Project acronym: **Go-myLife**

Project full title: **Going on line: my social Life**

AAL Joint Programme

Call for Proposals AAL-2009-2-089

D3.3 Final Platform Architecture and Design

Author: Idoia Olalde (Andago)

Version: 1.0

Date: 20/03/2012

Deliverable Number:	D3.2
Contractual Date of Delivery:	31/01/2012
Actual Date of Delivery:	20/03/2012
Title of Deliverable:	Final Platform Architecture and Design
Dissemination Level:	Public
WP contributing to the Deliverable:	WP3
Author(s):	Idoia Olalde (Andago)
Participant(s):	Ioannis Kouris (ICCS) / Fabio Tumiatti (Atos) / Sergio García (Atos) / Mikel Serrano (Andago) / Roberto Calvo (URJC)

History			
Version	Date	Author	Comments
0.0	05/01/2012	Idoia Olalde (Andago)	Draft version
0.1	07/02/2012	Ioannis Kouris (ICCS) Mikel Serrano (Andago) Sergio García (Atos) Roberto Calvo (URJC)	Initial contributions
0.2	27/02/2012	Ioannis Kouris (ICCS) Mikel Serrano (Andago) Sergio García (Atos) Roberto Calvo (URJC	Contributions
1.0	30/03/2012	Idoia Olalde (Andago)	Final version

Approval and Sign-off		
Date	Name	Sign-off

Abstract

This document contains the final architecture design of the Go-myLife platform and the interfaces it offers: User Interface API, an interface to manage the content stored in the Go-myLife Social Engine; External Social Networks API, an interface to connect to existing online social networks and allow a bidirectional communication between the platforms; Services Integration API, an interface to integrate content from external information sources; Communities Integration API, an interface for the construction of the communities and the management of their content; Data Analysis API, an interface to analysis the data, obtain anonymously statistics and detect tendencies to be offered through a Report Dashboard for 3rd parties.

It also summaries the functionalities that final prototype of Go-myLife will have and it presents the final design of the web application which end-users will use.

This last design is based on the technical requirements gathered during the workshops with end-users and the feedback of the pilots.

Keywords

Go-myLife architecture, Go-myLife platform, API, Go-myLife prototype

Table of Contents

Table of Figures

1 Introduction

1.1 Summary

The aim of the deliverable D3.3 “Final Platform Architecture and Design” is to provide the final architecture and prototype of Go-myLife. The first prototype of Go-myLife platform was designed and implemented during the first phase of the project and tested during the trials. The feedback from these pilots helped the consortium to redesign and adjust Go-myLife to the real needs of the users. Along the document, the Go-myLife architecture is specified as well as its relationship with end-users and business models. The Go-myLife platform provides a set of tools and application programming interfaces (hereinafter API) for the inclusion of services and content from external providers and the construction of communities.

1.2 Role of this deliverable

The deliverable D3.3 “Final Platform Architecture and Design” is the outcome of the task T3.3 “Final Platform Architecture and Design” of the work package WP3.

In this task, it has been designed the final Go-myLife architecture and prototype in accordance to the architecture design specified in D3.1 “Initial Platform Architecture and Design”. The prototype design was explained with greater detail in D3.2 First Prototype Design and implemented during T4.1 and the feedback from the pilots with end-users. These pilots were carried out in United Kingdom and Poland as part of WP6, Evaluation and validation through scenarios.

This deliverable tries to summary the final Go-myLife architecture, the final prototype and the provided tools and APIs for the building of the local life communities that will be carried out during WP5 – Communities constructions and customization.

All technical partners have contributed to this deliverable.

2 Feedback from pilots and workshops with end-users

This section describes the features and improvements collected in the workshops and pilots that can have a potential impact in the design of the Go-myLife architecture and prototype. These workshops where placed in UK and Austria, and pilots in UK and Poland. Were carried out two types of workshops, one was conducted to assess existing Social Networks and the second one to investigate the communication patterns of older people in Social Networks.

2.1 Findings in the workshops

This section collects the findings of the workshops with end-users that took place in United Kingdom and Austria.

Requirement	**Design impact**
Forums	Users can discuss different themes expressing their thoughts and feelings. It is important for the users to have the option to publish something in the forum anonymously or using real name.
Local communities	Another kind of group that is highly appreciated by the senior citizens and will be implemented in Go-myLife. UI interface to interact with groups or associations that are relevant to their local community and neighbours.
Friends groups	Section in Go-myLife to manage friends in different groups of friends. Example: family, work, etc.
Platform usability	The platform must be simple, easy to use, well-structured, consistent and not overwhelming.
Attractive layout design	An appealing layout and attractive design are the key factors that would make elderly people to use it regularly.
Easy navigation	The navigation of the platform has to be simple, clear and logically structured, to avoid confusion. Breadcrumb navigation and a "backward" button in every page should be used by the platform to help users track their actions.
Reliable search	Implement and design search engine in greater depth, to ensure that it will offer significant and relevant

	results to the user.
Avoid banners/flash/icons/pop-ups	Because elderly demand an easy navigation, no pop-up advertisements will be displayed; the usage of flash and banners will be omitted and the layout will be as simple as possible. The icons of the social network will not be changed because it can cause irritation and confusion.
Large font	The font size should be large and clear for elderly people.
Avoid scrolling	The content of each page should be fitted within the screen size, avoiding the need for scroll-down in the desktop. The scroll-down should be transformed to a “book-like” layout, because old people find it easier to use the platform like a book and “turn” their pages to navigate.
Sharing options	Users need to be able to control the sharing options and these must be displayed in an easily discoverable and understandable way.
Help	Help can also come from explanatory videos and text that guide the user step-by-step through the platform and explain how to use the different functionalities of Go-myLife. Instructions must be clear.
Simple profile and registration	The registration and profile completion must be simple. Because many users don’t like being required to provide a great deal of personal information, the majority of fields must be optional.

2.2 Findings in the pilots

This section collects the improvements and requirements made by the end-users during the trials. These improvements and requirements are based on the Polish pilot. UK pilot is taking place as the time this deliverable is being written. The outcome of the trial will be incorporated once it is finalized.

Requirement	**Design impact**
Step by step help/manual	Provide a step by step clear help to avoid confusion and facilitate the use of the social network.

Bigger letters	Use big letters en each section of Go-myLife, including the main navigation menu.
No warnings	Delete warnings or alerts of user executed actions description. Design UI to include labels where the message will be posted.
Password recovery	Implement a password recovery option because older people are more susceptible to forget their username or password.
Edit mandatory profile data	The users want to change profile data such as first name, last name and email.
Pending event invitations alert	Make more visible when the user have pending event invitations.
Accepted friendship requests alert	Make more visible when a friend accepted friendship request sent by the user.
Pending friendship request alert	Make more visible when the user have pending friendship requests to accept.
View group before join	View more group information before join.
Privacy on sharing content	Option to share content (news, photos ...) with only a group of friends. Design UI to select a group when the user wants to post a note or upload a photo.
Language selection	Apply UI modifications to change language inside the social network when you are using the mobile phone.
Delete profile information	Make changes in profile information. The older people don't want to fill and show all optional information that is displayed because they think that it is too much information.
Avoid scrolling	Older people don't like scrolling, and there are some screens where scroll is necessary to see everything. Most annoying thing to the users is horizontal scrolling, so the UI should show information without scrolling.
Set maximum file size to upload	If the size of the photo to upload is too big, it requires a lot of waiting time while uploading and maybe it will crash or older people think that it crashed. Establishing a maximum file size can avoid these

	problems.
Layout colours	Older people don't like the colours of the layout in Go-myLife desktop version. They want white background and use new colours appropriate for them.
Enter key	Use the enter key besides the mousse will be easier for older people.

3 Final Platform Architecture

The final architecture of Go-myLife platform is shown in the Figure 1. However, significant changes of the existing components will be implemented to address the missing gaps and the improvements identified during the trials.

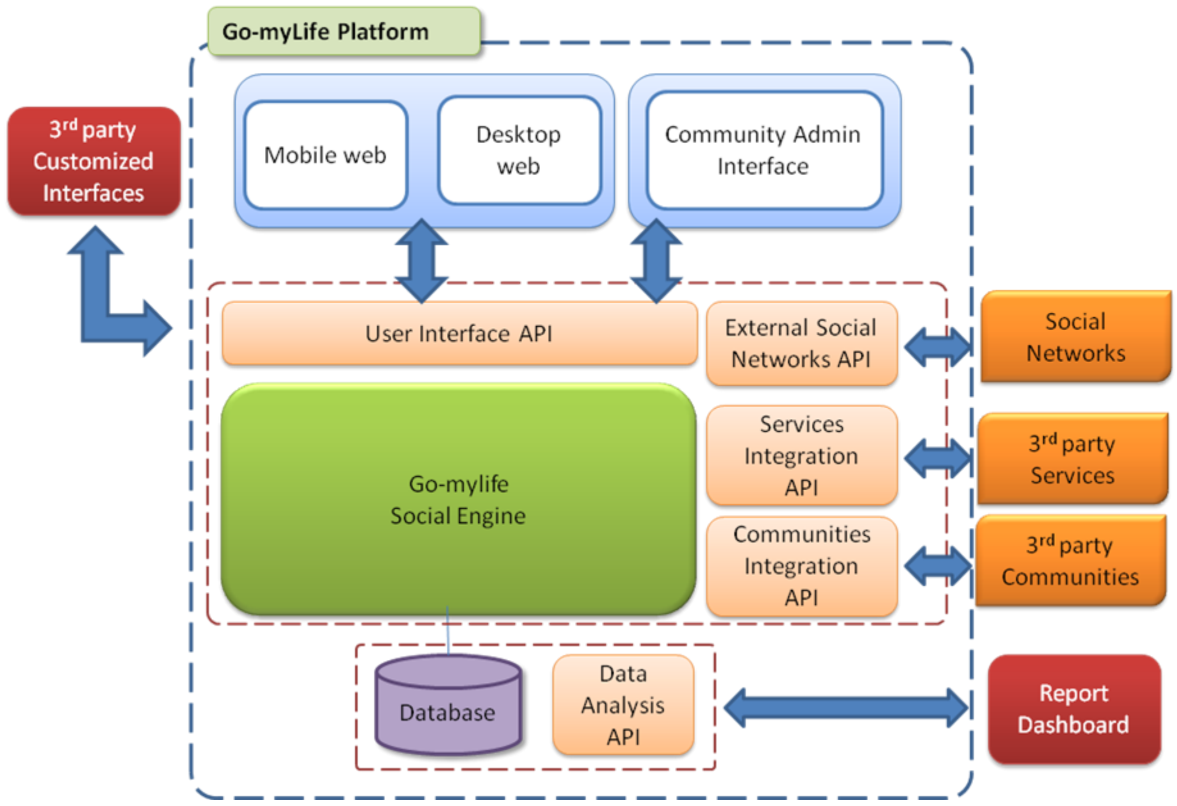


Figure 1: Platform Architecture

3.1 Go-myLife components

Hence, the Go-myLife platform architecture, as it is illustrated in Figure 1 is composed of the following components:

- Web client application: At the user interface level, a web application, that includes a Mobile web form as well as a Desktop web, was developed using Google Web Toolkit. The Web client application provides users with a web based access to the system to interact with their peers, share contents, connect to other social networks and be active members in the communities. The web client applications will be redesigned to include the feedback of the pilots in UI terms.
- Community Admin Interface: a web interface that provides to administrators web-based access to capabilities for the administrators of the communities that include:
    - Manage Communities
    - Manage Communities News

  - Manage Communities Photos
  - Manage Communities Events

  This administrator page will be implemented for the second prototype.

- Go-myLife Social Engine: The Go-myLife Social Engine is the core of the Go-myLife platform and is responsible for the management of the content, its relationships and privacy. It is based on the open source LibreGeoSocial framework. New functionalities have been identified as well as improvements/enhancements in existing ones.

- User Interface API: an interface to manage the content stored in the Go-myLife Social Engine. This interface is called from the UI level to create the different functionalities offered in the client application. According to the new functionalities that the Go-myLife Social Engine will be developed, new methods will be added to the API.

- External Social Networks API: an interface to connect to existing online social networks and allow a bidirectional communication between the platforms. In the scope of Go-myLife project, different connectors will be developed for the interaction with each of these networks. All of them follow a schema to guarantee its integration without unnecessary changes at the user interface.

  In this second phase of the project, a redesign of this API has been taken in order to adapt it to the real requirements of the existing social networks. A Twitter connector will be implemented for the second prototype.

- Services Integration API: an interface to integrate content from external information sources. This module allows an easy integration system for sources of information; it qualifies the contents and provides to clients a good abstraction model. This API remains the same.

- Communities Integration API: an interface for the construction of the communities and the management of their content. This API remains the same.

- Data Analysis API: an interface to analysis the data, obtain anonymously statistics and detect tendencies to be offered through a Report Dashboard for 3rd parties. This API will be generated for this second version.

- Database: is the database software used to store all the Go-myLife related data.

3.2 APIs, SDKs and tools

Along with the Go-myLife web applications, the consortium provides a set of APIs to facilitate the construction of communities and the integration of external services and social networks inside Go-myLife platform.

In this second phase of the project, the consortium has redesigned some of the APIs to reach the user needs.

3.2.1 User Interface API

User interface API, is a REST[1] interface to manage the content stored in the Go-myLife Social Engine. This interface is called from the UI level to create the different functionalities the client application offers. Calls to the interface are sent via HTTP protocol and using the petitions POST and GET with JavaScript's XmlHttpRequest.

- GET method: with this method the data of a resource is obtained. Example: the call http://rest.gomylife.libresoft.es/social/user/data/?format=JSON gets the user's data in a JSON[2] file. When the response is received, the client parses it and shows the corresponding data in the user interface.
- POST method: with this method the client sends POST parameters to the URL of the resource and the REST interface modifies the DB data or creates an entry of content on it. Example: the call https://rest.gomylife.libresoft.es/social/user/login/?format=JSON needs POST parameters username and password. For this method, the JavaScript's FormData object is used to reproduce <form> submission mechanism. So, a file and/or a set of key/values pairs are added to this object and sent using XmlHttpRequest. In the previous example, keys "username" and "password" with corresponding values will be added and sent to REST interface to complete login functionality.

The user interface API is divided in several modules to manage the content. *Each module implements a task or a list of functionalities (summarized below the description)*:

Export management: module that allows clients to receive the information using different formats of data. The received information could be JSON or XML file, included in the HTTP Response. To obtain information in XML file, the resource URL is used, and when the client wants to receive it in JSON format, the parameter "format=JSON" will be added to the URL. With this option, when the client receives the JSON file, it parses the information and it shows it in the UI.

Node management: REST API module to cover standard functionalities over the different nodes (A node represents an object of these content types: user, photo, note, video, link, audio and event):

- Change node position
- Tag a node
- Remove node tags

---

[1] http://en.wikipedia.org/wiki/Representational_state_transfer

https://www.ibm.com/developerworks/webservices/library/ws-restful/

[2] http://www.json.org/

- Change node availability dates

Certificates Management: REST API module that can be used to manage security certificates of the application. It only has one functionality:

- Redirect after accepting the certificate

User Management: REST API module to manage user's information and relationships with other users. The below list summarize the functionalities:

- Create new user
- Modify user data (registration data)
- Modify GML profile (user optional data)
- Log in
- Reset user password
- Get user data
- Get my data
- Delete user
- Get friends
- Get friendship invitations
- Set user position
- Near people
- Near friends
- Set user status
- Set user avatar
- Unset user avatar
- List all users
- Search users
- Start relation between users
- Finish relation between users
- Get sent friendship requests

Notes management: REST API module for managing notes.

- Create a new note

- Get note data
- Delete a note
- List all notes
- List all user notes

Photos management: REST API module for managing photos.

- Upload a photo
- Get photo data
- Get photo image
- Get photo thumbnail
- Delete a photo
- List all photos
- List all user photos
- Edit photo data

Sounds management: REST API module for managing sounds.

- Upload a sound
- Get sound data
- Get sound file
- Delete a sound
- List all sounds
- List all user sounds

Videos management: REST API module for managing videos:

- Upload a video
- Get video data
- Get video file
- Delete a video

Links management: REST API module for managing links:

- Create new link

- Get link data
- Delete a link

Events management: REST API module for managing events:

- Create a new event
- Get event data
- Delete an event
- Change the attendee’s subscription
- Modify event data
- Invite users to the event
- Get pending event invitations

Layers management: REST API module to manage information about layers. All the contents managed by LibreGeoSocial are represented inside a layer. The layers are the abstraction used in LibreGeoSocial to classify the information. The layers manager provides an interface to access to the different information sources supported in the server, allowing users and applications to interact with nodes from several sources. These sources could be internal sources (with nodes stored in LibreGeoSocial server data base) or external sources (retrieving information through web services, for example: Panoramio). Both layers (external, internal) provide the same features using a unique interface.

- Get the layers list
- Make a layer search
- Get layer's icon
- Get layer's info
- Get layer's category
- Create a layer
- Delete a layer

Walls Management: REST API module for managing user's walls. Internally, a wall is a layer of type wall, so the working is very similar. They use the same interface with special features. This interface is used to get user's wall or search nodes in that wall.

- Get the walls list
- Make a wall search
- Get wall's icon

- Change wall's icon
- Get wall's info
- Get wall's category
- Get user's wall

Comments management: REST API module that manages comments on nodes. The comments are shown in nodes data (for example, comments posted on a note).

- Add new comment
- Delete comment

Messages management: REST API module that manages private messages between users.

- Send a message
- View message
- Get received messages
- Get sent messages
- Get unread messages
- Delete message

Privacy management: REST API module that manages the privacy of the nodes (notes, photos,...) and layers of Go-myLife social network. The privacy options can be: private (only the user can see it), friends (the user’s friends can see it), friends of friends (the friends of the friends of the user can also see it), public (everybody).

- Show privacy status (nodes, layers)
- Permission change (node, layer)

Groups of friends’ management: REST API module that manages groups of friends created by the user to arrange his contacts into categories.

- Create group of users
- Add users to a group
- Delete users from a group
- Edit group info (name)
- Delete group

- Get group's users
- Get my groups of users list

3.2.2 Communities Integration API

The Communities Integration API allows the interaction of third party providers with the Go-myLife social network. With this API, local communities or associations can create their own groups, share content with their members and manage it. The API is implemented using a REST interface and can be called over HTTP protocol, using GET and POST requests. In that sense, the administrator of the community may create, update or delete content related to a community.

To facilitate the creation and management of these local groups, a Community Admin page will be implemented. The potential third party administrator must have available the access key, provided by the administrator of the Go-myLife platform upon verification, in order to access the Community Admin page.

