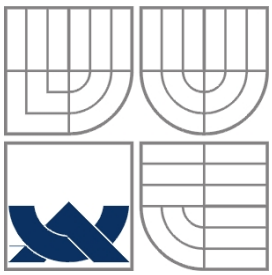
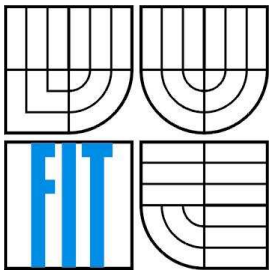


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

USER MOBILE APPLICATION WITH LOCATION-SENSING ASPECTS

UŽÍVATEĽSKÁ APLIKÁCIA PRE MOBILNÉ ZARIADENIA S LOKALIZAČNÝMI ASPEKTAMI

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — PETER MARCELY
AUTHOR

VEDOUCÍ PRÁCE — DOC. DR. ING. JAN ČERNOCKÝ
SUPERVISOR

BRNO 2009

Abstrakt

Táto práca sa zaoberá analýzou, návrhom a implementáciou sieťovej aplikácie napísanej v Jave. Táto aplikácia umožnuje klientom v uživateľskom rozhraní, založenom na Google Maps API a JavaScripte, sledovať pozíciu ich priateľov a ostatných užívateľov pripojených prostredníctvom 7DS do danej bezdrôtovej siete. V uživateľskom prostredí je možné k určitému miestu pripojiť galériu obrázkov. Užívatelia sa môžu pripojiť na komunitný server Facebook, čím získaju prístup k importovaniu a exportovaniu fotografií z tohto komunitného servra.

Abstract

This work deals with analysis, design and implementation of network application written in Java. User's front-end of this application, which is based on Google Maps API and JavaScript, allows following the position of their friends and other users connected to network by 7DS. There is a possibility to put pictures to a specific location on a map. Users can also connect to community network Facebook what allows them to import and export pictures from this community network.

Klíčová slova

Java, JavaScript, Google Mapy, Facebook, 7DS, Geolokalizácia, Fotogaléria

Keywords

Java, JavaScript, Google Maps, Facebook, 7DS, Geolocalisation, Photogallery

Citace

Marcely Peter: User mobile application with Location-sensing aspects, Brno, FIT VUT v Brně, 2009

User mobile application with Location-sensing aspects

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením prof. Marie Papadopouli a doc. Černockého.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Peter Marcely
30.4.2009

Poděkování

I would like to thank to my supervisors prof. Maria Papadopouli and doc. Černocký who advised and allowed me to work on this project. Also, I would like to thank to all professors and employers of the university who allowed me to study abroad at the University of Crete and to my family for their support.

Contents

1 Introduction

When I got the opportunity to study and develop my Bachelor thesis abroad, I decided to take advantage of this unique chance. As I am mostly interested in computer networks, I was trying to find a project which can allow me to improve my skills. Photojournal social network extension at the University of Crete under the supervision of prof. Maria Papadopouli became that project.

Geolocalisation and community networks like Facebook are fields in computer science which development is enormous. Idea of the project was to combine these two subjects and create a novel network application with localization and community aspects. As there was a project called Photojournal at the University of Crete, it allowed me to have a great foundation-stone, on which I could build up this idea.

In the second chapter, there is an analysis of all projects and main parts which lead to Photojournal extension. 7DS offers its ability to share the information within different types of network, Photojournal its structure for good start and Facebook its social network along with its API.

In the third chapter, we will be going through used technologies. Java provides the inter-operability on different types of operating system, JavaScript and AJAX give us the possibility to change the web page in browser dynamically without changing layout and structure. The storage of data of users and their photo galleries is made by XML. Google Maps are used for embedding the map into Photojournal client and Facebook API for implementing the functionality of this social community network.

Next chapter is project design. This chapter explains the design of Photojournal, following by design Photojournal's extension including user's positioning, facebook's gallery management and simple chat.

In fifth chapter, the implementation of this previous design is written. The diagram and the tables shows the procedures running between parts of Photojournal application in detail.

Measurements and model case are described in sixth chapter.

The last chapter is conclusion, which includes the summary of the project and it possible extension.

2 Analysis

As my application is based on Photojournal which is based on 7DS, the best will be to describe the 7DS architecture.

2.1 7DS

7DS [1] is an architecture that allows peers to share information. Let us assume that we have a wireless ad-hoc network with many connected peers, some of them have a connection to the Internet and some of them have not. When a user A does not have a proper connection to the Internet, he will send a broadcast message called query to all his neighbours asking for the information from the Internet. All the neighbours try to search the information in their cache, and those whose cache contains this information respond back sending desired information.

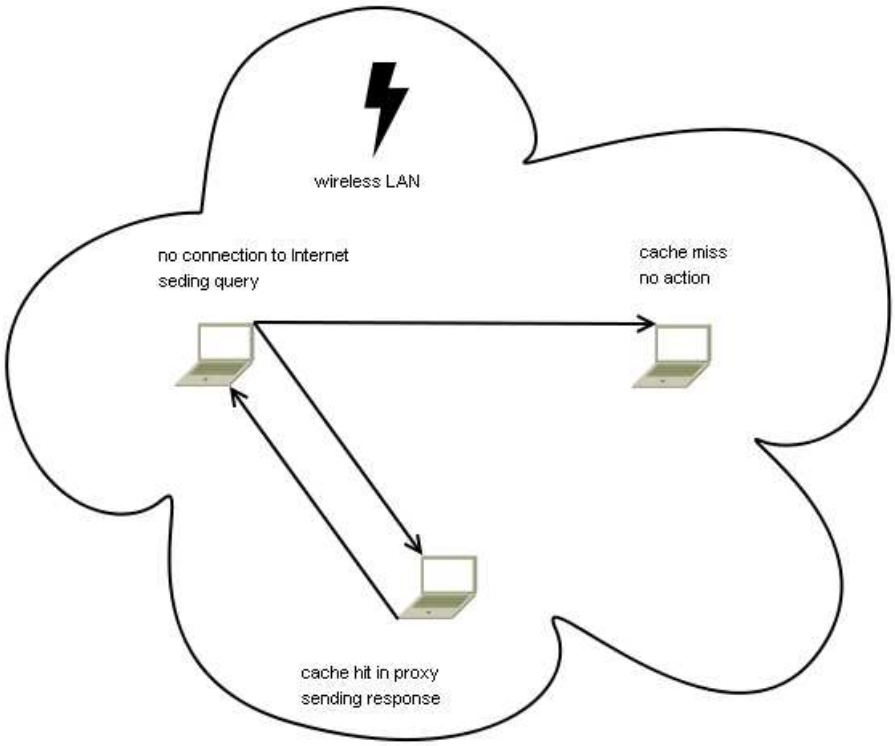


Figure 2.1: Example of 7DS structure [1]

Figure 2.1 shows the basic principle of 7DS. The implementation is made of proxy servers which are caching the information (usually web pages) while the user is connected to the Internet.

2.2 Photojournal

Photojournal [1] is a novel application which uses 7DS in order to share the information within peers connected to wireless network. Photojournal uses Google Maps API in order to allow the users to create, manage, comment and share their photos placed on the map. It is a client-server application.

Server which runs on Java is using 7DS to load the parts of the map even when the user is not connected to the Internet. It also stores information about the user and photos in an XML file, communicates with other Photojournal users and exchanges the galleries and comments on photos placed on the map.

Client is running in ordinary web browser using proxy server of his local Photojournal server which makes the interaction between him and server. The web site is coded in JavaScript, and is using AJAX to create the requests from client and receive the responds from server.

The functionality is as follows. User can place pictures from his hard drive and notes into the map, make them public or private, view, edit, delete and comment them. When there are more connected users using this application, he can search at the selected area and try to find the pictures and notes shared by these users.

Photojournal also expects the position from extern application using some position mechanism, either CLS or GPS.

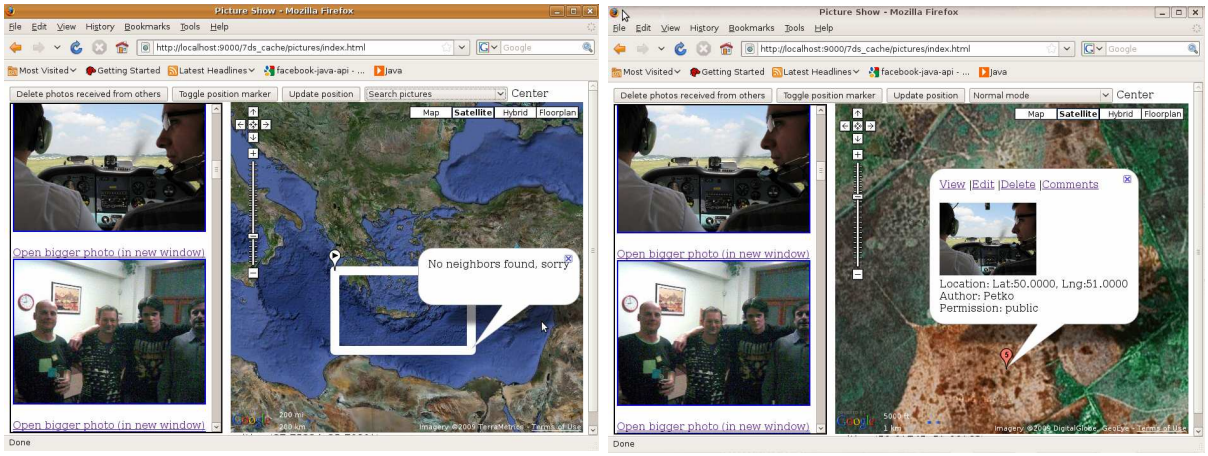


Figure 2.2: Examples of Photojournal application

2.3 Social networks

Social network [2] is a structure (group) of nodes (people) that has some similar characteristics. Internet social network services are focused on online communication between people who have common interests, opinions or visions. These services also group people working for the same company, living in the same area, or people who attended same schools. People are willing to share various information from comments to photos and videos.

