Development of a new tool in e-Catalunya

Master’s Degree in Information Technology Project September 2009

Jordi Diaz Albuixech

Director: Josep Casanovas, Co-Director: Rosa Mª Martín Santiago, Tutor: David Carrera
# Content table

1. **INTRODUCTION** ............................................................................................................. 6
   1.1 Background .................................................................................................................. 6
   1.2 Motivation ..................................................................................................................... 7
   1.3 Project Objectives ...................................................................................................... 8
      1.3.1 Study .................................................................................................................... 8
      1.3.2 Practical Implementation ..................................................................................... 9
   1.4 Expected Benefits ..................................................................................................... 10
   1.5 Document Organization .......................................................................................... 11

2. **THE e-Catalunya PROJECT** ...................................................................................... 13
   2.1 Introduction .............................................................................................................. 13
   2.2 Portals and groups .................................................................................................... 14
      2.2.1 Privacy and participation control ....................................................................... 15
   2.3 The e-Catalunya role system .................................................................................... 16
   2.4 Tools .......................................................................................................................... 17
      2.4.1 Wiki ...................................................................................................................... 17
      2.4.2 Blog ....................................................................................................................... 17
      2.4.3 Calendar ................................................................................................................. 17
      2.4.4 Repository ........................................................................................................... 18
      2.4.5 Photo Album ....................................................................................................... 18
      2.4.6 Mailroom ............................................................................................................. 18
      2.4.7 Mailing list .......................................................................................................... 18
      2.4.8 Forum .................................................................................................................... 19
      2.4.9 eBugTracker ....................................................................................................... 19
   2.5 Social and Knowledge Network ................................................................................ 19
   2.6 Infrastructure Software and development environment .......................................... 22
      2.6.1 Webservers .......................................................................................................... 22
      2.6.2 Data Management ............................................................................................. 23

3. **STUDY** ...................................................................................................................... 24
   3.1 Description & Study Methodology ............................................................................ 24
   3.2 Importation / Exportation Content Services ............................................................... 25
      3.2.1 Existing Service Providers .................................................................................. 26
      3.2.2 Importing and exporting Content applications .................................................. 28
3.2.3 Standard application technology (widgets or gadgets) ........................................ 35
3.3 Enhancing e-Catalunya with Google Friends & Facebook Connect ......................... 49
  3.3.1 Google Friends ........................................................................................................ 49
  3.3.2 Facebook Connect .................................................................................................. 50
3.4 OpenSocial Layer ....................................................................................................... 55
  3.4.1 Technical issues and implementation .................................................................... 57
  3.4.2 OpenSocial Specifications ..................................................................................... 61
3.5 OAuth Authentication .................................................................................................. 62
  3.5.1 OAuth for Web Applications .................................................................................... 63
  3.5.2 Token Management ................................................................................................ 66
  3.5.3 Signature methods ................................................................................................ 67
  3.5.4 E-Catalunya OAuth Consumer: Service Integration .............................................. 68
3.6 OAuth Protocol in Shindig ........................................................................................... 69
3.7 SocialSite ...................................................................................................................... 74
  3.7.1 Social functionalities ................................................................................................ 75
3.8 Technology Viability Analysis ..................................................................................... 76
  3.8.1 OpenSocial Layer in e-Catalunya ......................................................................... 76
  3.8.2 SocialSite in e-Catalunya ...................................................................................... 77
  3.8.3 Gadgets & OpenSocial vs. Facebook applications in e-Catalunya ....................... 78
  3.8.4 Google rendering server vs own rendering server container ............................... 79
  3.8.5 Facebook Connect and Google Friends in e-Catalunya ......................................... 79
3.9 Preliminary study summary for Generalitat ................................................................. 80
  3.9.1 Technology analysis conclusions ......................................................................... 81
  3.9.2 Reviewed Objectives & Conclusions .................................................................... 84
4 FURTHER STUDIES .......................................................................................................... 88
  4.1 Service Externalization .............................................................................................. 88
    4.1.1 PhotoAlbum .......................................................................................................... 89
    4.1.2 Storage Services .................................................................................................. 93
    4.1.3 External surveys services ..................................................................................... 96
5 PROTOTYPE IMPLEMENTATION ..................................................................................... 103
  5.1 e-Catalunya Domain Layer ....................................................................................... 103
    5.1.1 Catalunya Services ............................................................................................... 104
  5.2 Gadget Implementation .............................................................................................. 106
    5.2.1 Configuring the services ...................................................................................... 108
<table>
<thead>
<tr>
<th>8.4.1</th>
<th>Gadget</th>
<th>158</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.2</td>
<td>Data Management</td>
<td>159</td>
</tr>
<tr>
<td>8.4.3</td>
<td>Communications</td>
<td>160</td>
</tr>
<tr>
<td>8.4.4</td>
<td>Web based interface</td>
<td>161</td>
</tr>
<tr>
<td>8.4.5</td>
<td>Digital signature</td>
<td>163</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

This master thesis aims to offer a state of the art technological vision of the current internet services provided and used by the top reference social network platforms, as well as their these services’ capabilities. The study is intended to offer a viability analysis to attach these services in the e-Catalunya platform.

The reader should know that this project is focused in providing a study. Although this project’s main goals were not to implement the studied technologies, I carry out some pilot tests for technology and service validation.

The following chapters will introduce the motivation that drove the development of this Master Thesis as well as the context in which it took place.

This document is intended to provide an idea about the work done within this project, as well as the methodology followed to achieve the project’s purposes.

1.1 Background

This Master Thesis has been developed inside the e-Catalunya frame project, an initiative promoted by Generalitat de Catalunya in collaboration with the Barcelona Faculty of Informatics (FIB). The project is developed by LCFIB (Laboratori de Càlcul de la facultat d’Informàtica de Barcelona) where I did all my thesis work.

E-Catalunya is a web 2.0 platform setting a common workspace that encourages collaboration between different groups within the Catalan society (e.g: Biblioteques judicial, E-moderadors, medició familiar, veïns). The user owns a set of tools to interact, organize and share opinions about different topics of interest with other members, contributing to the creation of the social knowledge networks. The platform is structured in portals where both professional and social groups can create social communities. The E-Catalunya main objective is to offer to
professional groups the tools to enhance the efficiency, quality and teamwork projects’
communication.

The e-Catalunya project started by the end of 2004 with the aim to create an innovative e-
government platform based on open source software and social networking capacities. The
first pilot implementation was run in September 2005 and more than ten new releases have
been deployed since then. In the present the platform has already reached 56 portals, more
than 1,200 groups and 16,000 users are currently registered in the platform.

In the few years have appeared a large quantity of companies offering services to their clients
through the internet. Google, Yahoo, MySpace, Linkedin are already offering mechanisms to
export their user’s information, for example like contact information. But not only this, more
and more of these giants are also offering high level functionalities provided through the
software as a service (SaaS) model.

From this point of view, it seemed interesting to see which possibilities there are in order to
integrate e-Catalunya with these services by establishing synergies with their platforms.

1.2 Motivation

The LCFIB was interested in the “new” services offered by the social network, which have
gained popularity in the last few months and its properties. They wanted to know what
possibilities could be integrated to e-Catalunya and their pros and cons. The target of this
project is to provide a specific study to answer to this lack of information and then make a
decision about how to adapt the platform to the current times. To perform this update I will
focus on content exportation and importation capabilities.

The main requisites to carry out this study are that e-Catalunya becomes through the studied
technologies a more present social network on internet. Offering the same possibilities to
export its content and applications to other websites, like massive social networks already do.
Another main objective of the study I carried out was finding out if e-Catalunya could get
integrated easily with other existing social networks. The objects of study were the different
mechanisms that could be used to create interaction between the platform and other
networks, and which were the limitations of them.
To add these services to e-Catalunya would mean to open its doors and allowing other communities and organizations to interact together with it. As this project is focused in giving a technical description about large set of services’ functionalities and their implementation only a few of them will finally become implemented in the platform.

1.3 Project Objectives

This project has two differentiated main goals. The first one is providing LCFIB and Generalitat de Catalunya with a study about how technologies are being used nowadays by social networks to achieve their social purposes. The second goal is making a decision about which capabilities would be interesting to be aggregated to the platform in order to fulfill current needs and challenges. Finally, a prototype will be developed to analyze the viability of the decisions made.

1.3.1 Study

e-Catalunya already covers many areas required to be a social network, like collaboration tools, friend network, syndication systems ... However the social network world is enlarging very fast and its capabilities are growing amazingly. Some of these new features are missing in the platform, and due to the impossibility to integrate all of them a study to define which of them are worth, the effort is not desirable but mandatory.

Once I did the study of these technologies, I arrived at some conclusions and exposed my point of view about what parts could be added in e-Catalunya. I showed to my supervisors and people who make decisions in the platform. Taking my studies and considerations into account they got clearer ideas about what could be done with the platform and finally suggested I focus my job at developing certain parts in which I proposed enhancements.

The detailed results of that meeting will be explained in the following pages, but now I will sum up the main points that were extracted from it. This will guide the reader in order to know exactly what the implementation purposes in this project are.
1.3.2 Practical Implementation

The second part of this project focuses on doing the appropriate changes to the platform to integrate the capabilities that the e-Catalunya project managers determined appropriated to improve importation and exportation content towards and from the platform. This was done by means of a pilot test in which such new capabilities were added.

To achieve my purposes, I saw that gadget technology was essential. A gadget is a mini-application to be executed at the end-user browser while rendering whatever website it is embedded in. We will learn more about them in coming sections.

The idea of this pilot test is to extend the current infrastructure with the necessary changes in the platform to enable the addition of gadgets from external sources. For instance: translation, news, dictionaries and whatever kind of “gadgerizable” application. On the section “Implementation” we will see how this was accomplished by means of gadgets.

At the end of this project, e-Catalunya will provide a mechanism to make gadget integration possible in those platform’s sites where integration studies demonstrate it enhances user experience as well as contributes to complementing the content posted by e-Catalunya members. Users will have some freedom to set their mini-applications (standard gadgets) where they prefer.

The final target was not just making e-Catalunya become a consumer of information, but also to be a producer of it. By also providing gadgets’ support on e-Catalunya the social capabilities were improved and promoting users being aware about what happens in the platform from outside too.

On behalf of this, the platform’s project managers suggested I to implement an application that could be embedded into any website and export activities from e-Catalunya. It should use some of the technologies I had previously studied. The concrete task was the development of a gadget that made the group calendar information available beyond the platform borders; this would help users to organize their personal agenda attending to e-Catalunya compromises and other source activity channels coming from whatever existing online source in “ical” format.

More detailed information about the practical objectives will be given later.
Project objectives:

• Creation of appealable and useful application exporting content from e-Catalunya.
• Users from different networks want to merge and centralize their information inside and outside e-Catalunya.
• Bring outside participation towards e-Catalunya.
• Applications connecting to Service Providers in order to retrieve information on behalf of a user. The idea is doing it from e-Catalunya and towards the platform as well (see section “Existing Service Providers”).

1.4 Expected Benefits

To illustrate the benefits that would be provided within the project objectives, I give some case studies use cases that may help to understand them better.

“Marc is a middle-age man who usually uses information technologies. However as many other people nowadays do, he knows how to manage some services. He uses Facebook to contact his friends, e-Catalunya for professional purposes and he joined iGoogle to personalize his own main page with his favourite feeds and applications.

Marc connects to each of these environments’ preferred personal information into his iGoogle. However he must go expressly to e-Catalunya to check his group calendar because he must attend to an e-Catalunya event and he doesn’t remember exactly when it is. The same can happen with some other desired information from the platform. Marc’s main browser page is set to his personal iGoogle so he tries to manage his life and get to know what is happening the world through this website. Thus he would like to add events from his own e-Calendaris in e-Catalunya into his personal Google or whatever other personal environment he uses.

Exporting e-Catalunya social information (e.g. contacts) could also be useful to have some social applications like birthday calendars or applications using Marc’s friends’ activities. “

“Peter has used e-Catalunya so much a lot since the beginning. He takes advantage of the collaborative tools and he wants to improve the quality of his contributions. He would like to add mini-applications (gadgets) to his posts to make their content more useful and understandable. All this would enhance support to collaboration tools by offering extra
understanding mechanisms. Peter would also like e-Catalunya to provide him a place where his chosen internet applications (like those from iGoogle) are shown. He is not subscribed to any other social network like Hi5, iGoogle or MySpace so providing an environment where he could manage applications (add, remove ...) would be useful for him”.

“Catherine is a psychologist specialized in managing Penitentiary institutions and she is not a member of the platform. She is looking for some professionals to contact who can help her to solve some issues. To address this purpose she could go directly to e-Catalunya and consult the e-Catalunya portals’ tools. However, in this case Catherine would take advantage of an application allowing her to post comments from websites other than e-Catalunya. This means that the tools from e-Catalunya would be opened to other website by means of embedded applications that would allow posting comments on an external website. At the same time, the content of these applications should be added into the thread from e-Catalunya which talks about the topic.”

“Antoni is another user who is new to the platform. He is not new to the social networking world, as he is already on many of them. Thanks to this, he has added many contacts on his contact lists from the others. Every time Antoni joins a new network he, wants to avoid typing all his contacts again and he tries to import them by means of to the Service Provider public services. Antoni wouldn’t like to add the contacts manually and he would like to use in e-Catalunya public services from third parties’ networks and retrieve contact information.

In the same way, he expects to use other kind of already existing importation data services that are currently available. For example, importing pictures systems, comments, book marks ...

1.5 Document Organization

This section explains in general terms how this document is structured and what the topics of each chapter are.

The current chapter introduces the subject of this thesis, explains the context in which this project takes place, its main goals and the motivation behind them.
The second chapter explains the e-Catalunya platform reviewing along with the software functionalities, its basic supporting technologies and the tools it provides.

The third chapter exposes the study of technologies I have carried out and it reviews the main concepts and standards used to import and export content through applications. It offers at the end a comparative study and conclusions about what of them can be useful and integrated to the e-Catalunya.

The fourth chapter presents the study done about external services offering storage, Image and survey services. A valuation of each one attending to their features and how they could match e-Catalunya needs is done. Finally the prototype implementation objectives are defined to be implemented.

The fifth chapter explains the implementation work to be done, in order to build a prototype using the technologies seen in the previous chapter.

The sixth chapter concludes this master thesis by offering some project conclusions and referring to the achieved objectives, the project costs, the initial time planning and its modifications and exposes possible future enhancements to be done related to this thesis. Personal conclusions are also include in order to know in what level the master thesis contribute in my professional development.

The tenth chapter presents the bibliography used during the development of this thesis.

The appendix contains a glossary for this document.
2 THE e-Catalunya PROJECT

This thesis is immersed in the e-Catalunya project. Before focusing on the subject of research, the goals and the followed method to carry it out, the reader should be given the context. Thus this chapter provides a little introduction to the platform functionalities, objectives and the current infrastructure that supports it.

2.1 Introduction

E-catalunya is a platform of virtual communities set up by the Generalitat the Catalunya and developed by the LCFIB at the Barcelona School of Informatics. The project is aimed at providing these communities with an appropriate framework for knowledge exchange, people process participation and teamwork by means of creating community portals.

Its main purpose is the creation of social, professional and knowledge networks between the users of the platform. e-Catalunya is composed a set of different community portals providing room for discussion, research and organization between its members, allowing them to generate, manage and exchange knowledge and experiences on the net.

E-Catalunya is intended provide a service to all those professional groups that take advantage of teamwork, i.e. doctors, teachers, etc without caring about distance restrictions. However, it is also meant for social groups built up from various communities such as patients, students, artists, etc, needing a space for information organization.

It aims to facilitate communication, interoperability, secure information sharing and collaboration on the World Wide Web. The platform comprises web tools to achieve these purposes such as wikis, forums, blogs, calendars, etc, and to establish communication environments attending different community needs.
Multiple portals have been created inside the platform to give support to different communities and providing feedback about functionality to the development team. As bugs are corrected and new functionalities and enhancements are added, new stable versions are periodically released to the public.

Some of the portals have experienced a large growth and are currently very active. For instance, the Justice portal that started in December 2006 with a pilot group, agglutinated in Jul 2007 1,669 users working in 32 different groups and 2,813 users and 59 groups by the end of June 2009.

2.2 Portals and groups

e-Catalunya is made up of different community portals providing a framework fulfilling various purposes. Each portal expects to be the nest for social activity from its users. These portals can include such a large diversity of issue that specialization and classification of content maybe don’t become necessary but mandatory for managing information. Otherwise due to the large number of topics talked about it, it can lead members to confusion. To give response to this fact, Catalunya includes portal group jerarquization and tools for managing it.

As far as it helps administrators to organize content, it also restricts users’ visibility to those groups that belong the information and it supports teamwork describing a workspace that hosts a set of tools for its members. Thanks to these tools members from the group exchange contents with each other.
Every portal can be public or private. All not logged in users will only be able to see and look up content on the public portals. On the other hand, those portals set as private will give access permission to those e-Catalunya members who had joined that portal before. Therefore, the platform provides a user account system in order to register and authenticate members.

Following the same fashion of work, groups’ visibility can also be public or private. Groups belonging to a public portal can be set as public, visible for any user, or private in which case it will only provide access to registered members that group. On the contrary groups belonging to private portals will follow the normal case permission policy allowing just group members to access content.

Groups also include other interesting features like the formality level of the group (formal or informal) or duration time (temporary or indefinite), which define whether group activity is meant to finish at a certain known date or not.
2.3 The e-Catalunya role system

e-Catalunya as a social network copes with multiple kinds of users. It expects to be an open platform where users can perform social activities even if they are members or not. Furthermore, inside the platform, the administration of portals and groups must be supervised for somebody who can organize and attend the users’ demands. All these users perform different sets of actions, which sometimes overlap and sometimes not, establishing different security levels of privileges depending on what they are allowed to do.

In e-Catalunya, the following roles are defined:

**Member:** any user belonging to a portal or a group. They can access any content inside the groups they belong to and participate in.

**Moderator:** members having a moderation role. They can perform the same activities as a member and even more; they can control content by editing, deleting or publishing content as well.

**Administrator:** Users with this role can fully administrate resources of the platform: system, portal, group. They can create new users, new tools, etc.

**Guest:** Another role may be considered with those users that participate without being registered or logged in. Users with this role might only access the public content of public portals.

While a group administrator can manage a group’s resources, a portal administrator can administrate the whole set of portal resources, including those of any hosted groups. The same applies for group and portal moderation roles.

This grants to the group members over the group resources, relieving like this portal administrators and moderators from group workload. This feature is especially useful for communities, where a large number of groups are expected and such an amount of work makes it impossible for the administrators/moderators to manage them.
2.4 Tools

Tools are the means by which users can share and deploy content into the social network. As they belong to groups and portals, tools may be public or private. Public tools are visible and accessible for those who can access and visualize the group or portal where the tools are located. A private tool is only visible and accessible for group members, in case of a group tool, or for portal members, in case of a portal tool.

e-Catalunya provides for now the following tools:

2.4.1 Wiki

A wiki is a collaborative tool which enables web content production, edition and distribution along a community website. It sets a common environment for users to interact easily with others contents by posting opinions about its articles or documents.

