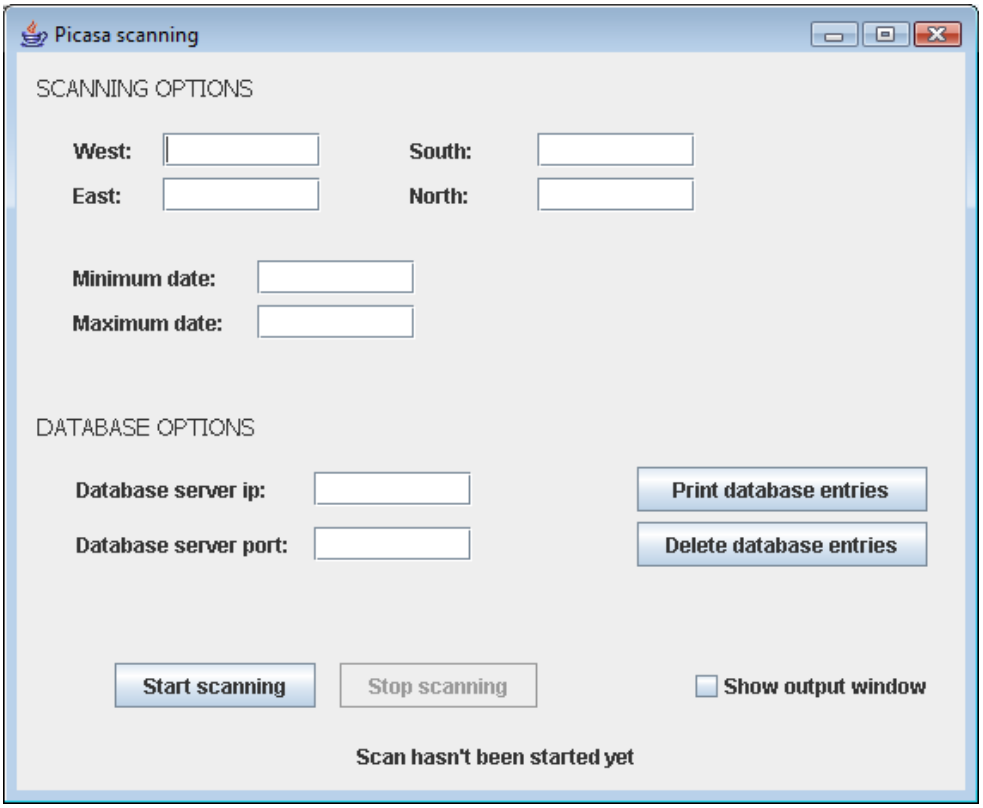
- v polje `Minimum date` vpiše datum, ki ga je pri zadnjem zajemu uporabil v polju `Maximum date` in
- polje `Maximum date` pusti prazno.

Potrebno je le še vedeti na katerem mrežnem naslovu (IP številka ali mrežno ime) in vratih teče Geoserver podatkovna baza. Ti vrednosti se vpiše v polji:

- `Database server ip` (mrežni naslov) in
- `Database server port` (vrata)

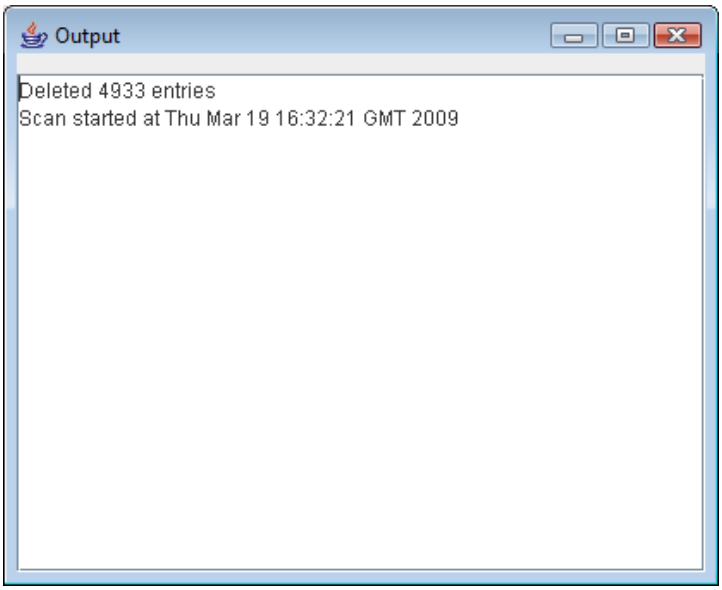
Ko so vnešeni vsi potrebni podatki, se pregledovanje lahko začne s pritiskom na gumb **`Start scanning`** in predhodno konča s pritiskom na gumb **`Stop scanning`**. Kaj aplikacija trenutno počne se lahko opazuje preko statusne oznake (ki sporoča koliko fotografij je