2.3.1 History

Social network services were formed since the foundation of Internet. Usenet, ARPANET or bulletin board services are identified as early efforts to support social networking. The first social networking websites were Theglobe.com and Geocities in 1994, and Tripod and Classmates.com aimed to keep bound between classmates in 1995. These communities bring people to interact with each other in chat rooms and share ideas and topics of interests through personal homepages. SixDegrees.com, founded in 1997, has as the first implemented the functionality which can be considered as a basic type of social network service nowadays. It provided creating of user profiles, managing friend list and sending messages to other users. Friendster, MySpace and Bebo [3] were the top three networks between the years 2002 and 2004 until MySpace began to rule and had even more page hits than Google in that time.

2.3.2 Facebook

In 2004, Mark Zuckerberg, student of Harvard University, founded The Facebook as a social networking website which was aimed for students of Harvard. After its success Facebook [4] step by step expanded to all universities in United States and Canada and was also opened for employees of large companies such as Apple or Microsoft. In the meanwhile, its name was changed from The Facebook into just Facebook after purchasing domain facebook.com for 200.000 dollars. After 26th September 2006, everyone over the age of 13 years and with valid email address can join Facebook.

Now, Facebook is a social networking web service, used by 200 millions of users from all over the world. After registration and filling his private information, each user can share in his network (within his friends) his ideas, comments, and photos, meet new people and expand his friend's network. Facebook can join together different groups of people who initially do not know each other. As a great example I can mention students studying abroad in the same city.

Despite its criticism for issues considering the privacy of users, Facebook is one of larges and most growing social networking website these days.

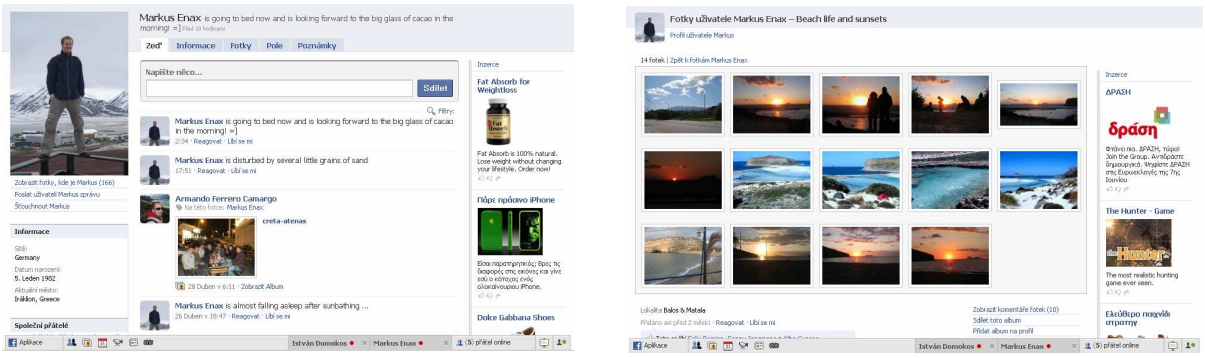


Figure 2.3: Examples of Facebook application

2.4 Google Maps

Google Maps allow the developers of web pages using JavaScript embed Google Maps in their own web pages. It does not depend on the fact, whether the web page is publicly available on the Internet or not. The only need is the connection to the Internet, or as in our case, 7DS can provide sharing maps within several connected users in wireless network.

Google Maps offer many different views, from basic map view to terrain and satellite views. Users are also able to plan their journey using Google Maps. It is widely spread all over the website of Internet. For example, when a company wants to show to their customer, where they can find it, the company will put embed Google maps into their sites with placemark of their location. User can then easily change the views and position on the map, and easily find the company.

Photojournal takes the advantages of Google Maps using its greatly supported and documented API. As the API key from Google website is requested, we had to sing up for it. Since is it absolutely free, there is not any problem. Then only one line of code is needed to include the Google Maps into Photojournal client and more preferences are available through JavaScript code.

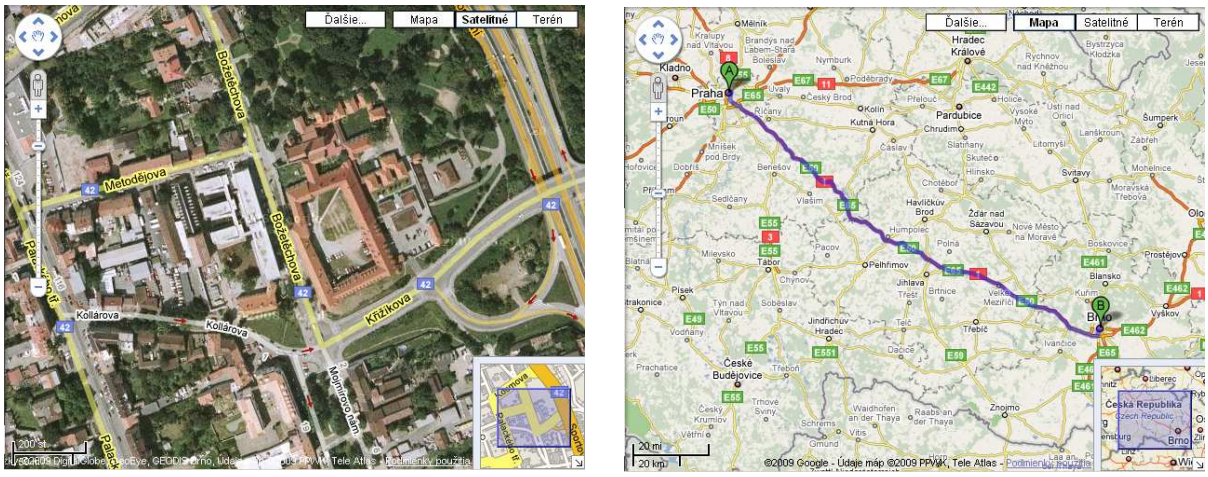


Figure 2.4: Examples of Google Maps

2.5 Photojournal social networking extension

Photojournal just fits for idea to create an application where users can see other users dynamically on the map. In order to make the project more interesting and more appropriate for Bachelor thesis, we decided to extend the project to be able communicate with social networking service Facebook.

This extension will give Photojournal functionality to load user's friend, showing their position on the map. These friends are expected to be in the same wireless network. Because of the fact that Facebook is also greatly used as a photo exchange server, it will be shame not to implement some photo gallery extension between Photojournal and Facebook. User will be able to export his photos

from Photojournal while adding the pictures to specific place and import the whole galleries from Facebook along with their positions. These imported galleries will be put to the specific place on the map and downloaded into local directory. We will try to manage comments and set the user's status message in Facebook.

The last missing functionality is chat implemented as broadcasting window, where users can send messages to all other connected peers.

3 Technologies

Photojournal combines many elements of computer science technologies from 7DS, wireless LAN, location-sensing to community networks. There are several programming languages and API needed to build such application and are described below.

3.1 Java

Java [5, 6] is one of the most used programming languages in the world. Java is object-oriented structured imperative programming language originally developed at Sun Microsystems and released in 1995. The programming language is similar to C++, but it has simpler object model and fewer low-level facilities. As it is compiled into byte code, it can run on and every Java Virtual Machine, so it is not depended on the computer architecture or operating system.

The whole Photojournal server is based on Java. It takes all its advantages which means that it should be theoretically able to run on every mobile device.

3.2 JavaScript

JavaScript [7] is a scripting language primary used as a client-side JavaScript for the development of dynamic websites. JavaScript, as it should be obvious, has some similarities with programming language Java. But the truth is, that JavaScript was influenced by many languages, and it is mostly derived from the Self and Scheme programming languages.

JavaScript is good choice for Photojournal, as it allows dynamic changes in the structure of web page which means the user do not need to refresh the web page after every change (new pictures, comments, etc.). The only problem is that basic JavaScript does not have the functionality to communicate with other running application, in our case server-side of Photojournal.