2.4.2 Blog

A blog is a web based tool with regular entries of commentary, descriptions of events, or other material such as pictures or documents. Blogs displays their content in reverse chronological order. It gives the viewer to post their comments about every of its entries.

2.4.3 Calendar

This tools shows a calendar and visually serves to keep track of the organized activities the user or group has on their agenda. This tool allow users inserting activities, joint and consult them graphically through the personal or group environment.
2.4.4 Repository

The repository tool creates a virtual space where users can share their files. It stores files from users and groups like a normal file system and allows the users to create folders and structure them in a hierarchy.

2.4.5 Photo Album

The Photo Album tool gives the user the chance to share pictures with another contacts inside the platform. Just like in the repository, in the photo album folders can be created and organised hierarchically. Users can comment every picture stored in the photo album.

2.4.6 Mailroom

The Mailroom is a tool for polling e-Catalunya users concerning different topics of interest that somebody may want to know other opinions about. It provides means to create questionnaires and to extract its results easily.

2.4.7 Mailing list

A mailing list is a collection of names and addresses to send e-mails to multiple recipients. Each group is provided with a couple of mailing lists by default: one that allows all users of the group to sent mails and the other that only administrators and moderators may use to send mails. A mail sent in any of the two lists will arrive to all the members of group.
2.4.8  Forum

A forum is another web based tool holding online discussions. They are used for managing user-generated content, sharing comments and experiences and making questions that anyone may answer. Forums in e-Catalunya are implemented using the phpBB2 software (php Bulletin Board).

2.4.9  eBugTracker

Thanks to this tool users can report easily the incidents, they may have when experiencing with e-Catalunya, to the group administrators. The eBugTracker is implemented using the Mantis BugTracker software.

2.5Social and Knowledge Network

The most interesting side of this platform according to my project’s objectives is the fact that e-Catalunya is a social network. Here I will offer some explanations about how e-Catalunya meets social requirements existing in any social network and what are some of the features that make it appealing to use between communities as a social means to cooperate and to achieve common team purposes.

Social networks are so characterized because every user has available a personal space where, among other things, they can detail their profile, search and store their contacts.

Any social network is mainly created by its people, the relationships established between them and the applications or features that enables them to mantain their social relationships "through" the internet. It can be seen as a graph where nodes are the individual actors within the networks, and ties are the relationships between the actors. In order to visualize these social relationships, e-Catalunya provides a graphical representation of a user and all his contacts. Any of the user contacts may have more contacts of their own that can be visualized as well on it building a network of relationships.
**Example of a social network**

While the social network is built using the user contacts, the knowledge network is built using the logged activities performed inside the e-Catalunya platform by them. So the network attempts to show to the user the portal members that according to their activities have similar preferences to him. For instance those members who use the same tools, read the same documents, do comments on the same topic articles ... summing up, those members that have more affinity to the user habits are shown. The more affinity a member has with the user, the closer this member is shown to the user. This means that the content consulted by those users appearing on the knowledge network are closer to the user will be proposed as interesting content for that user.
Example of a Knowledge network

Thanks to the knowledge network, e-Catalunya can extract some useful information from its users and suggest content to those users sharing similar interests. Some proposals like recommended documents are directly inferred from the reading activity the closer affinity members have, or even more give the chance to access the content created by these similar members.
2.6 Infrastructure Software and development environment

In order to allow the platform to work, it requires a set of software that composes the base where e-Catalunya runs. First of all I will describe all technologies involved in the e-Catalunya platform that are relevant to my project. At the time I joined the project, these technologies had already been chosen, thus no comparison with further technologies will be included.

Below there is a list of each of these components.

2.6.1 Webservers

2.6.1.1 Jakarta Tomcat

Most software infrastructure developed under the e-Catalunya project as well as the open source software added is programmed in Java. This is the reason why e-Catalunya needs an application server that allows the execution of Java code, for example Tomcat. This servlet container is responsible for serving the dynamic content of the platform.

2.6.1.2 Apache and PHP

Apache is a web server that serves static content very efficiently. Therefore, having an Apache + Tomcat will provide us with the necessary architecture for optimizing serving requests.

Apache has a large set of modules that enable the serving of other types of content, such as perl, php etc. One of the integrated tools in e-Catalunya is the forum which is implemented by the phpBB2 software.
2.6.1.3 **Apache Tomcat Connector**

All requests sent to the e-Catalunya platform go to the Apache server first. These connections sometimes require that Apache communicates with Tomcat so that it can process them. To enable this communication between both servers it is required the Apache-Tomcat connector.

2.6.2 **Data Management**

2.6.2.1.1 **MySQL**

MySQL is a well known GNU GPL relational database. As far as e-Catalunya needs a storage system for managing large amount of data and also provides reliability at servicing multiple connections simultaneously, MySQL is chosen for achieving these purposes. The large number of programming languages having libraries to use its API makes it a very good option to be used on systems based on multiple languages like ours. This DBMS apart from being one of the most used currently, doesn’t use too many resources.
3 STUDY

3.1 Description & Study Methodology

The steps I follow to carry out this project are the next the following ones:

I joined many social networks to understand how they work and what functionalities they offer. After getting some ideas, I started a more deeper study information flow mechanisms between websites. This information exchange required some protocols to be performed, thus I carried out a research on standards, API’s, specifications and infrastructures to understand exactly how this exportation and importation was done.

I focused my research on three main topics concerning content diffusion. On one side I looked for different ways to export information from e-Catalunya. The platform project managers wanted the e-Catalunya community to develop more than it had done until then, for this reason one way to make its content more available is by bringing it all around internet.

On the other side e-Catalunya needs to enhance its collaborative tools and sometimes there is the need to have external information available or some services to help our members perform their activities. In order to achieve this purpose I investigated what kinds of applications could be interesting for the platform and which were the requirements for them. In addition to this, I listed public services offered by the most well-known Service providers like: Google, Yahoo, MSN, MySpace, Linkedin ... These services allow users to retrieve their personal information and transmit it to whatever other personal environment they own. By using them, contact lists and agenda activities unification can be achieved and user heavy tasks like copying information from one place to another disappear.

Finally, after playing with these two previous possibilities I wanted to understand and investigate deeper what the real capabilities of these transmission information technologies were. Importing and exporting is great, but the idea was to learn if e-Catalunya tools could be converted into interacting applications in order to have them embedded inside any webpage.
The result of the study was successful and the studied technologies would allow achieving building applications in both senses.

Apart from the through study into these technologies, I was also advised to do some more studies about other aspects from the platform. The aim was the same, propose some ideas about how the platform could be changed or improved in the future. These other studies diverge from the main focus of my thesis, so the results of them are enclosed in the “Further Studies” section.

3.2 Importation / Exportation Content Services

The proliferation of different topic social networks promotes many internet users belong to more than one social network. People have different interests and they can feel themselves identified inside different groups of the society. All of a sudden we have seen how users have to cope with their profiles in many of them and this usually makes their profiles seems to be
unconnected and belonging to independent individuals. Since these profiles belong to the same individuals and normally contain the same features like; gender, age, profession, name, location ... and other information like contact lists information, the ability to transfer this information between its owner’s social sites is a great improvement.

This project centralizes its attention into exporting existing services provided by spread-used companies. In addition it takes a deeper look to dynamic content importation applications. The last studies Facebook Proprietary Applications API and finally it focuses on the main technology that will motivate the implementation of this project; Gadget Technology.

### 3.2.1 Existing Service Providers

More and more different social networks are offering new services transmitting their user’s private information. With this, users expect their network’s services to work together in order to accomplish their data unification. As no website does everything perfectly, users use different websites for different purposes (videos, pictures storage, e-mail accounts ...). The exportation possibilities are great because by its means any network community platform can provide users’ data wherever they required.

The appearance of these services also comes motivated because every time a user joins a new social network he must re-enter all his personal data to configure it. It comprises adding contacts, images, videos, bookmarks etc. By using public services, the signing up process is done more easily and data is coherent between user internet environments.

Not a long time ago social networks were reluctant to allow their users’ information to go beyond their bounders. They thought it would promote users moving to other networks. Opposing that, the growth of the number of users belonging to different networks forced interactivity between social networks. It was not just an enhancing feature for users but a vital necessity. With this on their minds they opened and adapted the platforms to make content available by means of public APIs.

Consequently in the present there exists many data Service Providers. Big companies like Google with Google Data, Yahoo with Yahoo! API, Microsoft with Microsoft Live services, MySpace ... already offer APIs based on REST calls that give support to content retrieval and
update mechanisms. Each of them provides a package of libraries to facilitate the job to developers who want to use them in different programming languages.

Google is the biggest public Service Provider at the moment offering its GoogleData API for reading and writing its user’s private information.

To integrate and get feed within these services, sites need to access the user resources from sites other than those hosting the service, and these are many times protected. Thus the strongest point for these services primarily relies on the communication security. It concerns mechanisms ensuring CIA, Confidentiality Integrity and Availability of the data.

As it happens, with many other technologies some companies implement their own systems to achieve secure data communication. This project will focus in OAuth protocol which is one of them. It has become adopted as a standard and it seems it is being consolidated on the thematic. OAuth also offer interesting capabilities to provide authentication for gadgets as we shall see later. Others like Microsoft use their own Windows Live ID Delegated Authentication systems to provide the same functionalities. For more technical information about OAuth protocol see on its own section.
3.2.2 Importing and exporting Content applications

It was not long ago when the web paradigm basically allowed users to browse the web searching for contents. In the present, this paradigm has been driven towards a more social side and it concerns more people. Today, users not only browse internet or send e-mails, but they are also encouraged to interact, contribute, share and participate with their own content. The internet has become a means of maintaining, creating and enjoying personal relationships between people who share similar interests, professions or likes. These new approaches of understanding the Web have revolutionized the WWW concept moving it beyond technology boundaries into a social phenomenon.

In order to produce more appealing content, different web platforms looked for a way to insert useful applications to their member’s spaces.

At the very beginning every platform began developing its own mechanism to integrate applications onto their platforms. They even encourage their members to create their applications by using their developing API. However, programmers could only display their applications in one determined environment as different platforms didn’t share same standards for rendering applications, nor API’s to get the data for feeding the application. That was not fair as the same logic of an application had to be re-implemented according to every proprietary API.

Following, the concrete case of Facebook proprietary applications will be treated apart. Once the reasons why a special chapter is devoted to this solution as a key issue from the core of this study are given, we will also proceed to the infrastructure and its application developing process explanation.

After this, we will go on explaining an alternative solution following a W3C specification. This specification allows platform-free application exportation by means of packaging software using a standard format known as gadgets.
3.2.2.1 **FaceBook**

Facebook is the most popular social networking website existing at the moment. A study carried out in January 2009 ranked that Facebook with its 250 million active users worldwide as the most used social networking. Thus it is not strange to say that it has become a reference for the social network world. Its popularity, number of members, social features and the aim of internet users to belong to the social network, where most of the people are inside, makes this platform one of the most suitable places on internet to spread for information to reach a huge audience.

Facebook was the first social network to offer developers the necessary environment and tools for using its proprietary API building applications. This fact created the explosive growth that Web applications have had recently. After it, many competitors followed its steps and they went their own way at creating their own proprietary API’s. As a result of this, many social networks’ API appeared diversifying completely the way to build applications. This encouraged developers specialize in different networks APIs and some applications were available in some networks and not others. The lack of a common API that establishes how to do the application exportation or rendering tied application’s usage to concrete platforms. This problem continuous until the appearance of gadget technology and OpenSocial.

As Generalitat requirements revealed in the beginning were mainly the platform content exportation around the web to as many people as possible, Facebook is a priori a very good option to achieve that point. However, although Facebook is the biggest social network there are many other big networks and if they all use the same technology to provide applications, the possibilities for them all to be useful grow very fast.

Gadgets have helped all those networks having their own APIs to develop applications to unify their efforts at integrating and exporting applications. With it their functionalities grow as the number of new gadgets are developed.

As Facebook has always been leading the social network technology development it is a model to be followed. Before gadget technology started to spread this platform had never cared too much about losing users because of the applications it was not providing and others did. Its managers were convinced that everybody wanting to spread an application would do it through their platform, because they were the easiest way to arrive to more people and it gave facilities to do it. But this way of thinking changed and finally they have opened their
platform by providing channels to publish its member information as well as they have recently integrated OpenSocket. It is an emulator that translates OpenSocial gadgets into Facebook applications.

We will focus here on the way Facebook develops applications. The Facebook Platform was released in 2006, providing a framework puts on for programmers to create applications that could interact with Facebook features. In order to do so, a new markup language called FML (Facebook Markup Language) was created to support it. This framework puts in developer’s hands the necessary methods to use the platform’s capabilities and building new cool applications. Rich applications can be created letting users interact with each other.

3.2.2.1.1 Application Registration

The process for submitting an e-Catalunya application in Facebook platform would be:

1) Join the Facebook Developer application to get access to applications creation environment.
2) Setting up a new application must be done from developer’s profile. Configure gadget data (e.g: callback function or url of the server where the application developed will be hosted) as well as required owner’s information. Two keys will be provided:
   • API key that identifies the application to Facebook and it is passed on API calls.
   • Application secret: used to authenticate with the requests done by the applicants.
3) Specify a canvas page URL. This url indicates where the application is deployed on Facebook. When users access to application they’re taken to that canvas page. Normally, this name references the name of e-Catalunya for example “e-CatFace Calendar”.

![Application Registration Process Image]
Process for building a Facebook application is explained in the next section.

### 3.2.2.1.2 Configuring a host for deploying a Facebook application

To set a Facebook application hosting environment in e-Catalunya some configurations must be done on the server. Firstly, it is required to set MySQL, however this is already done and we shouldn’t have to do anything. But there is still one more step to do which is uploading the Facebook PHP client library.

Once the host is configured a PHP generated file including the application’s API key and secret must be downloaded to the server from the Facebook developer website.

This file must be renamed into index.php and set it up in the same directory where the Facebook client library resides.

The logic of the application can be re-implemented to give the applications the desired behavior and functionality; e.g: implementing an e-Calendar exporting content application.

Users will be able to access that application through the canvas page URL on Facebook assigned to that gadget.

### 3.2.2.1.3 Facebook APIs

Facebook Platform is composed of a number of core components. By means of which developers can add applications to the platform and make them exportable.

- The proprietary API, which conforms to a set of methods that let adding social context to the applications. Facebook method calls are made over HTTP protocol by using GET and POST requests to the Facebook API REST server. Different libraries are created to support developers. For example: get friends, get mutual friends with friends ...

- FBML, Facebook Markup Language, it allows integrating applications into Facebook. It is a subset of HTML with some added elements specific to Facebook. It basically helps to hook the application into several points from the portal interface: profile, profile actions, Facebook canvas, News Feed and Mini-Feed ...
• XFBML incorporates FBML into an HTML page on a Facebook Connect site or an iframe application. At this time XFBML many tags are very similar to existing FBML tags.
• FQL, Facebook Query Language, allows quick and easy querying for Facebook user data without using the API.

FBJS, Facebook's solution for letting developers incorporate JavaScript into their applications. It allows accessing various features of Facebook Platform by doing, from any website, Facebook API calls through JavaScript. This fact helps creating AJAX applications easily (users must be logged in the platform to perform the calls).

By using them all, the platform’s content can be used to feed any of the applicant’s information requirements.

When creating an application there are two ways of configuring the application. One is using IFrames and the other FBML as the default for application’s canvas pages. So some differences should be considered:

FBML

• Fast on first page loads
• Easy access to lots of Facebook elements
• Sensible authorization mechanism

IFrames

• Let use the JavaScript, HTML, and CSS developing languages to which programmers are normally more used to.
• Applications are faster when they do a lot of AJAX calls, since the requests don't need to go through Facebook proxy.
• Debugging regular HTML and JavaScript is easier than for FBML and FBJS.

Some time ago, FBML applications were more frequently used than IFrame applications. With the appearance of gadgets, which already chose Iframes, and due to its popularity, Facebook has recently enhanced the chances by developing IFrame applications.
3.2.2.1.4 **IFrame Canvas Page renderization**

IFrame applications are very straightforward. When the user loads the application’s canvas page located in Facebook, Facebook renders a Web page that contains the Facebook chrome surrounding the IFrame and it provides information from the application. This IFrame contains the URL to the application host server callback endpoint. When the application server endpoint receives a request, it will perform the appropriate calls to the Facebook’s platform API to retrieve the required social content. Following, the server will render the application with that information and it will prepare it to be wrapped into an IFrame. Lastly the rendered code will be send back to the browser and the application will be displayed in the IFrame.

Here’s a simple diagram of how all this works:

![Traditional IFrame Canvas Page Diagram](image)

3.2.2.1.5 **FBML Canvas Page Renderization**

FBML canvas pages are different from previous ones. When the user requests the application canvas page the platform doesn’t send back a response at that moment. Alternatively, the platform does a HTTP POST request to the application server callback url function. Then Facebook wait until a FBML response is returned, and continuously it converts that FBML with embedded social content into HTML. Finally the code is sent back to the user’s browser.
Another difference with the IFrame canvas page applications is that in order to show names and pictures in FBML, it is not necessary to be done performing raw calls to the Facebook API. Instead, FBML tags can be used to retrieve data (e.g: fb:profile-pic, fb:photo, fb:mp3). For other specific kind of information calls to the API will still need to be done.

Here's a diagram of how an FBML canvas page works.

3.2.2.1.6 FBML Canvas Page

In order to develop Facebook Applications and cope with the core components, programmers are free to use the development environment of their choice. The platform officially supports PHP and JavaScript libraries, but unofficial libraries have appeared and provide the usage of FBML and others to write the logic of applications.
3.2.3 Standard application technology (widgets or gadgets)

3.2.3.1 Introduction

We have seen the concrete example of Facebook. This platform uses a proprietary API that allows it to integrate applications according to its specifications. But the development of applications following these standards doesn't reach application integration in any other platform.

To solve this problem, a standard for exporting applications all around internet was created. Applications following this standard should be rendered within any website without concerning what platform wrapped them. This was how the gadget initiative arose and sometime after has spread out.

The implementation of this standard helps developers to fix an historical external code integration process difficulty. Before gadget implementation arrived, the execution of dynamically included third-party applications in a website was achieved by using iFrames. This solution allowed developers to program websites so that the in case of loading and running external scripting code, they had to render it in a completely independent parent context (say the website containing that iFrame). This restriction avoided retrieved code to interact with the parent context host, for example by means of performing AJAX call to the server (cross-domain communication limitations). Because of this, the only thing that could be achieved was the rendering of external applications together in the same visualization page as the website, but that application couldn’t feed with the page host server information.

Gadget specification supposed the disappearing of separate iFrame context on remote application rendering. This meant that gadgets could then interact with the context of the social network where they were deployed even if the network and gadget domain was different.

These applications which are normally powered by scripting languages, like JavaScript, need some mechanism to get embedded and we will see later.
As has already been laid out, gadget specifications provided good opportunities for exporting applications into websites. Google and its partners did not remain passive in front of Facebook growing and thus they decided to come up with the implementation of the gadget W3C specification. The goal was clear, enlarging proprietary API application rendering capabilities and opening proprietary APIs.

We will focus specifically on one implementation of the standard carried out by Google and offering nice exporting and importing functionalities. The resulting implementation is called Google Gadgets and we will provide more detail later.