The Communities Integration API allows content creation/update and reception of the posted content. The reception of the content is performed using a GET method, e.g. https://rest.gomylife.libresoft.es/social/group/<groupid>/<parameters>/?format=JSON and content creation and update using a POST method, e.g. https://rest.gomylife.libresoft.es/socia/group/<groupid>/<parameters>/?format=JSON . Calls to the interface are sent via HTTP protocol using the petitions POST and GET with JavaScript's XmlHttpRequest. The *groupid* defines the specific local group, while *parameter* defines the action to be performed. The available operations are:

Communities' management: REST API to manage communities

- Get the communities list
- Create community (name, description, category, icon, availability, latitude, longitude)
- Get community data (name, description, category, icon, creation date, latitude, longitude)
- Get community members
- Show community elements
- Edit community data
- Make a search inside a community
- Delete a community
- List all user's communities
- Join a user to a group
- Delete a user from a group

Community news management: REST API to manage communities' news feeds

- Get community news list
- Send news content (title, news_content, available_to, available_from, latitude, longitude, altitude)
- Delete news

Community photos management: REST API to manage communities' photos

- Get community photos list
- Send photo content (name, description, photo_file, longitude, latitude, available_to, available_from, altitude)
- Delete photos

Community sounds management: REST API to manage communities' sounds

- Get community sounds list
- Send sound content (name, description, sound_file, longitude, latitude, available_to, available_from, altitude)
- Delete sounds

Community videos management: REST API to manage communities' videos

- Get community videos list
- Send sound content (name, description, video_file, longitude, latitude, available_to, available_from, altitude)
- Delete videos

Community links management: REST API to manage communities' links

- Get community links list
- Send links content (title, text, url, longitude, latitude, available_to, available_from, altitude)
- Delete links

Community events management: REST API to manage communities' events

- Get community events list

- Send events content (title, text, url, longitude, latitude, event_to, event_date, available_to, available_from, altitude)
- Delete links

3.2.3 Services Integration API

This section explains how to develop a new external channel, in order to integrate an external information source (such as Flickr, Panoramio, Youtube, etc.) into LibreGeoSocial layers system. It can be developed a channel to integrate any information source and this will remain integrated in LibreGeoSocial.

As mentioned before, all the contents managed by LibreGeoSocial are represented inside a layer and the layers are the abstraction used in LibreGeoSocial to classify the information. When a channel is developed it will be reflected as a new layer from the point of view of LibreGeoSocial server. The difference is that the social contents belonging to this layer won't be stored in the LibreGeoSocial database, so they will be requested to their original source every time the channel is load.

For the creation of a channel, it has to be created a new class that inherits from the class *GIC.Channels.GenericChannel* (Figure 2):

```
class <OwnChannel> (GenericChannel):
```

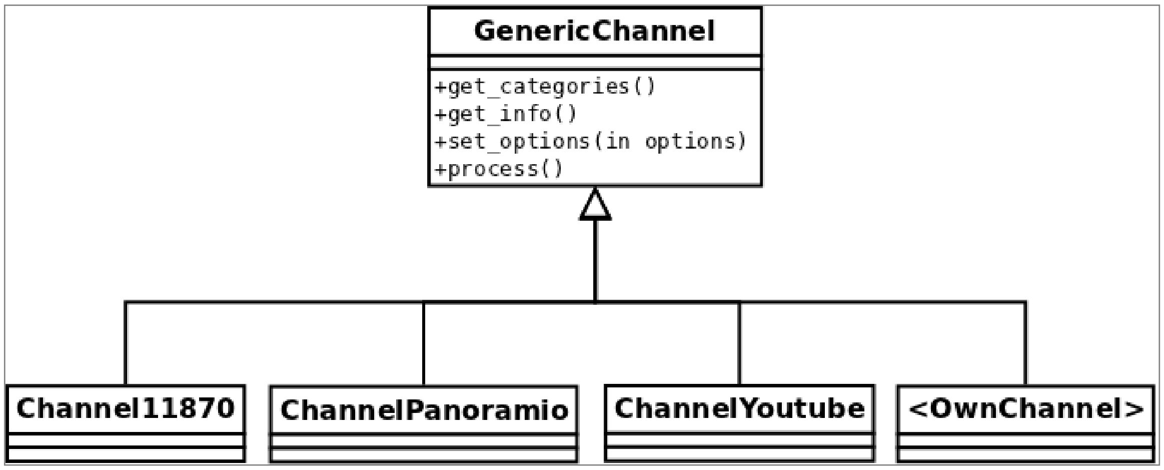


Figure 2:The External Channel Aggregator

Next, these are the methods of the API that have to be implemented:

- ***get_categories():***

  Description: This method has to return information about the different categories (if supported) by the channel. For example, in a video channel categories could be: “sports”, “movie”, “music”.

  Output: returns an array of dictionaries. Each dictionary will have the next keys for each category:

  - “id”: identification number

  - “name”: identification name
  - “description”: description of the category.

  This information will be useful for users and it will be showed in the LibreGeoSocial layers description.

- ***get_info():***

  Description: Returns the description of the channel. This information will be showed in the description of the channel in LibreGeoSocial.

  Output: a string with the channel description

- ***set_options():***

  Description: This method configures the channel regarding the different options supported. LibreGeoSocial needs a set of mandatory options and this function will receive, at least, these fields (use them as your convenience). It could be added other mandatory and/or optional parameters but it is not recommendable for accessing the channel through the LibreGeoSocial Standard Layer System. This system tries to call all the layers with the same options; so, new layers could be added to the system without needing to make changes in the client (which makes always the same request not worrying about the requesting layer).

  Input: a dictionary with the different options. This dictionary will be composed with pairs of options/values.

  Output: a pair of values with True/False if the options are correct and a returned message. This returned message could be used, for example, to refer a missing mandatory field: return False, “The parameter latitude is mandatory”

- ***process():***

  Description: This function is the core of the channel and it will make the search/request returning the matching information.

  Output: The matched information regarding different options and the information source have to be returned regarding the LibreGeoSocial content models. Currently, they can be returned the different information nodes through the LibreGeoSocial standard ones: notes, photos, audios and videos. So, this function return a pair values: the first value is a boolean (true/false) notifying if everything was right and the second one has to be an array of LibreGeoSocial nodes.

  In order to return LibreGeoSocial nodes in a standard way, they have to be used the classes designed for this purpose. These classes represent the LibreGeoSocial standard nodes that are used to return information with internal templates. Using another node’ models could not fit correctly with these templates. The classes to use LibreGeoSocial standard nodes can be find in: *GIC.Channels.Items*.

Here is an example of how to return an information node in the standard layers system:

```
from from GIC.Channels.Items import Note


note = Note()
note.title = "Title information"
note.text = "Text information"


return (True, [note])
```

This “simple” API would allow developers to integrate different information sources quickly.

Finally, to install a channel into a LibreGeoSocial server, it has to be created a directory inside the libs directory of the LibreGeoSocial server installation with the next convention: “channel<name of the channel>”. Here is an example for a channel called Panoramio:

```
cd /var/www/libregeosocial/libs
mkdir channelPanoramio
```

Then, put the python file/template inside the directory, the file has to be named as the directory with the first letter uppercase:

```
cp /home/user/development/panoramio.py

./channelPanoramio/ChannelPanoramio.py
```

The class implemented by this file/template/channel also has to use a convention: “Channel<name of the channel>”[3]. For example:

```
class ChannelPanoramio(GenericChannel):
```

With the channel class correctly installed it will have to be inserted in the layers system. There is a python script to manage this task under *social.layers.LayersConsistency*. It should be invoked inside a Django instance:

```
$> python2.5 manage.py shell
shell> from social.layers import LayerConsistency
shell> LayerConsistency.main()
```

[3] The system is case sensitive. For the Panoramio channel the directory will be channelPanoramio and then, the class name will be ChannelPanoramio. The first letter is important, so this will be used by the layer system to detect and to instance channels automatically.

This script will search new channels in the system.

3.2.4 External Social Networks API

This API provides a connection to other social networks analysed in previous deliverables. It allows a bidirectional communication between the platforms. Different connectors are developed for the interaction with each of these networks. All of them follow the next architecture schema to guarantee its integration without unnecessary changes at the user interface:

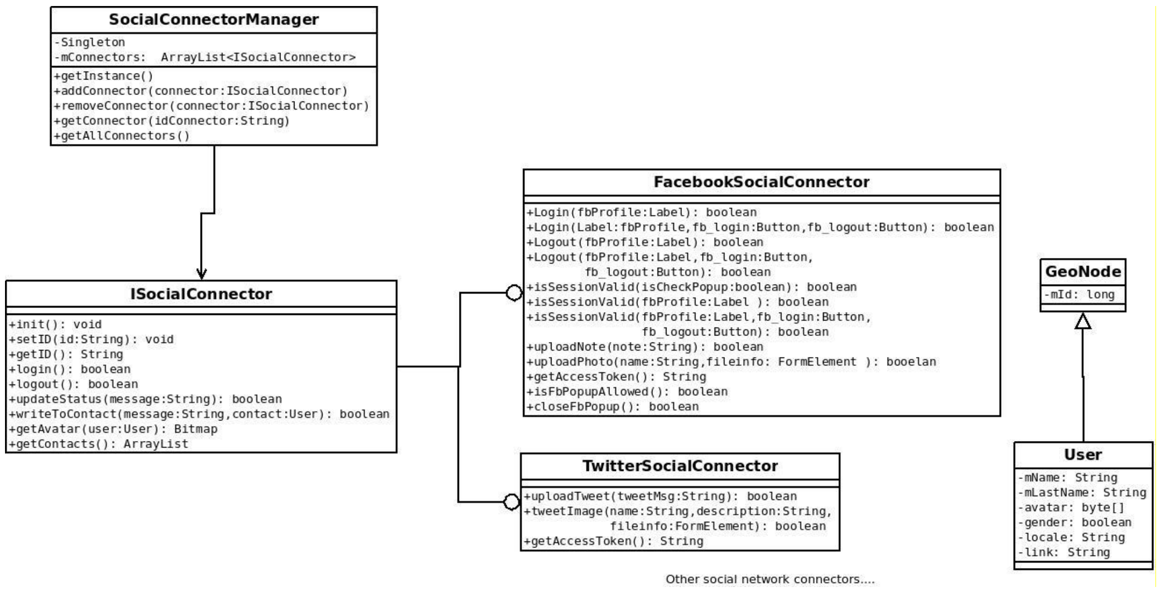

SocialConnectorManager
-Singleton
-mConnectors: ArrayList<ISocialConnector>
+getInstance()
+addConnector(connector:ISocialConnector)
+removeConnector(connector:ISocialConnector)
+getConnector(idConnector:String)
+getAllConnectors()
ISocialConnector
+init(): void
+setID(id:String): void
+getID(): String
+login(): boolean
+logout(): boolean
+updateStatus(message:String): boolean
+writeToContact(message:String,contact:User): boolean
+getAvatar(user:User): Bitmap
+getContacts(): ArrayList
FacebookSocialConnector
+Login(fbProfile:Label): boolean
+Login(Label:fbProfile,fb_login:Button,fb_logout:Button): boolean
+Logout(fbProfile:Label): boolean
+Logout(fbProfile:Label,fb_login:Button, fb_logout:Button): boolean
+isSessionValid(isCheckPopup:boolean): boolean
+isSessionValid(fbProfile:Label): boolean
+isSessionValid(fbProfile:Label,fb_login:Button, fb_logout:Button): boolean
+uploadNote(note:String): boolean
+uploadPhoto(name:String,fileinfo: FormElement): booelan
+getAccessToken(): String
+isFbPopupAllowed(): boolean
+closeFbPopup(): boolean
TwitterSocialConnector
+uploadTweet(tweetMsg:String): boolean
+tweetImage(name:String,description:String, fileinfo:FormElement): boolean
+getAccessToken(): String
GeoNode
-mId: long
User
-mName: String
-mLastName: String
-avatar: byte[]
-gender: boolean
-locale: String
-link: String
Other social network connectors....

Figure 3 Social Connector Diagram

The SocialConnectorManager is a factory class that creates all the instances of Social Connectors and manages them. With this API, the inclusion of new social networks in Go-myLife platform will be easy because only is necessary to implement the ISocialConnector interface for each social network. Implementing this interface allows the platform to obtain and post data to each specific social network, in this case, the data from/to Facebook and Twitter.

Using ISocialConnector interface, these options are possible: login/logout, update user status, get user information, get user profile picture, get user contacts, write to a contact, etc.

3.2.5 Data Analysis API

Data Analysis API of Go-myLife platform offers access to a series of tools that provide usage and performance analytics. Via the Report Dashboard, predefined and custom queries can be constructed to retrieve the desired information, using RESTful web services. Each query must be accompanied by a date range for which the query will be executed. The following table summarizes the types of the anonymised accessible data.

User demographics
• Age • Gender • Marital status • Has children • Has grandchildren
Geolocation related data (results can be sorted by country, city or province)
• Users per location • Platform access type (desktop or mobile) • Groups per location • News posts per location • Photos uploaded per location • Events per location
User activity
• Number of friendships • Number of networks • Number of local life groups • Number of photos uploaded • Number of comments posted • Number of comments received per post • Number of events created • Number of events attended
Platform access statistics
• Number of visits per user (total) • Number of visits per user (defined period) • Number of days between visits

• Percentage of new visits • Visit length   ○ Total   ○ Per section   ○ Average time (total)   ○ Average time (per section) • Day of the week and time of the day
Search content
• Search terms cloud • Types of search data

4 Security and privacy

As social networking sites such as Go-myLife become an increasingly important part of all of our lives, the information that we choose to share on these channels becomes exponentially significant and ‘risky’, inasmuch as what and how often we choose to share the things we are doing with the rest of the world. Apart from the privacy of the user content, this must not be accessible outside the social network and to send personal information like password (used in login for example) the HTTPS protocol to encrypt data must be used. Cookies will be used to save users sessions and the information of the cookie is going to be encrypted too.

4.1 Encrypted cookie

The use of cookies in our social network offers the possibility to don’t write again username and password if the user didn’t logout from Go-myLife. These cookies save username and password and web page reads that information to log in automatically if the user was previously logged in. When he/she logged out, the cookies fields where deleted.

Login mechanism needs the username and password, and then sends to the server using HTTPS encryption. The problem was that the password we obtain from the cookies will be the same we need to send using security protocol, or choose and alternative to don’t put the password visible in the cookie. Because of that, a two way encryption is a great alternative and it works in this way:

- User puts username and password and logs into the page
- Web page encrypts the username and password using two way fast AES (Advanced Encryption Standard) algorithm. This algorithm needs a key that is used to encrypt the strings, so that *key* is going to be the first 24 bytes of the MD5 hash of the user password. Now we save this values in the cookie:
    - username : encrypted username using AES and key
    - password: encrypted password using AES and key
    - mdp: md5 hash of the user’s password
- When the user didn’t logout from the social network and he/she wants to access to Go-myLife, cookies where checked.
- If fields exist in that cookie, we use AES decryption to obtain original username and password of the user:
    - Decrypt “username” field with AES and 24 first bytes of “mdp” field (md5 hashed password)
    - Decrypt “password” field with AES and 24 first bytes of “mdp” field too.
- Web page makes automatic logging using obtained username and password from the decryption.

4.2 Https

Hyper Text Transfer Protocol Secure (HTTPS) is a secure version of the Hyper Text Transfer Protocol (http). HTTPS allows secure ecommerce transactions, such as online banking. This protocol uses a Secure Sockets Layer (SSL) technology for encrypting data travelling between a web browser and a web server. To do that, an SSL certificate will be installed in the server.

The social network that the user is working with has made sure that no one can steal user private information. Using HTTPS, the computer/mobile and the server agree on a "code" between them, and then they scramble the messages using that "code". In that way no one in between can read them. This keeps user information safe from hackers when the password is required and used in the server calls to make login or modify user data in Go-myLife.

The steps we follow in Go-mylife to enable security connections are:

- Acquire and buy an SSL certificate for the website
- Install the SSL certificate on the server
- Identify calls to the server where a security connection is required and make changes to made them secure
    - login call (username and password is required)
    - modify user data call (username and password is required)

4.3 Privacy inside social network

Social networking sites have become very popular avenues for people to communicate with family, friends and colleagues from around the corner or across the globe. The privacy in this area is very important to offer the user the possibility for who wants to share information.

By default, the content that the user posts on Go-myLife can be viewed with all users inside the network, but only this content can be seen if the user is your friend or if a user makes a search or see nodes (photos, notes...) around his/her position.

The user can choose and share content with all the people inside social network, friends, and group of friends or establish as private content for personal use. With this option, the user can select the privacy of the content he/she uploads and share that content with all his friends or only a group of friends he/she selects on the posting/uploading page.

4.4 Location

Go-myLife is a location based platform. It links a location to each piece of content that is uploaded to the platform and retrieves the location of the user to offer him/her interesting content and service around him/her.

Tracking the location of the user is an option that must be controlled by him. When a user enters to the Go-myLife web application, a location tracking request appears and the user can select to approve or deny it. This information will be used to offer and discover services and content around the user.

It must be said that tracking the location of the user doesn’t mean to share it with the rest of the users. Go-myLife is aware about the vulnerability to the privacy of a user that sharing his location can imply so in consequence, it offers privacy settings where to control this option.

5 Accessibility and usability

Accessibility and usability are key components of Go-myLife. In that sense, the online social platform encompasses application of the Web Content Accessibility Guidelines provided by the World Wide Web Consortium (W3C)[4]. Making the platform more accessible to older people means that the users will perceive, understand, navigate and interact with the individual components of the platform.

Designing the platform by applying the feedback received by the users during the workshops from the beginning of the implementation, enhances the usability and the sense of equal access for each end-user. Including real people in the development process, real-life experience is provided to the developers, showing the human side of accessibility.

In order to make Web more accessible to a diverse range of people, especially the elder ones, the web accessibility and usability specifications provided by W3C are described in Web Content Accessibility Guidelines (WCAG) 2.0 and by the International Organization for Standardization (ISO)[5] in ISO 9241 part 151 (Guidance on World Wide Web user interfaces). In that sense, the following tables provide the specified guidelines and how they are met in Go-myLife platform implementation.