3.2.1 AJAX

AJAX [8] is shorthand for Asynchronous JavaScript and XML. It is fresh and still more popular technology firstly publicly mentioned in 2005. It is a group of web development tools used to create interactive web applications. AJAX allows web browsers to make a request to a server-side of application and receive data from the server asynchronously in the background without changing the layout and behavior of the current web page. Data is received by XMLHttpRequest object though the response does not need to be pure XML, HTML or plain text can be used too.

AJAX provides Photojournal the ability to communicate between its client’s and server’s side asynchronously which has a great impact to the interactivity of the whole process. User does not need

to take care of any web page navigation elements, as the web page remains always the same and only the dynamic elements like JavaScript and AJAX are responsible for the changes on the site.

3.3 XML

XML, eXtensible Markup Language [9], is a markup language which was developed and standardized by World Wide Web Consortium. It can store a wide spectrum of different types of data. The language is mainly used for exchange of data between applications and for publishing documents. The original language HTML was not appropriate for this usage due to its robustness and not very strict syntax. XML has no predefined tags, instead of HTML, and the syntax is much stricter.

XML can be used for the exchange of information between server and client of Photojournal, but the data are in most cases very simple so the usage of HTML language with just 2 main tags, html and body, is preferred. When we realize that XML has no predefined tags, we could even say, that we are using XML for the client-server communication.

Next field where the usage of XML is very helpful is storing data while the application is not running. We will store the information about photo galleries and its locations, user's information, list of facebook friends in XML file for the easy further usage.

3.4 Google Maps API

Let us describe initializing, setting the properties and running Google Maps in Photojournal client with sample parts of JavaScript code.

All the JavaScript code is usually put into the head of the web page. This line includes all the Google Maps API [10] code into our page. The version of API can vary due to the desired version. When version 2 is set, Google will provide us the newest version of API. Key is the API key aimed for developers.

```
<script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=abc"
        type="text/javascript"></script>
```

This part of code called document object model (DOM) has to be in the body part of HTML web page. It is a place on the display where the map is going to be located. The height and width can be easily set up.

```
<div id="map_canvas" style="width: 500px; height: 300px"></div>
```

This function in the head of web page shows the basic initialization of Google maps. If the browser is compatible with Google Maps, it will put the map to an element called map_canvas which was set in the previous step. Latitude and longitude are used for geographic localization of each place on the Earth. In this initialization these two values are user for setting the center of the map along with the level of zoom.

```
function initialize() {
   if (GBrowserIsCompatible()) {
     var map = new GMap2(document.getElementById("map_canvas"));
     map.setCenter(new GLatLng(34.4319, -124.1214), 11);
   }
}
```

The last necessary statement is as follows. After all elements of HTML web page are loaded, the onLoad event handler is called in body in order to initialize the map_canvas window. On unload there is used function which correctly closes the map and prevents memory leaks.

```
<body onload="initialize()" onunload="GUnload()">
```

These are the main and basic steps to make the Google Maps appear on the web page. There are more properties, events and utilities which can expand the usage of Google Maps. As it was mentioned before, there are more types of map views. Table 3.1 shows their utilization and commands the way how to change the type.

```
map.setMapType(G_SATELLITE_MAP);
```

Map	**Description**
G_NORMAL_MAP	Displays the default road map view.
G_SATELLITE_MAP	Displays Google Earth satellite images.
G_HYBRID_MAP	Displays a mixture of normal and satellite views.
G_DEFAULT_MAP_TYPES	Contains an array of the above three types, useful for iterative processing.
G_PHYSICAL_MAP	Displays a physical map based on terrain information.

Table 3.1: Possible map views of Google Maps [10]

Markers and info boxes are essential parts of Photojournal's idea. Users can put their pictures at particular places on the map and see their friends marked with the caption too. This example shows the window that is going to be opened in the middle of map. The simple "Hello, world" message will

be displayed, but instead of it, more sophisticated HTML code can be shown in the Info Window. Also the position can be changed by class GLatLng with constructor setting the Latitude and Longitude.

```
map.openInfoWindow(map.getCenter(), document.createTextNode("Hello, world"));
```

3.5 Facebook API

Facebook API [11] is probably the most complicated of all mentioned technologies. Its development is enormous, so the API goes through many changes. What is more, the Facebook API for Java is unsupported since May 2008 by Facebook developers. Luckily, its development continues as an open-source, as there are some complications in its use. The version 2.0.3 is appropriate for Desktop applications, although there are several newer versions of the API, but the login process always ends up with error message from Facebook.

Facebook API includes many tools, but only two of them we are going to need. The API which communicate with the application by GET or POST HTTP requests and responds and the Facebook query language (FQL) which provides the opportunity to form and execute queries and receiving the actual results.

In Facebook Java API, there are more choices of the choosing the client. The reason of selection XML client is that it is easy to handle and there are no complications with reading the results of more structured and huge queries. Here are the examples of request by calling a function in Java and response from Facebook. The unique identifiers of user's friend are received in structured XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<friends_get_response xmlns=http://api.facebook.com/1.0/
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://api.facebook.com/1.0/
http://api.facebook.com/1.0/facebook.xsd" list="true">
  <uid>222333</uid>
  <uid>1240079</uid>
</friends_get_response>
```

In table 3.2, there are main functions needed for getting the operability of Facebook. Names of the functions are overtaken from the original Facebook API documentation, the actual names in open-source Facebook Java API are slightly different.

API method	Description
auth.createToken	Creates an auth_token to be passed in as a parameter to login.php and then to auth.getSession after the user has logged in.
auth.getSession	Returns the session key linked to an auth_token, as returned by auth.createToken. More details are in the next chapter.
fql.query	Evaluates an FQL (Facebook Query Language) query and returns, in our case, XML response.
friends.get	Gets the unique identifiers of all user's friends.
friends.areFriends	Checks if a pair of people are friends.
photos.createAlbum	Creates and returns a new album owned by the current session user.
photos.getAlbums	Returns metadata about all of the photo albums uploaded by the specified user.
photos.upload	Uploads a photo owned by the current session user and returns the new photo.
users.setStatus	Sets the status message on Facebook.

Table 3.2: Important methods of Facebook API [11]

3.5.1 FQL

Facebook Query Language (FQL) [12] is SQL-style query language used for evaluating and executing queries. The application calls function fql.query with specified query and the result, similar to other API functions. Examples of FQL queries can be found below. The first example will return names of users with specified unique identifiers. The second gets the name of the groups of which is the user with uid 123456 member.

```
SELECT name, pic FROM user WHERE uid=211031 OR uid=4801660
```

```
SELECT name FROM group WHERE gid IN (SELECT gid FROM group_member WHERE uid
= ''123456'')
```

3.5.2 Login process

The login process in the desktop application is more complicated, so it needs to be well explained. The first step in order to create a desktop application which cooperates with Facebook is the registration of the application. Similar to Google Maps API, we are going to receive an API key representing our application and Application secret key authenticating our application. These key will be further used in the source code. The last preference in Facebook developers interface is the type of

our application. Facebook offers a great for website applications, but instead, Desktop application needs to be set up.

Now, when the preferences are adjusted and we have valid keys, the allowance of usage the application by user is necessary. If the user wants to use Photojournal extension, he must visit the page below and allow the access to the application. Facebook takes precautions against malicious application which could steal some private information without knowledge of the user.

```
http://www.facebook.com/login.php?api_key=d3a38c9e64efdca378a8dd5211731461&v=1.0&canvas=true
```

Now there is nothing standing against the login process in the application. Firstly, we need to call `auth.createToken` which will generate token used in the link below.

```
http://apps.facebook.com/d3a38c9e64efdca378a8dd5211731461/?auth_token=d793550e379a3d8a534f9cc51e76bc96&installed=1
```

User is going to fill his contact information in the web browser in login page of Facebook and after successful login, calling the function auth.getSession with previously generated token will return correct session which will provide the application valid communication channel with Facebook. As Photojournal is dynamically reloading web page without any directions to other pages, we have to think of other way of login that is using web browser. This procedure is described in the Design chapter.

After all the steps have been successfully executed, application uses its session until it does not call the auth.expireSession or just looses the current session value.

3.5.3 Extended permissions

The applications are limited to some restrictions which can be allowed by the user. The extended permission can be given at specific web page by user. The link below shows an example which will provide the application permission to set the status message. Table 3.3 shows some of the possible extended permissions, their codes and descriptions.