3.2.3.2 Gadget Technology

A gadget is a portable chunk of code that can be installed and executed within any separate web page by an end user without requiring additional compilation. It brings dynamic content from third parties’ sites without the website owner having to update or control it. Gadgets may be looked upon as downloadable applications which look and act like traditional applications but are implemented using web technologies: JavaScript, HTML and CSS. Other terms used to describe web gadgets include: widget, badge, module, webjit, capsule, snippet, mini or flake. The W3C uses widget notation and provides standard specification to standardize packaging its format. In this project we will use gadget notation instead of widget, the main reason is because we will focus on the implementation of widgets given by Google, which use this nomenclature. Gadgets are client-side applications that are authored using web standards, but whose content can also be embedded into web documents.

Technically speaking gadget specification relies on a XML configuration file, containing instructions on how to identify, verify, process and render the gadget. This specification must follow some standards and comply with XSD schemas. The gadget descriptor also declares metadata and configuration parameters.

The steps for processing a gadget package describe the expected behavior for the gadget presentation and means of error handling for runtimes while processing its descriptor file. Gadgets are deployed inside iFrames and its content comes via a separate HTTP connection with its own style sheet, cookies, etc. Final composition takes place in the browser as far as it is customizable.
Gadgets can be embedded into any website; however they are often used in customizable user homepages like IGoogle, hi5, WidgetBox, Netvibes, pageFlakes ... These portals normally permit creating a personalized homepage that contains a user personalized choice of gadgets. These gadgets come in lots of different forms and provide access to activities and information from all across the web, without ever having to leave the gadget portal page.

Some things which can be done with gadgets are: View the latest news, e-mails, check out weather forecasts, stock quotes, and movie listings, do quick search for words, dictionary or translator services, check daily weather etc.

3.2.3.2.1 Difference between Portlets and gadgets

At first sight it is easy that the reader can’t see the difference between a gadgets’ features and those given by portlets. The main difference between both is that while gadgets are web pages integrated by means of iFrames with their own CSS, cookies, etc, portlets produce HTML fragments assembled into a single HTML page sharing style sheets and cookies. In portlets the final composition takes place on a portal server, and a single page is delivered to the client browser unlike in gadgets.

Comparing portlets versus gadgets, we also find that portlets require fewer HTTP connections, they allow for common styling (all portlets in a website), they can communicate between each other and take advantage of common authorization features. (so that a user doesn’t have to sign on to each portlet separately).

Portlets use a window-manager metaphor that allows the portlet server to resize them, whereas iFrame-based gadgets don’t do any of that. They don’t require special portal servers, they can be embedded in more creative ways, offloading like this, the server of final client presentation process.

Portlets are often used in intranets and social networks, to provide a collection of applications on a single web page for their members. (e.g. news feed, calendar, expense forms, bug reports, etc.). On the other side, gadgets are used everywhere else not only inside company boundaries (e.g. embedding Google maps, Facebook applications, etc.). As far as they are hosted somewhere that’s not under the portal’s control, this can include some potential security holes.
In the future, as extra features of portlets may not be compelling enough to justify the extra cost of running a portlet server, it is intended that gadgets will substitute them in the short term.

The only portlet feature with compelling benefits is common authentication and when this issue is overtaken, gadgets will probably push them out indefinitely.

### 3.2.3.2.2 Gadget supporting infrastructure

Gadget technology implementation requires some infrastructure to support its functionality. These applications can be easily developed, but once the XML file descriptors’ are created, they must be submitted to a rendering process. In order to carry out this process, all rendered gadgets must be available on a public location, so that the server container can retrieve the gadget descriptor code. Before being displayed, the server will parse the file and later it will generate the appropriated gadget code to be interpreted by the end-user’s browser. The code will be a merge of html plus a scripting language like JavaScript that the browser will finally display in the place the websites indicates.

A context or site into which a gadget is embedded is called a gadget container. To deploy a gadget it is necessary to have a server container in charge of the rendering process for managing the gadgets’ layout and controls.

According to the environment gadgets are embedded in, there are two different approaches to render gadgets. The first option is their deployment on gadget containers portals, which are normally offered by Social Networks. The second is gadget rendering service provided to any website around the web. As we will see the requirements for each of them are different. After this study some conclusions will be extracted and a choice between them will be done. This election will outline the implementation work to be done in this project according to the e-Catalunya exporting and importing content needs.
At present many, to social networks (Orkut, iGoogle, MySpace, WidgetBox ...) have brought gadget service technology at their user’s scope. Most of them already offer already a gadget container by which users can interact and get to know about world and all that gadgets bring to their personal environment.

The developers’ point of view about these public personal containers is positive because it allows them to display their new gadget creations. Taking advantage of this technology and the paradigm “write on it everywhere” is very easy to spread interesting gadgets around the WWW. However, this approach is good when these programmers want their applications to be displayed on personal environments implementing a gadget container; these environments even allow them to take advantage of their social capabilities. But, when gadgets don’t care about personal usage and want to be deployed inside gadget-container-less websites a gadget renderer service must be understood separately from the gadget container. In this case the need of a container offering just rendering services is required. In the next section we will examine more closely the ways to obtain this service.

3.2.3.2.1 Gadget Container

Above gadget rendered inside a gadget container portal, below gadgets embedded inside any website.
Some examples are: iGoogle, Orkut, MySpace, Hi5...

3.2.3.2.2 Containers offering gadget rendering services

Gadget containers websites are attractive because they provide gadget visualization services and allow unification of different source information to a central place.

Not only social website providing a gadget portal may want to integrate rich content to their sites making them more attractive to the visitors. Aiming to spread gadget functionalities and keeping an eye on the difficulties that IT departments and developers can have to maintain a service giving support to this technology (time-costing), Google is present providing a free gadget rendering service. It puts audience disposal a gadget container service which allows developers to ignore gadget server maintenance and focus only on application development. However this service is only offered over gadgets submitted to its gadget directory and otherwise the rendering is not possible.

Google also supports and enhances gadget development by adding useful functionalities that programmers can use to build more appealing applications. When developing gadgets using these capabilities, it is important to keep in mind that if they are rendered in other server containers, problems can appear. Basically due to this libraries are owned by Google and are only available for its server container.

When third party rendering services, like Google’s, don’t satisfy the desired gadget features, meaning they do not provide the necessary JavaScript libraries, or a platform wants to integrate its own gadget container to provide social functionalities (by means of OpenSocial), it
is compulsory to set up a personal container. The personal set up of a server container will allow meeting the desired capabilities and requirements for the service aiming to be provided. Apache Shindig is an open source project whose code can be plugged into a server’s infrastructure (e.g: Tomcat) to host social applications (OpenSocial). This solution could be a perfect option to achieve these purposes.

(A period of time dedicated to this project was focused on providing the steps to integrate Apache Shindig with e-Catalunya platform. This information has been moved to that section because it contains high technical level information and it could difficult a non expert reader’s understanding. Please consult the project appendix if you want to get more information about how this process is done.)

3.2.3.2.3 Apache Shindig

3.2.3.2.3.1 Shindig Server Container Requirements

The java version implementation requires a servlet container supporting Web Application 2.3 or above and JDK 1.5 or above.

Apache Maven installation is needed. Maven is a software project management tool that can manage a project’s build. It is required to build the project once Shindig code has been downloaded.

3.2.3.2.3.2 Server Container

Shindig is a reference implementation of the OpenSocial API Google specifications. This server container extension is currently under development at the Apache Software Foundation. There are a couple of available versions one in Java and the other in PHP. I will focus on the present first one as it is the more mature. Shindig java implementation sits on top of servlet container layer.
Shindig provides a rendering engine for gadgets, a translation layer from private platform data to the OpenSocial data types, handle REST and RPC requests, and an implementation of the security communication protocols required for the gadgets to access that data (OAuth protocol).

(For more detailed information about the infrastructure of this server container please consult the project appendix. This information has been moved to that section because it contains high technical level information and it could difficult a non expert reader’s understanding.)

### 3.2.3.2.4 Google gadget implementation

Google has gone further away than providing just an implementation of gadget standard and it extended W3C gadget specification with the creation of the Google Gadget API. It offers, gadget.* API’s allowing developers to build cool applications easily by means of libraries supporting data management and presentation functionalities. To achieve Google gadget rendering, a server container must implement Google Gadgets API.

This API also provides support calls for OpenSocial layer which we will see in the section “OpenSocial Layer”. This API’s OpenSocial functionalities are supported only for those containers wanting to offer OpenSocial gadgets rendering.

As we can see, Google has firmly bet on this gadget technology, and thanks to it, is today providing one of the most extended personalized homepage services in the present, iGoogle. This portal allows users to personalize their content by adding those applications Google Gadgets API compliant they are interested in. This way user can get the information they require and those functionalities they may need inside an unified environment.
3.2.3.2.5  Google Gadget Anatomy

```
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="hello world example" />
  <Content type="html">
    <![CDATA[
Hello, world!
]]>
  </Content>
</Module>
```

This is a very simple gadget displaying “Hello world!”.

The first line from a Gadget descriptor XML file must be the standard way to start an XML file. This defines the XML version and the codification of its content. Later on the <Module> tag indicates that this XML contains a gadget.

The <ModulePrefs> tag contains information about the gadget such as its title, description, author, preferred sizing and other optional features. The <UserPrefs> section defines controls that allow users to specify their settings for the gadget.

The content tag specifies if the xml holds the content itself or has a link to external content. The <Content> section is where the gadget attributes and user preferences are combined with programming logic and formatting information to become a running gadget. It also contains the HTML elements that determine the appearance of the gadget.

The line <Content type="html"> indicates that the gadget’s content type is HTML. <![CDATA[ ...insert HTML here... ]]> is used to enclose HTML when a gadget’s content type is html. It tells the gadget parser that the text within the CDATA section should not be treated as XML. In this section the real work of the gadget happens. When writing a gadget, this is the start section.
The inserted code in this section assumes that all required functionalities or gadgets features are available for its renderization. The features a gadget need can be required or optional. An API must be consulted to know if the feature are available and ensure the capability is enabled. The code will finally be processed by a gadget server and rendered in an IFRAME.

</Content> signifies the end of the Content section.

</Module> signifies the end of the gadget definition.

### 3.2.3.2.6 User Preferences

This section defines controls that allow users to specify configuration and persistence settings for the gadget. Some gadgets need to give users a way of supplying user-specific information. The user preferences section in the XML file describes the user input fields that can be used into user interface controls when the gadget runs. User preferences are stored persistently during execution time. Gadgets may perform persistence on behalf of a user so that the gadget has access to them across multiple rendering requests. The responsibility of providing this persistence is the gadget container which may or not have implemented an interface for this.

### 3.2.3.2.7 Gadget Features

There are two different types of JavaScript libraries in the gadgets API: the core library, and the feature-specific libraries. The first one includes general functions that aren't related to a specific feature. These functions can be called over a userprefs object to manage its data, as well as methods that aren't specifically related to userprefs like those retrieving diverse data from remote locations.

**Core JavaScript API**

It is provided automatically by the server in such a way to conform the Gadgets API Reference (for more information consult [http://code.google.com/intl/ca/apis/gadgets/docs/reference/](http://code.google.com/intl/ca/apis/gadgets/docs/reference/)).
File   JsDoc

**prefs.js** Provides access to user preferences, module dimensions, and messages.

**io.js** Provides remote content retrieval functions.

**json.js** Provides operations for translating objects to and from JSON

**util.js** Provides general-purpose utility functions. (String managing, features check function and registration event functions)

On the other side, the gadget API includes the following feature libraries.

**Feature-specific libraries**

Gadget features are the primary extensibility mechanism employed by gadgets. They often direct a gadget server to make new JavaScript APIs available to rendering code, but may also manipulate the gadget's contents, for example to add extended syntax.

Most gadgets insert one or more features. The more features a gadget server supports, the more gadgets it supports. Thus, a gadget server should support as many features as possible. Special attention the next exposed features deserves as they have high usage rates. For this reason the default implementation of server containers already include support for them and no problem of rendering appears.

Each feature supplies a JavaScript API to be made available to the gadget. The table below lists and describes each feature and link it with its API:

<table>
<thead>
<tr>
<th>Feature/Library</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tabs</td>
<td>Allow the addition of tabs into gadgets, for info organization purposes.</td>
</tr>
<tr>
<td>minimessage</td>
<td>Allows user-targeted messages creation within the gadget: Status, Promotional or error messages</td>
</tr>
<tr>
<td>Flash</td>
<td>Embeds Flash content in gadgets.</td>
</tr>
<tr>
<td>Rpc</td>
<td>Provides operations for making remote procedure calls for gadget-to-container, container-to-gadget, and gadget-to-gadget communication.</td>
</tr>
</tbody>
</table>
Views
Provides operations for dealing with different gadget views capabilities.

skins
Provides operations for getting display information about the currently shown skin.

dynamic-height
Provides operations for retrieving information about and modifying the window the gadget is placed in.

Settitle
Allow to set gadget’s title programmatically. (comes with the same library than dynamic-height).

Gadgets can run on multiple sites and platforms and there are special tags and libraries that can be added to their codes. These extra functionalities can be optional or required when rendering a gadget and its usage is tied to the gadget container, which may or may no implement some of these functionalities. For instance these functionalities can attend all kind of purposes like: presentation, authentication, social capabilities etc.

<ModulePrefs
title="My e-Catalunya Calendar">
<Require feature="dynamic-height"/>
</ModulePrefs>

In this case, the gadget container would probably not have problems rendering the gadget. But, another gadget that was developed specifically for iGoogle, which for instance added the feature locked-domain, wouldn’t be able to be rendered, as this feature is specific to just this container.

The locked-domain library isolates a gadget from other gadgets running on the same page. This feature is recommended for those gadgets storing sensitive private user data. This feature is only supported on iGoogle ().

Some restrictions are applied to “url” type gadgets (see the next section) on using the gadgets JavaScript libraries. In this case they can't take advantage of all gadget API features, mainly those features concerning about HTML and JavaScript direct manipulation.
3.2.3.2.8  Gadget types

In this implementation, the XML gadget description files can contain all the data and code for the gadget, or they can be url-referenced and retrieve the necessary information from any source.

**HTML type**

With the html content type, all the code typically resides in the gadget spec. This includes the gadget XML, and any HTML markup and JavaScript.

**URL type**

These gadgets do not provide any code in the gadget specification section and all their content is on a remote web page referenced by a URL in the gadget specifications. In the remote web page all the HTML and JavaScript code resides. These are useful for those gadgets whose content is dynamically changing. This content type is not supported by the gadgets.* or OpenSocial APIs.

   <Module>
     <ModulePrefs ... />
     <Content type="url" href="http://www/cgi-bin/example/gadgets/mystats.cgi"/>
   </Module>

This type of gadget is very useful when turning existing web sites or applications into gadgets. Moreover it is also the best choice for gadgets requiring login to be implemented. Thanks to this, developers’ are allowed to use other scripting languages than JavaScript.

While HTML content type gadgets are automatically generated and rendered inside iFrames by a Gadget server, on url type programmer is responsible for managing how a gadget must be displayed (say graphical interface wrapper).

A Gadget server or server container is in charge of performing the rendering of the gadget. Behind a displayed web-side gadget there is always a server performing rendering job, we will see exactly what this server does later.
To perform renderization when the gadget URL is requested, gadget’s parameters are appended to the GET request as key-value pairs. User preferences are also passed like this. These parameters will be later read by a gadget using JavaScript, or by a server-side web application.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<Module>
  <ModulePrefs title="Preferences for __UP_mynname__" height="250"/>
  <UserPref name="mychoice" display_name="List Params?" datatype="bool"/>
  <UserPref name="mynname" display_name="Name" required="true"/>
  <Content type="url" href="http://www.examples.com/params.php"/>
</Module>
```

### 3.2.3.2.9 Gadget Views

Gadgets can have different implemented views. A view is a location and/or dimensions in which a gadget is displayed. Every view has a different characteristic in every gadget. Views are specified by multiple content sections inside a gadget XML. Each of them is labeled with one or more optional view identifiers (Profile, Canvas, Home, preview), which allows the gadget to behave or appear differently depending on the context in which it’s rendered. This context is provided by the gadget container and the most usual are:

**Home view:** The small view for a gadget, displayed with other gadgets. It shows all inside gadgets of a gadget container in their small display format.

**Canvas view:** The large view for a gadget, displayed alone without any other gadgets.

![Left top: gadget home view.](image1)

*Left top: gadget home view.*

![Right bottom: gadget canvas view.](image2)

*Right bottom: gadget canvas view.*
3.3 Enhancing e-Catalunya with Google Friends & Facebook Connect

3.3.1 Google Friends

Google Friends is a powerful gadget application enabling social capabilities to the community visiting a website. It is a mechanism allowing any user belonging to a Google, Yahoo, Aim, etc account to interact with whatever website including Google Friends Connect. This way, those websites are not providing user session control or they want to allow non-members to do contributions to do it. To support this social functionality, Google use its own gadget infrastructure.

This tool allows the management of comments from third parties’ users, without requiring the developers of that website any effort in maintaining it.

Google Friends allow anyone to join a website in order to comment or give their opinion.

It offers different capabilities through different kinds of gadgets when users/viewers log in. For example: invite their friends, rate content, post comments (translation feature allows any user to see others comments on their preferred language), sending recommendations of the page, monitoring last users activities, and providing activity monitoring on other social networks like Facebook or twitter.
No programming is required to integrate this service in e-Catalunya, just the addition of a few snippets of code. It will integrate with existing Google login systems to identify the user. As we are talking about Google, this will be done by implementing the OAuth protocol to achieve authentication.

Google Friends allows user not only interact with friends and social information coming from the account into he logged into the applications, but also provides mechanisms (libraries) to allow social information retrieval from the social network it is embedded in. Of course it requires a back-end that supports calls to this mechanisms and for these reason the server must be OpenSocial Layer compliant.

3.3.2 Facebook Connect

Thanks to this service, Facebook users are enabled to login to affiliated sites using their Facebook account and integrate into them the social power capabilities of Facebook. It is easy to integrate Google Friends by adding chunks of code on the target websites.
Once a user logs into a Facebook connect application in whatever website, these applications, gain access to a big amount of the user’s Facebook account data: profile’s data like user’s name, photos, data about a user’s friends, notifications, and others. The following capabilities can be offered with this data to the users navigating on the websites including Facebook Connect applications.

- Sharing website rich content; text, image, flash elements, embedded links and user-generated content, and user’s comments ... with other Facebook friends.
- Commenting on the website using user’s Facebook identifiers, and bring those conversations back to users’ friends on Facebook.
- Registration and log-in into these websites using Facebook account.
- Recommending websites to users’ friends on the platform.
- Providing users with a list of recent activity taken on that site. Showing which user’s friends have commented most recently and which articles.

In addition to these services offered on the web embedding Facebook connect applications any external action performed on these websites will be registered on the Facebook user’s activity stream.

This service can be understood as a single sign-on service like Google Friends. This means that when a user logs in once in Facebook it gains access to all systems, including these Connect applications, without being prompted to log in again at each of them.
Facebook Connect is created by a set of components, some of which can be independently and quickly integrated into any website. Here I show some of those social gadgets that could be interesting for e-Catalunya.

Comments Box is a standalone social gadget for any website easily allowing a website’s users to comment on it, once they are logged in. Comments can also be anonymous in case the user doesn’t want to log in. The application also allows showing all comments made in the Comments Box. Users can share their comment directly on their Facebook profile and in their Facebook asks for permission to import his friends.

User log in to the application embedded in a website.

User is enabled to perform any of the actions told before. In this example the user decides recommending the website to his friends.

Facebook asks for permission to import his friends.

User can see the profile from those Facebook users sign this website.
friends' home page streams. Each story on Facebook includes a link back to the page the comments box is on.

The Comments Box has a default appearance which can be personalized by defining a url stylesheet of the website where to be included.

Example of easy integrating the Comment Box

To add the Comments Box to e-Catalunya basic steps should be done.

- Setup a new Connect application and get the Facebook API key and specify a callback URL to your website.
- Upload into the website website the file xd_receiver.html:

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Cross-Domain Receiver Page</title>
</head>
<body>
<script src="http://static.ak.facebook.com/js/api_lib/v0.4/XdCommReceiver.js?v2" type="text/javascript"></script>
</body>
</html>
```
Add these snippets of code to each web file where you want a Comments Box.