Web Content Accessibility Guidelines 2.0	
Specification	***Success criteria***
Many older people require large text due to declining vision, including text in form fields and other controls	Resize text without assistive technology up to 200 percent without loss of content or functionality
Actions	
• Usage of relative font-sizes • Large fonts by default • Control on the Web page to allow users to incrementally change the size of all text up to 200 percent	
Specification	***Success criteria***
Text style and its visual presentation impacts how hard or easy it is for people to read, especially older people with declining vision	Visual Presentation includes requirements on text style, text justification, line spacing, line length, and horizontal scrolling
Actions	
• Italic text is avoided • Different styles on individual pages has been avoided	

[4] http://www.w3.org

[5] http://www.iso.org

• Sufficient inter-column spacing provided	
Specification	***Success criteria***
Most older people's colour perception changes, and they lose contrast sensitivity	• Colour should not be used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element • A contrast ratio of at least 4.5:1 for the visual presentation of text and images should be applied • Contrast ratio of at least 7:1 for the visual presentation of text and images
Actions	
• High contrast between the background and the text • High contrast between the background and the images • Text captions are used in conjunction with colour in different sections	
Specification	***Success criteria***
Some older people use text-to-speech (speech synthesis) software, which is becoming increasingly available in browsers and operating systems	For non-text content a text alternative is required
Actions	
• Image buttons include text descriptions	
Specification	***Success criteria***
Older people with declining eyesight may not be able to discern the characters in a Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHA)	Alternatives to CAPTCHAs should be provided
Actions	
• No CAPTCHAs are used throughout the platform	
Specification	***Success criteria***
Many older people need links to be particularly clear and identifiable due to declining vision and cognition	• A link can be determined from the link text alone, or from the link text together with its surrounding context • A mechanism should be available to allow the purpose of each link to be identified from link text alone

	• A visible keyboard focus indicator that shows what component on the web page has focus
Actions	
• The links are visually distinct (underlined) • The links and the controls are highlighted when the mouse hovers over them	
Specification	***Success criteria***
Many older people need navigation to be particularly clear due to declining cognitive abilities	• More than one way should be available to locate a Web page within a set of Web pages • Information about the user's location within a set of Web pages should be available • Web pages should have titles that describe topic or purpose
Actions	
• Search function provided • Breadcrumb navigation provided • Current location is indicated within navigation bars • A link to the home page is provided	
Specification	***Success criteria***
It is difficult for some older people to use a mouse due to declining vision or dexterity	• Focus indicators should be visible • Labels should be provided when content requires user input • Text alternatives for any non-text content such as form controls • Text should be resizable up to 200 percent
Actions	
• The links and the controls are highlighted when the mouse hovers over them • Large buttons and clickable areas are used • Large fonts are used by default • Font size is changeable	
Specification	***Success criteria***
Some older people cannot use a mouse well or at all and instead use a keyboard	• The content should be operable through a keyboard interface • Keyboard focus should be moved away from that component using only a keyboard • All functionality of the content should be operable through a keyboard interface • A mechanism should be available to bypass blocks of content that are repeated • Components should receive focus in an order that preserves meaning and operability

	• The keyboard focus indicator has to be visible
Actions	
• Usage of HTML form controls and links to ensure that users can use the form without the mouse • A link at the top of each page allows to go directly to the main content area • The interactive elements are put in an order that follows sequences and relationships within the content • A highly visible highlighting mechanism for links or controls when they receive keyboard focus is provided	

Specification	***Success criteria***
Some older people are particularly distracted by any movement and sound on web pages	• A mechanism for the user to pause, stop, or hide moving or blinking content • interruptions should be postponed or suppressed
Actions	
• Any distraction has been avoided, such as banners, advertisements, usage of flash content and animations	

Specification	***Success criteria***
It takes some older people longer to read text and complete transactions due to declining vision, dexterity, or cognition	• Timing should not be an essential part of the event or activity presented by the web page content (except for multimedia or real-time events)
Actions	
• No time limits are used	

Specification	***Success criteria***
Many older people are inexperienced web users without advanced browsing habits and therefore read the whole page, so good page organization is important	• Headings and labels should describe topic or purpose • Section headings should be used to organize the content • Techniques to help with text organization should be used
Actions	
• Each page purpose is described by its title, where the background colour is the same as the colour of the section button • The layout of the interfaces remains always the same. On the top is the navigation section and the underneath content is separated in menu options, content and actions (organised in vertical columns)	

Specification	***Success criteria***
Many older people find it particularly difficult to understand complex sentences, unusual words, and technical jargon	• A mechanism should be available for identifying specific definitions of words or phrases used in an unusual or restricted way • A mechanism for identifying the expanded form or meaning of abbreviations should be available • The reading level should not require reading ability more advanced than the lower secondary education level
Actions	
• The language used is easily understandable by a non-computer literate user • Help documentation and tips are provided on each section to give the user detailed information of the possible actions	

Specification	***Success criteria***
For people who are new to the web, and older people with some types of cognitive decline, consistent navigation and presentation is particularly important	• Navigation should be presented in the same relative order across the website • Components with similar functionality should be identified consistently
Actions	
• The navigation does not change throughout the platform, nor over time • Similar components are represented and behave in the same way • Navigation remains the same throughout the product cycle, uses the same icons and text for both desktop and mobile version and the current location is provided by the breadcrumb navigation bar	

Specification	***Success criteria***
Some older people experiencing cognitive decline can be confused or distracted by pop-ups, new windows, or new tabs	• When any component receives focus it should not initiate a change of context • Changes of context should be initiated only by user request or a mechanism should be available to turn off such changes
Actions	
• Pop-up windows, new windows or tabs are not used • Content changes only when the user has interact with a component, not by simply selecting it	

Specification	***Success criteria***
Some older people with declining vision or cognition can miss content that automatically updates or	• When any component receives focus, it should not initiate a change of context • Changing a setting should not automatically

refreshes in a page	change the context unless the user has been advised beforehand • Changes of context should be initiated only by user request or a mechanism should be available to turn off such changes
Actions	
• No automatic updates are performed, unless the user has been advised beforehand	

Specification	***Success criteria***
It is difficult for some older people to understand the requirements of forms and transactions	• Labels or instructions should be provided when content requires user input • Context-sensitive help should be available • Components that have the same functionality within a set of web pages should identified consistently
Actions	
• A help button is present in every web page in order to provide information relevant to the interaction of the user with the content • Similar items are grouped together • Labels, names and text alternatives are consistent for content that has the same functionality • Suggestions for text input • Expected data format and examples are provided	

Specification	***Success criteria***
It is difficult for some older people to use forms and complete transactions due to declining cognitive abilities	• The users should check and correct any information they submit • If an input error is automatically detected, the item that is in error should be identified and the error should be described to the user • If an input error is automatically detected and suggestions for correction are known, then the suggestions should be provided to the user
Actions	
• Feedback messages are provided following each action taken (such as posting a comment, upload an image, accept an invitation etc) • Inform the user what irreversible action is about to happen • Provide a text description that contains information about the number of input errors, suggestions for corrections to each item, and instructions on how to proceed • Error messages are easy to understand and distinguishable from other text in the web page	

• Text descriptions are provided to identify required fields that were not completed

Web Content Accessibility Guidelines 2.0	
Specification	***Success criteria***
Many older people require large text due to declining vision, including text in form fields and other controls	Resize text without assistive technology up to 200 percent without loss of content or functionality
Actions	
• Usage of relative font-sizes • Large fonts by default • Control on the Web page to allow users to incrementally change the size of all text up to 200 percent	
Specification	***Success criteria***
Text style and its visual presentation impacts how hard or easy it is for people to read, especially older people with declining vision	Visual Presentation includes requirements on text style, text justification, line spacing, line length, and horizontal scrolling
Actions	
• Italic text is avoided • Different styles on individual pages has been avoided • Sufficient inter-column spacing provided	

ISO 9241-151 Ergonomics of human-system interaction on the World Wide Web user interfaces							
Clause	**Guideline**	**Applicability**		**Conformance**			
		Yes/ No	**Reason not applicable**	**Yes**	**Partially**	**No**	**Comments**
6	**High-level design decisions and design strategy**						
6.1	General aspects						
6.2	Determining the purpose of a Web application	**Y**		**X**			
6.3	Analysing the target user groups	**Y**		**X**			
6.4	Analysing the users' goals and tasks	**Y**		**X**			
6.5	Matching application purpose and user	**Y**		**X**			

	goals						
6.6	Recognizing the purpose of a Web application	**Y**		**X**			
6.7	Prioritizing different design goals	**Y**		**X**			
6.8	Applying ICT accessibility standards	**Y**		**X**			
6.9	Applying software accessibility standards	**Y**		**X**			
6.10	Applying content accessibility standards	**Y**		**X**			
6.11	Identifying the website and its owner	**Y**			**X**		
6.12	Coherent multi-site strategy	**N**					
7	**Content design**						
7.1	**Conceptual content model**						
7.1.1	General						
7.1.2	Designing the conceptual model	**Y**		**X**			
7.1.3	Appropriateness of content for the target group and tasks interface look and feel	**Y**		**X**			
7.1.4	Completeness of content	**Y**		**X**			
7.1.5	Structuring content appropriately	**Y**		**X**			
7.1.6	Level of granularity	**Y**		**X**			
7.2	**Content objects and functionality**						
7.2.1	General						
7.2.2	Independence of content, structure and presentation	**Y**		**X**			
7.2.3	**Selecting suitable media**						
7.2.3.1	Selecting appropriate media objects	**Y**		**X**			
7.2.3.2	Providing text equivalents for non-text media objects	**Y**		**X**			
7.2.3.3	Enabling users to	**N**	No time				

	control time-dependent media objects		dependent content				
7.2.4	Keeping the content up to date	**Y**		**X**			
7.2.5	Making the date and time of the last update available	**N**					
7.2.6	Enabling communication with the website owner	**Y**					
7.2.7	Accepting online user feedback	**Y**					
7.2.8	**Privacy and business policies**						
7.2.8.1	Providing a privacy policy statement	**Y**					
7.2.8.2	Providing a business policy statement	**N**					
7.2.8.3	User control of personal information	**Y**					
7.2.8.4	Storing information on the user's machine	**N**					
7.2.9	**Individualisation and user adaptation**						
7.2.9.1	General						
7.2.9.2	Taking account of the users' tasks and information needs	**N**	No possibility to individualize the interface				
7.2.9.3	Making individualization and adaptation evident	**N**	No possibility to individualize the interface				
7.2.9.4	Making user profiles evident	**N**	No user profiles available				
7.2.9.5	Allowing users to see and change profiles	**N**	No user profiles available				
7.2.9.6	Informing about automatically generated profiles	**N**	No user profiles available				
7.2.9.7	Switching off automatic user adaptation	**N**	No user profiles available				
7.2.9.8	Providing access to	**N**	No user				

	complete content		profiles available				
8	**Navigation and search**						
8.1	General						
8.2	**General guidance on navigation**						
8.2.1	Making navigation self-descriptive	**Y**		**X**			
8.2.2	Showing users where they are	**Y**		**X**			
8.2.3	Supporting different navigation behaviours	**Y**		**X**			
8.2.4	Offering alternative access paths	**N**					
8.2.5	Minimizing navigation effort	**Y**		**X**			
8.3	**Navigation structure**						
8.3.1	General						
8.3.2	Choosing suitable navigation structures	**Y**		**X**			
8.3.3	Breadth versus depth of the navigation structure	**Y**		**X**			
8.3.4	Organizing the navigation in a meaningful manner	**Y**		**X**			
8.3.5	Offering task-based navigation	**Y**		**X**			
8.3.6	Offering clear navigation within multi-step tasks	**Y**		**X**			
8.3.7	Combining different ways to organize navigation	**N**	Avoidance of multiple navigation ways				
8.3.8	Informative home page	**Y**		**X**			
8.3.9	Directly accessing relevant information from the home page	**Y**		**X**			
8.3.10	**Splash screens**						
8.3.10.1	Avoiding unnecessary splash screens	**Y**		**X**			
8.3.10.2	Skipping splash screens	**N**	Non applicable				

8.3.11	Avoiding opening unnecessary windows	**Y**		**X**			
8.4	**Navigation components**						
8.4.1	General						
8.4.2	Providing navigation overviews	**Y**		**X**			
8.4.3	Maintaining visibility of navigation links	**Y**		**X**			
8.4.4	Consistency between navigation components and content	**Y**		**X**			
8.4.5	Placing navigation components consistently	**Y**		**X**			
8.4.6	Making several levels of navigation visible	**N**	Non applicable				
8.4.7	Splitting up navigation overviews	**N**	Non applicable				
8.4.8	Providing a site map	**N**	Not needed				
8.4.9	Providing cross linking to potentially relevant content	**N**	Non applicable				
8.4.10	Making dynamic navigation links obvious	**Y**		**X**			
8.4.11	Linking back to the home page or landmark pages	**Y**		**X**			
8.4.12	Going back to higher levels	**Y**		**X**			
8.4.13	Providing a “step back” function	**Y**		**X**			
8.4.14	Subdividing long pages	**Y**		**X**			
8.4.15	Explicit activation	**Y**		**X**			
8.4.16	Avoiding dead links	**Y**		**X**			
8.4.17	Avoiding incorrect links	**Y**		**X**			
8.5	**Search**						
8.5.1	General						
8.5.2	**Search function**						
8.5.2.1	Providing a search function	**Y**		**X**			

8.5.2.2	Providing appropriate search functions	**Y**		**X**			
8.5.2.3	Providing a simple search function	**Y**		**X**			
8.5.2.4	Advanced search	**N**	Non applicable				
8.5.2.5	Full-text search	**N**	Non applicable				
8.5.2.6	Describing the search technique used	**Y**		**X**			
8.5.2.7	Availability of search	**N**	Only selected content is searchable				
8.5.2.8	Search field size	**Y**		**X**			
8.5.2.9	Shortcut to search function	**Y**		**X**			
8.5.2.10	Error-tolerant search	**Y**		**X**			
8.5.3	**Search results**						
8.5.3.1	Ordering of search results	**Y**		**X**			
8.5.3.2	Relevance-based ranking of search results	**N**	Non applicable				
8.5.3.3	Descriptiveness of results	**Y**		**X**			
8.5.3.4	Sorting or filtering search results	**Y**		**X**			
8.5.4	**Using search functions**						
8.5.4.1	Scope of a search	**Y**		**X**			
8.5.4.2	Selecting the scope of a search	**Y**		**X**			
8.5.4.3	Providing feedback on the volume of the search result	**Y**		**X**			
8.5.4.4	Handling large result sets	**Y**		**X**			
8.5.4.5	Showing the query with the results	**N**	Non applicable				
8.5.5	**Repeating and refining searches**						
8.5.5.1	Giving advice for unsuccessful searches	**Y**		**X**			
8.5.5.2	Repeating searches	**Y**		**X**			

8.5.5.3	Refining searches	**N**	Non applicable				
9	**Content presentation**						
9.1	General						
9.2	Observing principles of human perception	**Y**		**X**			
9.3	**Page design issues functions**						
9.3.1	General page information	**Y**					
9.3.2	Consistent page layout	**Y**		**X**			
9.3.3	Placing title information consistently	**Y**		**X**			
9.3.4	Recognising new content	**Y**		**X**			
9.3.5	Visualising temporal status	**Y**		**X**			
9.3.6	Selecting appropriate page lengths	**Y**		**X**			
9.3.7	Minimise vertical scrolling	**Y**		**X**			
9.3.8	Avoiding horizontal scrolling	**Y**		**X**			
9.3.9	Using colour	**Y**		**X**			
9.3.10	Using frames with care	**Y**		**X**			
9.3.11	Providing alternatives to frame-based presentation	**N**	No frames used				
9.3.12	Providing alternative text-only pages	**N**	No frames used				
9.3.13	Consistency across related Web sites	**N**	Non applicable				
9.3.14	Using appropriate techniques for defining the layout of a page	**Y**		**X**			
9.3.15	Identifying all pages of a Web site	**Y**		**X**			
9.3.16	Providing printable document versions	**N**	Not supported				
9.3.17	Use of "white space"	**Y**		**X**			
9.4	**Link design**						
9.4.1	General						
9.4.2	Identification of links	**Y**		**X**			

9.4.3	Distinguishing adjacent links from each other	**Y**		**X**			
9.4.4	Distinguishing navigation links from transactions	**Y**		**X**			
9.4.5	Self-explanatory link cues	**Y**		**X**			
9.4.6	Using familiar terminology for navigation links	**Y**		**X**			
9.4.7	Using descriptive link labels	**Y**		**X**			
9.4.8	Highlighting previously visited links	**N**	Not used				
9.4.9	Marking links to special targets	**N**	Not used				
9.4.10	Marking links opening new windows	**N**	Not used				
9.4.11	Distinguishing navigation links from controls	**Y**		**X**			
9.4.12	Distinguishable within-page links	**N**	Not used				
9.4.13	Link length	**Y**		**X**			
9.4.14	Redundant links	**N**	Non applicable				
9.4.15	Avoiding link overload	**Y**		**X**			
9.4.16	Page titles as bookmarks	**N**	Non applicable				
9.5	**Interaction objects**						
9.5.1	Choosing appropriate interaction objects	**Y**		**X**			
9.5.2	Making interaction objects identifiable and understandable	**Y**		**X**			
9.5.3	Providing keyboard shortcuts	**N**	Non applicable				
9.6	**Text design**						
9.6.1	Readability of text	**Y**		**X**			
9.6.2	Supporting text skimming	**N**	Non applicable				
9.6.3	Writing style	**Y**		**X**			
9.6.4	Text quality	**Y**		**X**			

9.6.5	Identifying the language used	**Y**		**X**			
9.6.6	Making text resizable by the user	**Y**		**X**			
10	**General design aspects**						
10.1	**Designing for cultural diversity and multilingual use**						
10.1.1	General						
10.1.2	Showing relevant location information	**N**					
10.1.3	Identifying supported languages	**Y**		**X**			
10.1.4	Using appropriate formats, units of measurement or currency	**Y**		**X**			
10.1.5	Designing presentation of text in different languages	**Y**		**X**			
10.2	Providing help	**Y**		**X**			
10.3	**Making Web user interfaces error-tolerant**						
10.3.1	Minimizing user errors	**Y**		**X**			
10.3.2	Providing clear error messages	**Y**		**X**			
10.4	URL names	**Y**		**X**			
10.5	Acceptable download times	**Y**		**X**			
10.6	Using generally accepted technologies and standards	**Y**		**X**			
10.7	Supporting common technologies	**Y**		**X**			
10.8	Making Web user interfaces robust	**Y**		**X**			
10.9	Designing for input device independence	**Y**		**X**			
10.10	Making the user interface of embedded objects usable and accessible	**Y**		**X**			

6 Final prototype

6.1 Functionalities of the prototype

The final prototype will offer the following functionalities:

News (Wall): A user will be able to post, comment and delete notes/news in his/her wall. Other members of his/her social network will be able to see what they post (sharing) and make comments about the shared content.

Media: A user will be able to edit/delete/upload media content such as photos providing some information about it (title, description, location …). The members of his/her network will see the shared media and could comment on it.

Events: Users will be able to create events selecting the place, date and a list of friends they want to invite to the event. Also, users can modify or delete created events and add more friends to the invited list (these options only if the user created the event). Every time the user has a pending event to accept, it will appear a notification.

My network: Through “My network”, users will be able to arrange their contacts in groups such as close family, neighbors, friends, etc. The rest of the content in Go-myLife (news, photos, etc.) will be displayed by these groups and they can share their photos, news, etc. with a selected group. The user has the option to create group of users, delete them, add or delete friends to/from a group and delete relationship with a friend. Also in this section, the user can accept/reject pending friendship invitations and every time the user has a pending friendship invitation to accept, Go-myLife will show a notification.

Local Life: The workshops *(see Chapter 2: Feedback from pilots and workshops with end-users)* also remarked the importance for the elderly of participating in the social life of the neighborhood. This could be done through “Local Life” section, where the user could access to local associations/groups. Each group will have its own news, media, events, etc.

Messages: Users of Go-myLife will be able to exchange messages among them. They have the option to show, delete or reply to the received messages.

Search: The elder people sometimes find overwhelming the content that people upload to the social networks. In order to help them in that task, there will be a search field. They can search for new users.

Forums: Users will be able to start a discussion about any particular topic or answer an existing one. Users could post anonymously or not.

Me: a profile section where the user could view and edit his/her personal information and set a profile picture.

Location: A key factor of Go-myLife project is the geolocated information. The content shared by users and even the own users will have a location associated, so content could be displayed in a map. Sections such as “Around me” will display relevant information and friends near the user.

Privacy Settings: A section to define the settings of sharing and private information exchange. A user can allow or avoid that users inside the social network can see their location.

Help: To assist the elderly to navigate through the application, there will be a “Help” section with tutorial videos or explanations of the screens and workflow.

Other social networks: The content from other social networks (such as Facebook and Twitter) will be integrated in the existing sections. For example, news/photos that the user posts will appear in these social networks.

LibreGeoSocial framework, which Go-myLife will be based on, gives the possibility to consume the content of third-parties. Go-myLife will take advantage of this featuring to provide to the users a greater experience. In that way, in mobile section, there will be a special section for **Panoramio**, displaying the photos in a map to show interesting information around the user.

6.2 UI design

A set of mockups were done before creating the UI. They represent the most important screens in the Go-myLife desktop version. We used Balsamiq (www.balsamiq.com) mainly for two reasons: it is a free on-line tool, so there is no need to install any application and could be used from several operating systems; and the designs are exportable both in PNG and XML formats (for latter editing).

