Project acronym: **Go-myLife**

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

AAL Joint Programme

Call for Proposals AAL-2009-2-089

D3.1 Initial Platform Architecture and Design

Author: Idoia Olalde (Andago)

Version: 1.0

Date: 31/03/2011

Deliverable Number:	D3.1
Contractual Date of Delivery:	31/03/2011
Actual Date of Delivery:	31/03/2011
Title of Deliverable:	D3.1 Initial Platform Architecture and Design
Dissemination Level:	Public
WP contributing to the Deliverable:	WP3
Author(s):	Andago
Participant(s):	ATOS, ICCS

History			
Version	Date	Author	Comments
0.1	09/03/2011	Idoia Olalde (Andago)	Draft version
0.2	14/03/2011	George Karkalis (ICCS)	Draft version
0.3	31/03/2011	A.McDonough (ATOS)	Draft version
1.0	31/03/2011	Idoia Olalde (Andago)	Final version

Approval and Sign-off		
Date	Name	Sign-off
31/03/2011	Idoia Olalde (Andago)	approved

Abstract

This document contains the initial architecture design of the Go-myLife platform. It also summaries the functionalities that Go-myLife will have as social network.

It describes the different components that compose the overall system: LibreGeoSocial framework as the Social Network Engine, the Social Connector Manager and the specific APIs to connect with external social networks such as Facebook and Twitter and the Web Application through which the user will interact with Go-myLife.

Keywords

Architecture, Design, Functionality, Social Network Engine, Social Network Manager, LibreGeoSocial, Web application

Table of Contents

Table of Illustrations

1 Introduction

1.1 Summary

The objective of this work package is to create the basic architecture and design for the Go-myLife platform. Within its lifetime which will be mainly segmented in two iterations, this activity, in the first cycle, will work towards a preliminary definition of the Go-myLife architecture based on the requirements gained from the use case analysis.

1.2 Role of this deliverable

The role of this deliverable *D3.1 Initial Platform Architecture and Design* is the definition of the initial architecture of the Go-myLife system. During this phase, we have identified the components, design details of the implementation and integration of the elements that make up the final platform.

All the tasks carried out in the WP2 related to the identification of users' needs will be reflected in the work of the WP3. This deliverable summaries the study of the technical requirements carried out during the WP2 and describes the different components and subsystems that take part in the overall architecture. A description of the requirements agreed and decisions made during the task *T2.5 Technical Requirements' Analysis* and reflected in its deliverable *D2.4 Technical Requirement's Analysis* will be included.

This initial design will be the base for the WP4, where the development of the prototype will be carried out, and WP5, where Go-myLife community will be built using the core platform services and capabilities.

2 Design considerations

2.1 Technical requirements

This section outlines the required technical functionality of Go-myLife and what is considered essential functionality in social networks for the intended beneficiaries. The functionality described in this section is generally available within social networks and as such may already be familiar to the user. We present below a brief description of each function. If you require more information please refer to the deliverable *D2.4-Technical Requirements Analysis*, where you will find a more detailed description.

2.1.1 Creating a profile

Go-myLife will provide the user profile creation with the facility to allow users to describe themselves, show their interests and friends. It should also allow a profile picture or video to be uploaded.

2.1.2 Uploading photos

The functionality should provide an "Image Gallery" where the user can upload photographs that have been taken from a mobile or are stored in a computer. It should allow the possibility to organise photos by title and/or date. It will allow photographs to be tagged with date, description, and title as well as geolocation information.

Photographs may be shared with other members of the social network provided they meet privacy parameters set by the user.

2.1.3 Uploading videos

Go-myLife will provide a "Video Gallery" where the user can upload videos from a video camera in a mobile phone and also videos stored on computer. The user may organise the videos by title and date. Tagging useful information to the video such as title, description, date and place will be required.

2.1.4 Finding family and friends

Go-myLife will include the ability to find friends, family members or other people that the user knows within Go-myLife or on an external social network. To find family or friends the user must enter an agreed search criteria. The search criteria may consist of name, email address, company name, school or university.

If a friend or family member is found, the person found should first accept the user's request before being added to the user's friends list.

2.1.5 Sharing content

Photographs, videos and public content can be uploaded to the user profile. This functionality will permit the user to share the content by setting privacy settings. Privacy settings permit access to the user only, selected friends, selected groups, all user's friends, friends of friends, Go-myLife users and social network users.

The user may send a URL to members of Go-myLife to share content or allow external users via a security token.

2.1.6 Groups

Go-myLife users may create special interest groups. The type of group may be hidden, private or public.

Groups may share text, image, music and video information and specify privacy settings for access.

2.1.7 Messages

Go-myLife users may exchange messages with other users. The message type supported will be wall, status, private and group.

2.1.8 Connection to other social networks

Go-myLife will have the ability to connect to other social networks via there API´s. The functionality will be dependent on the public API provided. Common features of interconnecting social networks include status upload, share content, messaging and finding friends. This functionality is described in greater detail in chapter 2.3.2 of this document.

2.1.9 GeoLocation

Go-myLife will able to track the location of users and store geolocated information for the different types of content. Information will be provided as to where friends and family are located in the near vicinity.

2.2 Other requirements

This section outlines other requirements which are technical by nature but whose focus is not the direct interaction with the user but impacts indirectly. They are requirements that are necessary to Go-myLife for its correct functioning in a social network environment.