aplikacija do določenega trenutka zajela) ali preko **`Output`** okna (slika 9). To okno prikažemo, če označimo označevalno polje (ang. checkbox) **`Show output window`**. V oknu lahko vidimo, katero območje aplikacija trenutno preiskuje, morebitne napake ter sporočila, ki označujejo npr. začetek in konec pregledovanja.



Slika 8. Grafični vmesnik aplikacije za zajem fotografij

Ko se pregledovanje konča, je uporabnik obveščen tako preko statusne oznake (ang. label) kot tudi preko **`Output`** okna. Statusna labela je prikazana na dnu okna (slika 8).



Slika 9. Output okno, ki kaže sporočila aplikacije

Uporabnik ima možnost izpisati in zbrisati stare zapise v Geoserver podatkovni bazi. Prvotno smo to funkcijo dodali le zaradi enostavnejšega testiranja, vendar se je potlej izkazalo, da bo ta funkcija prav prišla tudi v okolju, kjer se je končana aplikacija preizkusila. Ob uporabi izpisa mora uporabnik še potrditi, da to res želi. To je varnostni mehanizem, ki je potreben, saj lahko izpisovanje traja zelo dolgo, odvisno od števila zapisov v podatkovni bazi. Enako velja za brisanje zapisov iz podatkovne baze. Uporabnik je obveščen o posledicah (zapisov ni mogoče povrniti, brisanje lahko traja dolgo) in mora potrditi svojo odločitev.

Slika 9 nam kaže, da je uporabnik ob zagonu aplikacije najprej izbrisal zapise iz podatkovne baze, nato pa začel s zajemanjem fotografij.

4. Analiza aplikacije

4.1. Težave pri izvedbi

4.1.1. Zajem slik

Če preberemo poglavje 3.3.2 bi na prvi pogled sklepali, da bi lahko izbrali naslednje parametre in takoj dobili vse želene fotografije, ki so označene z geografsko dolžino in širino:

- `max-results=` zelo visoka številka (večja kot število geooznačenih slik na PWA),
- `start-index=1`,
- `bbox=-180,-90,180,90` ter
- `published-max=` današnji datum (oz. datum pregledovanja).

Vendar temu ni tako, saj ima PWA API določene omejitve in lastnosti:

1) maksimalno število slik na eni strani je 500
2) maksimalno število vseh slik, ki jih PWA API vrne iz zahtevanega območja je 1000
3) nenavadno obnašanje pri mejnih vrednostih zahtevanega (`bbox`) področja npr. -90 in 90
4) `bbox` s koordinatami 0,0,0,0 vrne več kot 1000 slik

Zato je bilo potrebno sprejeti nekaj ukrepov:

- področja morajo biti dovolj majhna, da bo PWA API vrnil manj kot 1000 slik. Za to smo poskrbeli z manjšanjem področij, kar je opisano v prejšnjih poglavjih.
- pri področjih, ki se jih ne da dodatno pomanjšati bomo vzeli prvih 1000 slik, ki jih PWA API vrne. Taka področja so npr. 0,0,0,0 ali pa popularne turistične destinacije (Paris, New York...). Do tega pride, ker uporabnik PWA ne določi točnih koordinat svojih slik, ampak izbere samo kraj slikanja. V tem primeru PWA sam določi privzete koordinate za določen kraj.
- mejnim vrednostim se bomo izognili (-90, 90).