6.2.1 Desktop mockups

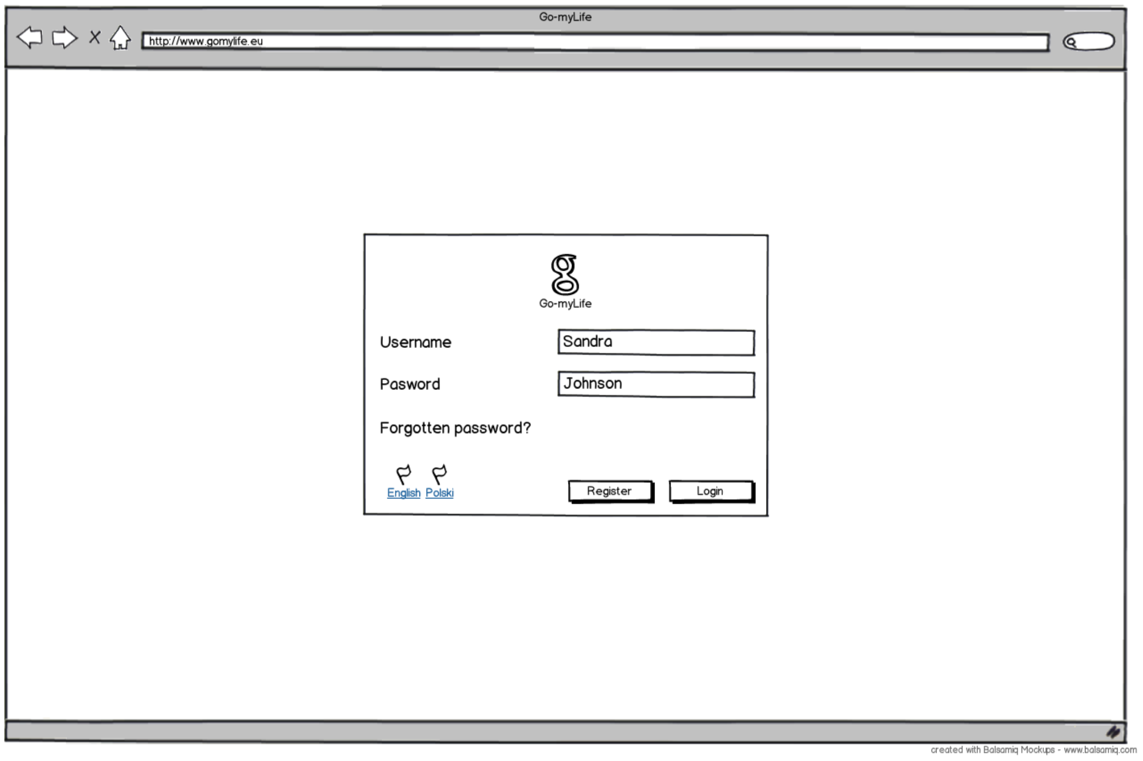
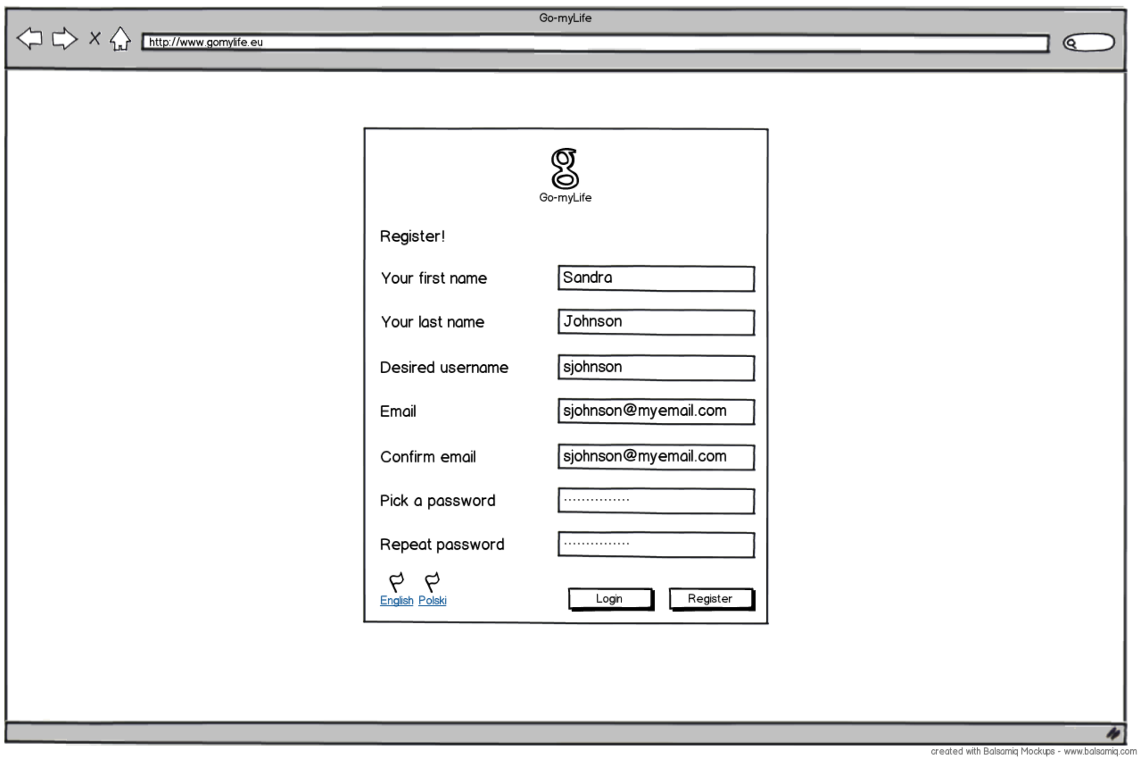


Figure 4: Login form mockup



Go-myLife

http://www.gomylife.eu

Go-myLife

Register!

Your first name	Sandra
Your last name	Johnson
Desired username	sjohnson
Email	sjohnson@myemail.com
Confirm email	sjohnson@myemail.com
Pick a password	···············
Repeat password	···············

English Polski

Login Register

created with Balsamiq Mockups - www.balsamiq.com

Figure 5: Registration form mockup

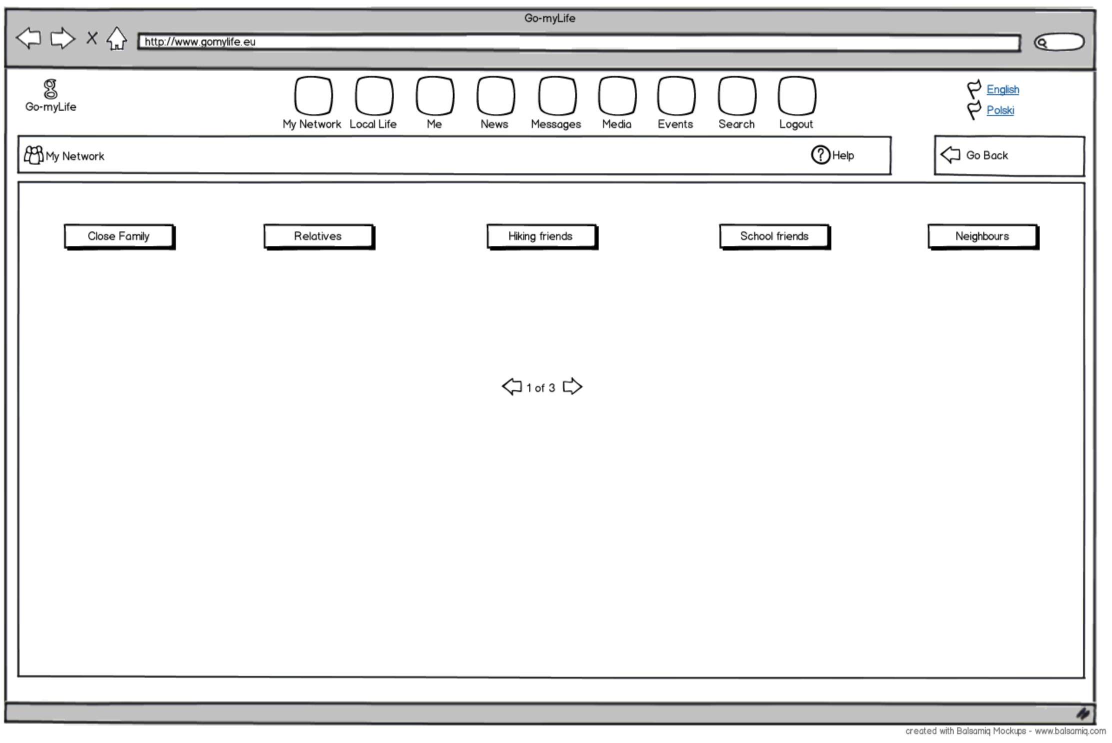
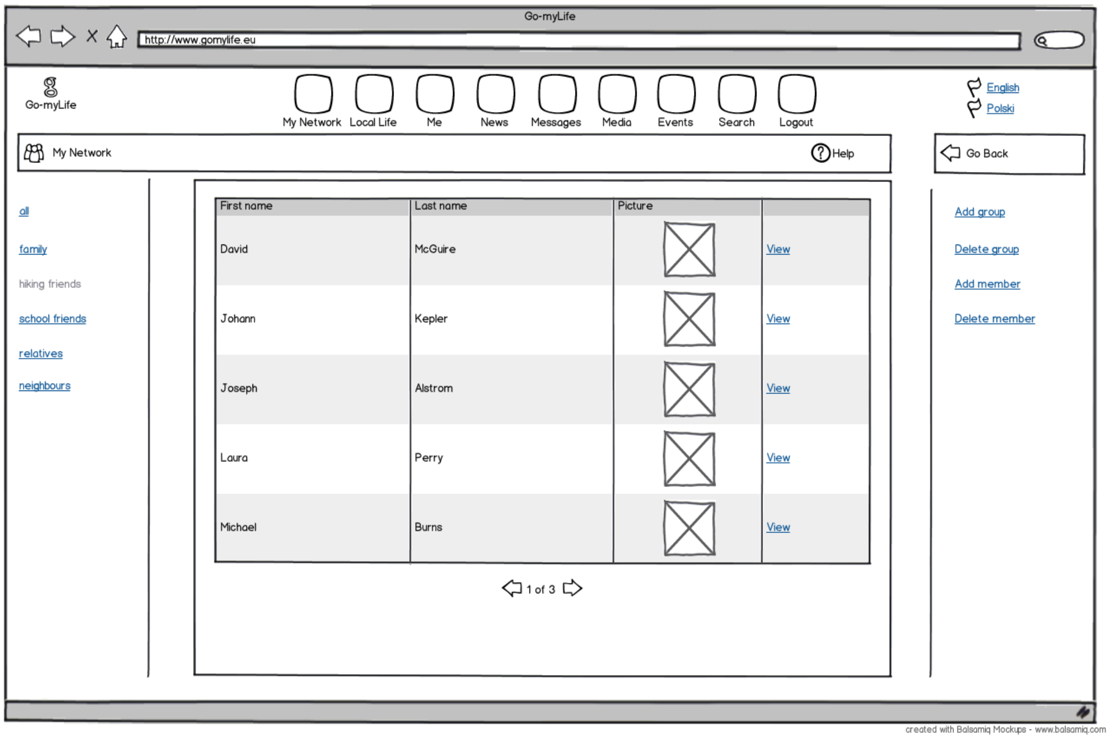


Figure 6: My Network section mockup



First name	Last name	Picture	
David	McGuire		View
Johann	Kepler		View
Joseph	Alstrom		View
Laura	Perry		View
Michael	Burns		View

Figure 7: List of friends mockup

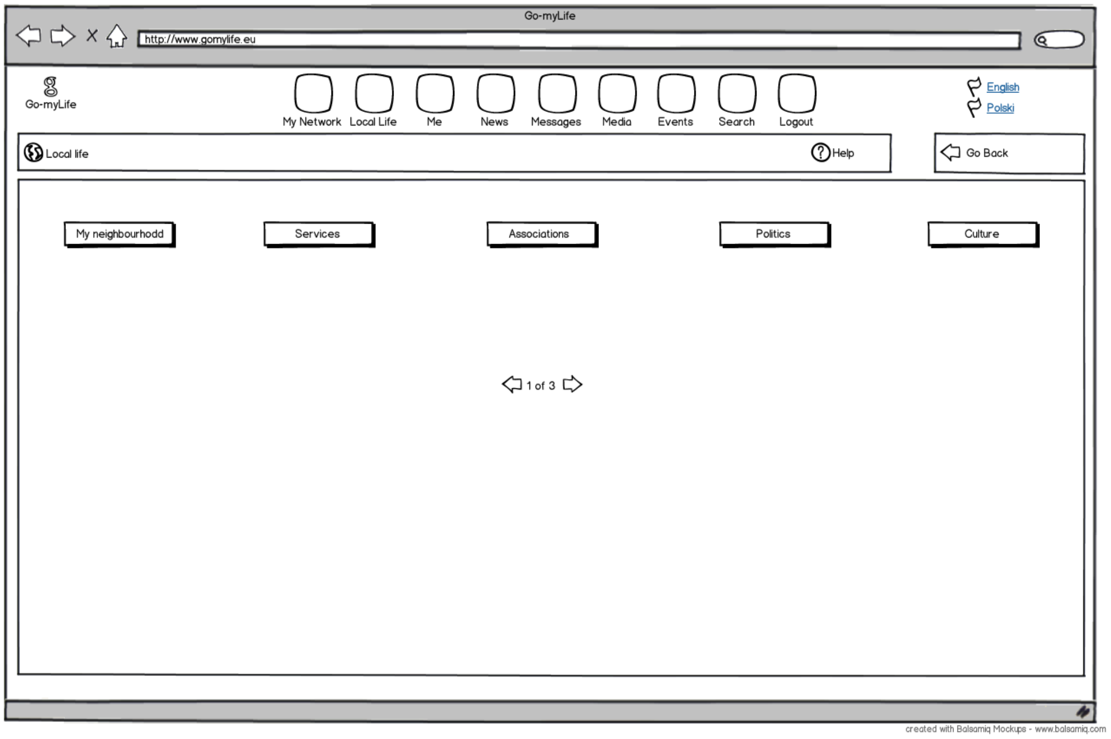
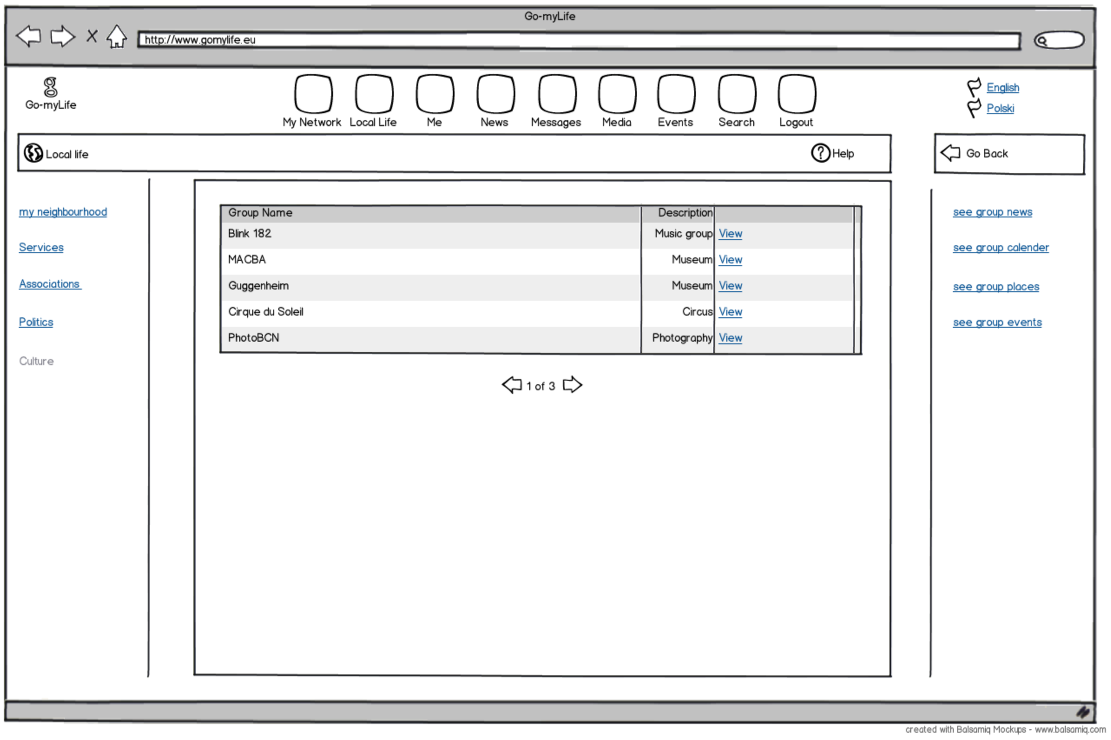


Figure 8: Local Life section mockup



Group Name	Description	
Blink 182	Music group	View
MACBA	Museum	View
Guggenheim	Museum	View
Cirque du Soleil	Circus	View
PhotoBCN	Photography	View

Figure 9: List in Local Life mockup

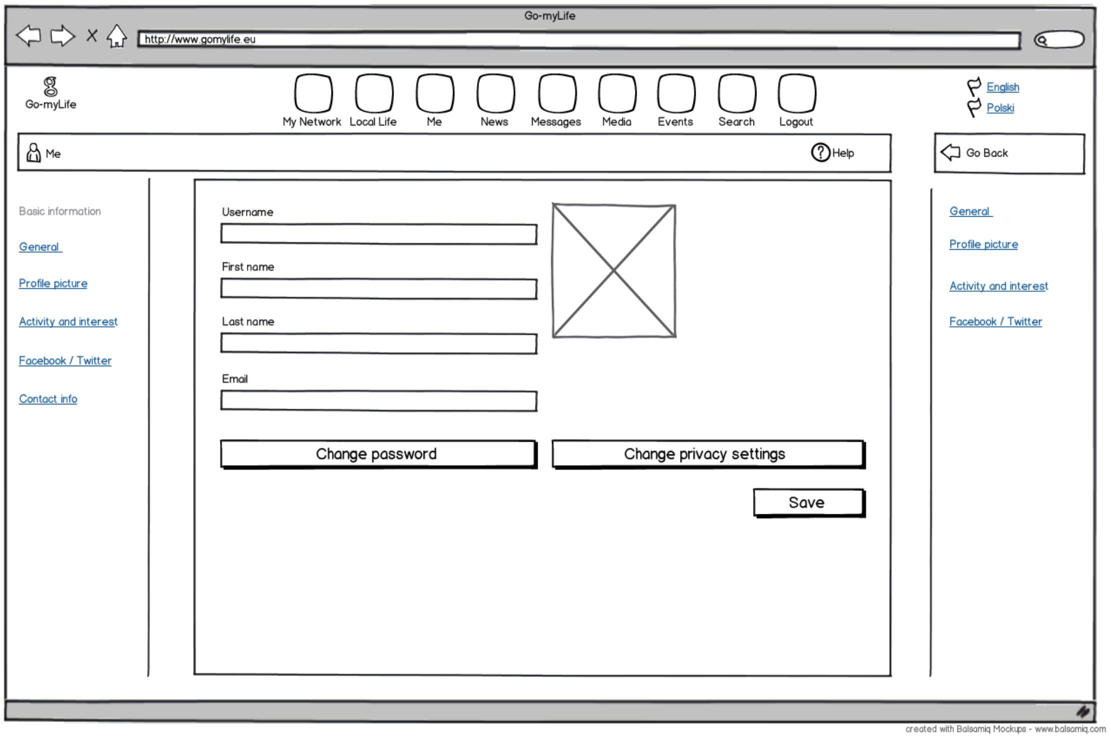
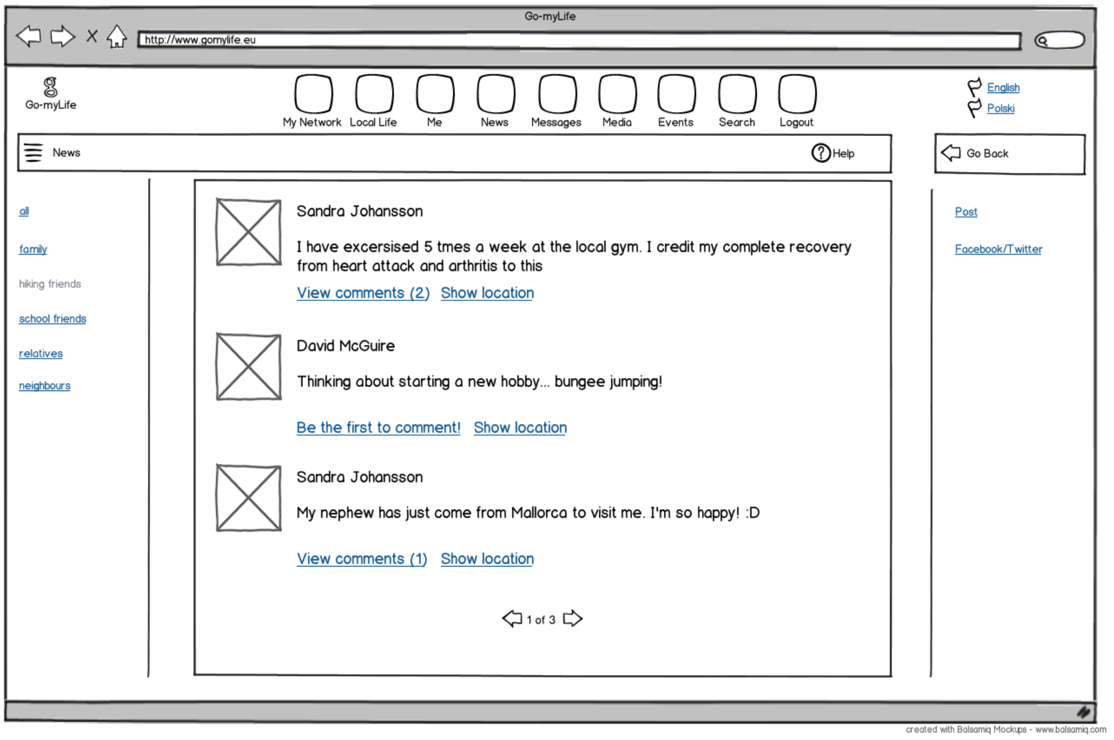