2.2.1Trust and Privacy

Go-myLife will be based on a trust on-line Social Network, that users can develop new relationships and share information without the fear of been cheating by fake users or malicious information. The users will have guaranteed their privacy and will have the possibility to define the privacy degree of their information. They will have the option to select what they want to share and for whom (visibility of their information).

2.2.2Scalability

One important thing that needs to be taken into account when designing and developing a new on-line social network is the platform scalability. Being able to accommodate a growing demand of new users without the need of redesigning the platform is something rely important and will be take into account in the design and development

of the Go-myLife platform. The platform will be designed to be easily scaled up or scaled down without the need of new development.

2.2.3 Performance

Go-myLife should provide a quality of service adequate for its intended use as a social network for elderly users. It should have a response time which is deemed reasonable for a non-critical system and which does not impede normal use.

The Go-myLife platform should not present the user with technical problems and give expected results at all times. It should be functionally reliable and not be prone to unexpected behaviour.

2.2.4 Security

Go-myLife should take on board the recommendations made by Cyber Security Standards designed to improve security and minimize the risk of cyber security attacks. Relevant standards such as ISO27001 will be adhered to assuring that Go-myLife users can rely on confidentiality, integrity and availability of their information. Incorporating the recommendations of best practices is seen as essential to provide a long term solution and ultimately certification by an accredited body.

2.2.5 Extensibility

Extension to Go-myLife can be through the addition of new functionality or through modification of existing functionality. There should be little impact on the overall structure of the Go-myLife project. It should be possible to add new services to the services already provided without an impact on the architecture.

Go-myLife will conform to relevant standards thus enabling the system to be extended as well as being extended to incorporate other social networks. The user should not be restricted to a social network in particular but be able to link to wider communities using the same Go-myLife interface.

2.2.6 Accessibility and Usability

In human-computer interaction, computer accessibility refers to the accessibility of a computer system to all people, regardless of disability or severity of impairment. It is an important concern for both software and web designers. They have to consider standards in order to produce applications that enable the use of a computer or a mobile device by every person, independently of any possible disability, and any special device (Assistive Technology) that they have to use.

Go-myLife will be designed around the interests and requirements of elderly people taking into account their usability and accessibility requirements. It will follow the WAI-W3C accessibility guidelines (WCAG 2.0) and ISO 9241 for meeting those requirements.

2.3 Selected technologies and social network integration approach

During the WP2, a deep analysis of existing social networks engines was carried out in order to evaluate them and select the one that better satisfies the functionalities expected for the Go-myLife platform.

We also evaluated some social networks in order to decide which of them will be connected to the Go-myLife platform. Our decision was based on the functionalities they offer through their APIs as well as their social impact in the elderlies' context. Taking into account these aspects, Facebook and Twitter were selected to integrate in the first version of Go-MyLife.

2.3.1 Social network engine: LibreGeoSocial

After our analysis of the different social network engines, LibreGeoSocial was selected. In the following tables, there is a comparison between the analyzed social network engines.

The Table 1 collects how the different engines fulfill the basic expected functionalities of a social network. Basically, all of them cover all the features.

	LIBREGEOSOCIAL	LIFERAY	ELGG	PINAX
Access to videos	yes	yes	yes	yes
Messaging service	comments	yes	yes	yes
Share pictures	yes	yes	yes	
Blog		yes	yes	yes
Communities	Groups based on gelocation and social network information	User groups, communities, organizations	yes	yes

Table 1 Comparison of the basic functionalities of a social network

The Table 2 summaries the features expected for Go-myLife, analyzing the integration of external sources of information and social networks and the adaptation to the requirements of the elderly in terms of accessibility and usability.

	LIBREGEOSOCIAL	LIFERAY	ELGG	PINAX
Connection to other SN	yes	yes	yes	
Plug-ins, add-ons	yes	yes	yes	yes
Accessibility		Theme-dependant	no	
Usability		yes	yes	
Multichannel access	yes	yes		no
Geolocated info	yes	no	no	no

Table 2 Comparison of features expected in Go-myLife

Go-MyLife will be a platform based on mobility and location awareness, not only focused on providing location based information but taking this step further by providing greater contextual awareness of whom and what is around the user. This will then allow serendipitous meetings with friends and family, help users to find out interesting information about places, add their own comments for the benefit of others, and help them to have greater security, knowing that it will be easier for them to get help and support when they get into difficulty.

For this reason, geolocation feature is a key aspect in the selection of the social network engine and among all the options, only *LibreGeoSocial* manages user's information as nodes with geolocated information. Some developments will be necessary to be done in order to satisfy the functionalities but it is considered more efficient to integrate these changes in LibreGeoSocial than integrating geolocated information in any other social network engine. LibreGeoSocial is only the engine of the platform which manages the information. Above it, it will be developed web applications which will have all the expected functionalities with a usable and accessible interface.

2.3.2 Social Networks

The API of each social network must be prepared to interact with the Go-myLife platform, developing a module (a connector) in the middle between Go-myLife and the specific social network (Facebook, Twitter, etc.).

In order to satisfy this requirement, a study of the API has been done.

2.3.2.1 Facebook

The Facebook Platform is divided into the following concepts:

- Social Plug-ins: embeddable HTML code that can be integrated in your website to add social features. The plug-ins are personalized for all users who are currently logged into Facebook, even if they are visiting your site for the first time.

Most *Social Plug-ins* can be integrated with your site by simply including the iframe tag for the plug-in within your page. Some of them require the use of *XFBML* (eXtended Facebook Markup Language). *XFBML* is a set of XML elements that can be included in your HTML pages to display *Social Plug-ins*. When your page is loaded, any *XFBML* elements found in the document are processed by the *JavaScript SDK*, resulting in the appropriate plug-in being rendered on your page.