```
http://www.facebook.com/authorize.php?api_key=d3a38c9e64efdca378a8dd5211731461&v=1.0&ext_perm=status_update
```

Permission	Description
`offline_access`	Grants an application access to user data when the user is offline or does not have an active session. Unluckily, this permission is not supported by Facebook Java API in specific used version.
`sms`	Allows a mobile application to send messages to the user and respond to messages from the user via text message.
`status_update`	Grants Photojournal the ability to update a user's status message.
`photo_upload`	Allows the application to use the mechanism for uploading photos, and tagging them.
`video_upload`	Grants permission to the application to upload video.

Table 3.3: Examples of extended permissions of Facebook application [11]

4 Project design

Let us focus on the design of the Photojournal. Firstly, we will describe the Photojournal's structure and design, then will go through design of Facebook's extension and finish with chat messages.

4.1 Photojournal

As it was mentioned before, Photojournal is taking advantage of 7DS architecture. That is why the transfers of parts of the map between the nodes (users) should be obvious. What is not obvious is how the pictures are shared between users and the storage of data. In next chapters, these functionalities along with its storage in XML structure in main file `db.xml` will be described.

Figure 4.1 shows how the Photojournal's client and server interact with other all elements and devices.

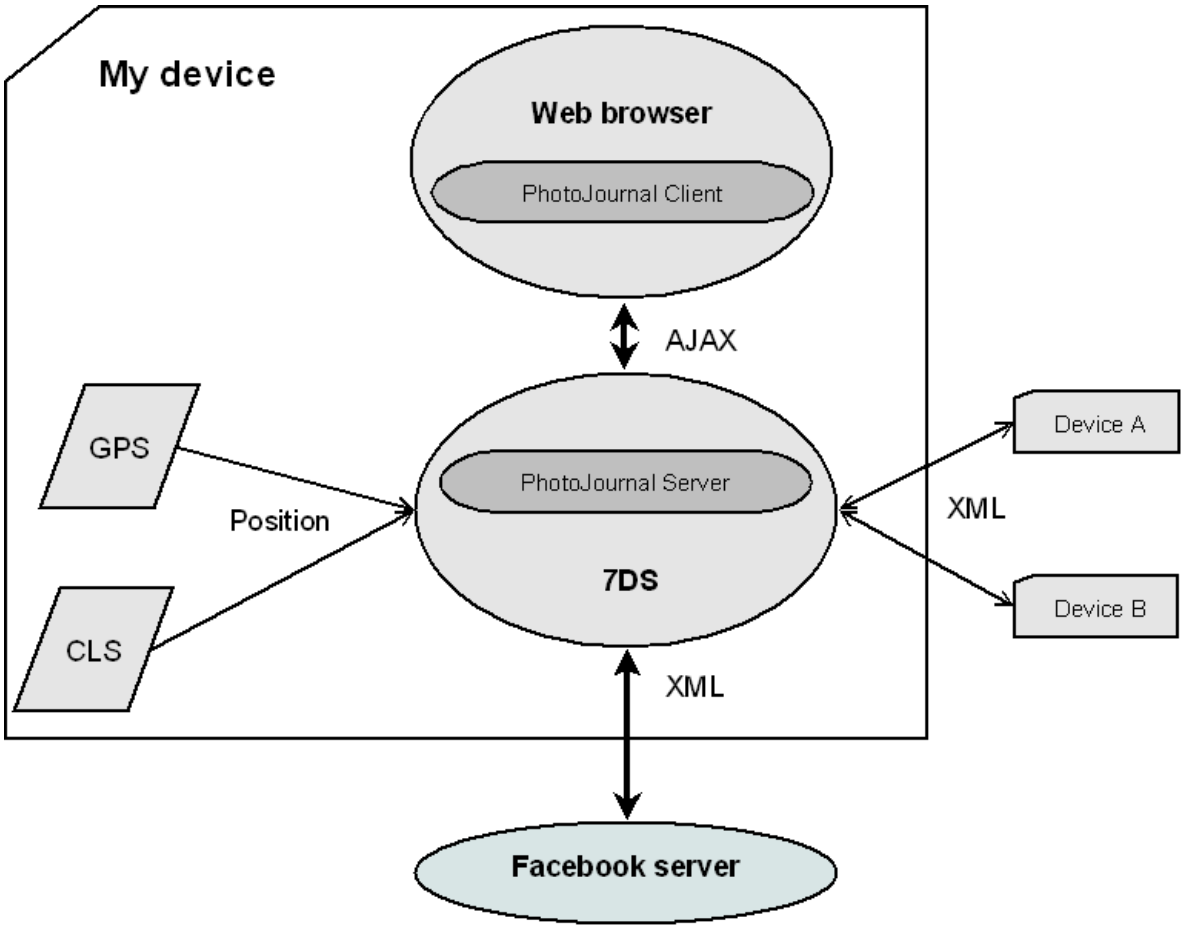


Figure 4.1: Structure of Photojournal and its extension

4.1.1 User's information

Each user is expected to enter his private information. Selecting the Edit profile from Photojournal menu allow user to save his name, email and website. Actually, only his name is important for the interaction with other users. Email and website are just informative data.

The XML structure shows how the data are stored. There is also a node called location which value is grabbed from file `movement/position.gps` periodically. This file is generated and updated by external application using positioning systems like GPS or CLS.

```
<user>
  <firstname>Peter</firstname>
  <lastname>Marcely</lastname>
  <webpage>http://stud.fit.vutbr.cz/~xmarce00/</webpage>
  <email>xmarce00@stud.fit.vutbr.cz</email>
  <location>35.30518,25.07242</location>
</user>
```

4.1.2 Photo galleries

Photojournal's main idea is to store or share pictures placed on a map to specific place. Starting from the beginning, user is able to select the "Add pictures" from main menu. After clicking on desired position on the map, user will be prompted to select the pictures from his hard drive. Unluckily, user is restricted to use just the Photojournal pictures directory. This is caused due to the JavaScript security restriction. The browser can not read the whole path of file, just his filename. The whole path is passed when the form is sent, but because of the fact that it uses AJAX, there is probably no way how to implement it. This complication makes the only unfinished function in Photojournal application.

Each photo from the gallery can be managed. It means that user can see the detailed information about it. If the user wants to make the pictures private, so no other user can download them, he will set the photo's attribute to private. If vice versa, he can leave the photo public and other users will be able to download this photo. Each photo can be rated by its owner and others can select and download photos with specific rating. Other basic functions are viewing each photo in new window in its original resolution, deleting the photo or the whole photo set, or deleting photos and notes based on the location by deleting the whole region.

Last functionality is commenting pictures. Photojournal offers users comments users the photos, download comments from other online user and manage them.

All the photo sets are stored in main storage file db.xml. This is the sample photoset saved in XML file.

```
<photoset>
 <latitude>45.5001320277979</latitude>
 <longtitude>-73.56552064418793</longtitude>
 <photo>
   <author>Petko</author>
   <file>http://%host%/7ds_cache/pictures/n1032970736_125129_7471.jpg</file>
   <rating>3</rating>
 </photo>
</photoset>
```

If a user wants to download pictures from other users in selected region, he will select Search pictures and mark the region. He can filter his request by the rating of pictures. When neighbours with selected pictures are found, user will select those which photos are wanted. If the same photos are downloaded again, they will not be appended to storage file based on their same location as previous photos.

4.1.3 Notes

The other functionality ate notes which can be attached to unique position on the map. The other functions remain the same as it was in photo sets. Notes are aimed just for the local users so there is no change of getting notes of others.

4.1.4 Queries

Queries are the way how the nodes of server-side Photojournal application communicate with each other. It could be easy to understand that there are tree types of queries. The first is a query which provides requesting pictures from selected region and sending the response created of found pictures in XML structure. In the example of XML file, we can see the `%host%` value of the host's IP address. It is because of the fact, that the IP can be changing over the time, so the actual IP address is changed for `%host%` and send. The other node can then easily download it due to the running web server.

The other one is used for refreshing list of neighbours. This query is called periodically from the browser in order to update the list.

The last one is a query that provides update of comments of other users on the selected photo. The example of syntax of request and response is as follows.

```
XPath: /root/user/firstname | /root/user/location

<root>
  <firstname>Peter</firstname>
  <location>10,20</location>
</root>
```

As there can be more different requests, the new XML structure file just with the root node and the requested sub nodes is sent.

4.2 Facebook extension

Let us now focus on the design of new functionality starting with the setting the communication process, following by location of friends, and ending with the import and export of photo galleries.

4.2.1 Login process

The theoretical login process was described in the last chapter. Now, the different types of connection will be discussed.

When the user has a connection to the Internet, he will easily type his login information into login box and confirm them. AJAX request is sent and the server side will create a valid connection and synchronize friends and by response, the browser will be notified. The UID (user's unique identifier), session key and user's full name will be obtained. This is the example of structure of Facebook node in XML storage file.