- Adding this code wherever it is desired to appear the Comments Box.

```html
<fb:comments></fb:comments>
<script type="text/javascript">
FB.init("YOUR_API_KEY_HERE", "<path from web root>/xd_receiver.htm");
</script>
```

The Live Stream Box gadget lets users visiting a site where it is included to share activities and comments in real time. The Live Stream Box works in real-time and it also provides interesting social capabilities like live Web chats (supporting millions of simultaneous users), posting content, webinars, mass-multiplayer games, and more.

As Comments Box it shares users’ updates directly on their Facebook profile and in their friends home page Streams, where, each of them includes a link back to the source sit.
Apart from integrating all these existing services to websites, users can also develop their functionalities to include Facebook social capabilities inside a website as they want. They can take advantage of Facebook login system to authenticate users and after when they create content add Facebook’s rich social context on the site. They can register some actions performed on the website to the activity flow register from facebook, or update the user status or even retrieve user’s profile picture to do any action on the web requiring that image. They can also show which of your users’ Facebook friends have accounts on that site by checking permission allowed with their friends to the website before.

### 3.4 OpenSocial Layer

The reason why I introduce this technology in my project’s study is because it offers very interesting functionalities for achieving social information diffusion (profile, activities, contact ...).

More and more, the need for internet applications is increasing. Due to this recrudescence, the social networks must spread the information though their own API.

At the beginning of my study, I took a look at several proprietary API’s enabling data-portability but their possibilities were limited, after I gave a wider perspective to my study and I became straight inside the OpenSocial common API. Going through a thorough study of its capabilities I realized the large amount of possibilities implementing OpenSocial and combining it with gadget technology could be reached by e-Catalunya. Since that point my job was figuring out how e-Catalunya could use the already commented technologies to promote the platform social aspect.

Sometime ago, in order to contribute enhancing social network’s features, networks started developing applications which maximized capabilities on their member’s information and gave interesting functionalities about them. More than just letting the users exchange opinions and interact, many interesting applications were developed using owner’s contact list. For example applications showing upcoming events, friends’ activities, birthday calendars or giving advice from several user activities inside that network.
The capabilities of the social networks improved notably thanks to these applications. Applications depended on the data coming from the social network they were embedded in. For this reason in order to retrieve social data, an internal channel of communication (API) between those applications and social network’s back end needed to be set. Thus with this new fashion towards more social applications came a growing list of site-specific APIs that developers had to learn. The proliferation of APIs across so many social networks forced developers to choose which ones to write applications for, and spend their time writing them separately.

The main point of this was the need for social applications to have global access to user’s data. They needed to be fed with different standard-compliant platforms which didn’t share same API. To overcome these difficulties common procedures for data retrieval needed to be set between social platforms. Some social networks took the initiative and established a common interface for developers wishing to create social applications accessing social data transparently to the application server backend. As a result of this OpenSocial API basis were set for social applications across multiple websites and platforms got quickly to work on it.

Google led the development and promotion of the API together with other big social networking companies, like including hi5, LinkedIn, MySpace, Netlog, Ning, orkut, and Yahoo!

OpenSocial provide a Social model which can somehow be modified to cover social platform model class diagrams. The basic classes are Person and Activity.
What is it?

OpenSocial is a common initiative aimed to be an API reference for social networks in order to make social-application developers’ job easier.

What does it offer?

It is intended to give mechanisms to standardize social services and force social information to flow across the Internet. Developers can create far-reaching applications that access a social network's friends and update feeds regardless of social network, web-applications or websites they are embedded in.

The OpenSocial APIs offer some methods for making social network members’ information available on internet through its API. It publicizes information concerning people, their friends, and activities from within the context or social network it is giving support to. This means that when a user runs an application on a social network, it will be interacting with his friends on that network, while running the same application on another container like MySpace the application will interact with his MySpace contacts.

This interface is used for other social applications (normally gadgets) which may want to import third party information about their user’s to achieve their purposes. More and more social networks are including in their user profile information monitoring their user’s activities on other social sites. OpenSocial opens any platform where it is deployed, to export its social content beyond its network’s borders and let information flow between users’ environments. As this layer lies on gadget technology, applications developed using this API are runnable in any Social Network implementing OpenSocial. Following the paradigm “write one, run everywhere”.

In short with OpenSocial layer, containers allow their gadgets to interact with the social network’s core database and provide personalized gadget social content.

**3.4.1 Technical issues and implementation**

Applications implementing the OpenSocial APIs will be interoperable with almost any social network system that supports
them. OpenSocial includes four JavaScript APIs supporting social applications to access data and core functions. Each API addresses a different aspect: one to request for people’s personal information and friends relationships, one for user activities, and the last offering persistence services over social user model (for example for saving results of a social game, or saving gifts a user sends to another).

Persistence API offers storage for those data and customizations done on gadgets by the users and they want to be saved stored. This storage takes place for each user on their social network account. These customizations can after be after retrieved when logging in again on the gadget and they suppose a big enhancement in front of not social gadgets.

The OpenSocial JavaScript API includes two namespaces: opensocial.* and gadgets.*.

Developing social gadgets implies using the OpenSocial API. The only thing that must be done is adding the following line of code in the XML gadget descriptor file, so that the OpenSocial libraries are loaded and the corresponding data retrieval actions towards the core from social network can be called.

This line to be added inside the ModulePrefs tag, is the next in bold:

```xml
<Module>
    <ModulePrefs title="Gifts part 0 - Boilerplate">
        <Require feature="opensocial-0.8"/>
    </ModulePrefs>
</Module>
```

Once this is done the developer only has to program the logic of the application by using the JavaScript OpenSocial API provided by the platform.
Here we have an example to show how a social gadget implementation is done. This gadget lists its viewer’s friends:

```javascript
function loadFriends() {
    var req = opensocial.newDataRequest();
    req.add(req.newFetchPersonRequest(opensocial.IdSpec.PersonId.VIEWER), 'viewer');

    var viewerFriends = opensocial.newIdSpec({ "userId" : "VIEWER", "groupId" : "FRIENDS" });
    var opt_params = {};
    opt_params[opensocial.DataRequest.PeopleRequestFields.MAX] = 100;
    req.add(req.newFetchPeopleRequest(viewerFriends, opt_params), 'viewerFriends');

    req.send(onLoadFriends);
}

function onLoadFriends(data) {
    var viewer = data.get('viewer').getData();
    var viewerFriends = data.get('viewerFriends').getData();
    html = new Array();
    html.push('<ul>);
    viewerFriends.each(function(person) {
        if (person.getId()) {
            html.push('<li>', person.getDisplayName(), '</li>');
        }
    });
    html.push('</ul>);
    html.join('');
    document.getElementById('friends').innerHTML = html.join('');
    gadgets.window.adjustHeight();
}

function init() {
    loadFriends();
} gadgets.util.registerOnLoadHandler(init);
</script>

<div id='main'>
Your friends:
<div id='friends'></div>
</div>
```

- This function loads the friend data. It creates a new data request object and populates it with specific types of data that will be returned: the viewer and the viewer’s friends.
- An IdSpec object is created and used because it is needed to specify one or more people in a group (in this case, the viewer's friends) to perform the friends request.
- Finally the request is sent to the server and the callback function specified.

The callback function, onLoadFriends, takes the server data returned, format it and display it on the page.

Div elements are used as entry points were data retrieved will be formatted and displayed.

Gadget XML file descriptor CDATA tag.
We can differ between the **gadget implementation API** and the **server-side implementation** of the OpenSocial network.

**A JavaScript API:**

- Provides client-side access to People, Activities and Persistence
- Enables embedding applications in a Social Network service's UI
- Built on top of Google Gadget technology

**And a REST API:**

- Provides server-side access and management to People, Activities and Persistence.
- Based on Atom Publishing Protocol
- JSON representation instead of Atom format XML

The addition of OpenSocial layer on the server container implies a deep study concerning the capabilities and services it has to provide. Depending on what the requisite analysis is, according to the functionalities given to the rendered gadgets, the configuration and requisites of the server container will be some or other. More features will have to be set up if gadgets attempt to perform third-party private data communication or if a persistence service wants to be enabled to save gadget preferences or data into user’s context profile. On the other hand, if only a rendering service is required, the implementation will be much easier.

As social networks’ current development hot issues point to the usage of this technology to make social information available cross-domain networks, it seems social data transmission mechanisms will go in the future, on that directions. So adding OpenSocial layer into the e-Catalunya infrastructure would set the basis for releasing e-Catalunya towards an open social network model, as well as giving the necessary services to the platform to adapt new capabilities in the future.

With the appearance of new social gadgets almost all current containers have integrated OpenSocial API support to them. In the next section we will have a look at how containers perform rendering process and which mechanisms must support for providing this service.
3.4.2 OpenSocial Specifications

To fulfil Opensocial gadget Container definitions, some features must be satisfied:

1. **The container must implement all methods in the JavaScript API Reference.** Both core library methods and those optional desired features gadget container must be implemented to be provided.

2. **The container must be able to integrate container-specific extensions.** These extensions may provide social network specific data retrieving, and the extraction of modeled objects not belonging to the default OpenSocial model.

For example, an extra activity or feature object field is specified e.g from e-Catalunya, it should be enumerated under the container’s namespace, and it should allow applications using the social layer to discover these fields and retrieve them through the API.

3. **The container must satisfy the Gadgets API Specification.** This means the server must be gadget-compliant. (See Server Container Compliance section).

4. **The container must support the RESTful Protocol Specification.** (See technological concepts section)

Containers must allow different formats data structure retrieving like JSON, XML ... In order to retrieve, update, create and delete data OpenSocial uses the HTTP protocol’s methods: GET, PUT, POST, DELETE in this order. In addition to this, they can also support the RPC protocol specification to give the tools to modify the social network database information.

*(For more detailed information about gadget rendering process please consult the project appendix. This information has been moved to that section because it contains high technical level information and it could difficult a non expert reader’s understanding.)*
3.5 OAuth Authentication

This protocol allows users to grant access to their private resources on one site called Service Provider, to another third party site called Consumer. OAuth basically verifies that requests made by an application are actually from the application and that it has permission to access potentially sensitive data. It allows a secure API authorization process which the applications must go through to gain access to the sensitive data. This step is preludes to any requests and queries over the waned data.

The basic aspect of this protocol is establishing a safe mechanism for exchanging a username and his password identity for a token one with defined rights over a service. OAuth is about providing access to user stuff without sharing. The main idea is that each access granted is restricted to a specific service from the user account. With this OAuth intends to avoid a user granting access to a third party server and this gets freedom to take whatever action on behalf of the user over that account.

As far as this introduces very interesting functionality at making personal information flow in between different online user’s environments, this adds serious potential security failure points too. HTTP requests are made insecurely and for this reason to allow users become safely authenticated, some kind of encryption method must be set.

Security and privacy are not strictly guaranteed by this protocol. In fact, privacy is not provided by OAuth at all, and it relies on other protocols achieving these purposes like, for example, SSL.

OAuth protocol eliminates three main weak points from the HTTP protocol’s authentication method called “Basic”. On one hand it transmits passwords unencrypted allowing everybody sniffing the net to capture and reuse those credentials. On the other, there is no link between the credentials and the request, so in other words they can be used without limitations if the message is leaked. Finally 'Basic' only supports a single username-password pair, whereas delegation requires being able to send both the credentials of the caller (Consumer) and those of the party delegating its access (User). Since data requests are made by the Consumer, it requires a way to authenticate itself in front of the Service Provider, but also to authenticate its authorization to access User data on behalf of him. This needs OAuth to support an HTTP
request with these two sets of credentials. OAuth protocol copes with all of them and addresses these limitations.

For insuring secure non-HTTP communications, OAuth signature methods were developed. HTTPS recommended a solution to prevent a man-in-the-middle, eavesdropping, and other security risks. However, HTTPS is often hard to setup and maintain. In the next section I will focus on some algorithms to allow encrypting data over insecure channels: HMAC-SHA1 and RSA-SHA1.

Before OAuth security protocol arrived, there were many proprietary methods for exchanging user credentials for an access token. For example: Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Upcoming API, Flickr API, Amazon Web Services API, etc. OAuth was created to standardize these protocols as well as improving them all by minimizing the transition existing services should have to perform to support OAuth.

One of the most important features of it is its direct handling of non-website services. OAuth supports all kinds of applications either mobile or desktop applications. Another essential thing is that this is an open source protocol, so here a secret shared key cannot be hardcoded into the retrieving software code as its vulnerability would be evident.

Here are some of the big companies already using this protocol on their public data services: Digg, Jaiku, Flickr, Ma.gnolia, Plaxo, Pownce, Twitter, and hopefully Google, Yahoo, and others.

### 3.5.1 OAuth for Web Applications

To explain the workflow required to acquire user authentication we will have a look at how Google supports the OAuth protocol for authorizing web applications accessing their users' Google services. Other platforms implement this technology in a very similar way, and the understanding of this will be extensible to the others.

All applications making OAuth requests must be registered on the SP so it authenticates them.

OAuth authorization involves a sequence of interactions between the application requesting the data and the SP. In the next diagram the workflow is illustrated.
The OAuth Authorization Process for Web Applications on Google

1. In a premature stage, the web application contacts the Google Authorization service, and asks for a request token for some determinate Google service.
2. Google verifies that the application making the request is on the list of registered domains allowed to use these services and responds with an unauthorized request token.
3. The web application directs the end user to a Google authorization page, referencing the request token and the service desired to gain access to.
4. On this page, the user is prompted to log into his account and then is either granted or denied limited access to the web application over the user’s Google service data by the web application.
5. The user decides then whether to grant or deny access to the web application by using his authentication (username and password). If the user denies access, he will be directed to a Google denial of access page and not back to the web application.
6. If the authentication is successful, the Authorization service redirects the user to a page designated by the third party application when registering to the SP. This redirect includes the now authorized request token.
7. Then the web application sends a request to Google Authorization service to gain access by exchanging it for a request token.
8. Google’s next step is verifying the request and then returning a valid access token.
9. The web application is now enabled to send a data request to the Google service. This and all requests will now be signed and include the access token.
10. Google service identifies the token and it supplies the requested data.
The OAuth Authorization Process for Web Applications on Yahoo

1. Sign up to get a Consumer Key (API Key)
   - Yahoo! provides a Consumer Key and a Shared Secret

2. Get a Request Token using get_request_token
   - Request parameters: oauth_consumer_key, oauth_nonce, oauth_signature_method, oauth_timestamp, oauth_version, oauth_lang_flag (optional), oauth_callback
   - Response parameters: oauth_token, oauth_token_secret, oauth_expires_in, oauth_request_oauth_url, oauth_callback_confirmed

3. Direct user to Yahoo! for authorization using request_auth
   - Request parameters: oauth_token
   - Yahoo! prompts user to provide authorization
   - User authorizes access to private data

4. Exchange the Request Token and OAuth Verifier for an Access Token using get_token
   - Request parameters: oauth_consumer_key, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_version, oauth_token, oauth_nonce, oauth_verifier
   - Yahoo! grants an Access Token and a Token Secret
     - Response parameters: oauth_token, oauth_token_secret, oauth_session_token, oauth_session_token_secret, oauth_expires_in

5. Use oauth_token in Yahoo! API Requests until Access Token expires
   - Refresh the Access Token using refresh_token
     - Request parameters: oauth_consumer_key, oauth_signature_method, oauth_signature, oauth_timestamp, oauth_version, oauth_token, oauth_nonce
     - Yahoo! grants a new Access Token and a Token Secret
       - Response parameters: oauth_token, oauth_token_secret, oauth_session_token, oauth_session_token_secret, oauth_expires_in, oauth_yahoo_guid

Step 1: Yahoo! API step
Step 2: OAuth flow step
Step 3: Requesting human entry
3.5.2 Token Management

Delegating access to private resources is done by using two sets of credentials: Consumer Key and Consumer Secret which identifies the Consumer, and Token Secret which identify the user. Each Consumer Key-Secret and Token-Secret can be thought of as a username-password pair, the first for the application and the second for the end-user. As the Consumer credentials work similarly to a username and a password, the User is represented by a Token.

OAuth uses two types of tokens: request token request and access token. The firsts are used to verify that the third party application is registered with the SP (Google, Yahoo ...).

Request tokens are used to verify SP registration gain end user authorization, which allows getting an access token. Access tokens are used later to request user data from the service over which the token has authorization.

The request token can be unauthorized and authorized. At the beginning, Google issues an unauthorized request token when asked. When the end user has logged in and has been granted access to the third-party web application, the request tokens is authorized. Authorized tokens are the only ones that can be exchanged for an access token. Request token are valid for one hour only, and can be exchanged only once.

Each access token from this protocol is specific to one user account and covers only the specific service requested for authorization in the original request. They may be contrary to the others long-lived, which means that can be reused at any time in the future by this web application. For instance Google uses long-lived access token, while Yahoo only provides validity for its tokens for one hour.

In case a token is hacked, a user is capable of revoking a Token without having to change passwords and break other applications. The decoupling of the user's username and password from Token is one of the fundamental aspects of OAuth.

Some features must be taken into account on those platforms using Google’s access token expiration policy. As access tokens can last for a long time, it is interesting that third parties interacting with these services may store securely user’s access key to their users’ services. This way whenever the user wants third party applications access to his data in other platforms on behalf of him, he will not have to repeat the whole authentication process again. To achieve this, the application must have a mechanisms to store access tokens for future use.
Google does not allow more than ten valid tokens per user and web application to be outstanding at the same time. This prevents getting a new token per each new access to Google Services.

A RESTful web service is a simple web service implemented using HTTP and the principles of REST. Such

The communication with public services APIs requires some methods to retrieve information, and they must be built safely. These requests are normally done over RESTFUL API based on HTTP methods (GET, POST), and may take a lot of time for developers to build them from zero. Mostly because of the time spent during signing process implementation. Thus requests to Yahoo! API or Google Data API (two examples of SP using OAuth and could be useful to add in e-Catalunya) can be done quicker by using the libraries that both SP offers. These libraries are provided for different programming languages what makes its usage extensible. E.g: Java, PHP, Perl, Python, and C#.