The plug-ins are: Like Button, Activity Feed, Recommendations, Like Box, Login Button, Registration, Facepile, Comments and Live Stream.

- Apps on Facebook: Apps on Facebook.com are web applications that are loaded in the context of Facebook. You can build your app using any language or tool chain that supports web programming, such as PHP, Python, Java or C#.

  Apps on Facebook.com are loaded into a Canvas Page. A Canvas Page is quite literally a blank canvas within Facebook on which to run your application. You populate the Canvas Page by providing a Canvas URL that contains the HTML, JavaScript and CSS that make up your application. When a user requests the Canvas Page, the Canvas URL is loaded within an iframe on that page. This results in the application being displayed within the standard Facebook chrome.

  Once the application is created, Facebook provides *Social Channels* to increase the popularity of the application. Some of them are automatically created: the application is added to the App Dashboard or Game Dashboard, a bookmark is created to enable users to easily navigate and a usage story is published to notify user's friends that he or she has started to use the application.

  The application can also publish directly to a user's News Feed and send Requests to their friends through the specific *Dialogs* explained in the next point.

  There is also the possibility to customize page tabs.

- Dialogs: *Dialogs* provide a simple, consistent interface to display dialogs to users. *Dialogs* do not require special user permissions because they require user interaction. *Dialogs* can be used in any type of application, whether on Facebook.com, a website, or a mobile application.

  You can integrate *Dialogs* into your application by constructing the URLs or by using a helper method in one of the Facebook Platform SDKs.

  Dialogs are all built to seamlessly run in a variety of display contexts (page, popup, iframe) on both the web and mobile.

  The existing *Dialogs* are: Feed Dialog, Friends Dialog, OAuth Dialog, Pay Dialog and Request Dialog.

- Social Graph: The Facebook core is the social graph that represents people and their connections. The *Graph API* presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph (e.g., people, photos, events, and pages) and the connections between them (e.g., friend

relationships, shared content, and photo tags). You can use this API to read data, search, publish and delete objects.

Every object in the social graph has a unique ID. You can access the properties of an object by requesting https://graph.facebook.com/ID.

All objects in Facebook can be accessed in the same way: Users, Pages, Events, Groups, Applications, Status messages, Photos, Photo albums, Profile pictures, Videos, Notes, Checkins.

All of the objects in the Facebook *Social Graph* are connected to each other via relationships (called connections). You can examine the connections between objects using the URL structure:

https://graph.facebook.com/ID/CONNECTION_TYPE.

All these requests return a response in JSON format.

- Other APIs: Other APIs exist in Facebook to add extra functionalities:
  - Credits API: The Facebook Credits API enables a user to use credits as a method for purchasing digital and virtual goods within your application.
  - Ads API: The Facebook Ads API lets you create and manage your own ads on Facebook programmatically, without using the Facebook Advertising Manager tool.
  - Chat API You can integrate Facebook Chat into your Web-based, desktop, or mobile instant messaging products. Your instant messaging client connects to Facebook Chat via the Jabber/XMPP service.
- Open Graph Protocol: The *Open Graph Protocol* enables you to integrate your web page into the social graph using Open Graph tags. If your web page represents real-world entities, through the Like Button, they can be directly included in the "Likes and Interests" section of the user's profile. With that feature, your web page behaves as a Facebook Page: it sends updates to the user, it appears in the same places that the Facebook pages show up around the site (e.g. search) and you can target ads to people who like your content.
- SDKs: Facebook provides Software Developers' Kits (SDKs) for different platforms:
  - iOS (Objective-C): https://github.com/facebook/facebook-ios-sdk
  - Android (Java): https://github.com/facebook/facebook-android-sdk
  - PHP SDK: https://github.com/facebook/php-sdk/
  - JavaScript SDK: https://github.com/facebook/connect-js
  - C# SDK: https://github.com/facebook/csharp-sdk
  - Python SDK: https://github.com/facebook/python-sdk
  - The community has also developed other libraries such as the GWT Facebook SDK: http://code.google.com/p/gwtfb/

With these SDKs, developers can build their own applications (mobile or native) and integrate with Facebook.

- FBL (Facebook Query Language): Facebook enables the use of a SQL-style interface to query the data exposed by the *Graph API*. It provides for some advanced features not available in the *Graph API*, including batching multiple queries into a single call.

Authentication

Facebook Platform uses the OAuth 2.0 protocol for authentication and authorization. It has different mechanisms to use within a website, mobile and desktop applications.

Facebook Platform supports two different OAuth 2.0 flows for user login: server-side (used whenever you need to call the Graph API from your web server) and client-side (used when you need to make calls to the Graph API from a client, such as JavaScript running in a Web browser or from a native mobile or desktop application)

Regardless of the flow you utilize, the implementation of the OAuth 2.0 involves three different steps: user authentication, application authorization and application authentication. User authentication ensures that the user is who they say they are. Application authorization ensures that the user knows exactly what data and capabilities they are providing to your app. Application authentication ensures that the user is giving their information to your app and not someone else. Once these steps are complete, the application is issued a user access token that you enable the access to the user's information and to take actions on their behalf.