```
<facebook>
  <uid>1642518549</uid>
  <name>Dezo Arsini</name>
  <session>2.V_HtVaWiUyvxnoWgoGVn_g__.86400.1241100000-1642518549</session>
  <hash>-761484553</hash>
  <connection>0</connection>
  <friends>
    <friend><uid>1032970736</uid><name>Peter Marcely</name></friend>
  </friends>
</facebook>
```

The toughest issue is, how to provide login to Facebook when the user has not a connection to the Internet. There is a 7DS which could be used in order to use the proxy server of other node which has a connection to the Internet. There are two complications.

The first is the security. The middle node can hijack user's session and take over his account. Second, more important and essential, Facebook does not support login through proxy which means we have to find another way.

The way consists of setting a hash code of user's email and password after successful online login into the XML storage file. When the user wants to login next time without the active

connection, he will confirm the button Work offline. He will be seen as a user in offline mode on the map and in list of neighbors in other user's clients.

All in all, there are three types of connections:

- Number 0 – user is not logged
- Number 1 – user is logged in offline mode
- Number 2 – user is logged in online mode

After the connection is lost, or user logs out, the connection value is changed to zero and a new login box appears in the client.

4.2.2 User's position

Showing friend's position on the map is the main idea of this project. Due to the fact, that we have all of the needed sources – external application setting the position, embed map and server side of application that enables the communication between nodes, there are no complications for design and implement showing positions on the map.

Photojournal's client in web browser periodically calls the function which is passed to server by AJAX. After broadcasting the message to find all online neighbors, these are going to respond with XML structured response which includes the type of connection, name of the user and his position. The position is loaded from file stored in `movement/position.gps` which is periodically updated by extern positioning application. The name of user is either the one loaded from Facebook, or it the connection is not active, name from user's profile is sent. The AJAX reply to the client is a list of friends consisting of Facebook online user's friends, Facebook offline user's friends and the rest of people who are neither user's friends or are not connected to Facebook.

Google Maps API does not have support to show marker along with the caption in the map, so we need to do some research on this issue. The API supports including various utilities with different functions. The Simple PopupMarker is the desired open-source utility. It allows displaying info-bubble next to marker with any HTML inside.

When the list is reloaded, neighbour users will appear in user's neighbour list and after selecting one of them, the map is centred to his position.

4.2.3 Exporting photo gallery

Once there is an established connection to Facebook, available API functions can be used. Those are the photo functions which allow managing photo albums on Facebook.

The current Photojournal's conception is that a user will select specific pictures from the pictures directory and these will be placed to a photo set at a specific location.

The only need is to find out, if the user wants to export the gallery also into his Facebook account in order to make the photos visible also for users not using Photojournal, or users who are not in the same wireless network.

After selecting whether the photos are going to be exported or not, and typing name of the exported gallery, the AJAX request is sent to Photojournal's server. After the valid connection is checked, the pictures are uploaded to Facebook with captions indicating the longitude and latitude together with the sign of Photojournal application. If the photo gallery with selected name exists, this one is selected, if not, a new photo gallery is created.

The XML storage file needs to be extended in order to save the information concerning Facebook gallery. Each user can see, where and if the photo set on Photojournal is also stored on Facebook. Here is example of extended photoset node.

```
<photoset>
  <latitude>45.5001320277979</latitude>
  <longtitude>-73.56552064418793</longtitude>
  <fbUpload>on</fbUpload>
  <fbUserName>Peter Marcely</fbUserName>
  <fbGallery>London</fbGallery>
  <photo>
    <author>Petko</author>
    <file>http://%host%/7ds_cache/pictures/n1032970736_125129_7471.jpg</file>
    <rating>3</rating>
  </photo>
</photoset>
```

Great idea of putting the geo location into the metadata of each picture failed, because Facebook is processing and changing the photos during upload. That is why the geo location in the header is lost and we cannot use this technique.

4.2.4 Importing photo gallery

As it was written before, the usage of sharing pictures among Facebook users is very popular. Users can tag people on photos which allow them to look for the pictures of certain users.

When considering the import of pictures from Facebook to Photojournal, the biggest issue is what position will be chosen for the imported pictures.

Let us assume we know the timestamp when the picture has been uploaded to Facebook. The timestamp is represented by the number of seconds elapsed since 1st January, 1970. This information is stored in file which name consists of unique identifier of user's Facebook account. This is the example of such file stored in `movement/1642518549.track`

```
1220437530;34.3232;54.3221;
1240917336;35.51874;24.02277;
1240917529;35.34440;25.13621;
1220788590;45.432;64.5768;
1240917646;35.21098;26.10787;
```

At this point more AJAX requests are needed. User is supposed to select “Import photos from Facebook”. If no movement file will be found, the center position of the map is used. The server will create a query asking for the names and identifiers of photo albums and send reply back to client. Client will prompt user to choose one of the albums, and new query for getting the names and time of creation of photos will be set. Each photo will be added to a specific photoset with location based on the timestamp. After successful import pictures are reloaded in the browser.

These are the two used queries. The first retrieves the identifiers as well as the names of albums and the other gets the pictures from selected photo album.

```
SELECT aid, name FROM album WHERE owner = '123456'

SELECT src_big, src_small, created FROM photo WHERE aid='456789'
```

4.3 Chat

Simple chat cannot be missing, that is why I decided to design and implement it. Facebook web interface support also chat messages, but since it is not implemented in the API and not all users of Photojournal are also users of Facebook, its interface is not used.

As the queries are implemented, there is no problem to design such simple broadcasting chat. The first user will send a message, server will get a AJAX request. Photojournal will send a query to other users who will process it and store the message into local file adding timestamp.

Import issue is, how to reload the chat messages window, when the AJAX request is sent only when the message is submitted. This is the only time, when the AJAX is waiting for the response. Of course, we can request the messages periodically in some interval using AJAX.

But there is also a better solution which does not overload the client – server communication so much. After each sent and received message, the chat file with messages in plain text(located in `7ds/chat.txt`) and another chat file with HTML styled messages(located in `7ds/chat.html`) are saved. Web browsers allow including web site element called `iframe`. This element is not dependent

on the rest of the site. We can then easily make it reload every 10 seconds. So when the message is sent, it is reacting dynamically.

There could be question, why is not the JavaScript used as it is on rest of the web site. The answer is that JavaScript has limited file handling due to security reasons, so it is much more easy to use the iframe element.

The following example is a query broadcasted to all users:

```
Chat: <b>Peter Marcely</b>: Hi folks!
```

And the example of `chat.html`:

```
<div style='width: 261px; font-size:70%;'>
     12:03:23 <b>Peter Marcely</b>: Fine!<br>
     12:02:14 <b>Andrej Rypak</b>: Fine, and u?<br>
     12:00:30 <b>Peter Marcely</b>: Hello how are you?<br>
</div>
```

4.4 Adding note as Facebook's status message

To show the performance and abilities of Facebook API, this simple function is going to be implemented. When user is adding the node and selected to upload it also to Facebook, the Facebook's status message is set together with note's coordinates.

5 Implementation

Programming language Java and IDE Eclipse was mainly used during the implementation. I am going to describe the structures, classes, functions and procedures used in Photojournal's extension.

The diagram on figure 5.1 shows the interaction between client's JavaScript functions and server's Java classes.

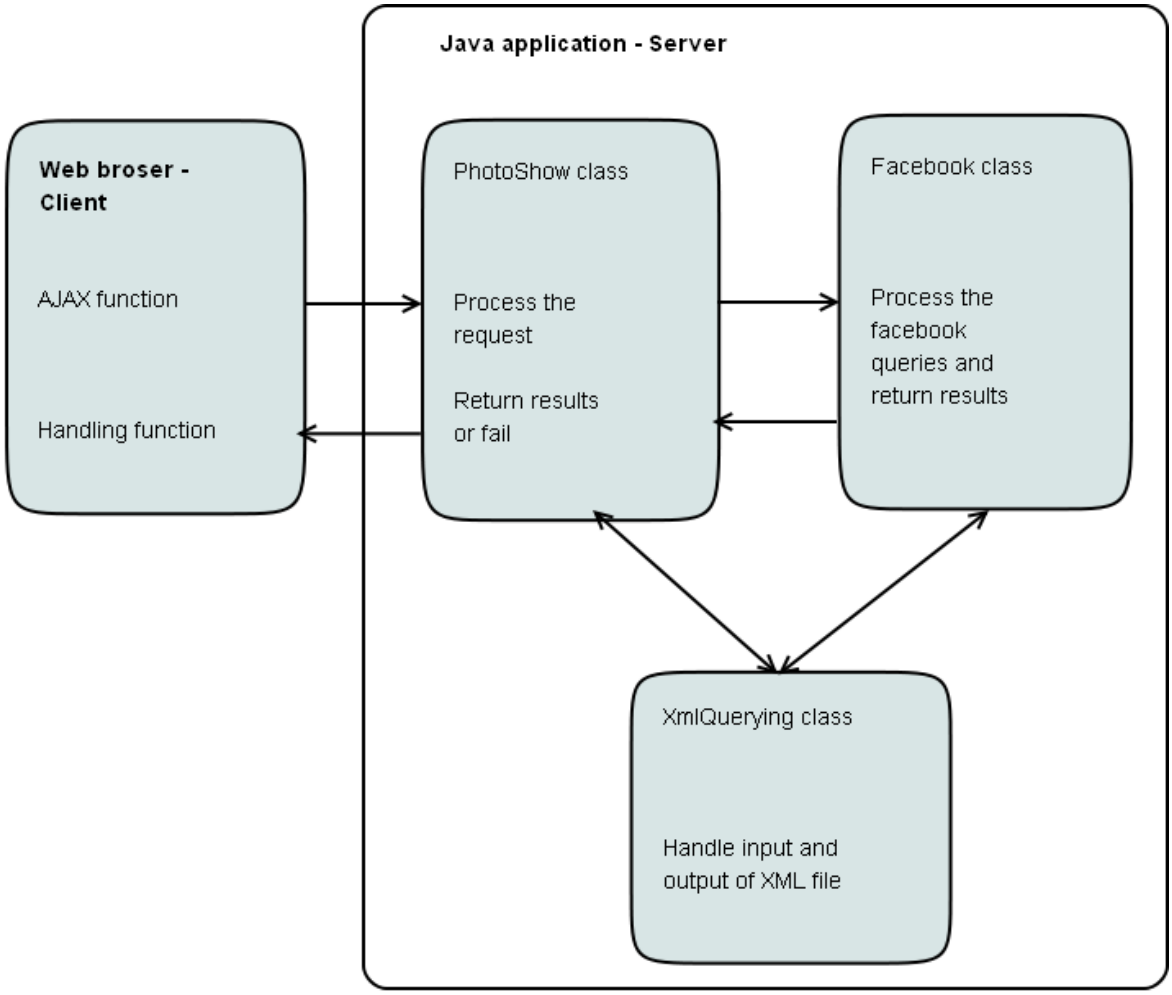


Figure 5.1: Internal structure of Photojournal application

5.1 Functions of Photojournal's extension

The implementation of most of the Facebook functions is based on this model. The picture illustrates how the processes running between parts of Photojournal are. Let us describe these functions more properly.

5.1.1 Login process

After analysis and design of login process, it should be easy to implement. Unfortunately, there are big complications on desktop applications using newer versions of Facebook Java API. After login, Facebook is returning exception of incorrect signature what means that the session is not active. What is more, when it finally starts to work with this appropriate version, the function which allows continuing session was not working too. The only solution was to start the instance of Facebook class in the main class and let it run all the time.

These complications could endanger the project, but finally the communication channel with Facebook was operating.

Let us now focus on the implemented login process:

1. Set the instance of Facebook XML client with valid application key and secret key.
2. Create authentication token.
3. Request the login page using apache's HTTP client and GET method where the application key and authentication token is passed.
4. Sending user's login information using POST method of HTTP client.
5. If login is successful, saving session. The communication can start.

In the following tables, the request is passing from the left to the right and the result is passing vise versa.

	Web Browser	**PhotoShow class**	**Facebook class**
Functions or structures	`facebookLogin` `loginBox`	`facebookLogin`	`login`
Request	`Email & password`	`Email & password`	
Result		`Success/fail`	`Success/fail`

Table 5.1: Implementation of login process to Facebook

5.1.2 Synchronizing friends

After login user's friends needs to be synchronized in the XML file.

	Web Browser	**PhotoShow class**	**Facebook class**
Functions or structures	`loginBox` `synchronize`	`facebookSynchronize`	`getFriends`
Request	`Update friend's list.`	`Update friend's list.`	
Result		`Success/fail`	`Success/fail`

Table 5.2: Implementation of synchronizing friends

5.1.3 Checking Facebook's connection

The connectivity is checked periodically, so the logout in browser can be made automatically when the connection is lost.

	Web Browser	PhotoShow class	Facebook class
Functions or structures	checkConnectionInterval checkConnection	facebookCheckConnection	CheckConnection
Request	Is active?	Is active?	
Result		Success/fail	Success/fail

Table 5.3: Implementation of testing the connection

5.1.4 Checking online neighbours

The online neighbours are checked periodically, the list of friends is retrieved and outputted.

	Web Browser	PhotoShow class	Facebook class
Functions or structures	checkOnlineUsers OnlineUsers	startListingNeighbors listNeighbors	friendsMap
Request	Who are my friends?	Is the user using Facebook?	
Result		List of friends	Success/fail

Table 5.4: Implementation of querying the neighbours