4.1.2. Zapis slik v podatkovno bazo Geoserver

Tukaj sta se pojavili dve težavi. Manjša težava je nastopila pri samem shranjevanju zapisov, kajti ključ slike iz PWA (`GphotoId`) je mnogo večji od največje vrednosti ključa, ki ga je znala sprejeti Geoserver baza (`entryId`). Težavo smo rešili tako, da smo uporabili SQLite podatkovno bazo za preslikavo med ključi. SQLite podatkovna baza je vsebovala tabelo z dvema stolpcema. Eden je vseboval `GphotoId` vrednosti, drugi pa vrednosti ključev, ki so se shranile v Geoserver bazo. Primer take tabele lahko vidimo v tabeli 2.

geoserverId	picasaId
1	5036746040772968722
2	5147857200211411906
3	5147855778577236322
4	5147848524377473234

Tabela 2: Stolpci SQL podatkovne baze za preslikavo med ključi

Druga težava je bila skrita napaka. Pri shranjevanju zapisov z enako dolžino in širino nekateri zapisi niso bili shranjeni. Geoserver podatkovno bazo odlikuje visoka hitrost delovanja, na račun tega pa obstaja omejitev števila zapisov za določeno zemljepisno točko. Težava je bila odpravljena po prijavi nenavadnega delovanja le–te.

4.1.3. Nepravilno in nepričakovano delovanje PWA Data API

PWA Data API ni vedno deloval kot bi moral in kot je bilo navedeno v dokumentaciji. Glavni razlog za to lahko pripišemo dejstvu, da je ob začetku dela, bil PWA Data API relativno nov izdelek (približno kakšno leto). Veliko hroščev je bilo še neodkritih, ker je bil PWA Data API takrat zelo malo uporabljen med razvijalci. Posledično tudi vse funkcije in zmogljivosti niso bile uporabljene v celoti.

Včasih smo po zahtevi na PWA dobili od strežnika odgovor `Internal server error – 500`. To se je rado zgodilo pri mejnih vrednostih zemljepisne širine (-90° in 90°), dogajalo pa se je tudi pri drugih območjih. Če se je to zgodilo, smo ponovili iskanje na istem območju. Hkrati smo s števcem opazovali, kolikokrat se je to že zgodilo. Če se je napaka zgodila na istem območju prevečkrat, smo območje preskočili in nadaljevali z naslednjim.

Omenili smo že, da je bilo potrebno v določenih primerih pri zajemanju slik območja manjšati. Tako smo pri razvijanju in testiranju opazili, da PWA Data API ne deluje pravilno pri zelo majhnih območjih. Če smo zahtevali slike iz območja, ki je bilo definirano recimo na sedem decimalk natančno, je lahko PWA Data API vseeno vrnil slike, ki se ne nahajajo v tem območju. Ni nam uspelo določiti zadnjega še veljavnega decimalnega mesta, saj je bilo včasih veljavno decimalno mesto na poziciji 5, 6 ali pa 7. Zato smo se odločili manjšati področja le na pet decimalnih mest natančno, shranili smo pa, kot že prej omenjeno, do prvih 1000 zapisov.

4.1.4. Posodobitev podatkovne baze

Kot je opisano v poglavju 3.3.5, aplikacija omogoča posodobitev podatkovne baze na omejen način. Omogoča le dodajanje slik v podatkovno bazo, ki so bile na PWA na novo dodane. Na ta način ne moremo dodati slik, ki so bile zasukane, ali celo izbrisati sliko v podatkovni bazi, ki je izbrisana na PWA. Po dokumentaciji PWA Data API naj bi bilo možno to ugotoviti z ustreznim klicem na PWA. Žal je tudi ta funkcija ena od funkcij, ki na PWA Data API ni delovala.

4.2. Možne izboljšave

Seveda je možno tudi našo aplikacijo izboljšati. Na nekaj možnih izboljšav smo naleteli že sami med pisanjem te naloge, marsikdo pa lahko najde še kakšno dodatno.

4.2.1. Obogatena varnostna kopija

Varnostna kopija, ki smo jo implementirali, vsebuje le podatke o fotografijah, ki so bili v času reševanja problema pomembni za nalogo. Če bi se v nekem produkcijskem okolju kasneje ugotovilo, da bi znal biti zanimiv še kakšen podatek o fotografiji (ki v prvotnem seznamu zahtev ni bil naveden), bi bilo potrebno vse slike še enkrat zajeti. Ni treba omenjati, da bi zato porabili ravno toliko časa kot pri prvem zajemu (če ne celo malce več zaradi novih dodanih fotografij).