2.3.2.2 Twitter

The Twitter API currently consists of 3 APIs:

- REST API: The Twitter REST API methods allow developers to access core Twitter data. This includes update timelines, status data, and user information.
- Search API: It is also a REST API that due to history remains separately. The Search API methods give developers methods to interact with Twitter search and trends data.
- Streaming API: The Streaming API provides near real-time high-volume access to Tweets in sampled and filtered form.
    - Streaming API: Public statuses from all users, filtered in various ways: By userid, by keyword, by random sampling, by geographic location, etc.
    - User Streams: Nearly all data required to update a user's display. Requires the user's OAuth token. Provides public and protected statuses from followings, direct messages, mentions, and other events taken on and by the user.
    - Site Streams: Allows multiplexing of multiple User Streams over a Site Stream connection. Once more than a handful of User Streams connections are opened from the same host or service, Site Streams must be used.

Methods to retrieve data from the Twitter API require a GET request. Methods that submit, change, or destroy data require a POST. A DELETE request is also accepted for methods that destroy data.

The documentation describes which formats are available for each method. The API presently supports the following data formats: XML, JSON, and the RSS and Atom syndication formats, with some methods only accepting a subset of these formats.

The community has created numerous Twitter API libraries.

There are also other features:

- Twitter Button: The Tweet Button is a small widget which allows users to easily share your website with their followers. There are three ways to add the Tweet Button to your webpage: using JavaScript, using an iframe or even build your own.
- @anywhere: It is a framework for adding the Twitter experience anywhere on the web. Rather than implementing APIs, site owners need only drop in a few lines of JavaScript. Developers can use @anywhere to add Follow Buttons, Hovercards, linkify Twitter usernames, and build deeper integrations with "Connect to Twitter."

Authentication

Although Twitter supports different mechanism for authentication (out-of-band/PIN code authentication, xAuth), it recommends the use of OAuth.

The OAuth request cycle is roughly:

- Retrieve a request token
- Request user authorization by sending the user to a Twitter.com login page
- Exchange the request token for an access token

The OAuth method will act in different ways depending on the status of the user and their previous interaction with the calling application:

1. If the user is logged into twitter.com and has already approved the calling application, the user will be immediately authenticated and returned to the callback URL.
2. If the user is not logged into twitter.com and has already approved the calling application, the user will be prompted to login to twitter.com then will be immediately authenticated and returned to the callback URL.
3. If the user is logged into twitter.com and has not already approved the calling application, the OAuth authorization prompt will be presented. Authorizing users will then be redirected to the callback URL.
4. If the user is not logged into twitter.com and has not already approved the calling application, the user will be prompted to login to twitter.com then will be presented the authorization prompt before redirecting back to the callback URL.

2.4 Architecture evolution in Go-myLife: iterative process

The design of the architecture will be an iterative process: analyze user's needs, design the architecture, implement a prototype, test the different modules and perform trials with real end-users which will give us feedback to a better design of the platform, identifying its weak points and make it more useful and adapted for the user. This cycle will be repeated within the life of Go-myLife project.

With this approach the elderly become the center of the design and Go-myLife will result in a platform designed for them.

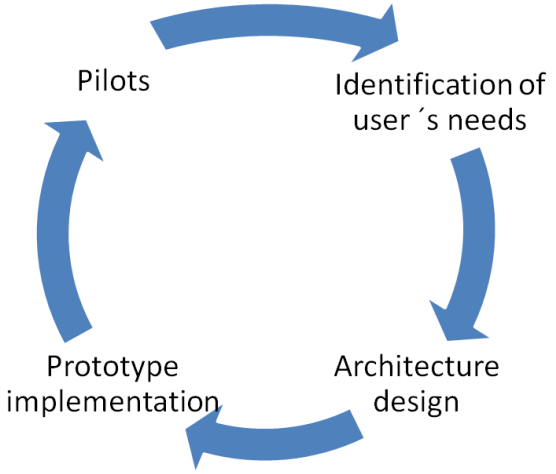

Illustration 1 Go-myLife Platform Iterative Design

This deliverable, *D3.1 Initial Platform Architecture and Design*, will be based on the deliverable *D2.5 Technical Requirement's Analysis*. *D2.5*, in its first version, collects the general functionalities expected for Go-myLife. With the overall architecture design defined, we will proceed to the design of the first prototype. The deliverable *D3.2 First prototype design* will describe a more detailed implementation of the platform and the client-side application that will be used on the pilots.

Go-myLife platform will be implemented during *T4.1 Platform prototype 1* that will be tested in the first version of the pilots.

A second version of *D2.5 Technical Requirement's Analysis* will be released once the workshops are carried out. In this new version, the functionalities and requirements will be updated to adapt them to the real needs of the end-users.

With these considerations and the feedback gained in the early pilots, the architecture designed will be refined. This iteration will result in the deliverables *D3.3 Final Platform Architecture and Design* and a second prototype will be developed in T*4.2 Platform prototype 2* to be tested in a new version of the pilots.

With this iterative approach, we try to reduce the high risk associated with research projects where several iterations allow a better control about the anticipated progress versus achieved progress to be better controlled by applying corrective measures.

3 Architecture design

This part of the document defines the architecture of the project, focused on the decomposition of the design of functional or logical components that expose well-defined communication interfaces.

Go-myLife will provide a platform where users can easily interact with other social networks. For that reason, Go-myLife will be based on a social network engine that integrates content from existing social networks and on which specific services designed for elderly people will be offered.