Figure 10: Basic information mockup



Sandra Johansson

I have excersised 5 tmes a week at the local gym. I credit my complete recovery from heart attack and arthritis to this

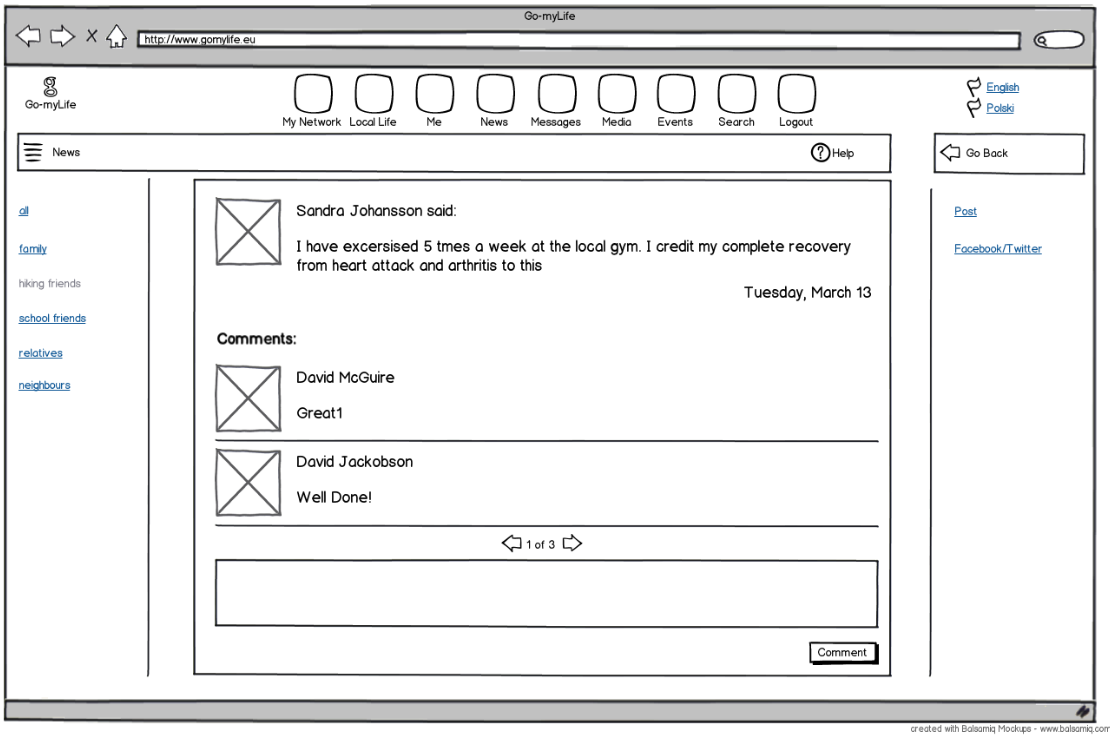
David McGuire

Thinking about starting a new hobby... bungee jumping!

Sandra Johansson

My nephew has just come from Mallorca to visit me. I'm so happy! :D

Figure 11: News list mockup



Sandra Johansson said:

I have excersised 5 tmes a week at the local gym. I credit my complete recovery from heart attack and arthritis to this

Tuesday, March 13

Comments:

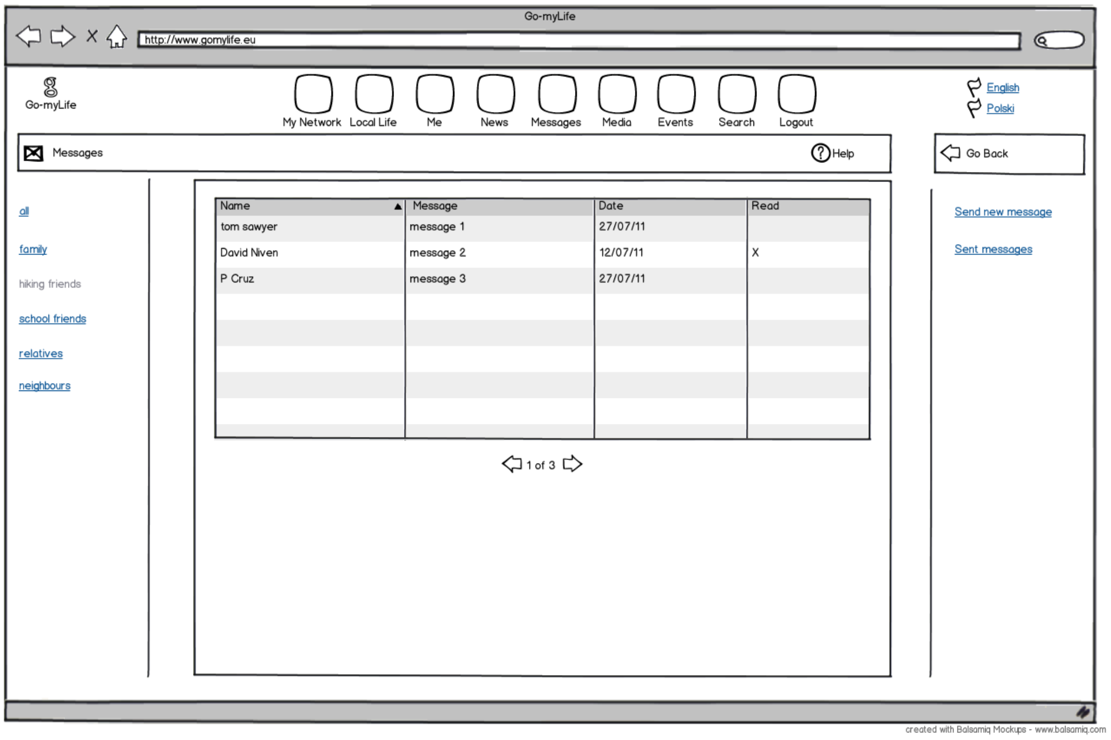
David McGuire

Great1

David Jackobson

Well Done!

Figure 12: Comments for a piece of news mockup



Name	Message	Date	Read
tom sawyer	message 1	27/07/11	
David Niven	message 2	12/07/11	X
P Cruz	message 3	27/07/11	

Figure 13: Messages list mockup

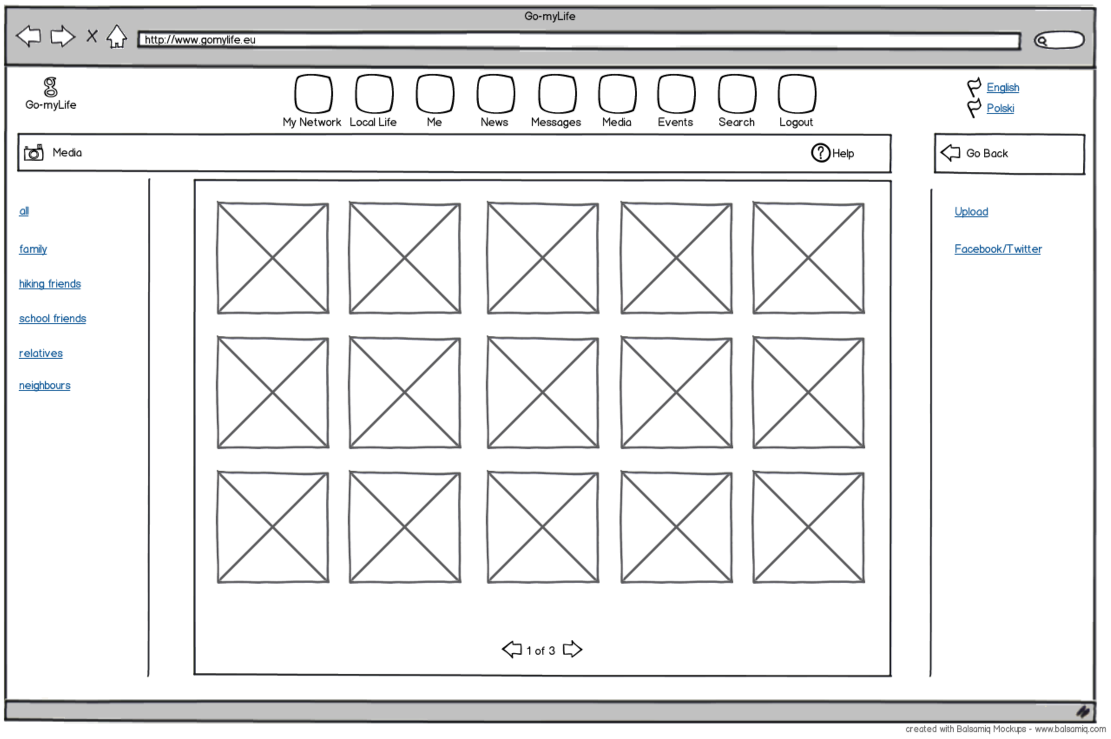
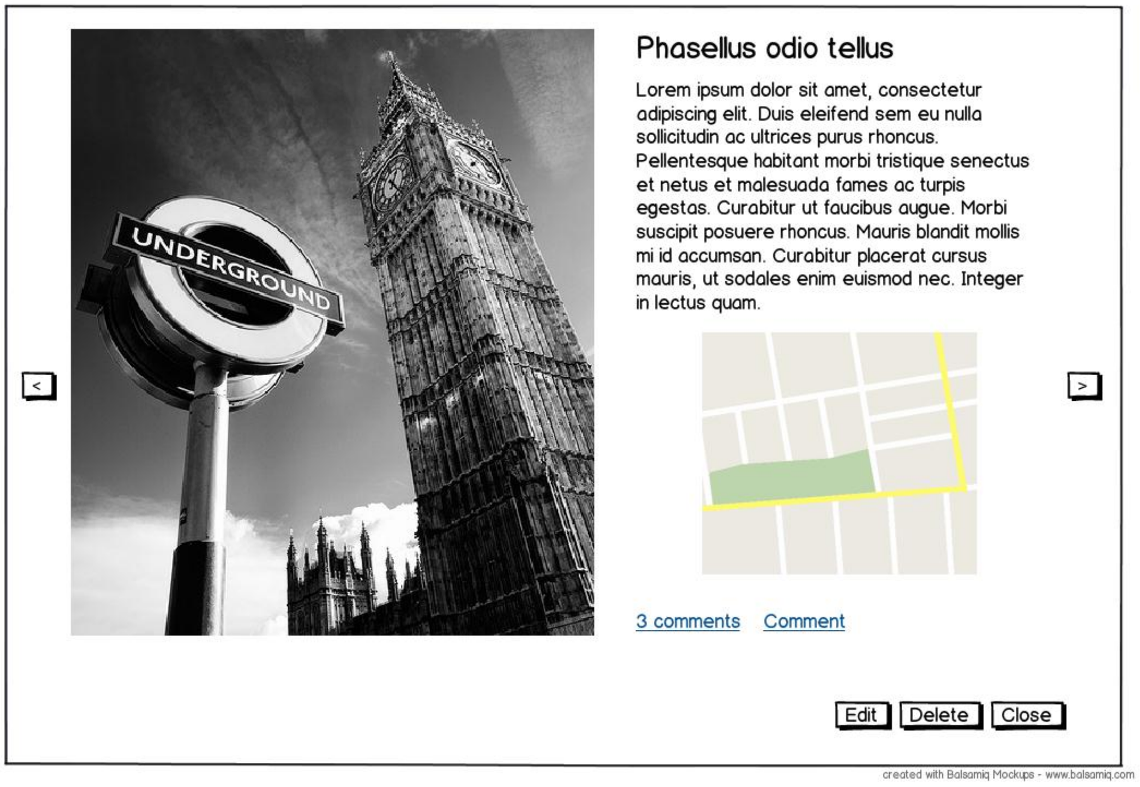
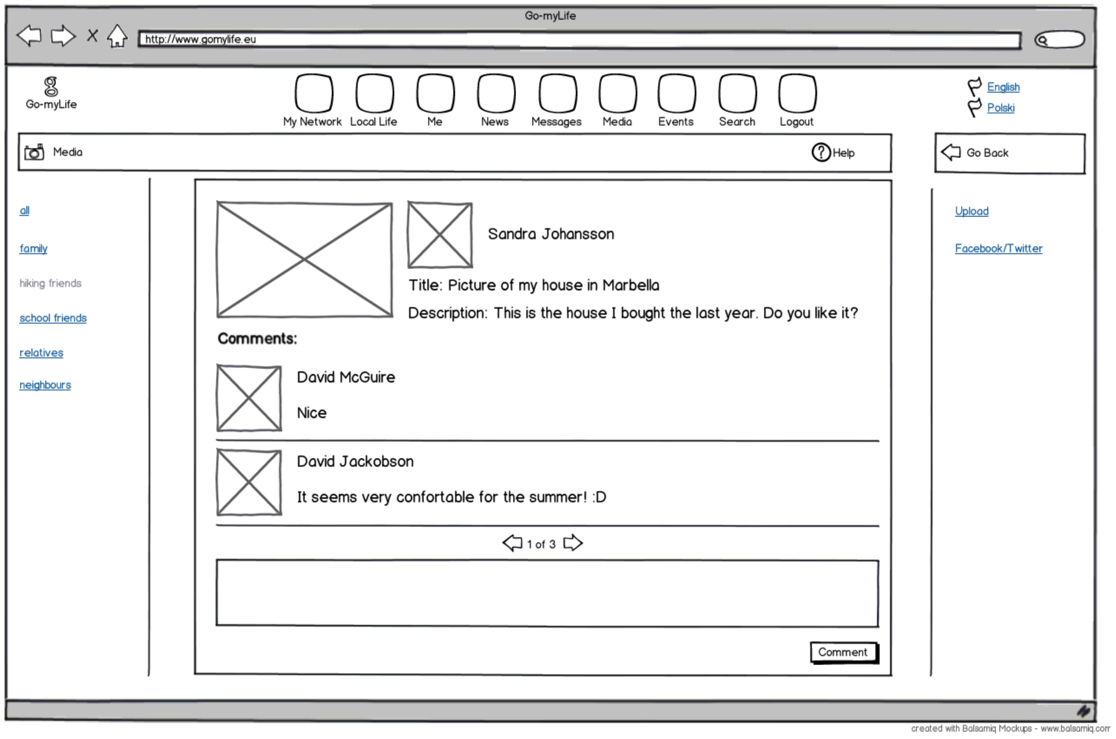


Figure 14: Media grid mockup


Phasellus odio tellus

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis eleifend sem eu nulla sollicitudin ac ultrices purus rhoncus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Curabitur ut faucibus augue. Morbi suscipit posuere rhoncus. Mauris blandit mollis mi id accumsan. Curabitur placerat cursus mauris, ut sodales enim euismod nec. Integer in lectus quam.

Figure 15: Photo details mockup


Sandra Johansson

Title: Picture of my house in Marbella

Description: This is the house I bought the last year. Do you like it?