Vse podatke, ki jih lahko izluščimo iz objekta PhotoEntry, lahko shranimo v XML dokument. Ker to spet terja procesorsko moč in pomnilniški prostor, bi lahko objekt `PhotoEntry` kar shranili v datoteko. Še bolje kot to, bi bilo, če bi tja shranili kar cel `PhotoFeed` objekt ali pa seznam, ki vsebuje objekte tipa `PhotoEntry`.

4.2.2. Nadomestna glavna podatkovna baza

Geoserver podatkovna baza ima svoje prednosti, predvsem hitro delovanje in transparentnost. Je pa poleg vsega tega zaprtega tipa in nedostopna javnosti. Njene prednosti lahko uporabi le podjetje samo. Namesto te podatkove baze se lahko uporabi marsikatera prostodostopna zastonjska alternativa. Lahko bi se recimo uporabila kar SQLite podatkovna baza, ki je v implementaciji rešitve problema služila le kot preslikovalna podatkova baza.

4.2.3. Določanje pomembnosti fotografije

V poglavju 3.3.2 je že omenjen proces določanja pomembnosti/popularnosti fotografije. Ker se čas izvajanja aplikacije s tem drastično podaljša, bi lahko aplikacijo zasnovali drugače (primerno za kakšen drug problem s podobnimi zahtevami). Aplikacija bi lahko delovala v dveh fazah:

- s prvim zagonom bi zajeli osnovne podatke kot je opisano v prejšnjih poglavjih, vendar brez ugotavljanja števila komentarjev za določeno fotografijo. Namesto tega bi določili število komentarjev za vsako fotografijo na vrednost nič.

- drugi zagon bi zajel število komentarjev oziroma določil število komentarjev za vsako fotografijo v podatkovni bazi in jo pri tem ustrezno ažuriral.

Tak način delovanja bi ustrezal nekomu, ki mu popularnost fotografije ni tako pomembna, zato bi se lahko izognil drugemu zagonu ali ga odložil za kasneje.

4.2.4. Pohitritev izvajanja aplikacije

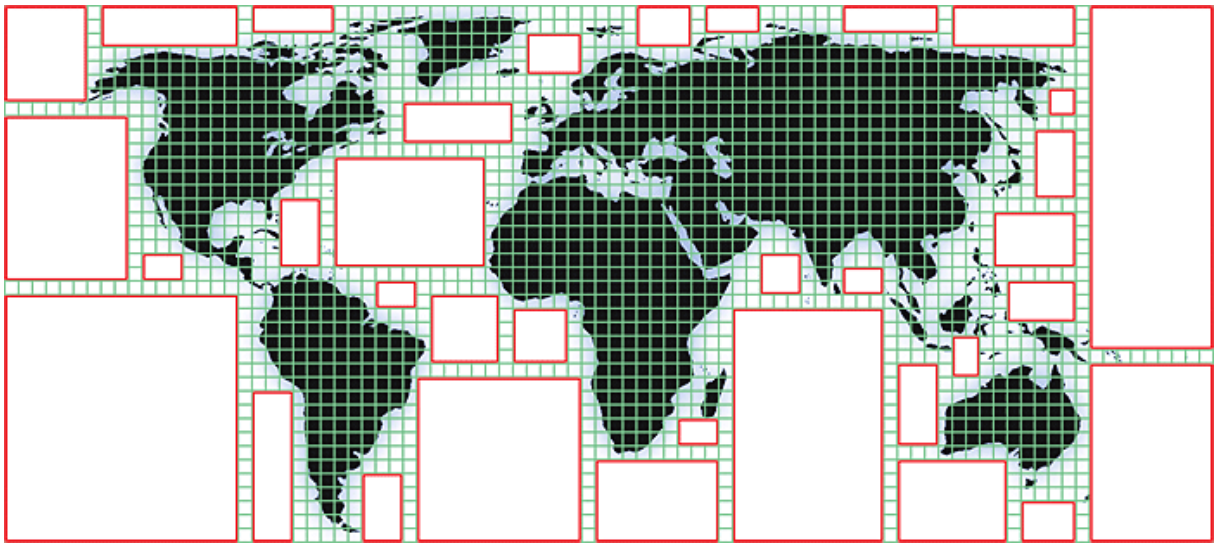
Izvajanje aplikacije lahko pohitrimo z različno velikimi območji pregledovanja, ali pa z zagonom več aplikacij naenkrat.