**3.5.3 Signature methods**

OAuth uses digital signatures instead of sending the full credentials. They allow verifying that the content of the request hasn’t changed during the communication and its source is the one is supposed to be.

In order to add encryption the transmitted data SPs provides different algorithms to calculate the signature of the request and including it with the request.

The signature is based on two main parts. The first one consists of using a hash algorithm, which allows data integrity verification. Through this method it obtains a unique identifier value for a data block. Using the same hash algorithm and the same data, the same value will always be found. If data has changed when it arrived to the receiver, it will be able to detect it when calculating the hash and doesn’t match. This algorithm is one way and it doesn’t provide the data reversely from the hash. Data integrity verification is achieved, but the identity of the sender is not verified, so it is needed to add the another mechanism. In order to verify that the request comes from the claimed sender, a shared secret, known only to them, will allow them encrypt and unencrypted the data.
Differently from HTTP authentication mode the secret is never sent with the request. The secret is used to sign the request but it is not part of it, nor can it be extracted.

Most of the SP service using OAuth defines 2 signature methods used to sign and verify requests HMAC and RSA combined with SHA1 hash method.

HMAC-SHA1 is a very common algorithm that uses a symmetric shared secret. On the other side, RSA-SHA1 provides enhanced security by using key-pairs, but it is more complex and requires key generation and a longer learning.

The Shared secret in OAuth, depends on the used signature method. While in the HMAC-SHA1 the shared secret is a combination of the Consumer Secret and Token Secret, in the RSA-SHA1 the Consumer Private Key is used exclusively to sign requests. RSA-SHA1 uses an asymmetric shared secret which means using public and private keys.

Private Key, must be kept under high security measures, if they are compromised, all Tokens are at risk. This is not the case with the other methods where if a Token secret or Consumer Secret is compromised, it does not allow access to other resources than the one for which the service was addressed to.

Although these are two common spread algorithms, it is interesting to see that different SPs can implement, or not, both of them. For instance, Google allows two algorithms to encrypt: RSA-SHA1 and HMAC-SHA1, while Yahoo allows only a consumer secret key encryption (HMAC-SHA1).

### 3.5.4 E-Catalunya OAuth Consumer: Service Integration

These are the steps one should go through to set up an application using authorization service protocol with OAuth on e-Catalunya. The following guideline is the one provided by Google but not many things change with others studied like Yahoo.

The first thing to be done is to register the application with the SP who is providing the data services. In the registration process, some information will be required like the application’s domain, kind of signature method to be used in the encryption (RSA-SHA1 or HMAC-SHA1) and...
any other file required to carry it out (e.g: security certificate). The SP should ask for proof that servers belong to the person who is registering the applications and a chunk of code can be asked for to be added in the header of domain’s main webpage.

In case it was desired to use a long-live token OAuth service, a token storage mechanism should be set up (this is out of the scope from this project). Tokens should be treated as securely as any other sensitive information stored on the server.

Both Google and Yahoo SPs requires specifying the kind of access they will allow. This access is expressed in URL form, which is more restrictive or less according to its path. Here the most restrictive url must be set, in order to make available just the required information. Example: http://www.google.com/calendar/feeds/.

The next step is setting up the mechanism to request and receive OAuth tokens. To do that, it is required that the Consumer gets SP API libraries, and performing calls will be possible. Thanks to them the application can generate well-formed, signed token request calls and handle the responses for the following sequence:

For instance in GData API the calls to manage tokens are the following:

- Get a request token (OAuthGetRequestToken)
- Get the request token authorized (OAuthAuthorizeToken)
- Exchange the authorized request token for an access token (OAuthGetAccessToken)

The developed application must be able to associate a token with its user, as OAuth tokens are specific to it. The way to do this is issuing a cookie to the user before making the token request. When Google redirects the user back to third-party site with an appended token, the application can read the cookie and associate the token with the correct user identification.

### 3.6 OAuth Protocol in Shindig

We have seen how Service provider use OAuth in order to authenticate connections with third-party networks users. But the development of gadgets can also make use of this protocol in order to create min-applications that want to interact with stored data in any Service provider. In order to do so this chapter is intended to explain how the server container must be configured to allow this authentication go across it.
The steps to follow on the application of this protocol over gadgets are the as those studied with normal applications requesting for data to SP (see OAuth protocol section). However, to perform the same operations with gadgets some other problems must be fixed.

As we have seen the OAuth protocol requires that all data requests are digitally signed. This is great for security, but in the case of JavaScript gadgets, managing private keys for creating digital signatures is not secure. Apart from this, browser security models prevent gadget from making cross-domain requests, which force to take into account extra complications.

These issues are solved by taking advantage of a gadget server feature called OAuth Proxy. The OAuth Proxy performs cross-domain information calls on behalf of the end-user or browser. The Proxy is in charge of signing data requests instead of the gadget, doing the heavy lifting for the developer.

The OAuth Proxy is built on top of OAuth and included on Apache Shindig. Using this implementation, any OpenSocial gadget container can add support for OAuth-gadgets for establishing communication with different Service Providers.

In order to manage this proxy and perform this communication, the gadget’s API allows them to remotely fetch content by means of JavaScript calls. This is achieved by using gadget’s API\texttt{makeRequest} function for retrieving and operating on remote web content.

When \texttt{makeRequest} is called, it is signed by the container which also adds additional data to the request before forwarding it to the remote website. This data contains information such as the identifier of the OpenSocial current owner of the application, the ID of the application making the request. Additional information can be also attached so that the server can use it to verify that the information added was not manipulated or to have a better knowledge about the sender application, its viewer and its context.

gadgets.io.makeRequest call. If opensocial_viewer_id parameter was specified in the request it will return information about that user.
In order to fetch SP services information, these data must be returned in a standard format. Thus SPs must set REST, JSON API end-points for requesting data calls. These will be referenced by gadgets inside a OAuth tag section (in ModulePrefs). To do these requests safely, different Service Providers demand different extra checks during the authentication process.

The main idea is that every SP has registered those gadgets or applications that will be able to perform calls to it. Thanks to this registration every time a connection wants to be established the consumer key is checked to validate who is doing the request.

### 3.6.1.1 Enabling OAuth in Shindig

In order to fully enable OAuth support in shindig, some mechanisms to store client secrets and authorization tokens must be implemented. Shindig can offer OAuth authentication for gadget data requests, but a lot of work is needed to achieve adding secrets.

### 3.6.1.1.1 Configuring Shindig for supporting remote fetch calls

In this section I will go through the steps that must be followed to provide a server container with full remote fetching system capabilities: performing remote requests to other Service Providers or even providing those SP services itself.

As mentioned told before consumer secrets are essential for gadgets or applications desired to set up communication with a SP service. That is why they must be provided by to the OAuth Proxy. This is done by listing these secrets in either the file “config/oauth.js” file from Apache Shindig or a production quality store.

In the gadget file descriptor, for those using this protocol, three websites must be specified like follows

```xml
<OAuth>
  <Service name="Service Provider">
    <Request url="http:// SP url providing the request_tokens" />
    <Access url="http:// SP url providing the access token" />
    <Authorization url="http:// SP url providing authorization token"
```
Then in the oauth.js the following entry is needed:

```javascript
{
    "Nom del fitxer descriptor del gadget .xml" : {
        "Service Provider" : {
            "consumer_key" : "gadgetConsumer",
            "consumer_secret" : "gadgetSecret",
            "key_type" : "HMAC_SYMMETRIC"
        }
    }
}
```

Where "consumer_key", "consumer_secret", and "key_type" were negotiated during the application registration process with the Service Provider. The container's role in this is to store this data when the gadget is registered on the server. After adding this information, the gadget should be able to make OAuth data requests to a SP.

### 3.6.1.1.2 Enabling OAuth for production use

When enabling an OAuth protocol for a production platform like e-Catalunya the usage of a robust OAuth Data Store become compulsory and it must store the sensitive data to make communications secure. To do so the implementation of the OAuthStore interface is needed.

Shindig acts as an OAuth consumer, using OAuth tokens to talk to OAuth SP on behalf of the gadgets it is proxying the requests for. An OAuth consumer needs to continuously store gadgets added on that container, and retrieve the appropriate tokens when proxying a request for a gadget. An OAuth Data Store stores three kind of information:

1. **The three endpoint’s url** that define OAuth providers (in OAuthStore.ProviderInfo): request_token, access_token, authorize.
2. **Consumer keys and secrets** that gadgets or maybe containers can have negotiated with OAuth SP’s. It requires that the SP information already has been stored in the OAuth store (defined in OAuthStore.ConsumerKeyAndSecret). The
authentication data will be saved like (key, secret, key_type) indexed by (gadget, server) pairs. Shindig will be only allowed to perform reads to this store and not any modification.

A gadget can be used by many people, so if each user gets access to their SP account, each gadget’s usage should be registered with the information of its user and token to be granted access to the user’s SP private data. For this reason a third point is defined.

3. It will write, read and delete OAuth access tokens indexed by (owner, viewer, gadget, server) and their corresponding token secrets (defined in OAuthStore.TokenInfo). This information will only be readable by the container infrastructure and it can expire according to each Service Provider policies. Notice that only secret tokens are stored and not request ones.

Things get a bit more complicated on social networking environments containers when the person viewing the gadget may not be the same as the data owner who added the gadget. That is the reason why the type of viewer may be very important, so when rendering a gadget performing remote OAuth operations can access different private data depending on the type of viewer. Contemplating these cases requires a specific study of the platform that for the moment e-Catalunya doesn’t have. So we will just focus on the aggregation of OAuth services to the platform without troubling over who is visualizing that application.

Once the implementation of this store is done, it needs to be registered in a new Guice module. Finally to integrate it the project must be rebuilt again. (The implementation of the stores is out from this project’s scope).

Some more modifications will have to be done if additional parameters to OAuth signed requests beyond the standard OpenSocial parameters want to be set.

3.6.1.3 Enabling OAuth on REST API

This issue will allow Shindig not to allow making OAuth calls through it towards third-parties OAuth compliant SP’s, but become one of these SP and provide the necessary logic to provide information services. Shindig is now understood also like a SP.
In order to enable the REST API, OAuth support is required. Here Shindig must just authenticate and interact with the client performing the request over its REST endpoint. The basic idea is:

- The gadget may want to request or send data to the container. This is done by sending a OAuth-signed REST call identifying the gadget with its identifier as well as the viewer identifier.
- Then Shindig validates the OAuth call.
- Shindig checks authorization. This authorization depends on the platform and can be more specific or less. For example the mechanism should at least check that the user has installed the gadget. This could be done by doing a matching of their respective identifiers with those stored on the Store. A part from it more checks could be done.
- At the end the request goes on to the rest of shindig and the data is returned.

Both key and secret that gadgets use must be generated by the container and provided to the author at gadget registration time. The second and third steps require Java implementation. The class implementing OAuthLookupService needs to be written to grant access to OAuth Data Store which keeps the gadget's key and secret, to validate the call, as well as implement any extra logic to process the authorization policy.

Of course this is not enough to achieve REST API working on our Shindig. The gadget registration process must be designed to provide credentials information to the users so they can access to the services provided and store them in the OAuth Data Store.

### 3.7 SocialSite

SocialSite sets a social network environment and its own implementation for a social network infrastructure. A part from customizing the dynamic websites looking and running some scripts for generating and configuring the network database system, developers can easily start providing typical social network services (Registering users, user invitations to groups, sending messages to other people, manage user information profile, tracking friends' activities, gadget integration system ...).
Moreover it comes with a very intuitive graphic interface that allows configuring user profiles and social data as well as managing the platform by means of web interface. With it, social capabilities like gadget integration administration can be achieved easily. Users are provided with a big amount of functionalities to submit their gadgets into platform and manage their profiles by means of the interface, and administrators do the corresponding with their management and other aspects from the platform.

SocialSite adds extra social networking features to the already existing OpenSocial layer provides a server to manage network’s social graph offering cool social functionalities. Global mapping of users is performed according to their relationships (friendship, business, membership), while OpenSocial doesn’t support it. Such social applications offer creating groups that share common interests or affiliations, and track user activities.

![Image of SocialSite interface](https://via.placeholder.com/150)

3.7.1 **Social functionalities**

The SocialSite extends utilities to manage user profiles, customize their properties and user relationships, and limits access to user profiles.

SocialSite extends utilities to offer:

- **Integration gadget management** by using a graphical interface and back-end.

- Social Graph repository in order to get advantage of social information anywhere on the portal.

- **Extended data availability with JavaScript and REST API** for accessing to the social Graph.
Some of the added features to OpenSocial capabilities are: more restricting user control over his content (profile or activities), contact information retrieval and updating from each user, users’ friendships information management, support any group creation, management or invitation capabilities to invite people to groups, messaging capabilities are also improved, and specific search over social data is achieved, between many others.

3.8 Technology Viability Analisys

In this section, I expose my personal reflections about the integration of each one of the transmission information technologies studied in this project. I also provide a comparative between the different technologies in order to establish some conclusions about the study done. These considerations were fundamental for orientating the proposals I did to my supervisors and e-Catalunya project managers.

3.8.1 OpenSocial Layer in e-Catalunya

As I have already said in the introduction, social networks are establishing all kind of channels to import and export whatever kind of information from different remote sources. What at the beginning just covered contact importation/exportation or multimedia data exchange like pictures videos now has become into social user information like their profiles, activities, status, their friends profiles on internet and all kind of information to be aware of other people’s activity. More and more new internet applications and social networks exploit them to provide new kind of content or social applications, and for this reason easy access mechanism have to be set to retrieve this information.

Most of the current social networks are addressing their development at providing services to achieve their social information flow outdoors an indoors. As we have seen gadgets can be one example of exploiting these social information and they can be useful to push user information towards where he desires. Providing like this networks content diffusion.

One of the use cases at the beginning of this master thesis focused on exporting information from e-Catalunya to a personal customizable gadget container. According to this, OpenSocial
API complies with the requirements of e-Catalunya managers to make the platform’s content reachable from everywhere desired.

OpenSocial offer a layer to make e-Catalunya member’s information spread and feed possible applications. However if we think about what kind of information managers desire to promote and export, that information concerns about public groups content: registered new activity information, new topics of discussion, new e-blogs, forums, groups’ activities in the e-Calendar etz. e-Catalunya wants to keep being a serious professional content and its user activity is not so strictly interesting as information concerning tools promoting participation is. So, the platform must focus on this information, and although social knowledge about users can improve a user’s experience the development of the platform must be tied to empowering content diffusion. This fact reduces notably the advantages OpenSocial can offer the platform, as member’ activity information sustains the usage of this technology. Even more, some of the activity information provided is redundant, because the platform already belongs mechanisms to advise activity on the collaboration tools their members. For this reason we see that OpenSocial provide easy mechanism for social information exportation, but also that the useful information exported is already reachable within the platform by other mechanisms which are already internally implemented.

Other kind of information that OpenSocial concerns, like user status or managing friends to obtain their activity, are contemplated as not so useful to the platform purposes (exporting gadgets) and they would move the platform’s synergy to the people and not to the collaboration purposes.

3.8.2 SocialSite in e-Catalunya

As I have already mentioned as OpenSocial inside the platform, the integration of the SocialSite into e-Catalunya addresses enhancements that exceed the platform’s needs. Some of the new functionalities contributed by this software escape from the strictly required collaborative environment in which e-Catalunya needs to work.

The addition of the SocialSite to the existing software implies a complete restructuration of e-Catalunya infrastructure.
3.8.3 Gadgets & OpenSocial vs. Facebook applications in e-Catalunya

After the study of both technologies attending to e-Catalunya requirements, and once the pros and cons were evaluated, gadget technology was chosen to carry out the implementation work of this project. Here I will explain why this decision was made according to the strengths and weaknesses of each of them.

Facebook, and OpenSocial empower the world's developers to make the web more social for users but they achieve similar purposes in different fashions.

Facebook with two hundred fifty million users is the biggest social network existing on internet. Its size and the fact it hosts most of the web applications are developed, cause many developers think about this network in order to spread their applications to multiple users. Developing Facebook target applications grant their application to be present in just one social network, but being in the biggest one.

OpenSocial standard was designed to develop a mechanism to fight against Facebook application hegemony and provide the small networks with tools to offer quality social applications. The main difference between Facebook and Gadgets+OpenSocial is that the latter provides networks with the implementation of standard APIs used to access social data homogeneously from any deployed social gadget. This allows any programmer developing social gadgets to deploy them in whatever OpenSocial compliant network and without modifying any line of code and achieve the applications work with that network’s social data. Facebook applications, on the other hand, are platform dependent and they can only run within the Facebook environment. This is because the API feeding with social data is proprietary, meaning that its APIs are specific to Facebook and languages such as FBML and FQL don’t work anywhere other this platform.

OpenSocial is nowadays supported by many sites which already host social applications developed by any programmer who has learnt OpenSocial API. The learning of this unique API promotes developers programming their applications for these networks forgetting like this to learn an API for each platform they want to build a gadget for. Facebook API proprietary model is only good because the platform is already a reference for all the others and they have the enough popularity to force developers using their APIs to develop applications for its
community. However, this is not entirely true as a new utility OpenSocket has been developed by Facebook in order to integrate social gadgets with its data core.

Developers want to program as fast and the easier as possible. Unlike Facebook, OpenSocial’s gadgets don’t require to be marked up in any proprietary language and developers can build to standard HTML and JavaScript. This supposes a big enhancement.

### 3.8.4 Google rendering server vs own rendering server container

A compelling reason to use Google's server container is that it supports most of the feature libraries required for the majority of gadgets. This means that not many gadgets will present problems at being rendered, because the utility of the gadgets is to be available for everybody and if iGoogle cannot render it, other containers will hardly render it.

In addition to all this we must be able to consider that many of the gadgets developed today are using the tools and specifications offered by Google (to provide gadget visualization improvements or mechanism to interact with other services it belongs). This means that many other platform specific gadget features are added and can be used on this server gadget rendered. Taking this into account, it seems advantageous to take advantage of this service, as at least it will provide the same results as our own implementation of Shindig, without requiring any implementation efforts.

However, the implementation would leave the doors open to a possible OpenSocial integration into the platform, which would bring many features to e-Catalunya.

### 3.8.5 Facebook Connect and Google Friends in e-Catalunya

As we have seen, Facebook and Google are leading the competition at socializing the web. Each of them offers, different solutions providing very interesting capabilities for e-Catalunya. Taking advantage of these applications, they would enable e-Catalunya to open its
participation to the users’ of these social networks. The power of these applications would
improve the diffusion of the platform as well as the topics treated on it, motivating more users
to be aware of its content and enlarging the possibility that more people will be interested in
collaborating.

Moreover, some of the capabilities contributed to the commented applications are included in
the platform’s roadmap to integrate in the future. For example: the web’s chat and the rating
content application. And some others are already included in the platform logic, like sending or
recommending posts to other users.

However, something to take into account is the public to which our platform is addressed.
Spreading out the platform and giving it popularity can be great if people make contributions
are serious and have knowledge the treated topic. But, it also brings to many people the
possibility to post their opinion on a platform whose target is helping professionals to develop
their talent and professional activity. Thus although the idea is great, these tools providing
collaboration from external users should be restricted to some specific groups or portals,
preventing others to be made public. In these other cases people who really were especially
concerned about the idea of getting in touch with other in order to solve doubts and
collaborate at spreading knowledge and helping the others, could always send to the project
managers of the platform a joining request.