5.1.5 Logout

Logout means that user's client will be lost and session deleted.

	Web Browser	PhotoShow class	Facebook class
Functions or structures	facebookLogout	facebookLogout	logout
Request	Do logout.	Do logout.	
Result		OK	OK

Table 5.5: Implementation of logout from Facebook

5.1.6 Import of pictures

These functions import pictures from Facebook and put them on the correct place. There are three steps of import. Firstly, user sends a request to import photos. Secondly, he will be asked to choose the gallery. Finally, the reload of photos is made.

	Web Browser	PhotoShow class	Facebook class
Functions or structures	importFacebook showAlbums importAjax afterImport addPhotos	facebookGetAlbums facebookImportGallery LoadLocalPhotos	getAlbums importPhotos
Request	Give me albums	Give me albums	
Result		List of album	List of albums
Request	ID of album	ID of album	
Result	Reload photos	Download photos and send success/fail	Hyperlinks to photos

Table 5.6: Implementation of importing photos from Facebook

5.1.7 Export of pictures

When user enters the name of gallery and the Facebook's attribute is set on, the pictures of this photo gallery are exported also to Facebook.

	Web Browser	PhotoShow class	Facebook class
Functions or structures	addPhotos collectPaths fileuploaded reload	uploadPhotos	uploadPhoto
Request	Submitted photos	Submitted photos	
Result	Reload photos	OK	OK

Table 5.7: Implementation of exporting photos to Facebook

5.1.8 Setting note as status message

When note is added with Facebook's attribute set on, Photojournal will set this status on Facebook.

	Web Browser	PhotoShow class	Facebook class
Functions or structures	AddNote reload	AddNote	updateStatus
Request	Add note	Add note	
Result	Reload elements	OK	OK

Table 5.8: Implementation of setting status message

5.2 External libraries

Photojournal uses large number of libraries as it is a huge application depending on many functions. Here is the list of libraries used during implementation of Photojournal extension, which are included in the package of Facebook Java API:

- Activation-1.1.jar
- Commons-logging-1.1.1.jar
- Facebook-java-api-2.0.3.jar
- Facebook-java-api-schema-2.0.3.jar
- Jaxb-api-2.1.jar
- Jaxb-impl-2.1.3.jar
- Json-20070829.jar
- Stax-api-1.0.2.jar

6 Model case

This model case shows the usage of Photojournal application. There are 2 connected peers via ad-hoc wireless network. One has valid online connection to Facebook, another one has valid offline connection. In the left side, there is facebook's login window, neighbour's window in which he can see his friends and chat's window. User's friends are also placed on the map, updating periodically. Each user with a valid connection can export and import pictures from Facebook.

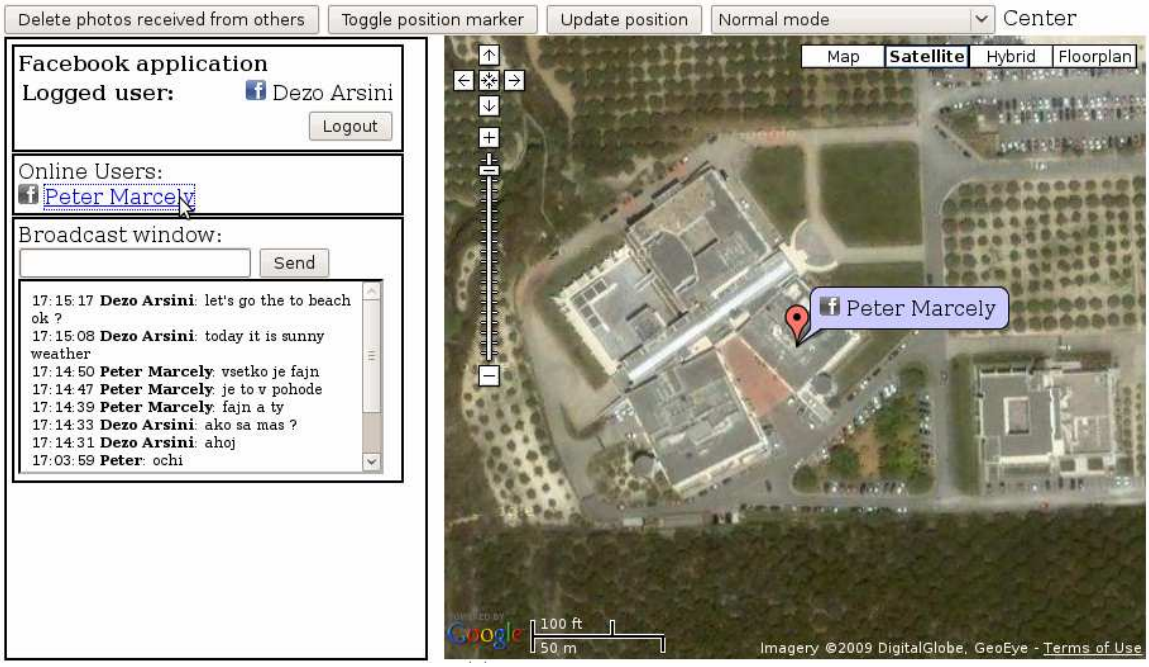


Figure 6.1: Example showing the position of a friend

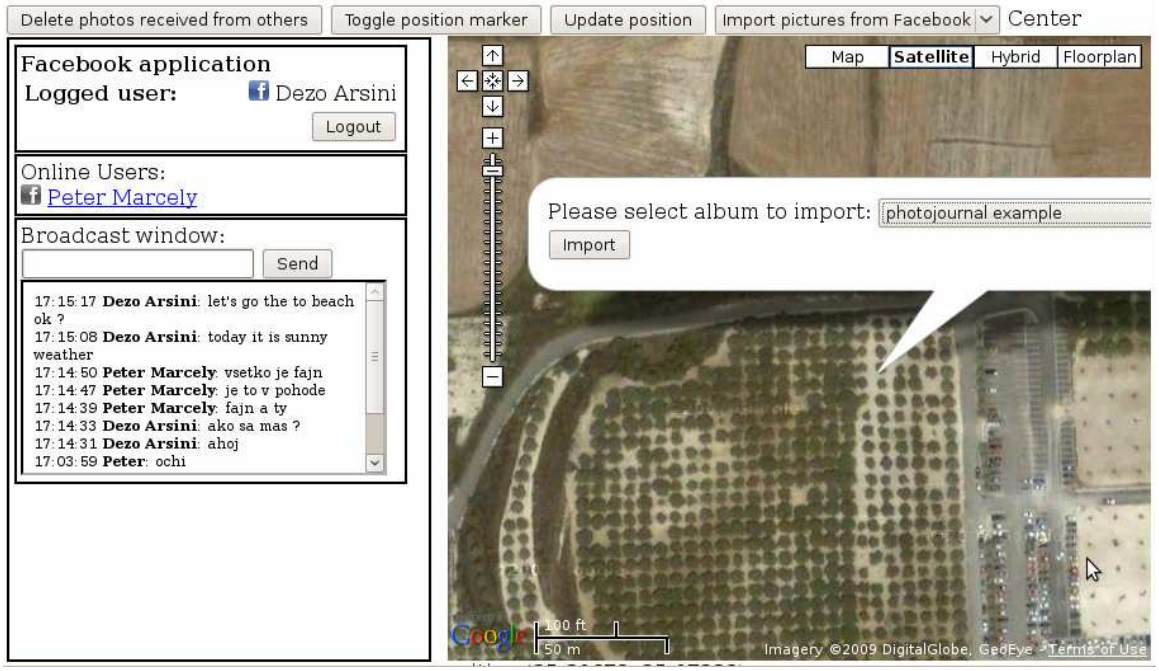


Figure 6.2: Example showing the choice of imported album

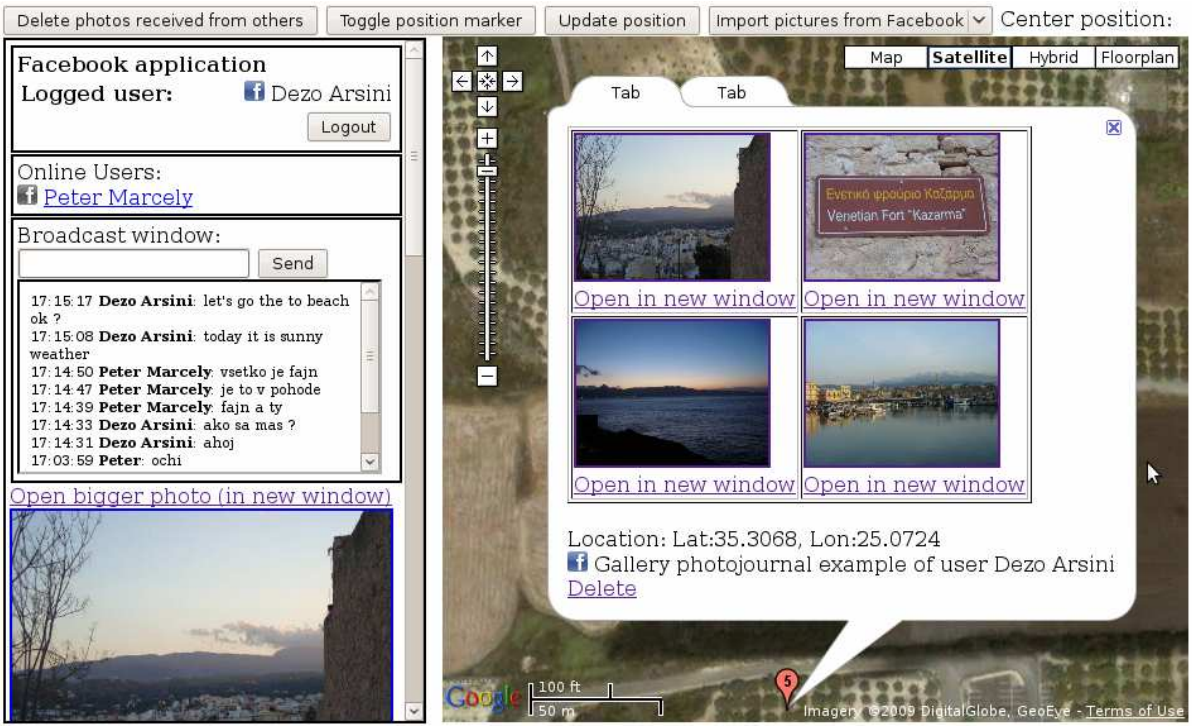


Figure 6.3: Example showing the imported gallery along with the photos

After export user's facebook gallery will be updated with these pictures. After import other users can access this gallery.

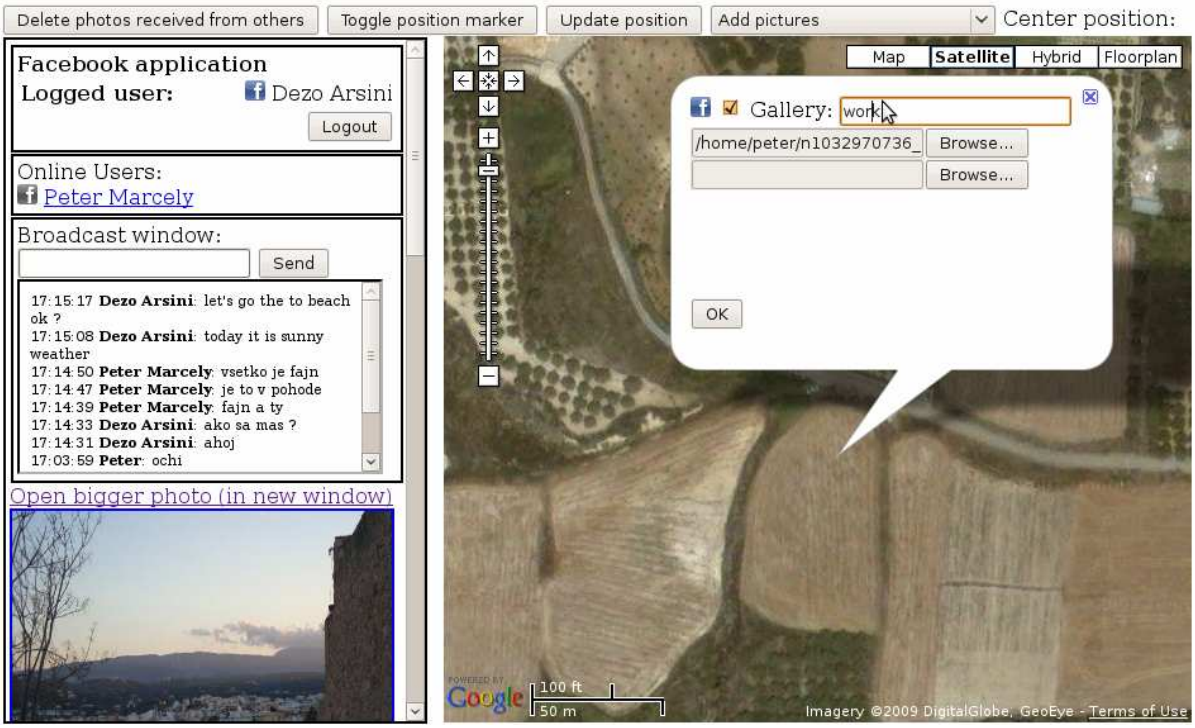


Figure 6.4: Example showing the ability of uploading and exporting pictures

7 Measurements

7.1 Principle of measurements

The principle of measurements is capturing the messages between the Photojournal's client and server. Since the communication between them is based on AJAX, we can use the network capturing software called Wireshark to capture the AJAX messages on local loopback. In fact, the AJAX messages are just HTTP GET request packet and HTTP response packet.

This is the example of capturing login messages and calculating the delay. The first captured packet shows the AJAX request, second AJAX response with result "1" which means success.