The following figure represents Go-MyLife ecosystem:

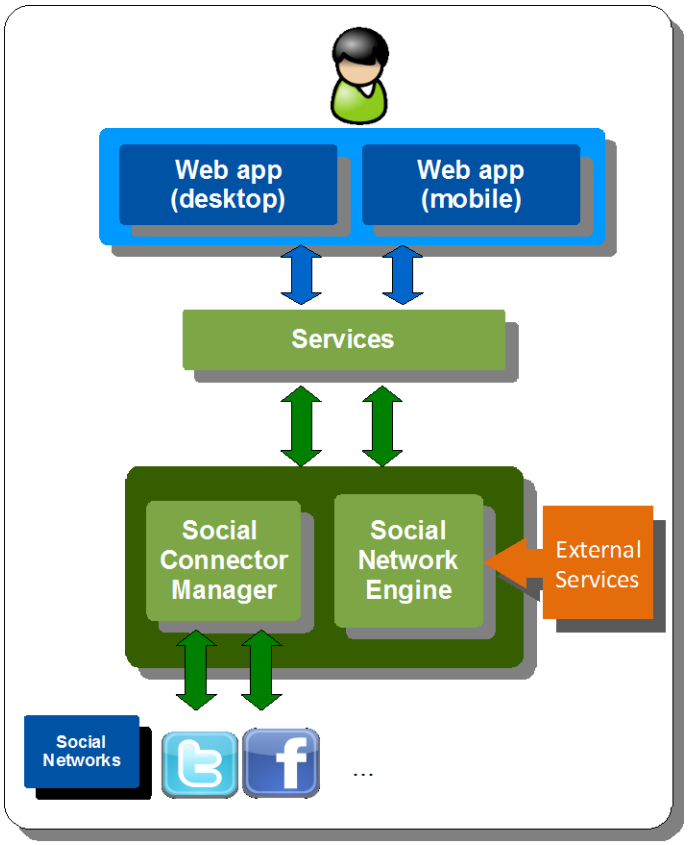

Illustration 2 Go-myLife schema

The Illustration 2 shows schematically the architecture of the project Go-myLife. In the core of the system, it will reside the *Social Network Engine* to manage the platform and connect to external sources of information and the *Social Connector Manager* that will connect to the different social networks. Users will access to Go-myLife through a web application (mobile or desktop version) that will offered specific services.

The following sections delve into all the parts of the Go-myLife platform.

3.1 Overall design

The deliverable *D3.1 Initial Platform Architecture and Design* presents a first version of the platform architecture of Go-myLife. This first version of the architecture is described in greater detail in the deliverable *D3.3 Final Platform Architecture and Design* In Illustration 2, it is shown the reference architecture as a basis for the development of the project of Go-myLife.

Below is a brief description of each of the modules of the architecture.

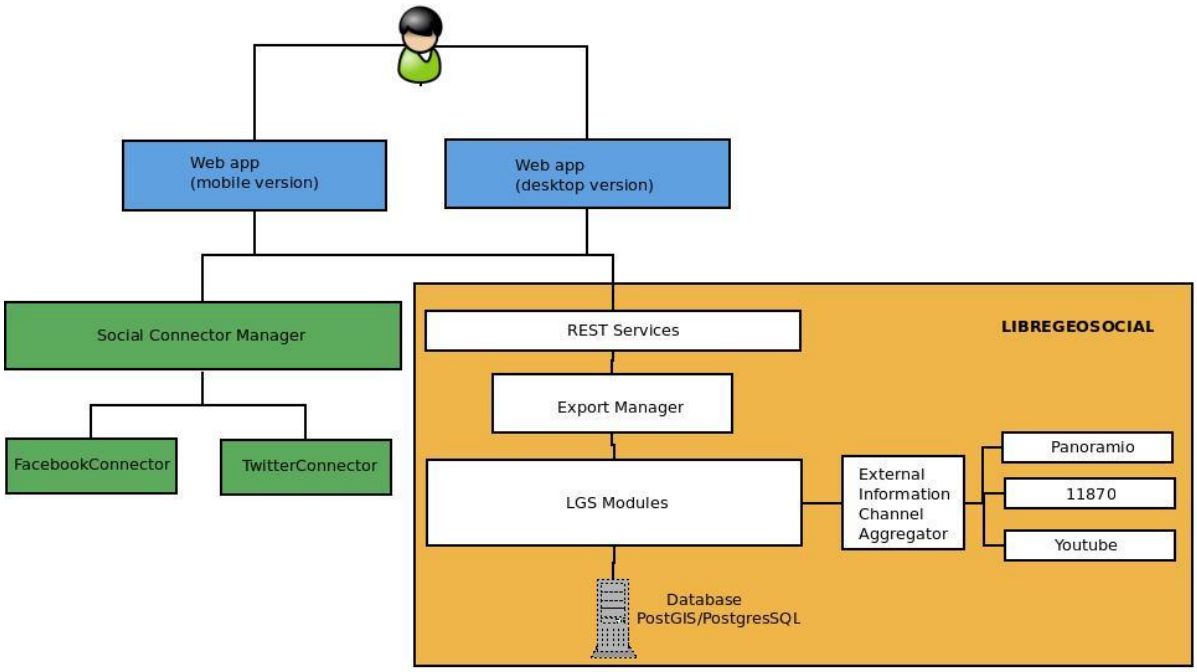

Illustration 3 Go-myLife architecture

Go-myLife Server Core: The core of the platform where the logic resides. It will be based on the framework *LibreGeoSocial*. It manages all the user information as geolocated nodes through different modules.

LibreGeoSocial framework can export its data in two different formats through the Export Manager: JSON and XML. It offers a REST API to access and manage it.

It has an external channels aggregator to retrieve information from third-parties (e.g. YouTube) and offer it to the user.