These limitations also apply to the impossibility to make public some discussion topics, which
concern about private information on people or subjects. For this reason the addition of these
technologies should be studied according to the topic of each group or portal.

3.9 Preliminary study summary for Generalitat

During the realization of this project, several meetings have been done with e-Catalunya’s
managers. The aim of these meetings was establishing a communication channel by which I
could get direct feedback from my supervisors (by means of internal meetings) and project
managers (doing some meetings with people from Generalitat de Catalunya). Thanks to these
meetings I was presenting the news about my study periodically. This way I could refine the
list of requirements for my project (iterate process), as well as observe which were the most interesting options to cover in my study.

As I commented before, the aim of this study is outlining the future lines for e-Catalunya, enumerating those more interesting functionalities. By doing iterations over the requirements, I better defined them and finally I could suggest some proposals that allowed aligning the future perspectives for e-Catalunya with the main strategy of my project. By establishing these synergies I ensure that my project objectives will match those e-Catalunya managers

I exposed the project managers the technologies and services could be interesting to integrate in the platform and I explain in this document. People from Generalitat told us that they were immersed into a reflection process of the platform. On one side they wanted to take a contemplation time to reflect about the evolution of it along last months. They wanted to wait before giving green light to the ideas I was purposing. That was one of the reasons why, it took so much time to me to be able to meet them to talk about the topic of future implementation in this project. In spite of this, after the meetings conclusions were extracted and main implementation objectives to develop defined. Thanks to them I finally could set my thesis implementation goals and start my practical work.

In the next section, I point out some of the available services, at the moment, which could enable great new capabilities to the platform. I sum up some of the issues we talked about with the platform managers and at the end I will expose the conclusions and objectives we agreed.

3.9.1 Technology analysis conclusions

e-Catalunya project managers main interest was focused in finding the way to export platform’s content all around internet. They were interested in giving the users the possibility to export information from inside the platform to anywhere they wanted. This reduced a lot the scope of technologies that could be used, as proprietary APIs were excluded from that selection.
However, project managers were aware from Facebook popularity and they evaluated positively having done the study about this platform. We were discussing about bringing content to this platform, but finally this option was rejected for limited extensible reasons. They considered more interesting to develop applications that could reach a wider range of public.

They also wanted the platform to reach their users beyond e-Catalunya borders by means of providing some kind of useful personalized information. They didn’t have very clear ideas about which content to export, but I create a document for them with the exhibition of some prototypes, so they could finally take a decision. Those documents were not included finally in annex of this project, because this information is internal to the LCFiB, and doesn’t require to be published. They found one of the proposals very interesting: The chosen one was the exportation of the e-Calendari tool to an exporting application. It would allow any user to consult his public groups’ activities all together from the environment of his choice. They asked for an application that could be parametrized by the user in order to show on it the information from those groups he belongs and wanted. This parameterization should be done within the “Personal Zone” in e-Catalunya. This web interface would provide a set of the groups the user belongs to and the calendars should feed the application. From the same interface the number of results shown by the application, when getting rendered, should be customizable. The final software should provide a description and an easy interface to consult the information.

I proposed some improvements to the e-Calendari exportation application, as other ideas came to my mind in this direction which I let them know. I also proposed making the application even more powerful by adding external public source activities information. This meant the application would combine information from the public e-Calendar tools with activity information coming from other public calendars like Google Calendar or any other platform using a standard transfer activity file format. This merge of activity information could be exported to an ical format file instead of sending the output to the exporting application format.

The platform’s managers liked my proposal and after speaking about graphical visualization of the user interface they decided to accept it.

Another point they were interested in, was adding functionalities into the collaborative environment. Content on the platform is usually technical and can require from some external help to make understandable for viewers. The managers said that supporting tools
to make concepts clear, translating words or text in case the post were done in another language from the author, or simply adding media to make the content more appealable would be a great enhancement. A priorly they didn’t put any content restriction for these applications.

The integration into the platform of functional applications could also be provided by their introduction into collaborative tools. The project manager said that giving users the possibility to integrate application or customizable content by means of any of the studied applications would be something that would really change the perception they have from the platform. At the same time that would make it more appealable for them.

I discussed about the extra functionalities OpenSocial would bring to the platform with them. As we have already described its features I am not going to explain them again. As a general conclusion I determined that this layer and SocialSite technologies over passes the needs e-Catalunya have today. It is true that everything seems to point to that direction on how social networks tend to work in the future, but by now the implementation of this layer it is not a priority.

Google Friends and Facebook Connect technologies were received enthusiastically by the people from the department. They thought it was a good idea to promote participation from non platform members on public content. However they don’t give it so much importance, as e-Catalunya already provides anonymous user comment post support.

Talking about e-Catalunya gadgets’ implementation, we determined with them that the e-Calendari exporting application would suppose a pilot test to know if this technology was bringing the so expected results. For this reason issues about implementing new gadgets for exporting e-Catalunya abroad the platform were leaved aside.

Contact importation services have been explained on the former study, and they were interesting for project managers in order to make the addition of new contacts and their information easier. They believed it to be an implementation enhancement to have into account in the future. The main service we spoke about were those offered by bigger social networks like Google, MSN or Yahoo.
3.9.2 Reviewed Objectives & Conclusions

In order to meet platform requirements and attend the project manager considerations I extracted some conclusions. According to them and to my study, here I will explain the derived objectives from my work, and I will make some decisions about which studied technologies will be chosen and which will not in order to achieve my goals.

At the end of the meeting, project managers agreed that one of the priority issues they have on their minds was that information reached e-Catalunya users outside from its borders. They focused on the e-Calendari tool to perform this content exportation. Basically they required an application that would show the activity flow resulting from merging different activity sources. On one hand the activities provided from e-Calendari tools inside the platform, and on the other hand, those activity publically available on an ical activities file format.

To achieve this purpose, I chose the gadget technology for developing an application that could be deployed in another internet location different from e-Catalunya and could perform the work. Choosing technology gadget, audience was guaranteed and gadget integration into any social network or website where the gadget wished to be deployed too.

The ease development of gadgets and the fact that proprietary applications’ APIs, like Facebook, limit the audience of the applications to their own community, made me reject proprietary implementation technologies. Google Gadgets is a more social network extensible technology and big companies like Facebook already support solutions to integrate these applications on their environment (like Facebook OpenSocket to integrate social gadgets).

As I explained before, one of Generalitat’s motivations was exporting e-Catalunya content all around internet. To achieve this purpose they were interested in adding a layer to the platform would distribute collaboration information from the platform. The goal of setting this infrastructure was to set the basis for sharing collaboration knowledge with other social networks. The idea was to empower e-Catalunya tools in order to reach other collaboration environments and achieve interaction with them.

To do so, I have studied the possibility to add OpenSocial layer providing social information to those applications asking for it. Digging inside I saw that it offers several mechanisms for spreading networks’ social information where required. However, the exported informationis
more concerned on more users’ **social information** rather than strictly on the platform’s collaboration content, which is the main content e-Catalunya wants to forward. This content refers to user participation e.g.: posts in the e-blog, e-forum, e-wiki etc. **Social capabilities provided by OpenSocial** are not those on the top of e-Catalunya managers’ **priority** list: most of them, if integrated, would be useless. On the other side, the interesting **information** that it does make **available** can also be retrieved by means of the internal mechanisms of the platform which help reaching the same application feeding purposes. Another point that **doesn’t match** with project managers’ s objectives is the fact they were just **interested in** offering **information feeding services** to developed services by e-Catalunya developers (e.g: gadgets, internet services) and the **proposed OpenSocial model offer this information to everybody**. For all these reasons the integration of OpenSocial was left on the background and the adaption of this layer was rejected.

For the same reasons **SocialSite technology is directly rejected** too. The integration of this software would imply a completely redesign and re-implementation of the platform, project managers decided to have into account this solution for a possible new 2.0 version of e-Catalunya.

e-Catalunya project managers wanted to **export platform’s applications** to other web environments and the usage of **gadget technology** is our choice to proceed. As a matter of fact aiming to perform **gadget rendering a server container is required**. My **decision to deploy gadgets inside the platform and outside was to user the server container offered by Google**. I took this option because of the numerous advantages it offered in front of integrating our own gadget container inside e-Catalunya infrastructure.

**As I rejected integrating OpenSocial layer, Apache Shindig on e-Catalunya was unnecessary.** In addition to this, Google’s server container implements some extra functionality that gadgets being rendered with it will be able to take advantage of. **Otherwise, in the implementation of an own container I should implement those functionalities or deny support to those gadgets supporting them**. For these reasons and due to the high **Shindig integration time-cost**, the chosen option to render gadgets inside the platform was the Google rendering service.

e-Catalunya project managers **concluded** that the functionalities provided by Facebook **Connect and Google Friends** technologies were interesting. The **inclusion** of them would allow implicit **spreading the platform**, for example by means of having all user activity in e-Catalunya be registered in Facebook user’s wall. These two kinds of applications would bring new social capabilities to the platform with not such a big effort required. Their **integration** would also let
the number of identified users in e-Catalunya increase in a very notorious way, which is one of the platform's managers objectives. This growing would motivate more users are aware of the platform's contents and also enlarge the possibility of more people being interested and collaborate. By granting Facebook and Google members participation capabilities offered to unknown users in the platform are improved. Up to now, the only way to interact with the platform for non-members is to access public tools and post a comment without being identified (as an anonymous user). These capabilities would allow non-members to interact and at the same time be identified by an e-mail account or name. However the usage of these services is tied to those members who have a Facebook or Google accounts. This restricts access to non members from these networks. Opening the community is a priority very good, but it must be somehow controlled because it can bring attached a content confidentiality problem or establish an entry point for undesirable user contributions.

For this reason and because it is desirable having having people who are interested in e-Catalunya be register as members, I conclude to skip those applications bringing social functionalities by now.

Two objectives were established in agreement with Generalitat de Catalunya. The first objective was to build an application to allow users to combine their chosen public groups’ e-Calendar activities with other coming from any public ical format activity file. The mixture of activities would export a combination of calendars’ activities wherever around interned by means of a gadget technology. Nevertheless this information is also desired to be exported to desktop applications. Thus managers proposed that the service providing this information would return the requested merged activity information in a human-readable format for gadgets or it would convert it into a downloadable ical activity file otherwise. By means of these services exportation in e-Catalunya will be achieved.

This gadget should be multiplatform, this means it should be deployable in most of the browsers, with an attractive easy interface which allowing visualizing events as well as their description. As a merge of activity sources will be done, a mechanism to differentiate between this information channels was also essential.

Finally, the other objective I agreed to achieve with platform managers was the integration of external gadgets into e-Catalunya (content importation). In this case The target was to add the gadgets inside the platform wherever they could be more interesting and worth. Thinking about a possible point, two options came to me. A place where they could be integrated should be somewhere where participation takes place and external help could be useful to
understand them. In short gadgets could be added into e-Catalunya tools. Apart from giving support to comments and user collaboration, another place would be user’s “Personal Zone”. The addition of gadgets inside the user “Personal Zone” would provide user with great customization capabilities. Useful functionalities could be added to manage users’ knowledge necessities within their e-Catalunya environment. This point is also included as an objective of this project.

In order to achieve this integration, a gadget system manager should be developed, so that users could add and remove the gadgets of their choice inside their collaborative tools or personal environment. The system should provide a directory of gadgets where users would choose which ones to add in the tools of e-Catalunya. This way, portal administrators would add those gadgets they believe most interesting for the portal’s purposes and at the same time they could ensure that gadget usage is strictly professional and escapes any idle purposes. In addition to this, the system should also provide a gadget submitting process for those gadgets not added in the e-Catalunya directory of gadgets. By means of this, users could ask for gadget inclusion to portal administrators.

Defined Objectives:

- **Exportation services:**
  - e-Calendari gadget (shows multiple calendar source activity combination)
  - Ical activity format file provider (exportation activity file service)

- **Integration of gadgets**
  - Gadget Manager (Gadget directory and administrator’s management)
  - Gadget integration
    - Personal Zone (gadget container)
    - e-tools gadget integration (gadgets supporting e-tools)
4 FURTHER STUDIES

The theoretical part of my project doesn’t finish with content importation and exportation, as I was required to do some more studies about other functionalities. These studies consisted in taking a look to the services that are currently maintained on e-Catalunya by LCFIB. I was asked to provide a detailed feature list of outsourcing Service Providers that offered similar capabilities that the platform is doing, in order to save up time and problems by externalizing some services e.g.: file repository (storage service), survey tool, Photo Albums …

Special attention was also deserved to single sign on authentication services. The integration of a federated login, like OpenID system (check the appendix for more information), would provide everybody with a third party network account in the federation where e-Catalunya is, to enter into the platform with his/her third party credentials. A lot of more people could access to e-Catalunya and use its services thanks to it.

4.1 Service Externalization

e-Catalunya is today a growing popularity social portal which is offering a big amount of services to its increasing number of users. In order to offer relievable functionalities and manage all of its services efficiently, the platform requires having a good storage system as well as tools that help to develop its purposes.

The main goal of this study is looking for some reliable services which could download the platform from possible failure points at the same time that maintaining our social network functionalities. This way we could conserve energy directed at the competencies of the platform like promoting the social network and improving members’ tools by improving teamwork efficiency. Thus I started collecting information about different online services that are in the present deployed in e-Catalunya, and could be moved to external third-party services providers.
The final goal is clear, why does the platform has to do some job that other external services can also do even better without worrying about their maintenance? By joining some of these services, fewer resources will need to be managed and modularity will be achieved. Every service will be on charge of tolerating or repairing service’s failures itself. Of course this decision is also made in the interest of reducing costs.

This study covers the features from different outsourced storage system. Any comparison can’t be provided with the current system because the contract Generalitat has with T-Systems is a contract including a software development package and I couldn’t get the specifications of each service provided by e-Catalunya. Another motivation for externalizing data storage is to getting this data processed outside from the platform as well as getting extra presentation capabilities in order to show information in a more intuitive way to the users. For instance, the current tool e-Poll is currently maintained by e-Catalunya and it could be done by a third-party offering better benefits like extracting and inferring more information from each surveys’ responses.

In the next pages the reader will find the description from the studied external services with its specific technical information accompanied with the corresponding economic costs of some of them.

What services we want to externalize? PhotoAlbum, File repository, profiles (using OpenSocial), polls ...

### 4.1.1 PhotoAlbum

Pictures are a big part of social data shared in social networks nowadays. A social space is reserved in almost any network for storing and presenting users’ images. E-Catalunya is not an exception and pictures are also shareable between members.

Many companies have appeared to provide these services out from social network environments. Google and Flickr have been successful on it and they provide their users with pictures storage and an attractive web interface for their visualization. Parallel to these services SPs have created new APIs to allow web application to access any user’s personal pictures. Updating pictures, modifying metadata and deleting them are some of the functionalities that they are available.
In my study I took a look to these services because more and more people joining e-Catalunya platform are already users of Picassa or Flickr services. If e-Catalunya could take advantage of this, the platform would be able to retrieve its users’ pictures from these SPs and finally display them inside the portal. But similar capabilities must be provided to all users and those people who don’t own a Google or Flickr account cannot be excluded from e-PhotoAlbum tool. For this reason if administrators decided to integrate Flickr or Google services, the current e-PhotoAlbum (as it is known today) should still be provided to those users who don’t own an account with any of these service providers. At this time, other could use their account with Google and Flickr to storage their pictures. When a user image content was required to be shown e-Catalunya would consult its own database or otherwise remotely retrieve the images.

The recommendation system would work the same way than until now and recommendation to SP images would be done through the public links to those pictures. Obviously all the pictures showable in the platform should be public on Google or Flickr.

4.1.1.1  Google & PICASSA

Picassa is a web photo-album storage service that allows their users to upload multimedia content (pictures, videos) and provides an easy web interface to show them. This service belongs to Google and it comes with a public API by which users can retrieve their content through any application using Google Data libraries.

Google allows using this service to any organization that doesn’t get profit directly from the use of Picasa, claim authorship of the software or any of its components. As e-Catalunya satisfies these premises we could add this service whenever we want.

The Picasa API would allow e-Catalunya to store all its members’ pictures and videos as well as those from its groups inside. The outsourcing of this service would download the platform from this data maintenance. At the same time Picassa offers public RSS about this content for use by feed readers. Users could store and see their pictures through e-Catalunya and also could import them thanks to RSS feed channels.

To integrate Picassa Album with the e-Imatges tool it is necessary to save the multimedia information on the Picassa Service Provider (SP), instead of storing the data in a database service like it is done until now. The platform must perform this persistence by doing calls to
the API. Every operation would be called with a concrete visibility value according to permissions the user is performing the request has.

The methodology followed to integrate the service would be to create a directory for each user of the platform and groups hosting e-Imatges tool. These directories would be managed remotely by e-Catalunya. It also would control user’s quota as well as set its data visibility to their desired value. Technically speaking the usage of this service in e-Catalunya would suppose the substitution of the e-Imatges storage interface’s implementation. The new implementation would use Google Data API java library to perform remote storage and content retrieval operations. Obviously all these communications would be done under security transmission protocols to ensure the data confidentiality and integrity (OAuth protocol).

Picassa offers two configurable features by which the account project managers to specify information’s retrieval type and visibility according to his purposes.

**Visibility** allows requesting data proceeding from different sharing levels.

- When a request has its visibility value set to “public” (does not require authentication) it will ask for media content defined like public.

- If this value is set to “private” (client authentication required) the Service provider will only send back private information.

- If the value is set to “both” then public and private data will be returned by the server. Visibility requests are closely linked with user’s authentication. For this reason e-Catalunya will act like referee giving access to directories to those users owning any of the multimedia content belonging to them.

Projection value indicates what elements and extensions should appear in the feed being requested. “base” projection value would set the representation of the answer as basic Atom feed, suitable for display in an Atom reader. On the other hand “api” value allows defining values to get specific media extension files.

Calls to this API also allow retrieving specific content according to file metadata information like uploaded time or to the album it belongs.
Purchase additional storage

Picassa comes with one gigabyte free storage account. However Google provides extra storage packages to enlarge this quota by paying some money yearly.

The storage plans are as follows:

- 10GB $20/yr
- 40GB $75/yr
- 150GB $250/yr
- 400GB $500/yr

While carrying out this study, some abstracts and posts were read talking about a supposed new massive storage service will be provided by Google. GDrive, how the rumorology call it, will provide anyone a universally accessible network share that spans across computers. This product could extend the storage service to all kind of files and not only to text files’ format or image and video files. By now nothing about the release of this product has been revealed, however it could be a good opportunity to store the e-Catalunya’s files in a near future.

This service provides a web interface by means of which users can also access to their galleries with the browsers and dispose from interesting capabilities for sharing, showing and other capabilities with images. The explanation of how using this service is included in one of the project points.

4.1.1.2 Flickr

This other photo manager provides a REST, XML-RPC and SOAP APIs providing picture retrieval and updating content services for web-based and non-web-based applications. Although not provided by the SP, there are many existing libraries supporting access to the APIs from multiple programming languages.