```
     0.000658       127.0.0.1    127.0.0.1    HTTP   GET
http://localhost:9000/7dsPhotoShowFacebookLogin:::petkom@gmail.com:::password:::
HTTP/1.1
     8.807743       127.0.0.1    127.0.0.1    HTTP   Continuation    or    non-HTTP
traffic: <html><body>|1|</body></html>
```

The computed delay is difference of captured times, 8.80785 seconds.

7.2 Conditions

The measurements were performed in FORTH (Foundation for Research and Technology) during two days in the afternoon. There were made thirty different attempts for each type of measured process. My personal laptop and another different personal computer were used in order to do the measurements. After all, Matlab was used to draw the graphs.

7.3 CCDF

The plots show the complementary cumulative distribution function of time needed to login to Facebook, to export photos and to import photos. CCDF [13] is complementary function to cumulative distribution function. CCDF shows how often the random variable is above a particular level.

7.4 CCDF of login time

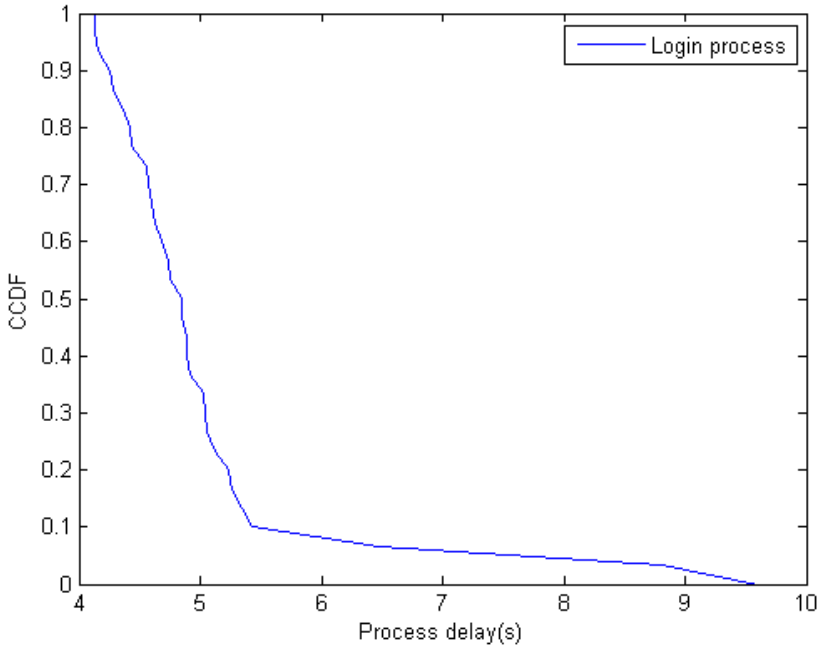


Figure 7.1: Complementary cumulative distribution function of login time

7.5 CCDF of time of exporting photos

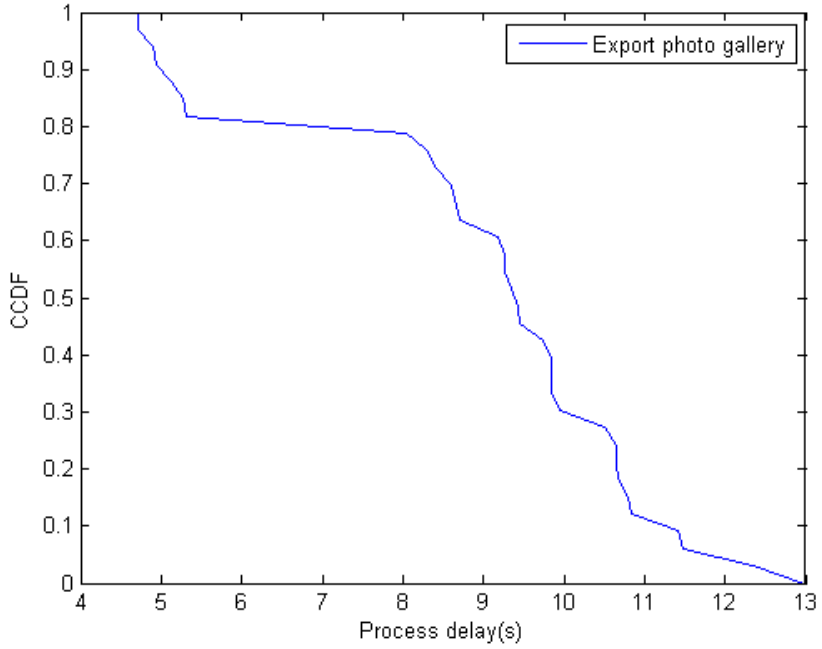


Figure 7.2: Complementary cumulative distribution function of exporting photos delay

7.6 CCDF of time of importing photos

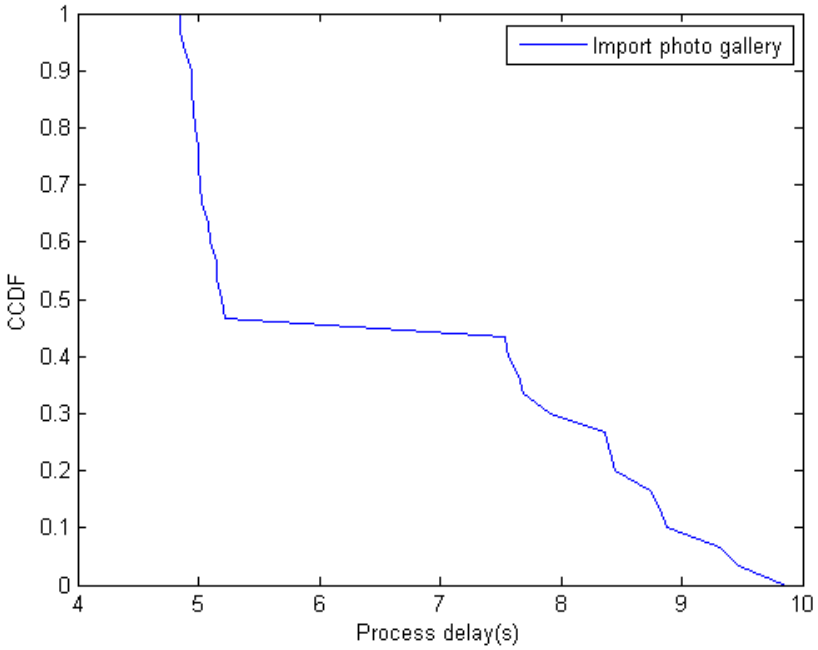


Figure 7.3: Complementary cumulative distribution function of importing photos delay

8 Conclusion

Photojournal and its extension gave me a great opportunity to develop my programming skills. What is more, since I spent a semester working on this project at the University of Crete, I have got even more practice experiences such as great practice of English language, or my independence.

At the beginning of the project, I needed to understand the concept and structure of Photojournal code. Except from the initial problems with connecting to Facebook, the project's development was fluent. After all, the Facebook extension project has been finished and Photojournal has now more functionality, including chat, showing other's users position and Facebook's photos interface.

There could be several more extension in the future. I have in mind Photojournal extension to Flickr which is another social server sharing pictures. Due to the fact that it is aimed mainly on the pictures, the support of pictures is better, and that is why the user's functions are better on Facebook.

9 Literature

[1] Niko Kotilainen, Maria Papadopouli: You've Got Photos! The design and evaluation of a location-based media-sharing application [cit. 2009-04-29]. Available on URL: http://www.ics.forth.gr/netlab/mobile/publications/papadopouli.mobimedia08.pdf

[2] Wikipedia.org, Social network [cit. 2009-05-01]. Available on URL: http://en.wikipedia.org/wiki/Social_network

[3] Wikipedia.org, List of social networking websites [cit. 2009-05-01]. Available on URL: http://en.wikipedia.org/wiki/List_of_social_networking_websites

[4] Wikipedia.org, Facebook [cit. 2009-05-01]. Available on URL: http://en.wikipedia.org/wiki/Facebook

[5] Java programming language tutorial [cit. 2009-05-01]. Available on URL: http://www.linuxsoft.cz/article_list.php?id_kategory=192

[6] Wikipedia.org, Java [cit. 2009-05-02]. Available on URL: http://en.wikipedia.org/wiki/Java

[7] Wikipedia.org, JavaScript [cit. 2009-05-02]. Available on URL: http://en.wikipedia.org/wiki/JavaScript

[8] Wikipedia.org, AJAX [cit. 2009-05-02]. Available on URL: http://en.wikipedia.org/wiki/AJAX

[9] Wikipedia.org, XML [cit. 2009-05-02]. Available on URL: http://en.wikipedia.org/wiki/XML

[10] Google maps API documentation [cit. 2009-05-04]. Available on URL: http://code.google.com/apis/maps/documentation/

[11] Facebook API documentation [cit. 200-05-04]. Available on URL: http://wiki.developers.facebook.com/index.php/API

[12] Facebook FQL language [cit. 2009-05-04]. Available on URL: http://wiki.developers.facebook.com/index.php/FQL

[13] Wikipedia.org, Cumulative distribution function [cit. 2009-05-05]. Available on URL: http://en.wikipedia.org/wiki/Cumulative_distribution_function

Attachments

Attachment 1. Attached CD includes technical report, source code and manual how to make the application work.