Social Connector Manager: This module manages the connection with other social networks, integrating this information inside Go-myLife. Different connectors will be implemented to connect to the specific APIs of the social networks and interact with them to retrieve information and show them in Go-myLife and to upload data directly

from Go-myLife to the specific network. In the first version of Go-myLife Twitter and Facebook will be integrated

It offers an interface to facilitate the integration of future social networks. Developers will only need to implement the connectors to specific social networks following that interface.

Web Application: this component is the client-side of Go-myLife platform. A web based application through which the user could update his/her status, share a photo, connect to friends, etc. There will be two versions with features and interfaces adapted to the respective devices: mobile and desktop version.

The interface will be specifically adapted to elderly needs taking into account usability and accessibility criteria.

3.2 Go-myLife subsystem and components

3.2.1 Go-myLife Server Core

The Go-myLife server core will be in charge of Go-myLife social network management. Functionalities covered by the server core are:

- Social network features:
  - Users management
  - Contents management
  - Classified contents management
  - Users relationships
  - Privacy
- Geolocation over contents and users in the social network
- Access to external contents (3rd parties information sources)
- API to fulfill all the client's needs.
- Ability to export the information in different formats

To develop this social core will be used the *LibreGeoSocial* framework. It is an open source framework that allows creating geolocated social networks and exports a REST API to manage all the functionalities.

3.2.1.1 Rest Services

The rest services exported with the *LibreGeoSocial* framework will be divided in different sections, regarding of the functionality.

Node management: REST API to cover standard functionalities over the different nodes stored in the server. (Nodes = Contents = (user, photo, note, video, audio))

- Change node position
- Tag a node
- Remove node tags
- Add new comment
- Delete comment
- Delete a node
- Get nodes for a layer
- Change node availability dates

User Management: REST API to manage the user's information, registration and relations with other users.

- Create new user
- Modify user
- Log in
- Get my data
- Get user data
- Delete a user
- Get friends
- Get friendship invitations
- Set user position
- Near people
- Near friends
- Set user status
- Set user avatar
- List all users
- Start relation between users
- Finish relation between users

Notes Management: REST API to manage content's notes.

- Create a new note
- Get note data
- Delete a note

Photos Management: REST API to manage content's photos.

- Upload a photo
- Get photo data
- Get photo image
- Get photo thumbnail
- Delete a photo

Sound Management: REST API to manage content's sounds.

- Upload a sound
- Get sound data
- Get sound file
- Delete a sound

Videos Management: REST API to manage content's videos.

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

Privacy Management: REST API to manage the privacy over the different contents created inside the server.

- Nodes privacy management
  - Show nodes' privacy status
  - Permissions change
- Layers privacy management
  - Show layer's privacy status
  - Permissions change
- User privacy management
  - Show user's privacy status
  - Permissions change
- Notes privacy management
  - Show note's privacy status
  - Change note's privacy
- Photos privacy management
  - Show photo's privacy status
  - Change photo's privacy
- Sounds privacy management
  - Show sound's privacy status
  - Change sound's privacy
- Videos privacy management
  - Show Video's privacy status
  - Change Video's privacy
- Privacy auxiliary functions
  - Get all available roles
  - Get all available permissions

Layers Management: REST API to manage information about layers. The layers managers provides an interface to access to the different information sources supported in the server, allowing users and applications to interact with nodes from several different sources. These sources could be internal sources (with nodes stored in Go-myLife Core Server database) 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
- Change layer's icon
- Get layer's info

- Get layer's category
- Create a layer
- Delete a layer

Layers' Queries: The layers system provides an interface to make specific queries. These queries will allow developers to access easily to special features in the layers system, such as, a search in multiple layers, something that also could be done with several requests with the standard layers interface.

- Multi layer search

3.2.1.2 Layer Manager

The *Layer Manager* is a module of *LibreGeoSocial* framework to manage and classify contents. Thus, all the contents are acceded through a specified layer or channel. These contents could be stored in the Go-myLife server (using Internal Layers) or in 3rd parties information sources (using External Layers).

For example, you could have three layers in your system: Go-myLife (internal), Panoramio (external), Picasa (external). The clients using Go-myLife server could access to the geolocated contents of these layers without worrying about the storing location of the contents. They will have a great abstraction and qualifying system. From the point of view of the client, there are no differences in the way you access and manage the information contents.

In order to add new layers to your system:

- You could create internal layers easily, using the Layers' API.
- You could integrate external layers implementing a specified interface in the server.

After that, clients will access the new information of the layers without making modifications in the client, thanks to the Layers' API.

This module allows an easy integration system for sources of information; it qualifies the contents and provides to clients a good abstraction model.

3.2.1.3 Export Manager

The *Export Manager* is a module of *LibreGeoSocial* framework that allows client to receive the information using different formats of data.

Currently, the *LibreGeoSocial* framework can export the data in two different formats through the Export Manager: JSON and XML. But, it could be added new formats.