Comments:

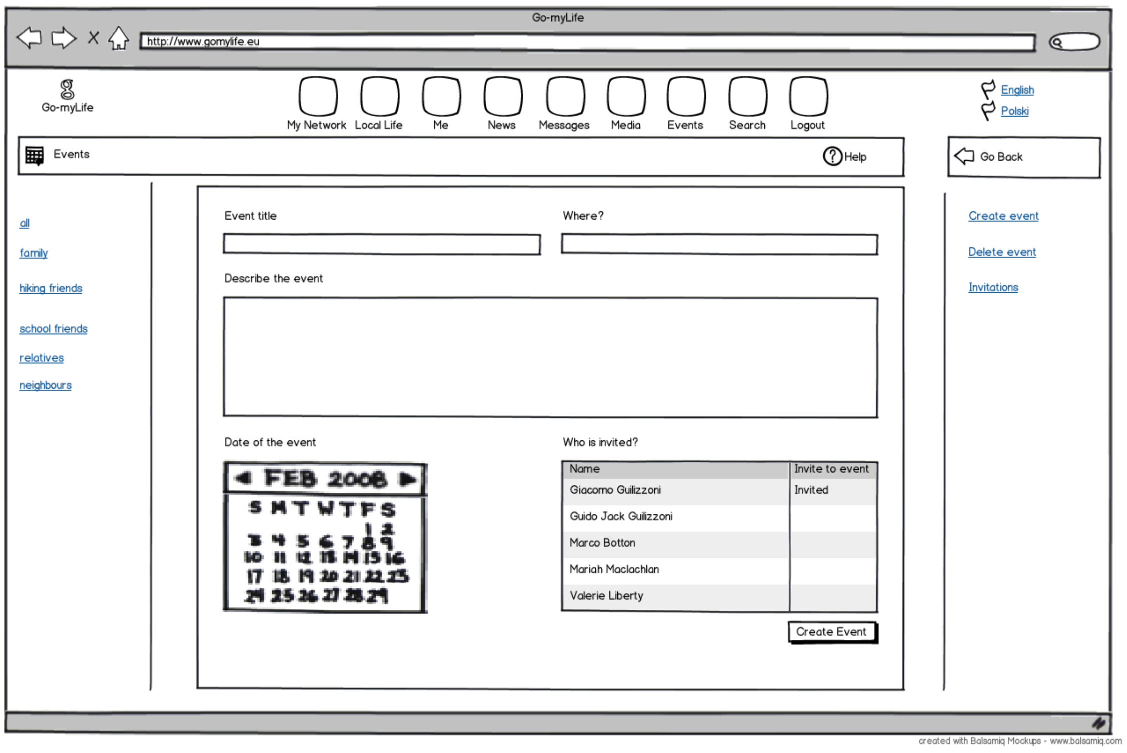
David McGuire

Nice

David Jackobson

It seems very confortable for the summer! :D

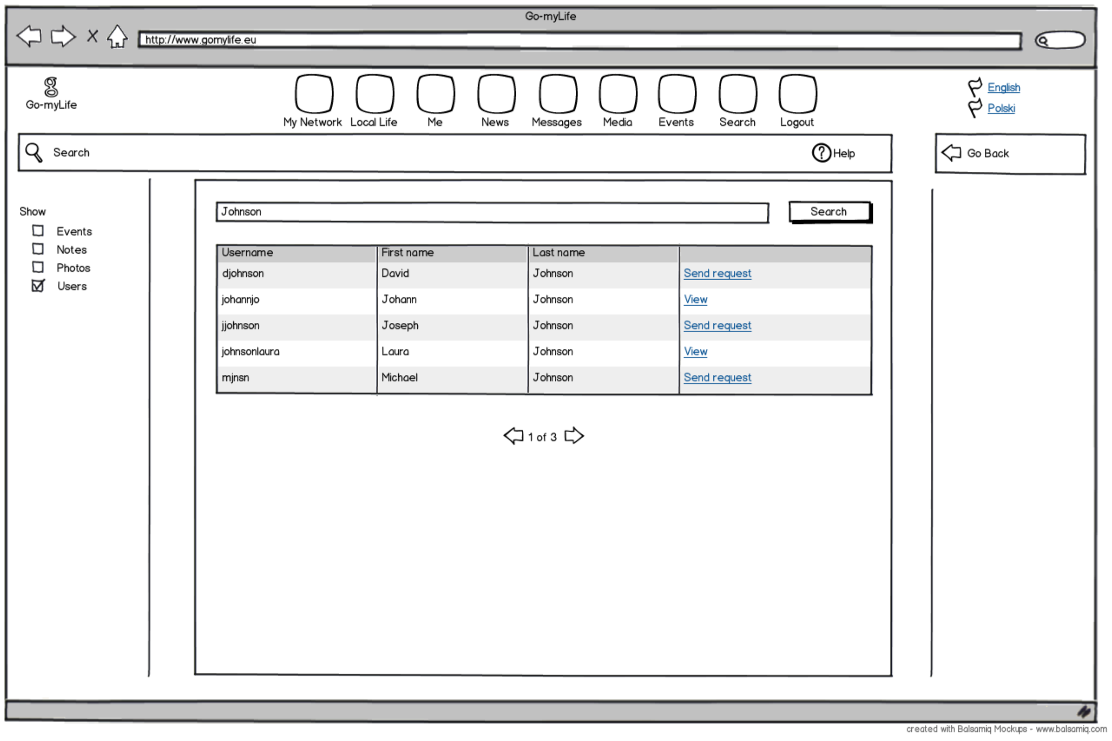
Figure 16: Comments for an image mockup



S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

Name	Invite to event
Giacomo Guilizzoni	Invited
Guido Jack Guilizzoni	
Marco Botton	
Mariah Maclachlan	
Valerie Liberty	

Figure 17: Create Event form mockup



Show: ☐ Events ☐ Notes ☐ Photos ☑ Users

Johnson

Username	First name	Last name	
djohnson	David	Johnson	Send request
johannjo	Johann	Johnson	View
jjohnson	Joseph	Johnson	Send request
johnsonlaura	Laura	Johnson	View
mjnsn	Michael	Johnson	Send request

Figure 18: Search display mockup

6.2.2 Mobile mockups

In the following section, the final mock-ups of the mobile user interfaces are provided.

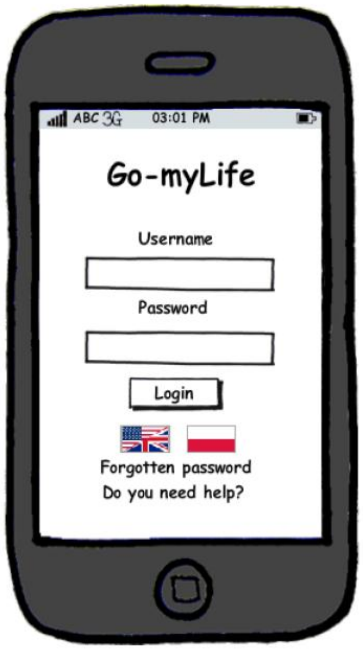


Figure 19 Login screen

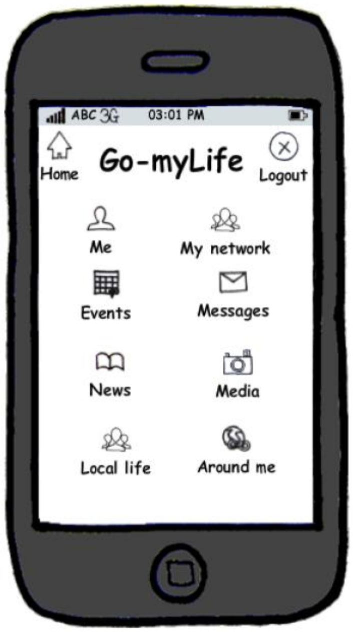
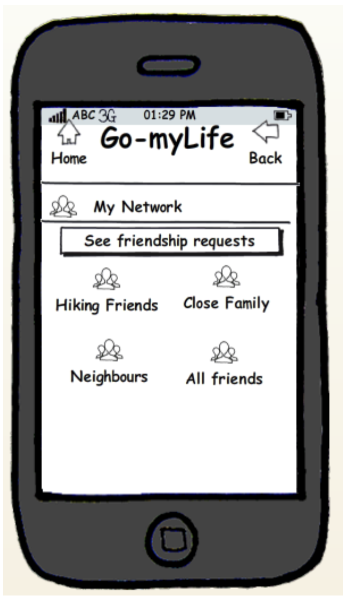
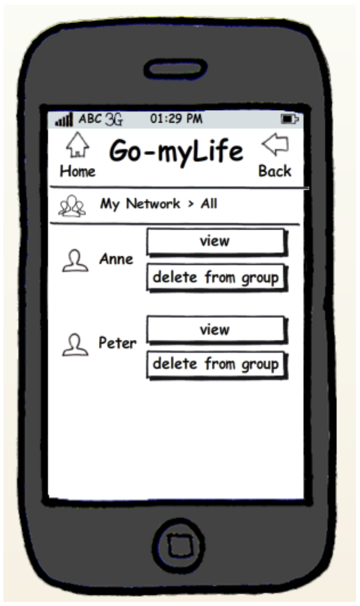


Figure 20 Main menu





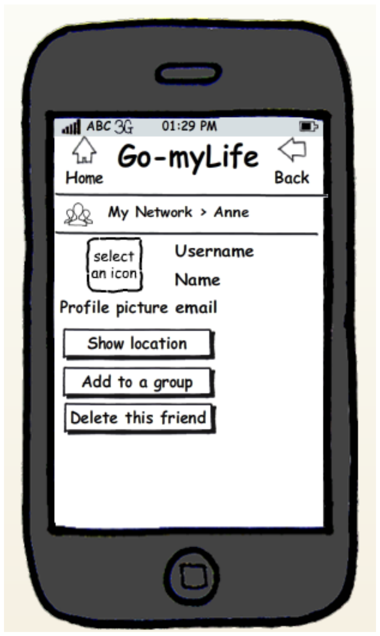
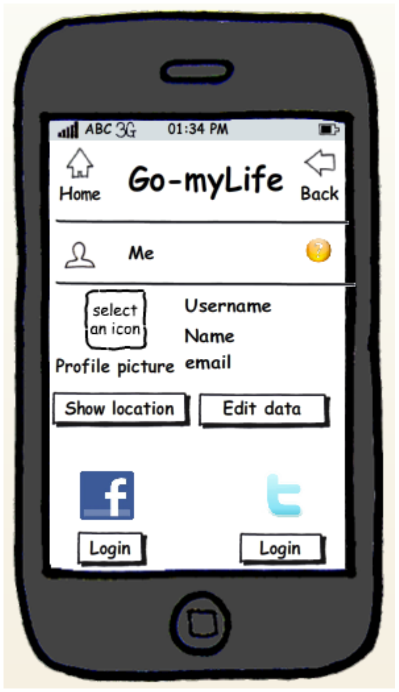
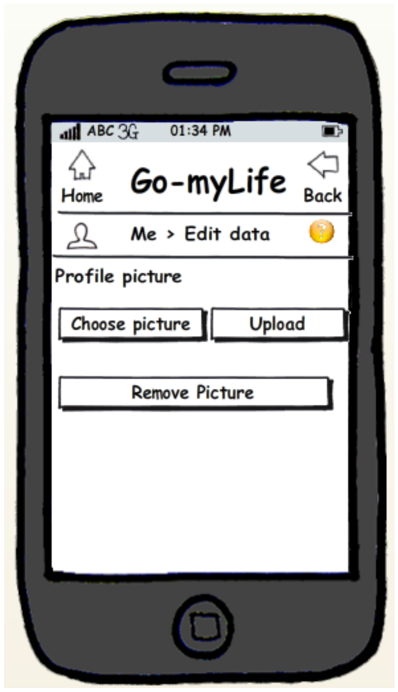


Figure 21 My network





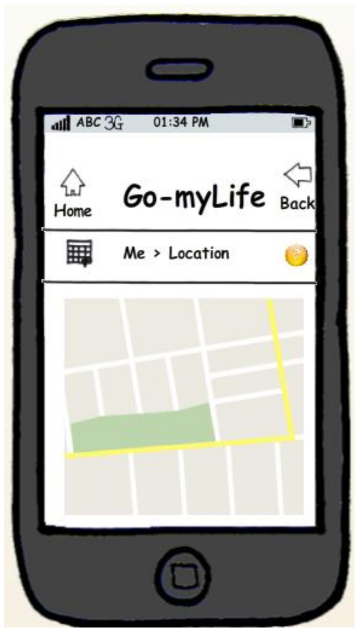
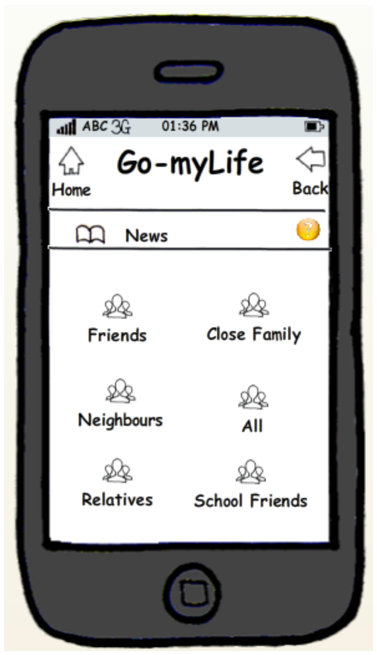
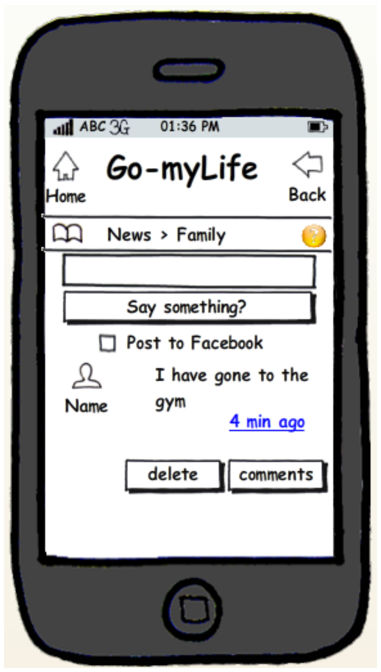


Figure 22 My profile





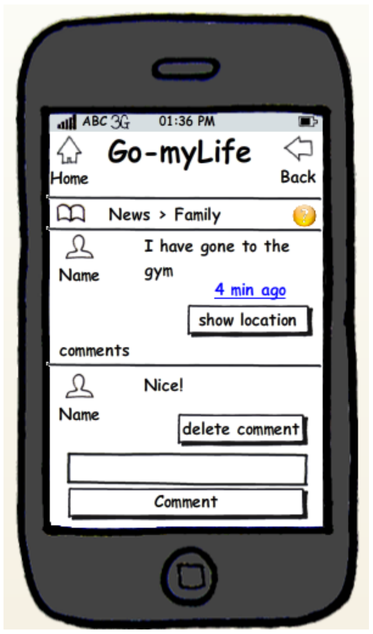
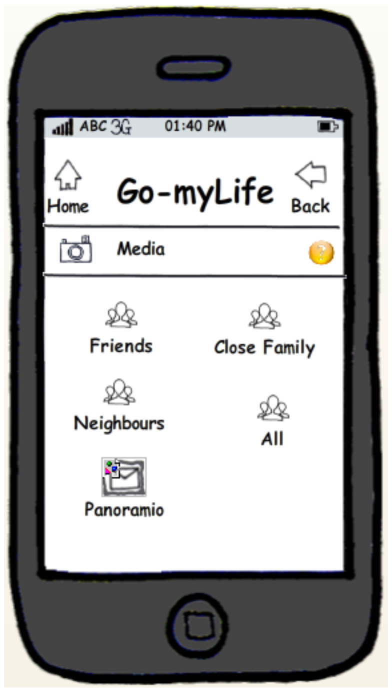
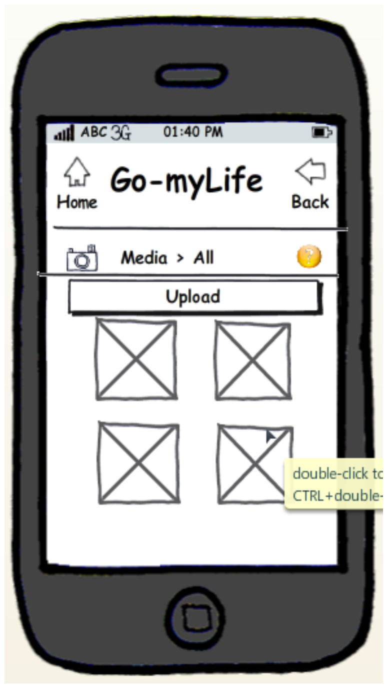
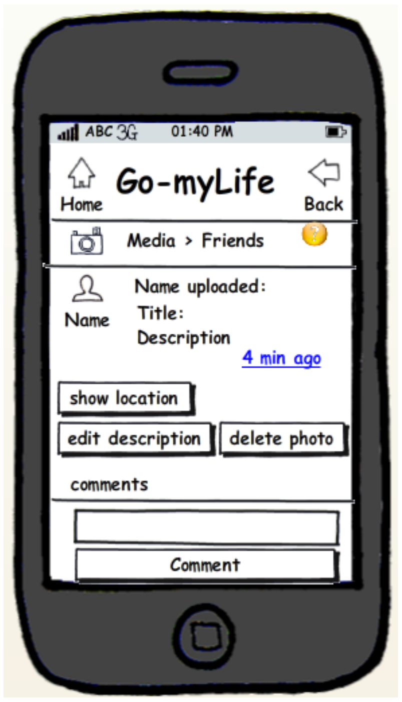
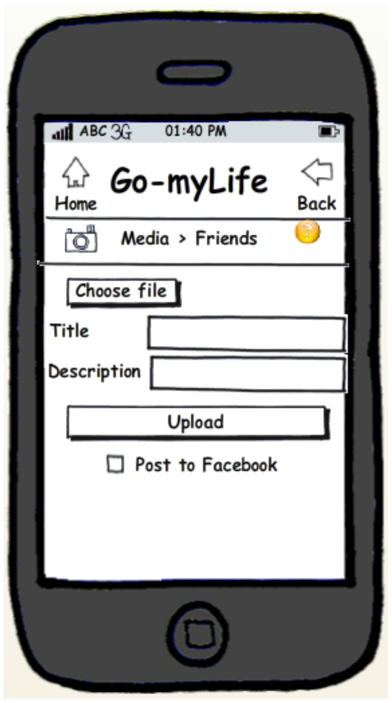


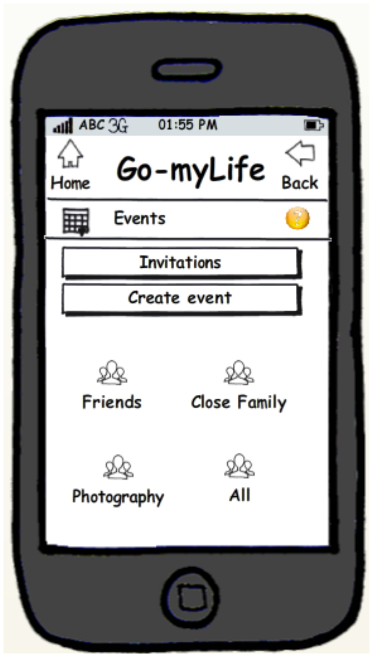
Figure 23 News

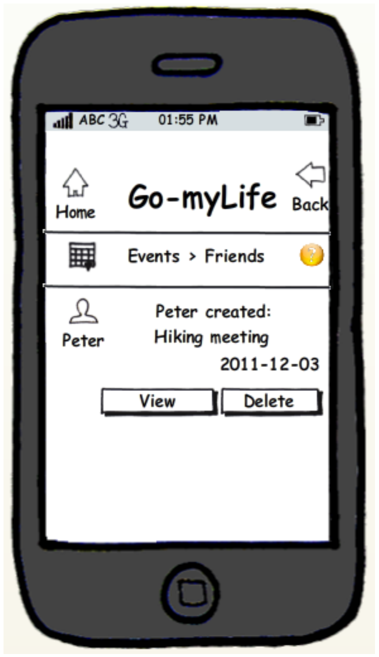





ABC 3G 01:40 PM
Home
Go-myLife
Back
Media > Friends
Name
Name uploaded:
Title:
Description
4 min ago
show location
edit description
delete photo
comments
Comment


ABC 3G 01:40 PM
Home
Go-myLife
Back
Media > Friends
Choose file
Title
Description
Upload
Post to Facebook


ABC 3G 01:55 PM
Home
Go-myLife
Back
Events
Invitations
Create event
Friends
Close Family
Photography
All


ABC 3G 01:55 PM
Home
Go-myLife
Back
Events > Friends
Peter
Peter created:
Hiking meeting
2011-12-03
View
Delete