4.2.4.1. Različno velika območja glede na lokacijo pregledovanja

Aplikacija zajema slike tako, da pregleduje le en majhen del Zemlje v nekem trenutku (kvadrat s stranico dolgo 0.2°). Ko s tem območjem konča, nadaljuje z enako velikim

območjem, ki se nahaja takoj zraven. Ta način se obnese v primeru, ko območje pregledovanja pripada celini oziroma delu, kjer je malo slik. Ker je slik posnetih na oceanih (verjetno) zelo malo, bi lahko taka območja preskočili (če take fotografije naročniku niso posebej pomembne).

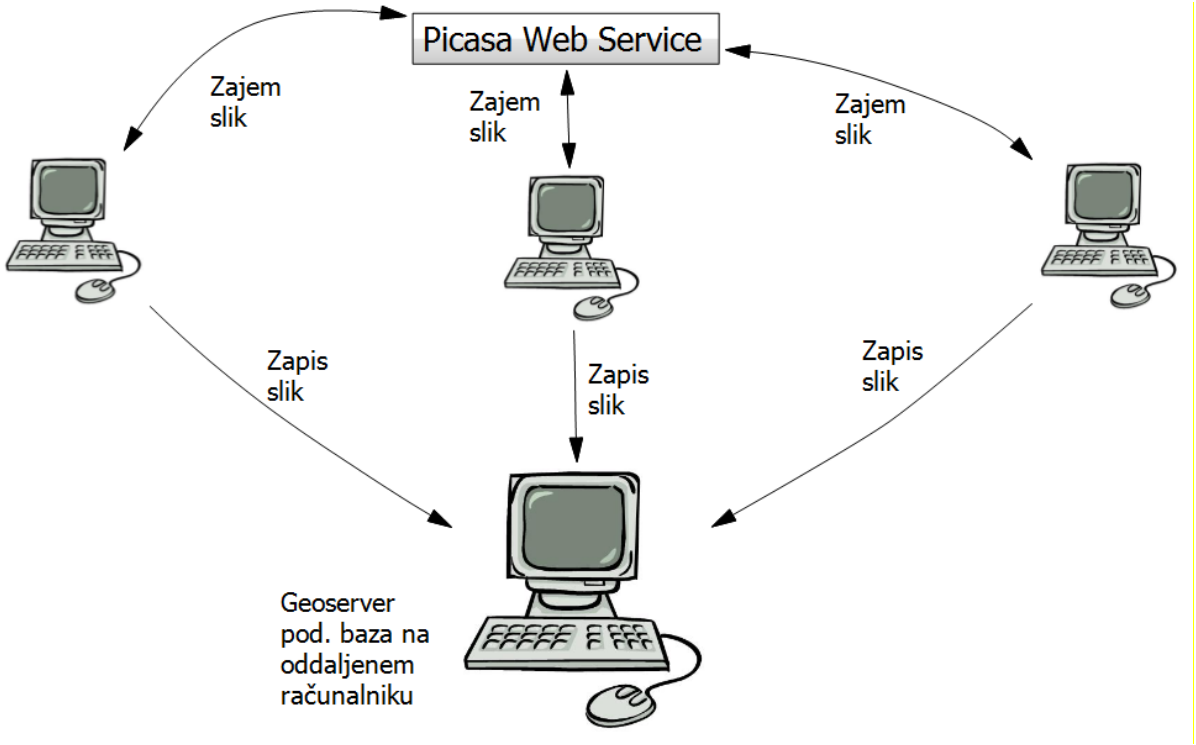
Aplikacija bi bila lahko zasnovana tako, da se območje, kjer je ocean preišče kot en velik pravokotnik. Hkrati bi aplikaciji dovolili večjo globino (`depth`) pri takem območju. Verjetnost, da neko majhno območje vsebuje več kot 1000 slik je majhna, večjo globino pa bi morali določiti predvsem zato, da lahko pridemo do morebitnega območja (otoka), ki vsebuje veliko slik. Ostala (oceanska) podobmočja, ki ne vsebujejo veliko slik, bi tako hitro obdelali. Za kopenska območja lahko pustimo enak sistem pregledovanja kot je bilo realizirano. Na sliki spodaj so območja z zelo majhnim številom slik označena z rdečimi pravokotniki.