Next, a class diagram of the module:

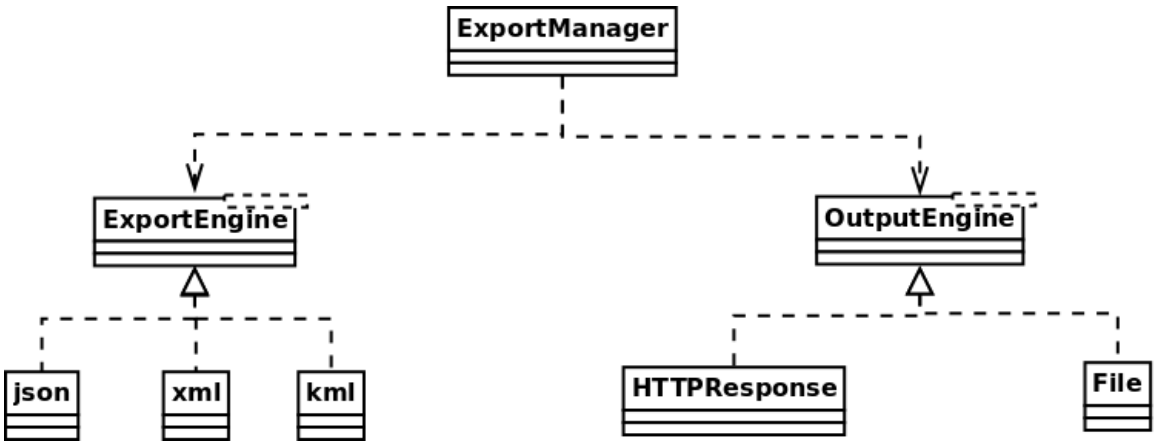

Illustration 4 Export Manager

The Export Manager not only converts the information to the desired format, but also returns the information with a specified response (usually, with an HTTPResponse).

In a typical workflow using *LibreGeoSocial* server, the client asks for information through the layers abstractions. Internally, the information is retrieved (internal or external). This information is returned to the client using the Export Manager, converting the data to JSON or XML, and after that, creating an HTTPResponse.

3.2.2 Social Connector Manager

To integrate external social networks like Facebook or Twitter, we have designed the following architecture:

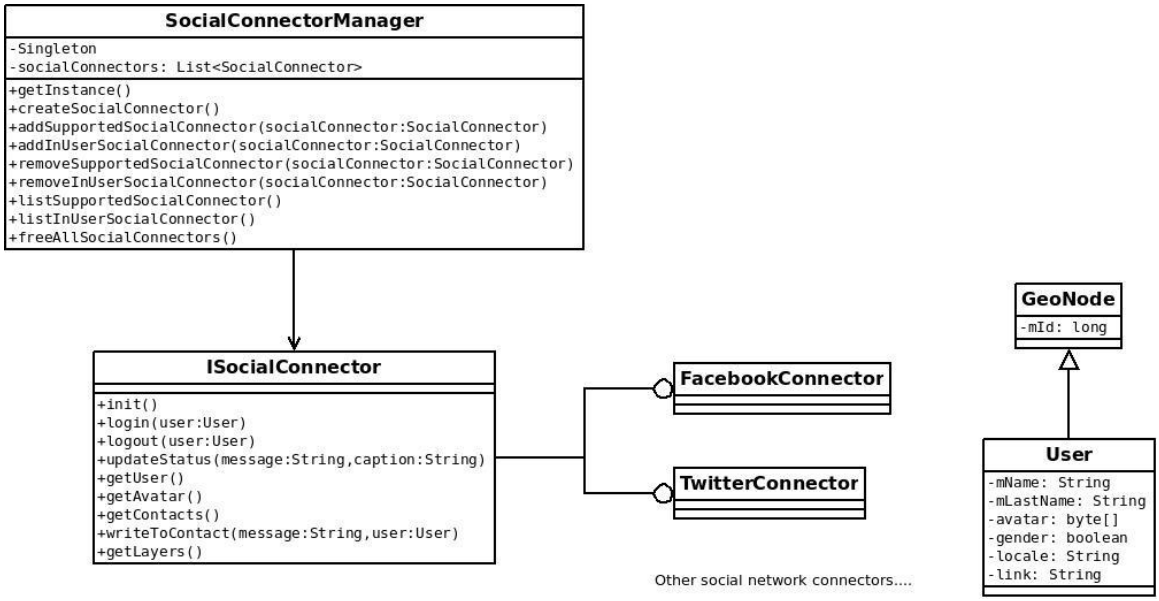

Illustration 5 Social Connector Manager

The objective is that users must be able to perform a two-way communication between these external social networks and Go-myLife at different levels:

1. Users of Go-myLife should be able to use their Facebook or Twitter accounts to authenticate into Go-myLife
2. Users of Go-myLife should be able to export content from Go-myLife to Facebook, Twitter, etc.
3. Users of Go-myLife should be able to import content from external social networks.

To allow an easy integration of social networks, the *SocialConnectorManager* class is a factory class used to create all the instances of Social Connectors. Adding new social networks will be as simple as implementing the *ISocialConnector* interface and fill the abstract methods with the code to obtain the data of a specific social network.

Using the *ISocialConnector* interface, it is possible to login/logout, to update status of users, to get user information, and to get all layers that this social network provides.

All the information in *LibreGeoSocial* framework is organized using layers that contain geonodes. For this reason, all social connectors must store the user information in layers. It should exist one social connector for each social network. They must respect the defined interface and return *GeoNodes*. This is how *LibreGeoSocial* framework can handle with the geolocated information.

3.2.3 Web application

The Go-myLife web application provides the end users a user friendly and accessible interface to Go-myLife platform. The web application will allow users to access all the functionalities provided by the system either through a desktop web browser or through a mobile web browser. Additionally through the web application the user will be able to download the Go-myLife Mobile Application and install it on his mobile device.

The architecture of the Go-myLife Web application is shown in Illustration 5.