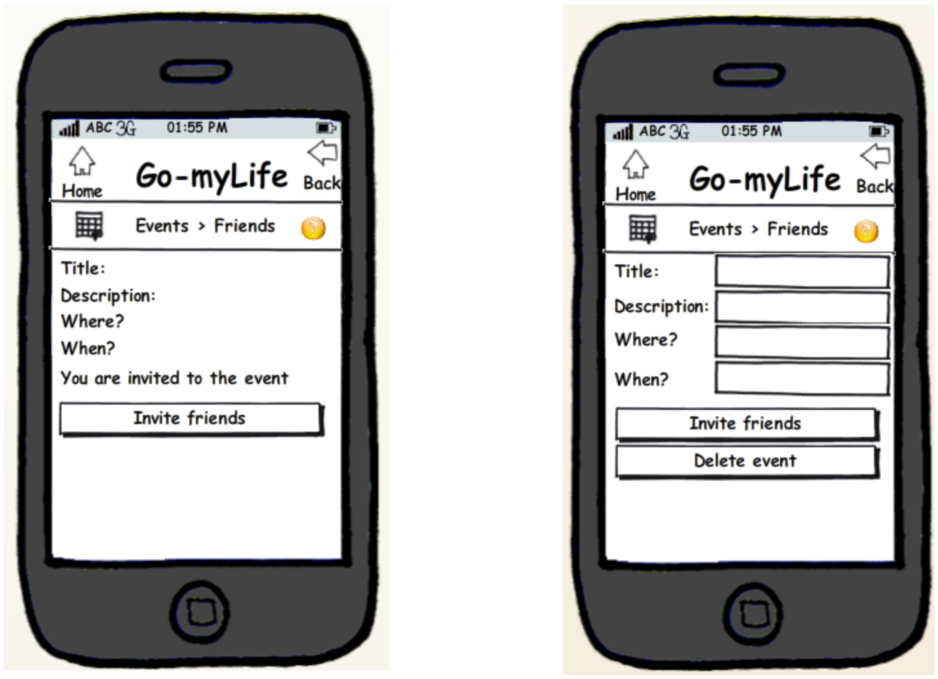


Figure 24 Eventos

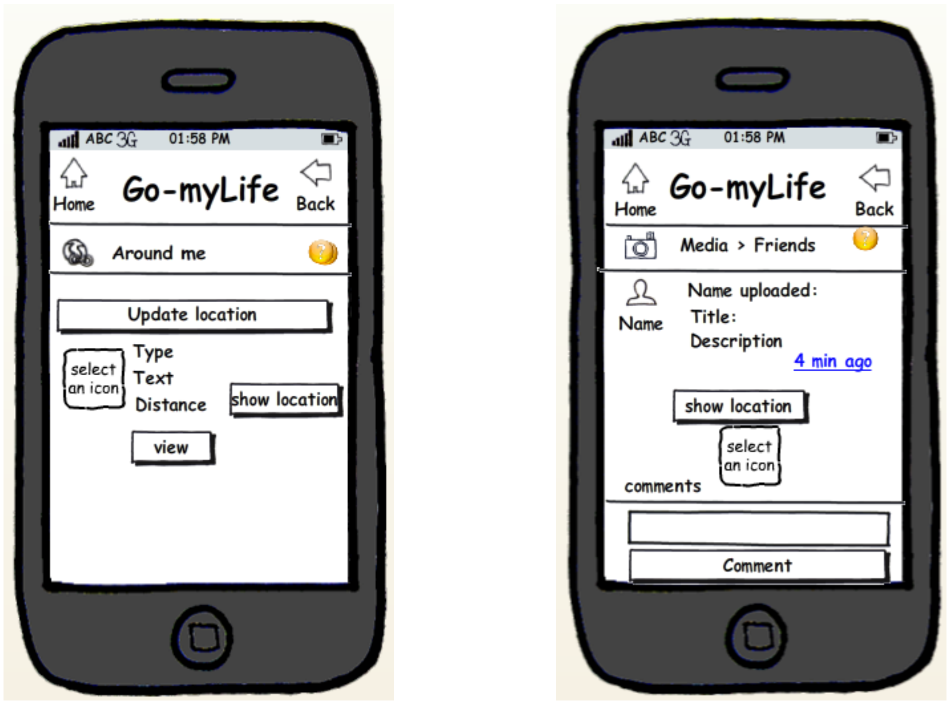
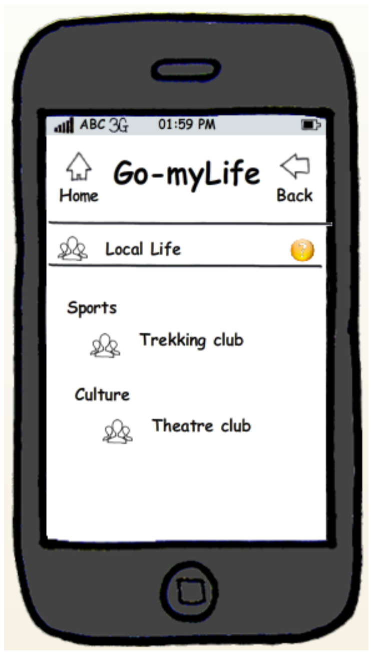


Figure 25 Around me



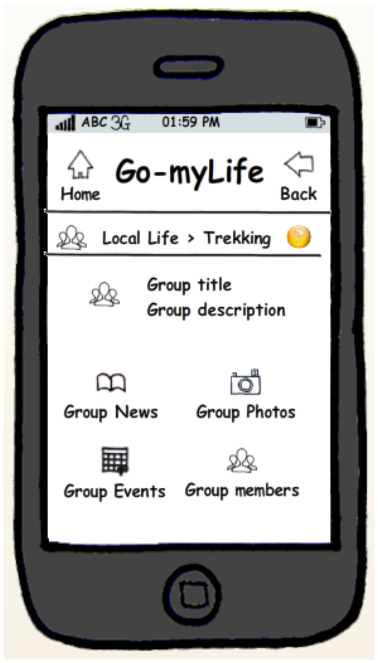
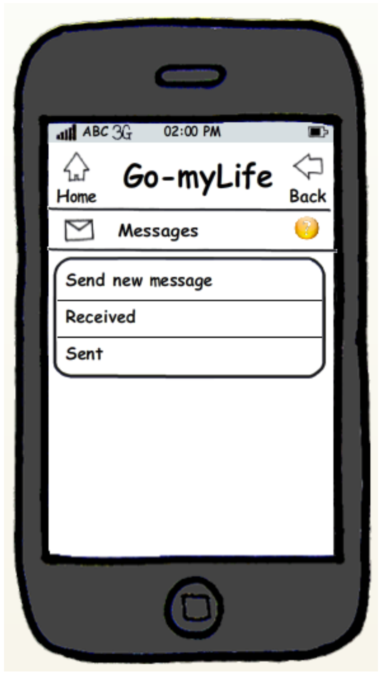
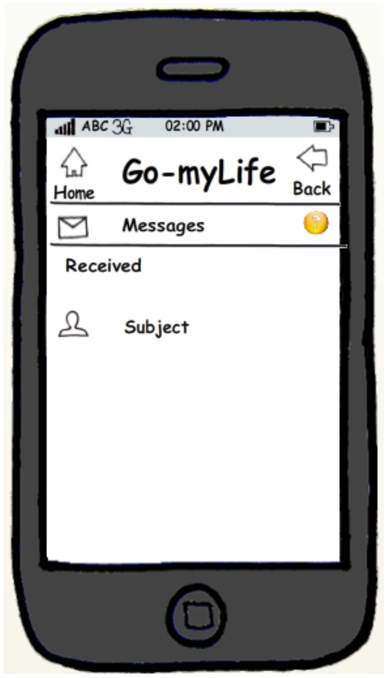
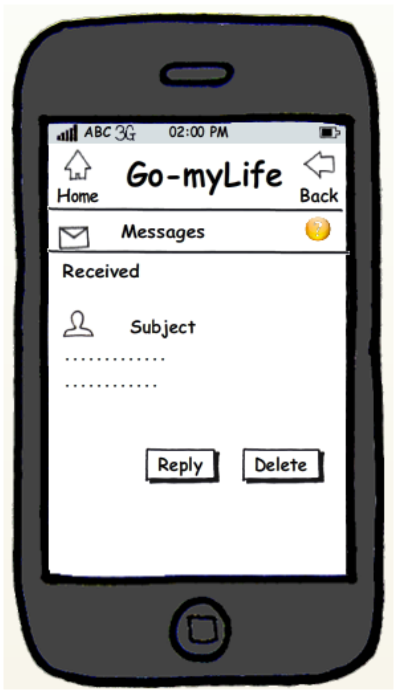
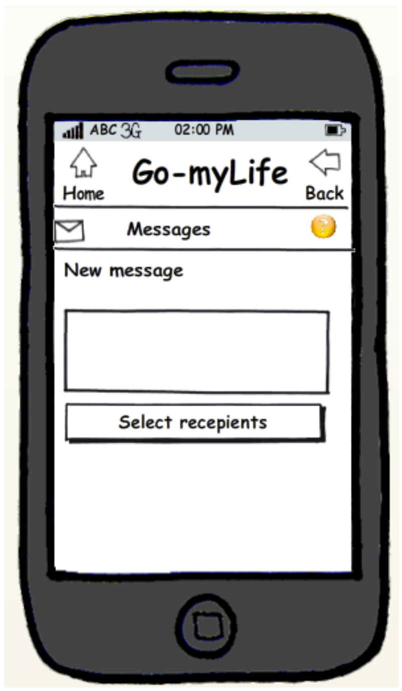


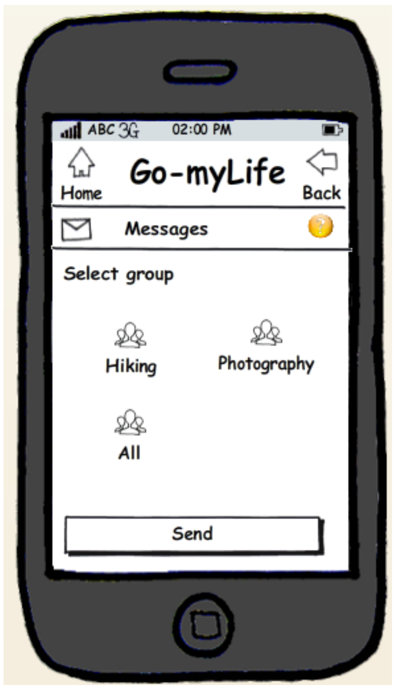
Figure 26 Local Life

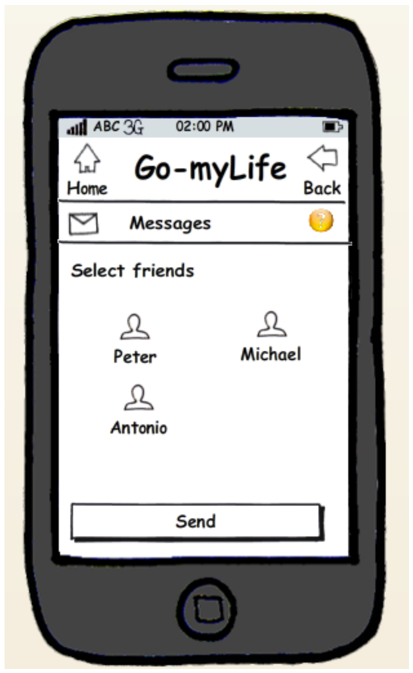





ABC 3G 02:00 PM
Home
Go-myLife
Back
Messages
Received
Subject
Reply
Delete


ABC 3G 02:00 PM
Home
Go-myLife
Back
Messages
New message
Select recepients


ABC 3G 02:00 PM
Home
Go-myLife
Back
Messages
Select group
Hiking
Photography
All
Send


ABC 3G 02:00 PM
Home
Go-myLife
Back
Messages
Select friends
Peter
Michael
Antonio
Send

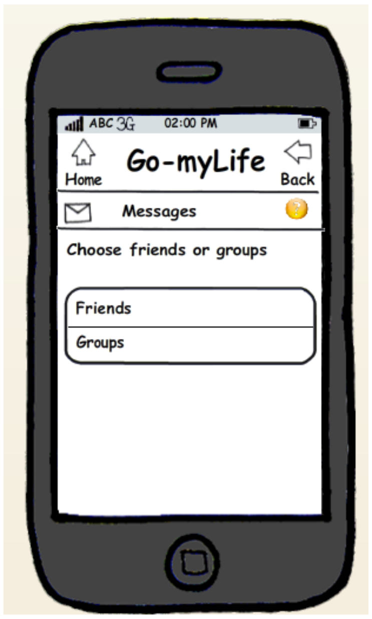


Figure 27 Messages

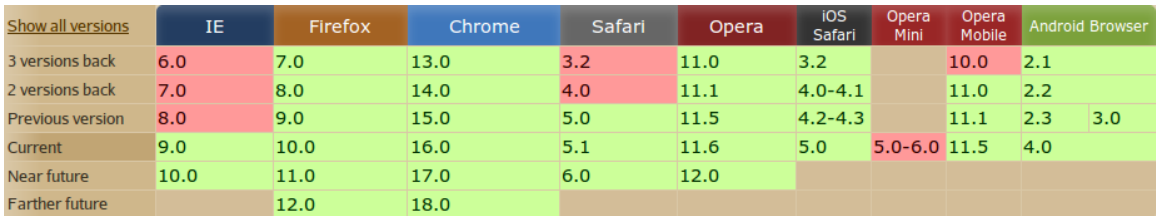
6.3 *HTML5 features*

In addition to specifying markup, HTML5 specifies scripting new attractive APIs like drag-and-drop, geolocation, history, File API, XmlHttpRequests2, CORS, etc. In Go-myLife social network it is necessary to use a few of them and explain why we need those APIS in our implementation.

6.3.1 Geolocation

One of the interesting and useful additions to the HTML5 specification is the Geolocation API. The Geolocation API allows users to share their location with web applications, so they can enjoy the benefits of various location-aware services.

Geolocation enables you as a developer or website owner to figure out where a particular user is located on the planet. This is useful in Go-myLife because it is a social network based on geolocation, where you can find out where your various friends are currently located or what is around you by getting latitude and longitude parameters from the browser using JavaScript.



Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
3 versions back	6.0	7.0	13.0	3.2	11.0	3.2		10.0	2.1
2 versions back	7.0	8.0	14.0	4.0	11.1	4.0-4.1		11.0	2.2
Previous version	8.0	9.0	15.0	5.0	11.5	4.2-4.3		11.1	2.3 3.0
Current	9.0	10.0	16.0	5.1	11.6	5.0	5.0-6.0	11.5	4.0
Near future	10.0	11.0	17.0	6.0	12.0				
Farther future		12.0	18.0						

Figure 28: browsers support for Geolocation API

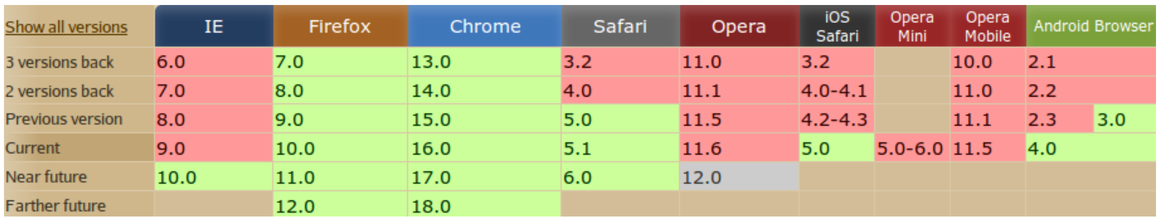
The previous figure shows that all of the actual browsers (except opera mini) have compatibility with this HTML5 feature, so the use of this capacity will not have any problems in the browsers.

6.3.2 XMLHttpRequests2

XMLHttpRequest is a JavaScript object that can send arbitrary HTTP requests from the user browser to a Web server in order to submit or retrieve information in the background. It is often used to develop the so called AJAX Web applications, i.e. applications that interact with Web servers in a faster way, as they usually exchange information with the Web server without having to load a new page in the browser. With this object we can make POST and GET methods to the REST server and obtain the response from it. Using POST method, post parameters data are included with a pairs of key/values (example: name=jon&lastname=aldridch) to send to the server. In Go-myLife the REST server response returns a JSON with information we need to parse and display/use in our social network.

The original XMLHttpRequest specification **did not support uploading files from the user machine**. The problem is that developers would have to craft the whole HTTP request with the contents of the files being uploaded. For that, they would need to have access to the actual contents of the files to be uploaded. Until recently, in JavaScript that was not possible due to security reasons.

With the release of HTML5, the solution to this problem comes with XMLhttpRequests 2. This new version adds support for the new FormData interface. FormData objects provide a way to easily construct a set of key/value pairs representing form fields and their values, which can then be easily sent using the XMLHttpRequest send() method. The best part of the FormData object is that it will also include file input content and we can upload files to the server using XmlHttpRequests methods.



Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser	
3 versions back	6.0	7.0	13.0	3.2	11.0	3.2		10.0	2.1	
2 versions back	7.0	8.0	14.0	4.0	11.1	4.0-4.1		11.0	2.2	
Previous version	8.0	9.0	15.0	5.0	11.5	4.2-4.3		11.1	2.3	3.0
Current	9.0	10.0	16.0	5.1	11.6	5.0	5.0-6.0	11.5	4.0	
Near future	10.0	11.0	17.0	6.0	12.0					
Farther future		12.0	18.0							

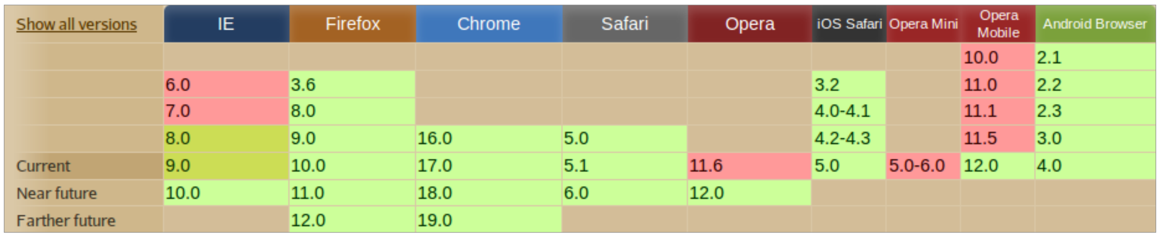
Figure 29: browsers support for XMLhttpRequests2

Right now, only Safari, Chrome, Firefox and Android +3.0 browsers have support for this specification methods, the option to send a file to the server from the user machine and get response from it. It will be important the implementation of this specification on the next version of IE because elderly people are most familiar with this browser.

6.3.3 CORS

One of the main problems that web developers found in the AJAX technology was the **same-origin policy**. This policy prevented from making HTTP requests through the *XMLHttpRequest* object to other domains than the current one. The most popular trick to avoid this setback was to implement a proxy module into their own servers that was in charge of making these communications. However, the community thought that it was better to have a browser native way to do this, so they wrote the Cross-Origin Resource Sharing (CORS) W3C Working Draft.

To use the CORS capabilities through the *XMLHttpRequest* object, they must to be added some new headers to the HTTP transactions. First, to tell the server that we are using CORS we must set the *Origin* header in a request, where we will specify the origin of the requesting page (host), and other headers to ask for further functionality, mainly: *Access-Control-Request-Method* to request the availability of some desired method and *Access-Control-Request-Header* to check the availability of some custom headers. Then the server determines whether allowing CORS or not. If it allows CORS, then he will set some new headers in the response, mainly: *Access-Control-Allow-Origin* to indicate the availability or CORS for the requested origin, *Access-Control-Allow-Methods* to indicate if the requested HTTP methods are allowed, *Access-Control-Allow-Headers* to indicate the availability of the custom headers requested, *Access-Control-Max-Age* to indicate the maximum time that the resource will be cached, and *Access-Control-Allow-Credentials* to indicate that the server will accept credentials (cookies, SSL certificates...). So this is the way we have to aboard the cross-domain communications natively with AJAX, without introducing proxies or other tricks.



Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
								10.0	2.1
	6.0	3.6				3.2		11.0	2.2
	7.0	8.0				4.0-4.1		11.1	2.3
	8.0	9.0	16.0	5.0		4.2-4.3		11.5	3.0
Current	9.0	10.0	17.0	5.1	11.6	5.0	5.0-6.0	12.0	4.0
Near future	10.0	11.0	18.0	6.0	12.0				
Farther future		12.0	19.0						

Figure 3: browsers support for CORS

As we can see in the previous figure, CORS is currently supported by almost all main web browsers, except Opera (standard and mini versions), so it should not be a problem for the elderly users' devices to use this technology.