Slika 10. Različno velika območja glede na lokacijo pregledovanja

4.2.4.2. Zagon več instanc aplikacije

Naslednji način pregledovanja bi lahko realizirali tudi tako, da bi zagnali več instanc aplikacije, kjer bi se vsaka aplikacija izvajala na svojem računalniku in s svojo širokopasovno internetno povezavo (slika 11). Zajete slike bi lahko še vedno poslali Geoserver podatkovni bazi, saj le–ta (lahko) deluje na oddaljenem strežniku. V tem primeru bi bilo potrebno opraviti še nekaj sprememb. Realizirana aplikacija deluje tako, da ustvari preslikovalno podatkovno bazo kar pri sebi. Če bi se vsaka aplikacija izvajala na svojem računalniku, bi morali ta del preseliti na strežnik, kjer bi se izvajal Geoserver. To je potrebno, da ne prihaja do konfliktov pri enolični identifikaciji slik.



Slika 11. Zajemanje slik z več aplikacijami/računalniki

Na ta način bi odpravili ozko grlo pri samem zajemanju slik, vendar to prinese ostale pomankljivosti:

- dodatna strojna oprema – več računalnikov in
- več širokopasovnih internetnih dostopov.

Pomankljivosti se kažejo seveda v večjih stroških, večji porabi prostora, ipd.

Kot vidimo ima naša aplikacija veliko možnosti za izboljšanje delovanja. Mnogo idej se je pojavilo med samim pisanjem diplomske naloge, nekaj pa že ob realizaciji nekaterih funkcionalnosti aplikacije. Ker je po metodologiji XP delujoča koda bolj pomembna kot vrhunska koda, smo se odločili optimizacijo opraviti na koncu. Možnih izboljšav je kar nekaj, vendar bi zanje porabili veliko več časa, kot ga predvideva diplomska naloga.

4.3. Analiza uporabe v praksi

Aplikacija je v območju z

- zahodno zemljepisno dolžino 0.1°,
- vzhodno zemljepisno dolžino 180°,
- južno zemljepisno širino -80° in

- severno zemljepisno širino -0.1°,

kar je malce manj kot čertrtina Zemlje, zajela 255.163 fotografij. V tem območju se gotovo nahaja več fotografij, saj vseh ni bilo možno zajeti zaradi težav omenjenih v prejšnjih poglavjih. Potrebno je poudariti, da so zajete samo fotografije s podatkom o zemljepisni dolžini in širini, ki so bile dodane na PWA do datuma 14.08.2008. Zajemanje je trajalo 1 dan 21 ur in 45 minut. Aplikacija kot tudi Geoserver podatkovna baza sta bila zagnana v virtualnem računalniku z naslednjimi lastnostmi:

- 1.2 GHz procesor,
- 256 MB pomnilnik in
- 8 Mbit/s internetna povezava.

Aplikacije žal ne moremo primerjati z drugimi, ker nismo nobene našli. Naša rešitev je še preveč enostavna, da bi jo ponudili na trg. Vsekakor pa je rešitev, ki je dovolj učinkovita in za zajem fotografij iz več spletnih storitev lahko odskočna deska za še bolj zanimiv izdelek.

5. Sklepne ugotovitve

V okviru diplomske naloge smo razvili aplikacijo za zajem fotografij iz storitve Picasa Web Albums. Aplikacijo smo preizkusili na četrtini Zemljine površine. Glede na rezultate menimo, da smo problem uspešno rešili, čeprav vseh zahtev ni bilo možno izpolniti. Izbrali smo primerno metodologijo, ki smo jo bili deloma tudi prisiljeni izbrati zaradi omejenosti s sredstvi in s časom. Izbira programskega jezika niti ni bila tako pomembna, saj pomembne razlike v hitrosti, pri današnjih zmogljivih računalnikih, ni. Ena od študij celo trdi, da delovanje aplikacije največ časa porabi pri vhodno-izhodnih operacijah.