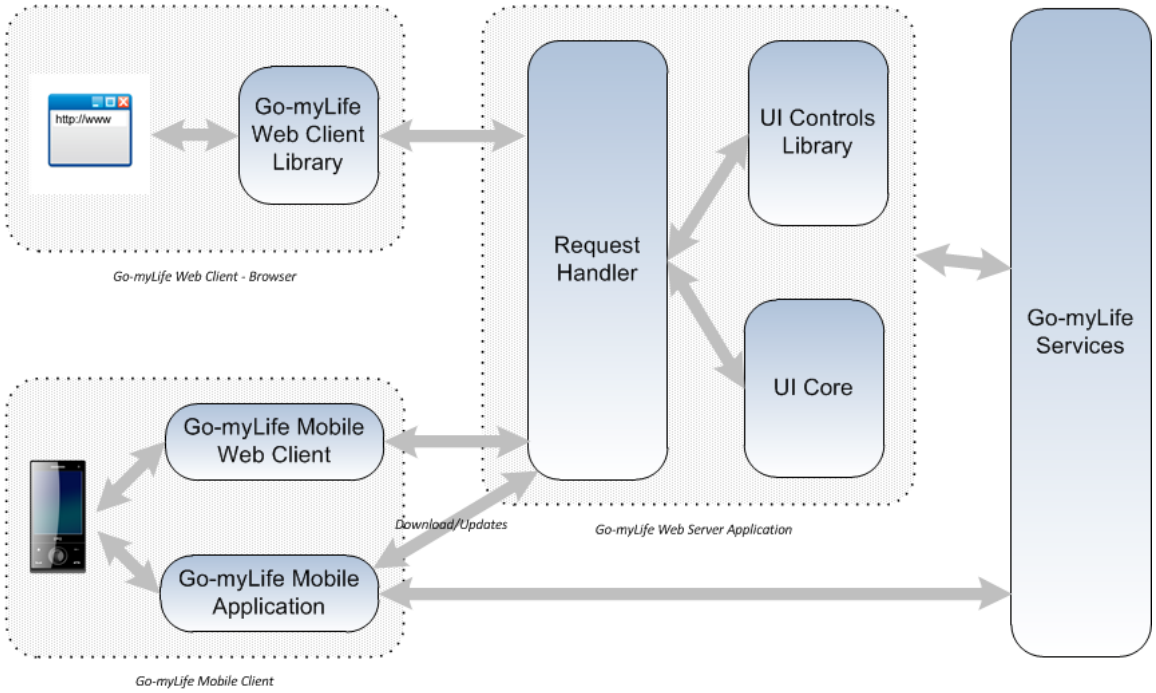

Illustration 5 Go-myLife Web Application Architecture

The Go-myLife Web Application consists of the following layers:

- The **Go-myLife Web Client** is the part of the application running on the web browser. It contains a JavaScript library component (Go-myLife Web Client Library) implementing the presentation logic of the application, providing rich user interaction capabilities and communicating with the server-side components.
- The **Go-myLife Web Server Application** running on a web server host implements the core functionality of the user interface and communicates with the rest Go-myLife services. It encapsulates the user interface rendering logic and the orchestration of user actions and application workflows.
- The **Go-myLife Mobile Client** provides access to Go-myLife from a mobile device either through a mobile web browser (Go-myLife Mobile Web Client) or through a mobile application installed on the device (Go-myLife Mobile Application). The mobile application can take advantage of advanced capabilities of the mobile device such as GPS functionality, camera and accelerometer.

The Go-myLife Web Application architecture is based on the Model-View-Controller (MVC) design pattern.

The Web Server Application contains the Request Handler component which receives requests from the browser (Web Client or Mobile Web Client), invokes the appropriate actions on the UI Core component, creates the corresponding user interface elements from the UI Controls Library component and renders the response back to the browser.

The UI Core component is implementing the core application logic and is responsible for interacting with the other Go-myLife components, such as the Social Networking Engine and the Social Connector Manager, through the services provided by the platform.

The UI Controls Library component is responsible for implementing the rendering logic of the user interface elements. Additionally this component will be responsible for implementing compliance with W3C's Web Content Accessibility Guidelines (WCAG) 2.0.

The Go-myLife Web Client will make use of JavaScript technology in order to provide a rich user-friendly browser based interface. This is the purpose of the Go-myLife Web Client Library which will utilize the jQuery library, an advanced and widely used JavaScript library facilitating web development and Ajax programming. Ajax will also be used in order to create a more interactive and dynamic interface.

The Go-myLife Mobile Web Client will be a simplified version of the web client optimized for mobile devices.

The Mobile Application will provide integration with the device's advanced capabilities and will make it easier for the user to access Go-myLife seamlessly during his everyday activities. The application will communicate directly with the Go-myLife platform services.

4 Conclusion

This deliverable describes the initial design for Go-myLife platform, explaining the different components that compose the overall system. It also collects the design requirements agreed during *T2.5 Technical Requirements' Analysis*.

The Go-myLife system consists of 3 main blocks: the Social Network Engine, the Social Connector Manager and the web based client application.

The Go-myLife server will be based on *LibreGeoSocial* framework which manages geolocated information, one of the bases of the Go-myLife project.

Connections with other social networks will be performed through the *Social Connector Manager* component, offering an interface to facilitate the integration of new social networks. In this first version, connectors to Twitter and Facebook will be implemented.

Users of Go-myLife will access to the platform through a web based application that will follow the W3C's Web Content Accessibility Guidelines (WCAG) 2.0 (mobile browser version and desktop browser version). Go-myLife will also offer a Mobile Application to provide integration with the device's advanced capabilities and Go-myLife platform.