6.4 Alternatives for Google Maps

Months ago, the Google Developer team announced that the era of unlimited Google Maps usage for free is officially over. Now, developers whose apps load more than 25,000 basic maps or 2,500 stylized maps per day will have to cough up some cash. So, the usage of Google Maps API in Go-myLife network is questionable. That's why we have studied options of mapping tools for our social network:

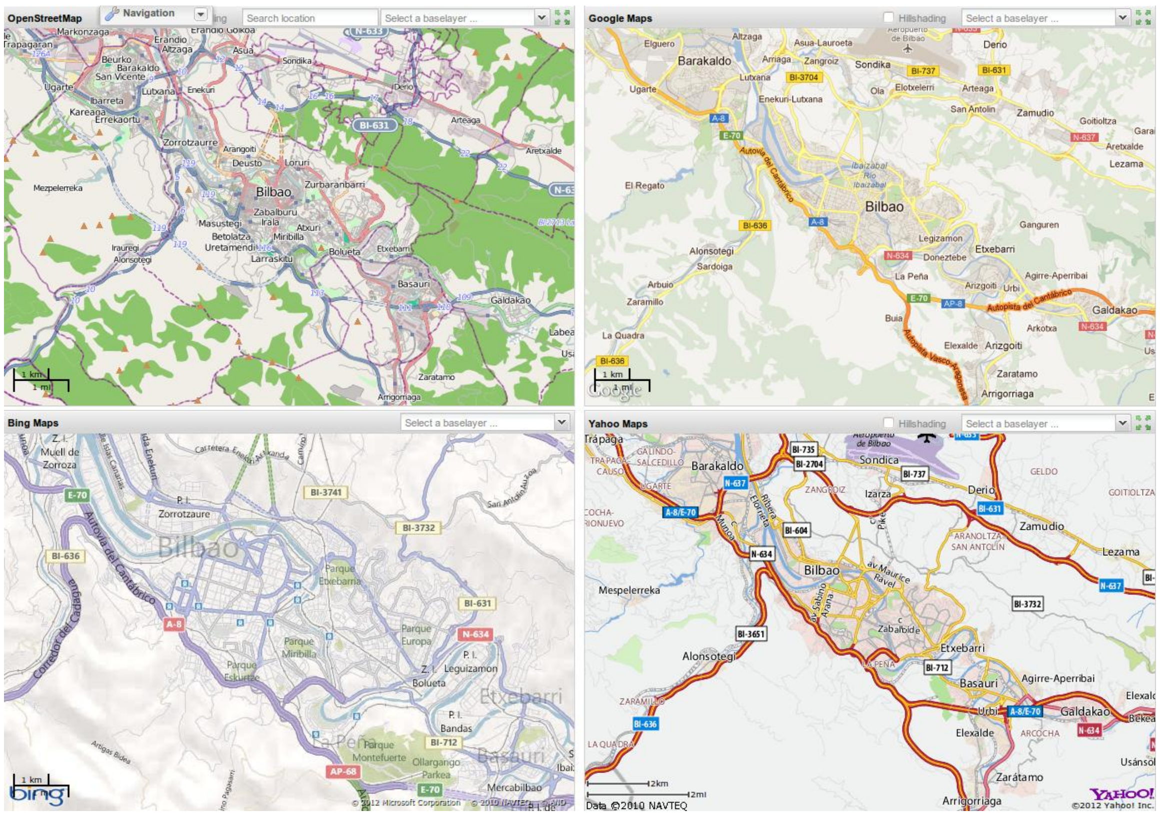


Figure 30: Maps comparison (1-OpenStreetMap, 2-Google Maps, 3-Bing Maps and 4-Yahoo Maps)

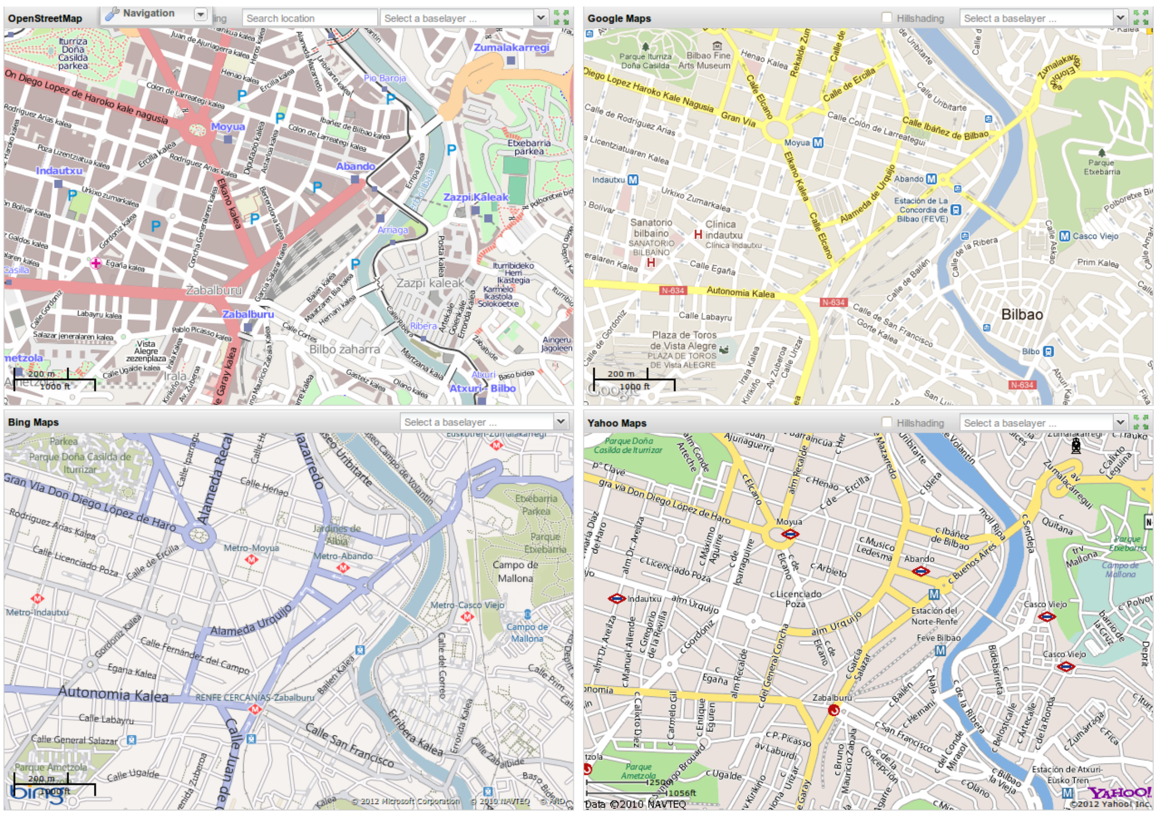


Figure 31: City comparison (1-OpenStreetMap, 2-Google Maps, 3-Bing Maps and 4-Yahoo Maps)

- Yahoo Maps: compared to Google Maps, it has similar interface and at street level buildings are more detailed. An application ID is required to use the Yahoo Maps AJAX API.
- OpenStreetMap: it is a free map done by users (uploading information from GPS or other sources) and updated every day. It has very good detailed information on the map and the style looks nice. Parts of the map are good differenced by colours, and it will be fine for elderly people. The perspective in Figure 30 looks nice, especially in green zones and roads information. Integration of OpenStreetMap can be done free and without a key registration using API.
- Bing Maps: similar interface compared to Yahoo but using non contrast colours on it. For elderly people this must be difficult to understand the map or differentiate elements on it. To use Bing Maps API, a key is required registering with Windows Live ID and getting it.

According to different maps and the corresponding APIs they provide, the principal aspect to take into account is the license issue with the API we use. OpenStreetMap is an open source map and we can use it with OpenLayers free API without key registering. Maybe in the future Yahoo and Bing do the same as Google Maps and they limit their maps visits without paying. Apart from that, OpenStreetMap style looks good and it has more detailed places every day thanks to the users.

6.4.1 OpenStreetMap

OpenStreetMap (OSM) is a free and editable map. Its community of users works to collect data and add/update maps so they can be made available to anyone for free. The location data comes from a variety of sources, such as recordings from GPS devices, free satellite imagery, and users’ own knowledge of an area. Any user can go in and add new information to a map or change existing information if they find errors in it (much like Wikipedia). The advantage of this is that everyday more information is updated and the maps will have more detailed information with users contribution. So, some parts are quite well covered by OpenStreetMap. In Figure 32 we can see a part of Bilbao in a map. It provides information about parkings, streets, parks, neighbourhoods, etc. The colour difference between the parts of the map makes it attractive, especially for elderly people.

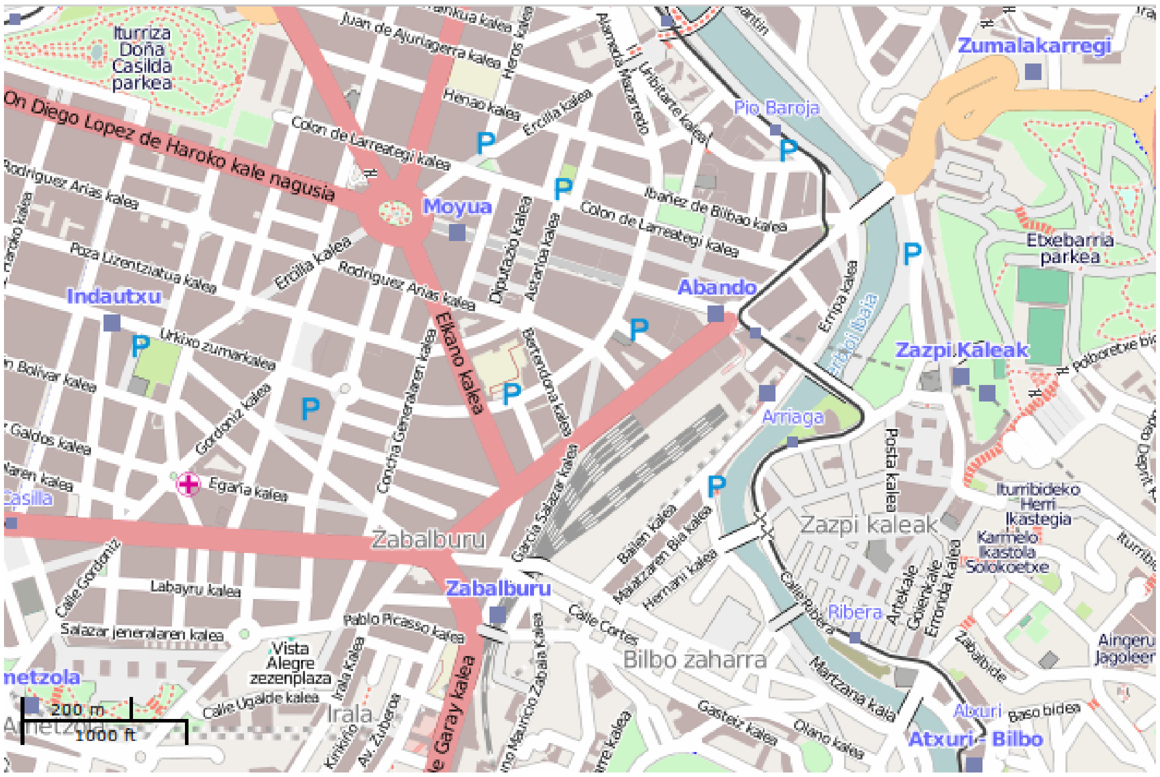


Figure 32: Bilbao map

6.4.2 OpenLayers

OpenLayers is a map viewing library, written in pure JavaScript. The OpenLayers library provides a *JavaScript API which makes it easy to incorporate maps from a variety of sources into your web page or application*. It can display map tiles and markers loaded from any source. The best part of this API is that Open Layers is completely free, Open Source JavaScript, released under the BSD License.

Furthermore, Open Layers implements industry-standard methods for geographic data access, such as the OpenGIS Consortium's Web Mapping Service (WMS) and Web Feature Service (WFS) protocols. As a framework, *OpenLayers is intended to separate map tools from map data so that all the tools can operate on all the data sources*:

- ka-Map
- TMS
- GeoRSS
- Google Maps (satellite, hybrid, streets), Yahoo, Microsoft, MultiMap, **OpenStreetMap**, ...

The common tools and features that can be used for every source are:

- Markers to mark a position in the map (use of a default image icon or the image we want)
- Texts to add something on a location of the map

- Map controls:
  - zoom
  - navigation
  - mouse controls
  - touch controls (mobile)
  - overview map
  - map sources selector: feature to switch map source. It is a selector where the user clicks on the map we wants to view (Google, Yahoo, …)

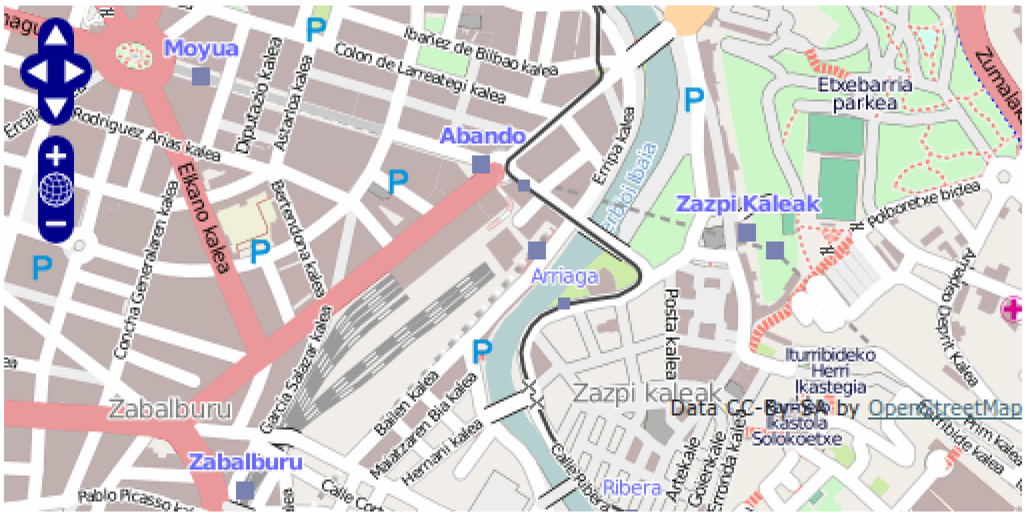


Figure 33: OpenLayers (data source: OpenStreetMap)

The Figure 33 displays a map using OpenStreetMap source. Navigation and zoom controls are default. OpenLayers API provides the option to change user interface and modify the CSS and use our own images for the buttons that can be displayed. Figure 34 shows a big zoom control modified to use in mobile devices, a marker and the overview map control on the right-down corner of the map. In Go-myLife social network, we are going to integrate OpenStreetMap using OpenLayers with the GWT library **GWT-OpenLayers**. GWT-OpenLayers is a Java wrapper for the OpenLayers JavaScript API. It allows GWT projects, like Go-myLife, to use the OpenLayers JavaScript API.

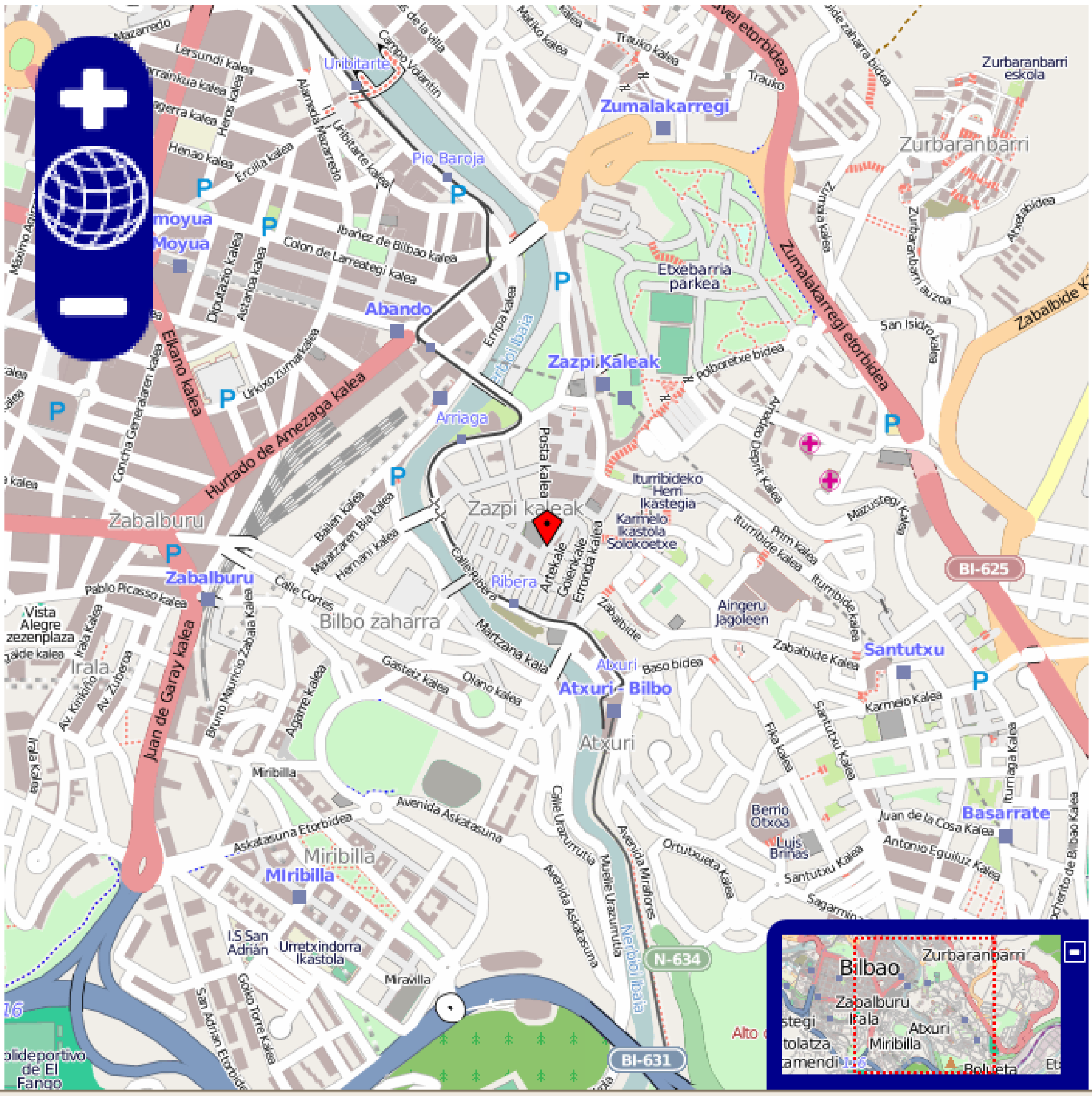


Figure 34: OpenLayers: OpenStreetMap (modified zoom button)

7 Conclusion

After the trials with end-users, the technical team made a depth analysis to detect the weak points and missing functionalities which led to a new set of technical requirements to meet users' needs. The trials and workshops detected challenges in the UI design to make it more intuitive for the elderly as well as usability and functional issues in terms of quality of the service.

With the feedback obtained from the different pilots and workshops, we have designed the final architecture of the Go-myLife platform, its APIs and the final prototype. The overall structure of the architecture remains unchanged but significant changes will be introduced in the existing components to improve and enhance the offered functionalities.

Go-myLife's technical team, during WP4, will develop and construct the final platform and prototype based on the guidelines detailed in this document. The final prototype will be tested again by real end-users to check its potential and show them the improvements made to reach their needs and suggestions.

This final platform and prototype will be the first commercially exploitable products that Go-myLife will generate.