These libraries can be used together with an authentication protocol which ensures security on the whole process. The protocol used in this case is Auth API, which is one of the protocols in which OAuth set its basis. Obviously to start using this API the user must be registered and use the provided API key provided to authenticate each connection.
In any interaction, an application should only request access to the strict path content it needs to access, by reading, writing or deleting personal pictures.

### 4.1.1.3 Conclusions PhotoAlbum Services

The integration of these web PhotoAlbums services and many other popular ones into e-Catalunya would suppose a big download for the platform at storing users’ data.

### 4.1.2 Storage Services

The storage service is vital for the platform purposes. Having a relievable service ensures 24 hour service providing. As it was previously introduced, e-Catalunya is growing considerably and every user has a storage quota to maintain his files on the system. One of the long future challenges for e-Catalunya is opening its doors to the citizenship and enlarges its community with people having special collaborative needs. Looking forward to achieving this purpose huge scalable storage policy must be taken into account. For this reason providing e-Catalunya with a storage service easy to manage by platform managers highly scalable and with a good ratio quality/price is something desirable.

Now I will present some current public services solutions:

#### 4.1.2.1 Box.net

Box is a web-based portal providing storage services for users and companies. It allows uploading and requesting content easily through its useful public interface.

Box.net provides an API, which can be used to create applications and Web sites that integrates with Box.net. The Box.net API is a collection of Web services, which means that applications using Box.net can be created in any programming language.

Box applications perform the following functions:

- Store and retrieve files from Box.net
- Organize files into folders
- Move, rename and delete files
- Share files

Although this storage service provides a complete API for content retrieval this alternative quickly stops to be good when having a look on the maximum capacity the solution offers. Due to its high prices it is difficult to integrate this solution.

**Pricing packages**

*Lite (Free)*

**Always Free 5 collaboration folders**
- 1 GB of storage
- 25 MB file uploads

*Individual*

€7.95 / month 5 collaboration folders
- 5 GB of storage
- 1GB file uploads
- 24/7 email support

*Business*

$15 / user / month Unlimited collaboration folders
- 10GB of storage
- 1GB file uploads
- Unlimited bandwidth

- Unlimited collaboration folders
- Customizable storage plans
- Up to 1 GB per file
4.1.2.2 **Amazon S3**

Amazon S3 is a quite new storage Service offered by Amazon. Amazon has basically taken the online storage infrastructure behind their core online services and make it public to provide storage service to whoever who required.

This highly scalable, reliable, fast, inexpensive service maximizes benefits of scale and passes those benefits on to developers. It’s not an online storage service in the same fashion as box.net. It is a service for programmers and not users.

It costs around $0.15 per gigabyte of storage per month and $0.20 per gigabyte of data transferred. This model is fair because you only pay for what you use, no more, no less, with no setup fees or monthly minimums.

**Here are the Amazon S3 service functionalities.**

Amazon S3 is intentionally built with a minimal feature set.

- It allows write, read, and delete an unlimited number of objects under 5 gigabytes payload each one.
- Each object is stored in a container and retrieved via a unique, developer-assigned key.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
- Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.
- Built to be flexible so that protocol or functional layers can easily be added. Default download protocol is HTTP.

**Pricing**

**Storage**

- $0.180 per GB – first 50 TB / month of storage used
- $0.170 per GB – next 50 TB / month of storage used
- $0.160 per GB – next 400 TB / month of storage used
- $0.150 per GB – storage used / month over 500 TB

**Data Transfer**
• $0.100 per GB – all data transfer in
• $0.170 per GB – first 10 TB / month data transfer out
• $0.130 per GB – next 40 TB / month data transfer out
• $0.110 per GB – next 100 TB / month data transfer out
• $0.100 per GB – data transfer out / month over 150 TB

Requests

• $0.012 per 1,000 PUT, COPY, POST, or LIST requests
• $0.012 per 10,000 GET and all other requests

Connections to Amazon S3 can be authenticated or not. If the connection is authenticated, the user must sign in using and access key id and a secret access key provided when signing up. While performing a request with Amazon, the request is assembled and security measures adopted and finally the request is forwarded to S3 service. Amazon verifies the Signature of each request is a valid hash of the request and, if authenticated, processes the request.

The service allows any user that is granted access to construct an HTTP URL that can be used to access that object or container through the query string authentication mechanism. This HTTP URL can be distributed to any user with a web client or embedded in a web page. To allow selected users to access data containers in S3 account, access control lists or query string authentication can be used.

4.1.2.3 Conclusions Storage Services

Amazon S3 service offers a very reliable service with big scalability possibilities and good ratio quality/price. From developer point of view, I believe it is the best option from those studied and it allows a big level of storage service customization.

4.1.3 External surveys services

This section goes through different existent solutions to externalize the tool e-Catalunya provide for building social surveys.
The “Processos Participatius” tool full-fills most of the functional requirements by which it was designed. It allows generating, deploying, modifying and managing surveys as well as presenting the results from users’ participation. In e-Catalunya every user can generate his surveys, although the process is carry out through a poor intuitive graphical interface. Once the surveys are published the tool doesn’t provide any attractive presentation web interface for showing the results, what makes the understanding of data really hard. Because of these reasons, the current tool it is not included into the platform production environment and just in the development one. The target of this study is to find an external solution improving this tool.

Platform administrators conclude through some stakeholders that the interface for survey creation is poor intuitive.

Poor result presentation without graphics

For this reason, it is desired to find a third-party service that enables users to create, deploy and store their own surveys creations in a better experience way. A great solution should
provide a graphical representation for the answered data that allowed extracting conclusions and make information more digestible.

Most of the online studied survey services are conceived to give support to high business enterprise management software components and feeding their making decision tools with users’ surveys feedback. All the solutions found are enterprise-based, and most of them intend to be a helping tool for the CRM (Customer Relationship Management). By now the current tool doesn’t infer any kind of information from the responses. However as most of the solutions studied provide some processing capabilities over its survey responses, I propose to add some

logic to e-Catalunya in order to match users with their level of affinity according to the responses they gave to surveys before (it is just a possible example).

Most common service’s found purposes are:

- **Automate Surveys**: Gather data following every customer interaction.
- **Satisfaction Reports**: Identify trends and establish a customer satisfaction index.
- **CRM Integration**: Integration with CRM to push the business ahead.

The following are the solutions that could match better our purposes.

**4.1.3.1 Hosted Survey™ Enterprise API**

It is software for building electronic surveys and questionnaires through a graphical interface. Hosted Survey™ is a fully web-hosted survey software developed for researchers and organizational improvement specialists. It’s used to create electronic surveys, feedback reviews, questionnaires, data analysis and reporting.
4.1.3.1.1 Survey Design and Editing Options

This website offers the possibility to edit new surveys or use existing ones, importing and their combination. All this can be achieved through an easy graphical interface. Optionally, they can embed HTML formatting and JavaScript commands into questions, answers, message text and emails getting personalized look and feel.

Hosted Survey has also an email invitation system which allows the administrators to invite participants using personalized messages. The system can also monitor who has responded and who has not.

In our case e-Catalunya users don’t push, by now, survey answering request. Nevertheless it’s a point we could have under consideration to be added in the system in order to allow the users sending surveys to specific contacts.

4.1.3.1.2 Enterprise API

Hosted Survey™ provides an API solution for giving support to any information request. It offers:

- Dynamic data collection, distribution of survey results.
- Automatic information response functionality which provide an AI system extracting and inferring information from surveys.
- Advanced online surveying AI functionality for dynamic survey adjustment in response of participant's responses and feedback).
- The service is scalable, secure and adaptable to client solutions.
- It just gives response to results retrieval data queries.
- It allows watching the survey results online using easy to read reports and ad-hoc queries.
The customer of the service is charged economically since responses exceed 50. Account allows creating unlimited surveys with unlimited questions and answers, but the user is charged by number of answered surveys.

This solution offers great capabilities for survey data management and response storing, however it provides many mechanisms that exceeds the needs from e-Catalunya.

4.1.3.2 **SURVEYGIZMO**

SurveyGizmo is an online survey software. It can create and launch great-looking online surveys along with website quizzes and polls, contest forms, registration pages, contact forms, and multi-page questionnaires with our survey software.

The public services are offered through an API that allows to:

1. Show to survey administrators a list of running surveys, the current responses, view reporting, edit the survey or promote it.
2. Embed survey into articles, blog posts or web pages through JavaScript publishing option.
3. Register an email campaign tracking who has taken the survey, abandoned it, or never clicked.
4. Search through response data and extract answers to survey questions and other tracking information like, ip, language and geo-location.

5. This API just allows data retrieval.

The API functions are called over the web by submitting secure http requests to the API. Like the others the survey building service is just graphical available and nothing can be done through the API. This is a black point that forces in case of its integration with e-Catalunya that users must go to the surveygizmo to build their surveys and after they just can visualize their results by using the API form inside the platform. This mechanism would imply that users answering should do it from outside the platform by clicking on a direct link that would send them to the answering survey page or being just redirected.

### 4.1.3.3 KeySurvey

Key Survey provides also easy powerful survey functionalities. Thanks to it, users are enabled to create, manage, deploy, dynamically manipulate surveys and present their responses and analysis.

Here we have another survey service oriented to give valuable feedback to the survey creators in order to establish mechanisms that reacts to survey response stimuli, for helping the company to take advantage from their users’ evaluation and improve the service offered.

Key Survey's API enables users to:

- Automatically send surveys after a pre-defined event has been triggered, in a company environment maybe a sale or customer service contact.
- Create relevant survey questions based on the individual customer's profile stored in company's database.
- Create custom reports for decision makers.
- Integrate historical customer survey responses into the company’s CRM.
Key Survey API supports a big number of languages: Java, .NET, Perl, PHP, Python, OCAML, Ruby, and XML...

4.1.3.4 Conclusions External Surveys

As I pointed in the beginning, all of the presented services offer similar capabilities that exceed the features and process capabilities we require for our surveys data in the platform. Although this wouldn’t be a problem if some other aspects didn’t miss to them.

The main problem for integrating any of these services to the e-Catalunya platform is the following. All the systems provide an API which helps retrieving response data statistics, but none of them allow uploading surveys. We would like to manage the creation of surveys from inside our platform and after send the templates to the service so it would deploy them. However there are no services offering this. For this reason, a more complex integration implementation should be thought as another solution. This would require some collaboration at developing a personalized service between e-Catalunya and the chosen survey SP to create the interacting mechanisms required for achieve “Processos participatius” goals.

A solution could be that e-Catalunya took the control of a single authentication process with the survey offering service and then let users access to an e-Catalunya restricted area where the user could manage its own surveys. This solution would imply a great development cost and the solution would be forever dependent on the third-party survey provider.

Seen the different possibilities and the difficulty to adapt them, I concluded that although maybe not very intuitive and graphically poor, the current tool to perform surveys inside the platform it is good enough. The studied options normally points commercial purposes and offer some extra functionality over the data our platform doesn’t require.

Due to these difficulties to integrate a service that allowed survey interface designing in e-Catalunya, I believe instead of integrating an external service, it would be better improve the one the platform belongs.
After the decisions were made in the meeting it was the turn to start developing what we had agreed. First of all I developed the services that would allow performing e-Calendar activity exportation. Later on I developed the integration system users could use to add gadgets inside the platform.

5.1 *e-Catalunya Domain Layer*

In order to achieve my implementation purposes I required some platform internal mechanisms offering to me access to some network’s internal data. e-Catalunya already provided some services covering this necessity by means of domain layer, which I used. This layer is mainly composed by two components:

- Domain model classes correspond to the classes from the e-Catalunya conceptual model. They represent the state of the system and maintain its properties.

- Service classes encapsulate access and control logic on the domain classes, thus acting as an interface for both the presentation layer and the Data layer.
5.1.1 Catalunya Services

The platform does not provide a unique entry point for the whole domain access (façade pattern), it rather offers a set of services having each one of them encapsulate an delimited area of responsibility. Distribution of responsibilities between services is conceptually driven: each service covers functionality for a determinate platform functional family.

The platform provided the following services:

UserService

Responsible for all user related functionality at a platform scope, including all methods responsible for:

- Creating/removing user accounts at the platform.
- Logging the latter according to law compliance issues.
• User account maintenance, including personal data edition, changing a user’s password...
• Finding user accounts according to different query criteria: by email, by name, filtering blocked/removed accounts, etc.
• Sending emails to users (via the platform’s MailService).

**PortalService**

It covers the portal framework functionality. This includes:

• Creating portals at the platform
• Adding/removing users to/from portals, maintaining member’s roles.
• Maintaining a user portal profile data (portal profile creation, edition and removal)
• Maintaining portal member’s tools (addition, edition, usage, and removal)
• Finding all the above mentioned data given different search criteria (find portal moderators, find a user’s role at a portal, find a given user’s tool set at a given portal, etc...)
• Maintaining the portal’s available tool set
• Maintain email templates at a given portal: (creation, edition, and removal).

**GroupService**

It is on charge for all group related tasks, including:

• Creating new groups at a given portal
• Adding/removing users to/from groups, maintaining member’s roles.
• Supporting group invitation creation, acceptance and refusal procedures.
• Supporting group subscription requesting and administrating procedures
• Maintaining group tools (addition, edition, usage, and removal)
• Maintaining group – taxonomy associations.
• Finding all the above mentioned data given different search criteria (find group moderators, find a user’s role at a group, find a given group’s tool set, find groups associated to a given taxonomy)
• Maintaining Group templates at a portal (creation, edition, removal)

**ToolService**

It will be responsible for tool integration, thus covering any functionality offered by the platform’s web tools. This service will be mainly used by the Portal and Group services in order to obtain tool support:

• Create/maintain/remove tool instances (both for groups and users)
• Finding tool instances according to different search criteria: a given group’s set of tools, a given user’s set of tools at a given portal, etc.

**HibernateService**

This service will cover data persistence related functionality, including:

• Storing, loading and updating model data from/to the platform’s database.

In the developed prototype I used most of them for retrieving the required information. (e.g: ToolService: In the case of e-Calendari I needed to retrieve user Calendar activity data, HibernateService: To store user registered gadgets in his “Personal Zone” ...).

### 5.2 Gadget Implementation

As I already told in the meeting conclusions section, Generalitat agreed offering e-Calendar exportation by means of gadget to e-Catalunya members. The service was intended to give the chance to every member of the platform to personalize his/her own e-Calendar gadget. The main idea was that they could feed the gadget with information about the public e-Calendars which they belong to inside the platform. But this wasn’t enough and adding activity information coming from other sources in internet was also desired. For this reason some
extra mechanisms had to be added to the logic of the application in order to retrieve this information whatever the remote source was.

The motivation that brings the implementation of this gadget was the incapability of the platform to centralize and display sorted user’s agenda activities. Many users belong nowadays to different networks offering sections to manage their local agendas. Sometimes these activities are specific from the portal and sometimes they are not. Helping the users to collect all these public agendas’ activities, merging them with those from e-Catalunya and finally sort them by date provide a great enhancement for users to manage their lives.

Apart from the gadget development, the project managers make noticeable the need that the platform could provide the same information was given by gadgets, by means of a standard activity file format. The chosen format was ical, as it was large used on the internet community for activity exportation purposes.

Done this distinction between gadgets an ical output, I will proceed to explain the use cases to carry out these functionalities.

The four main implementation tasks have been done to achieve the e-Calendari gadget purposes:

- Configuration service (web interface)
- Gadget’s file descriptor service. (Gadget providing gadget source code).
- Gadget design
- Ical activity service.
5.2.1 Configuring the services

Both gadget and ical format file services were planned to be configured from the same web interface. The reason is simple, the output from both information services will be in different format, but the information they content will be obtained the same way.

The following use case flow describes how this configuration will be done.

- The user enters into the gadget configuration tool.
- The user enters the kind of e-Calendar exportation he wants to do (gadget or ical).
- The user does a selection of public e-Calendars he has access to inside e-Catalunya.
- The user inserts his public calendars’ urls to be merged with those from e-Catalunya.
- The user inserts the starting date since the data will be returned. It defines the activity time window from the activities will be shown.
- The user selects how many activities he wants to be shown on the output since the starting point.
- The user pushes next button.

Once the user has introduced all the parameters and the next button pressed, the system will return an url which will either point to a gadget file descriptor or ical format file depending on what the user chose. The parameters arranged before will be hard coded into the url so that the server can configure the output. In the first case Tomcat will hard code those parameters on the e-Calendar gadget file descriptor template. On the second, the server will get all the parameters and will execute the logic to merge all activities from its sources and extract the appropriated information according to the time window specified by the user (starting date...
and number of elements to be shown). At the end it will returns the information in ical format as follows.

**Output iCal format file provided by the platform when ical format is chosen.**

```plaintext
```

Now I will explain the functionality of the two different outputs and the things can be achieved with them.

### 5.2.2 Gadget descriptor file service

The url the members get when asking for an ical output format is already the good to obtain the file desired. However when the user is asking for a personal gadget url some more things must be done. At that time, the web interface just gives the url of the servlet that send back a gadget file descriptor. As we have seen in the theory, gadgets need a server container in order to be rendered and this is the step we haven’t yet talked about. To integrate the gadget into any gadget container the url provided will be needed. So if for example we go to our personal iGoogle page and we want to add this new gadget, we will have to give the url to the adding gadget mechanism.
A while after, the Google Server will perform its checks to ensure the gadget file descriptor is right and error-free. As the gadget doesn’t belong to the general Gadget directory of Google a security message and advice will prompted to the user. By accepting it the gadget will be added to the iGoogle environment of the user.

After, we will be able to see our new gadget creation on the gadget container main page. The applications is rendered according to the gadget file descriptor, its css information and the JavaScript perform the appropriated calls to the e-Catalunya server in order to retrieve activities’ data.

I inserted my gadget on Hi5 social network to check it also worked well in other social networks accepting gadgets.
Jahrms:: Developer

My Applications

Sample App

Summary:

Description:

Translations:

Please set your default locale.

New Application

Enter the full URL to your OpenSocial application, then click submit. For example, http://mydomain.com/application.xml

Upload XML files

Select general application category

Select a general application category:

- Standard OpenSocial Applications
- Mobile Applications
- Desktop Applications
- External Websites

Submit

Sample App

Summary:

Description:

Translations:

Please set your default locale.

Sample App

Summary:

Description:

Translations:

Please set your default locale.

My Applications

e-Calendar

General | View Source | Language

App ID: 12219_60f089242777401

Category: Standard OpenSocial Applications

App URL: http://epages.fib.upc.edu/80/18/historietagadget.xml

Developer:

Add

Receive e-mail:

Yes | Edit

Di, 16 de Febrero

00:00: Actitud estudiantil

Di, 17 de Febrero

14:10: Actitud estudiantil

Di, 24 de Febrero

00:00: Cepatodos 2009

Mi, 2 de Març

00:00: Monocrom Copy

Di, 5 de Març

14:15: "La verdad se ve con pensamiento y nunca sentimental"

- Aquí es una descripción

Di, 7 de Març

07:05: Actitud estudiantil

Di, 10 de Març

00:00: Actitud estudiantil

Mi, 1 de Abril

00:00: Actitud estudiantil

Di, 16 de Abril

00:00: El diario no habla, (un)

Di, 23 de Abril

H5 Games
5.2.2.1 **Gadget design**

Special attention had to be put in the gadget interface design, to achieve some non functional requirements. Those requirements concern about data presentation and comprehension. The gadget displays so much information as the number of activities the user set up from his/her calendars. As every calendar come from different sources, it is important to make noticeable which activities come from one source and which come from others. For this reason I thought about giving different colors to each of the different sources of these calendars.