Med razvojem aplikacije nismo imeli velikih težav. Vse so bile relativno hitro prebrodene, saj smo imeli ob strani veliko bolj izkušene razvijalce, ki so mi znali svetovati.

Če bi problem reševali še enkrat, bi bila rešitev mnogo učinkovitejša/zmogljivejša iz dveh razlogov:

- veliko idej za možne izboljšave smo dobili v času pisanja diplomske naloge ter
- PWA API ima sedaj odpravljenih že veliko več hroščev, verjetno je dodana tudi kakšna nova funkcija.

Če bi bilo treba aplikacijo nadgraditi z določenimi funkcionalnostmi, menimo, da ne bi imeli večjih težav, saj smo se držali pravil/standardov kodiranja. Programsko kodo smo pisali razumljivo in jo bogatili s potrebnimi komentarji. Ob razvijanju aplikacije smo utrdili znanje programiranja, ki smo ga pridobili na fakulteti in v okviru študentskega dela. Prav tako smo pridobili nekaj novega znanja iz področja izbiranja metodologij in spletnih storitev, ki ga prej nismo imeli.

Tema diplomske naloge je vsekakor zanimiva za današnji čas, ko se doživlja razcvet lokacijskih storitev. Poleg same vsebine nekega dogodka sta pomembnejša tudi kraj in čas tega, vsebino dogodka pa najlažje prikažemo s fotografijo ali video posnetkom. Nad zajetimi slikami bi lahko opravili tudi marsikatero statistično raziskavo. Na podlagi števila zajetih fotografij (za katere ni nujno, da so geooznačene), bi lahko npr. ugotovili, koliko so digitalni fotoaparati razširjeni, kam in kdaj turisti najbolj potujejo, in podobno.

6. Viri

[1] (2008) M. Bajec, M. Krisper, Agilne metodologije razvoja informacijskih sistemov. Dostopno na: http://infolab.fri.uni-lj.si/marko/CRP2001/Clanki/Agilne%20Metodologije%20Razvoja%20IS.pdf

[2] S. Divjak, Middleware, Študijsko gradivo za uporabo pri predmetu Sistemska programska oprema, Ljubljana 2008, str. 2 – 5.

[3] I. Rožanc, Ekstremno programiranje (XP), Študijsko gradivo za uporabo pri predmetu Razvoj programskih sistemov 2, Ljubljana 2008, str. 27 – 45.

[4] (2009) Eclipse, Dostopno na: http://www.eclipse.org/

[5] (2009) Google Data APIs – pregled. Dostopno na: http://code.google.com/apis/gdata/overview.html

[6] (2009) Kaj je Java? Dostopno na: http://java.about.com/od/gettingstarted/a/whatisjava.htm

[7] (2009) Model – View – Controller paradigma. Dostopno na: http://borutb.blogspot.com/2005/12/model-view-controller-paradigma.html

[8] (2009) O Java tehnologiji. Dostopno na: http://www.sun.com/java/about/

[9] (2008) Picasa Web Albums. Dostopno na: http://picasaweb.google.com/

[10] (2008) Picasa Web Album Data API. Dostopno na: http://code.google.com/apis/picasaweb/overview.html

[11] (2008) SQLite. Dostopno na: http://www.sqlite.org/

[12] (2008) Uporaba agilnih metodologij pri razvoju J2EE projektov. Dostopno na http://cot.uni-mb.si/cotl/pomlad2004/brajak.html

[13] (2008) Vlada Republike Slovenije, Center Vlade RS za informatiko, Enotna metodologija razvoja informacijskih sistemov. Dostopno na: http://www2.gov.si/mju/emris.nsf/Emris?OpenFrameSet

[14] (2008) Zeroc ICE. Dostopno na: http://www.zeroc.com/

[15] (2009) XML. Dostopno na: http://sl.wikipedia.org/wiki/XML