Normally gadget default view is not canvas, so that means the space to render the applications is reduced and many times a lot of information must be displayed on it. For this reason the interface should be clear and very easy to interact with.

Before implementing the gadget information required to display on it was listed. As any agenda the features of each activity are: name, description, date and time.

---

*When an event click is thrown on the name activity, the description is drop down under the activity name.*
The gadget had to be completely configurable so that user could create it with as many resources (public agendas) as they liked it. Another important restriction had to be left up to the user was the number of results the gadget was allowed to show. All these features were configurable through the interface presented before.

In the current version of this gadget, the features are not modifiable once the user submits the url of the calculated gadget to the server container. This could be one enhancement to do in the future.

5.2.2.2 Gadget infrastructure

To achieve the gadget rendering, I had to first develop the necessary services that would bring the activity data to an endpoint where the gadget could get it from. The goal of these services is clear, getting the information from whatever source and sort activities coming from all of them by date. Once this ordination is done the application server must determine which data

![Diagram of gadget infrastructure]

This diagram shows the general scheme of services I implemented. 1st step must be configure with the user preferences which kind of data will be retrieved (1st point). Later on the user will be able to access to the data according to the format he/she chose.
is required for the user (gadget), and parse the information according to user specified parameters.

Activity information coming from e-Catalunya can be retrieved by means of already seen internal interfaces; this doesn’t happen when information is retrieved from remote source. In these cases to make the data received be meaningful, the server needs to use some ical libraries to allow java parsing.

The gadget will, after consulting that service, display its information wherever it is located. The gadget interface programming had to be done according to the non-functional requirements, but it also had to concern about data structure received when asking for activities.

The communication between the e-Catalunya server and the gadget, once it has been rendered, will be done using AJAX. This communication is made over HTTP protocol which supports text based format like XML. So the task from the gadget is performing calls to e-Catalunya servers to get the information and so on parsing it to display it graphically.

Once the gadget and its activities are loaded, the gadget provides a way by which users can ask to the gadget to retrieve more activities. If this mechanism is used, the gadget will perform new search through the servlet and new activities will be appended at the bottom of the agenda.

### 5.2.3 Ical activity service output

That e-Catalunya provides a service returning ical format data is useful to generate activity exporting files. Users can use these files to export their activities wherever they want: social networks accepting this format, desktop applications, mobile devices ... However this is not the only utility we can find to this service. As we are talking about urls pointing to servlets providing ical source activities, these urls can be also used to feed other internet services. One example of this is GCalendar. This Google service offers the possibility to manage users’ calendar in a centralized web interface. GCalendar offers the possibility to export calendars as well as integrate new ones using the standard format ical. Serveral tests were done with this service in order to feed remotely the e-Calendar gadget and do testing. But on the other way around some tests were also done. To import a calendar to GCalendar is a task that can be
done by providing a remote URL. I provided to GCalendar with the URL of the servlet returning iCal format information and the result was successful.

I present a couple of tests I did to check services worked as expected.

A) My first test was creating a URL that configured the servlet to return an iCal format file activities from a calendar “Test Calendar” on my GCalendar account. Once I check the output, I use that URL to import a calendar to GCalendar again. I could achieve my purpose successfully I compared the imported calendar coincide with the source calendar “Test Calendar” in GCalendar.
The next example shows how the same operation than before is done with a little modification. This time the service should merge “Test Calendar” with the calendar “e-Calendario” from e-Catalunya. The mixt is returned as an ical format output and imported once again into a new calendar from GCalendar. The result is shown on the figure below where in yellow letters can be seen the combination. While in orange is shown the “Test Calendar”.
“e-Calendario” is the e-Catalunya source calendar used to be merged with “Test Calendar”.

5.3 e-Catalunya Gadget Integration

The second practical purpose from this project was the modification of the platform to allow users adding gadgets on it. The integration of gadgets had to be done, but the election of the place where putting them was complicated and after the meeting was still unclear. This addition had to come tied to an enhancement in the way users could transmit their information through the e-Catalunya collaborative tools. So the most logic location to think about was setting gadgets within e-Catalunya tools. This allowed the users to take advantage of them while posting their comments. After evaluating the different tools e-Forum, e-blog, e-Calendari, e-Wiki I got to the conclusion that according to the utility of each tool the most suitable place would be in the e-blog.

By adding post in e-blog quickly interaction is reached and its interface allows seeing different comments in one shot. Integrating gadgets here goes after the same purpose of quickly interaction and quickly content understanding. The idea is to encourage users to complement their post comments with helpful application or content that could be useful for the readers.

Therefore I finally modified the forms used by the tool e-blog and I added the chance to add gadgets when editing post. The same way I also provided the necessary logic and interface elements to remove any in-appropriated gadget considered by the author.
Adding gadgets when posting on the e-blog form.

As you can see when adding gadgets the interface offers two possibilities to specify/address them. One option is providing the direction of the gadget file descriptor marked as “URL Gadget” and the other marked as “Codi Gadget”. The last one allows the user adding a html tag block which allow a server container rendering a gadget inside a given iframe in this site.

Snippet of code for embedding a gadget into a HTML webpage using Google server container.

I added the second option because for rendering gadgets inside the platform I used the server container provided by Google, instead of using our own Shindig. However Google just offer rendering for those gadgets which are included in its application directory (say gadgets submitted to be in its directory). For this reason, it would be possible that a user wanted to add a new gadget and the server container which renders it was a different one than Google server container.
The integration of gadgets was done by modifying the comments on the FCKEditor.

Form in e-blog to edit posts, it allows posted gadgets removal too.
The interface lists the names of the gadgets being rendered on the current post so that user can select them in order to remove when desired.

The e-blog websites is finally rendered as follows with the integrated gadgets.

After doing my personal study about gadgets and technologies I had to wait some weeks until I could do the presentation of it in Generalitat de Catalunya. During this time I was eager to understand how gadget technology was working practically, and I did some tests. I installed Apache shindig on my local environment and I implemented a basic gadget system. In short it consisted in adding a portlet into the “Personal Zone” to allow members manage his basic own gadget container. The functionalities of it were very simple and the members could just add and remove gadgets to their environment, but the experience was very useful to get to know internal e-Catalunya services for managing data when developing. This implementation
allowed people from Generalitat to understand a little bit better what I was proposing to them when speaking about gadgets.

Here I add the result of that aimed test implementation.
5.4 Implementation Issues

During the implementation some issues came up and I had to manage to solve them and go ahead with the implementation.

One of the problems I had was rendering the e-Calendar gadget with the e-Catalunya Shindig gadget container. The problem was that the servlet providing rendering services requires the url location from the XML gadget descriptor to render. As in our case this url contains some parameters separated by ‘?’ character, Shindig parse these parameters like parameters for the rendering serer and not parameters concerning the servlet providing the XML gadget descriptor. To solve this problem was easy, the output from the portlet providing the URL for the gadget descriptor was encoded by means of URL encode method from java. And when the servlet providing the descriptor was called, this first to any action it was on charge of performing the decoding, (url decode method). Like this was fixed the problem.

Another problem I had, was once I had already finished building the e-Calendar gadget infrastructure. I did some test and everything seemed to be working. Then I decided to create an URL for a gadget which includes many activity sources (internal calendars and remote source calendars) and the URL was generated successfully. However as you can remember the URL contains the appended parameters concerning the sources the gadget consults to feed
itself so in that case the URL length was very long. This wouldn’t be such a problem if when I added the XML gadget descriptor URL into the text field provided by iGoogle, to add gadgets to the gadget container, it will truncate the length of the given URL at 250 characters for security reasons. Obviously this fact created me a big problem. As e-Catalunya has a tiny-url system I thought about using this system to perform a url transformation into a tiny url that I could add to the iGoogle or Hi5 portal and like this those could accepted and render my gadget on their respective gadget containers. The tiny-url system consists in matching long urls with shorter ones, so that when somebody tries to access to the system by means of a tiny-url the systems detect and redirect the requester to the long url associated to that tiny one.

However when I tried to do it and the result was unsuccessful. When I tried to add the URL I got an error and investigating with the Live Http headers I saw that it was due to iGoogle didn’t allow redirection when adding gadget descriptor url. So I couldn’t manage this way.

Finally I come up with the solution, although I didn’t finally carry it out due to time limitation. The solution I found was developing time-costing and I decided to expose the solution and solve the problem when the solution will pass to production environment.

This solution consists in changing the logic behind the servlet providing personalized gadget URL so that it substituted the appended parameters passed to the old URL for a hash parameter. This matching between the parameterized string and the hash parameter should be stored in the database. After when when the servlet providing the gadget file descriptor is called so that parameter should be matched back to the initial parameterized string in order to generate the right XML gadget descriptor with the hardcoded information from activity sources.

This problem doesn’t affect to the correctness of the solution provided, as the service exporting activity source merge into ical format files was working properly for those gadgets feded by many resources.
6 PROJECT CONCLUSIONS & ANALYSIS

This chapter includes the accomplished goals, the enhancements for the project as well as its initial time planning compared to the final one. It also describes economical cost of the project and the personal conclusions of the author.

6.1 Achieved Objectives

One of the most important accomplished objectives on this project has been the creation of documentation and specific proposals for enhancing e-Catalunya through the studied technologies. The value of this study is very high as it has put in context platform project managers with current technologies used in the social network world, as well as it sets the basis for future actions over the platform. This documentation also contains technology comparison between them that could be useful for LCFIB in the future if same technologies want to be integrated.

Apart from the study done about importation and exportation technologies, the studies about external services providing similar services than those used in e-Catalunya and having a look on the OpenID technology the practical work of this project has been split in two parts.

Conclusively I can say that the biggest part of the two main objectives has been achieved. On one hand content exportation has been achieved with the developing of the e-Calendari gadget, which has been successful and it now import e-Calendar tool to different social networks. The service exporting the merge of different source activities into an ical format file was also finished. The testing stage of the last one has supposed for me a big satisfaction as the new service allowed existing internet service like GCalendar to be feed with important activity information from e-Catalunya that can be very useful for its users.
Secondly the integration of the most talked technology during the project, the Gadget technology, has been materialized in an enhancement in the e-blog tool for providing applications helping to understand content posted by users in e-blog, and it has also supposed the basic implementation of a basic gadget container inside users’ “Personal Zone”. This part has been partially accomplished as far as the time assigned to the project is limited and I had no time to implement the administrator’s management system to control which gadgets could be added and which not.

6.2 Future Lines

During the first stage of implementation an Apache Shindig was integrated inside the development infrastructure of e-Catalunya. Several tests were done with it, between them the deployment of the e-Calendar gadget or the rendering of some gadgets in the e-blog. Some gadgets worked properly while some others that were tried didn’t work, due to the lack of implemented libraries in our server. Although some tries were performed at configuring the Shindig OAuth proxy this work has to be postponed as other issues were more important. In the future these work could be finished, so that we could use our own Apache Shindig in case we want to implement some e-Catalunya functionalities for gadgets.

Other possible line for future work would be the adaptation and gadget support for whatever browser displaying the e-Calendar gadget.

A gadget management system could be offered as a new functionality of the e-Catalunya portal administrator customization,

6.3 Project Cost

The cost of the project is divided in human resources, hardware costs and software costs.

6.3.1 Human resources costs

The development of this project is divided in two distinguished stages. The first stage was the analysis which comprised the initial study, service externalization study, public service providers, study of the importation and exportation technologies, standard specification
research, protocol description, gadget technology and infrastructure, technology comparison and election, integration of exporting and importing mechanisms into e-Catalunya platform.

The second phase was the development of a pilot test integrating gadget capabilities as well as the information services feeding those applications to import e-Catalunya information beyond its borders.

The human resources costs have been calculated for both study and implementation stages, implying in this process the need of a system analyst and a developer. Next table shows the human resource costs detailed with information about the professional profile of the worker, accumulated work time and estimated per hour salary.

<table>
<thead>
<tr>
<th>Hours</th>
<th>Price/Hour</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyst</td>
<td>445</td>
<td>37 €</td>
</tr>
<tr>
<td>Developer</td>
<td>308</td>
<td>27 €</td>
</tr>
<tr>
<td>Total</td>
<td>753</td>
<td></td>
</tr>
</tbody>
</table>

*Human resources costs.*

### 6.3.2 Hardware costs

In order to carry out this project I only required a personal computer to perform the study and implementation of my pilot test. The next figure shows its cost.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>900 €</td>
</tr>
<tr>
<td>Total</td>
<td>900 €</td>
</tr>
</tbody>
</table>

*Hardware costs.*

### 6.3.3 Software costs

All the software used during the development of this project was free software. Next figure displays the software costs.
### 6.3.4 Total Cost

In the next figure the total cost of the project is displayed. The cost is divided into human resources cost, the hardware cost and the software cost.

<table>
<thead>
<tr>
<th>Resources</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human resources</td>
<td>24781 €</td>
</tr>
<tr>
<td>Hardware</td>
<td>900 €</td>
</tr>
<tr>
<td>Software</td>
<td>0 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>25681 €</strong></td>
</tr>
</tbody>
</table>

**Total costs.**

### 6.4 Project Planning

The time-distribution in this project has been split as the Gantt diagram shows. I dedicated a big part of my study to understand the social technologies. My background about exportation technologies was very poor so it took me a long time to understand how they work.

The redaction of documentation for internal purposes and the preparation of a presentation to my teamwork and supervisors as well as another for Generalitat de Catalunya required to me a lot of my time. An added difficulty to this was the fact all of these documents had to be written in English as I wanted to include them in the project and this took me extra time. However intermediate documents handed to the government were written in Catalan which is my mother tongue. The main reason why this information is not included in this project, is once again because of confidentiality purposes and project don’t want to be available publically.
The planning of this project was a little bit tricky to define. Generalitat ordered to LCFIB to perform an study to understand which were the capabilities that new importation/exportation technologies could offer to e-Catalunya. My knowledge about these technologies was reduced, although I had listened some terms related about technologies my idea about these technologies was completely unclear and un-connected.

6.4.1 First stage planning

The study part of this project was based into different topics. The first study I did covered different available services on internet that could allow e-Catalunya to externalize some of its services. This study takes me around two weeks and after I passed to the second and biggest study part of my project: importation and exportation technologies.

The scope of the study was so huge that at the beginning I was completely lost acquiring many concepts when I was doing the research. The scope of the research was unreachable as many technologies, services and standards come up when I was surfing on the net. Too many different services, standards, APIs and all kind of technologies were too confusing for me to get a general idea about what was the state of art. It took me almost one month until I started having a general idea about what the different possibilities were.

Once I had a clear idea about what where the requirements of my study, I did an initial planning about the study I should do on some technologies I had already started learning.

Since the objectives from my project wouldn’t be set until I presented the results of my study to Generalitat de Catalunya I could not plan my second stage.

This is the gantt chart that shows the initial first stage planning in which I basically did information research.
First stage Original Gantt Diagram
The following sections summarize the tasks defined in the gantt chart.

6.4.1.1  Externalization service study

Although it didn’t take me a lot of time, this initial study was laborious as I tried to find the services matching the platform’s objectives and I didn’t find all of them.

6.4.1.2  Importation and exportation technology study

In this part of my study, it is explained how Service Providers can be integrated by other websites. It introduces the creation of specific embedded applications into their specific social networks.

Although the general concept was not very difficult to grasp, the study of a concrete proprietary solution for building exportation applications was not straightforward at all.

6.4.1.3  Gadget Technology

This phase includes the description of a Gadget, its specification and its implementation as well as its motivations. In contrast to gadgets, proprietary applications only allow embedding applications in a particular social network domain. This stage helps telling the way for exporting data between social networks by following a more extensible application exportation standard.

Some confusion appeared here between the specifications of gadgets and the technological implementation carried out by Google and Google gadgets. This was traduced into a time delay in the estimated planning.

6.4.1.4  OpenSocial

The complicated software infrastructure supporting OpenSocial gadgets and its mechanism were difficult to understand. In order to make social gadgets work using the social network
where they are deployed, it made me spend a long time achieving comprehension on this point.

Moreover integration of this technology on e-Catalunya server container was a time-costing complex process to be understood.

6.4.1.1.5 Rendering services

This section defines different ways to achieve gadget rendering. Special attention deserves this point, as it outlines the future implementation will be followed to add gadgets to the platform.

6.4.1.1.6 Comparative technologies

After having a look to all the studied technologies, some comparison had to be established between them in order to choose which of them offer better capabilities for e-Catalunya. Later, the implementation of them will be carried out.

6.4.2 First Stage planning revision

At the end of this first stage some of the planned tasks were not achieved like it was expected. Some deviations on the initial planning forced me to delay the implementing stage start.

The disadjust in the planning was motivated by the time I dedicated to generate the documentation about my study for the government project managers.

I underestimate the time I would require to write low technical level documentation and handing in to Generalitat’s project managers. In the new diagram we can see how the task “Presentació Generalitat” is enlarged due to the lack of time to finish the documentation when I expected to do it. So these 9 days of delay force all the implementation process to move ahead 9 days.
Gantt chart with the real time spent on each task.
Once finished the first stage me and my supervisors in the LCFIB went to “El Departament de Direcció General de Ciutadania de la Generalitat de Catalunya” in order to expose my study. Since I could understand their desired requirements for the platform, by means of meeting repeatedly with them, I offered them some proposals to be integrated to the platform. Once they accepted I could define the technology I would use and then do the planning again with the new data.

**Second Stage planning**

Once the documentation was handed in to “El Departament de Direcció General de Ciutadania de la Generalitat de Catalunya” with all my proposals, complying the requirements they had transmit to me in previous meetings. Then I had to wait until the green light arrived. And when they finally accepted my proposals, I planned the second stage with another gantt diagram.
<table>
<thead>
<tr>
<th>Número de tarea</th>
<th>Tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Estudio externalització de serveis</td>
<td>5 días</td>
<td>lun 02/02/09</td>
<td>vie 06/02/09</td>
</tr>
<tr>
<td>2</td>
<td>Entrega Documentació externalització</td>
<td>5 días</td>
<td>lun 09/02/09</td>
<td>vie 13/02/09</td>
</tr>
<tr>
<td>3</td>
<td>Estudi de l'estat de l'art</td>
<td>60 días</td>
<td>lun 16/02/09</td>
<td>ven 08/05/09</td>
</tr>
<tr>
<td>4</td>
<td>Presentació Generalitat Ce</td>
<td>12 días</td>
<td>mar 09/02/09</td>
<td>mar 09/02/09</td>
</tr>
<tr>
<td>5</td>
<td>Integració de gadget i plataforma</td>
<td>6 días</td>
<td>mar 10/02/09</td>
<td>mar 10/02/09</td>
</tr>
<tr>
<td>6</td>
<td>Desenvolupament</td>
<td>6 días</td>
<td>mar 10/02/09</td>
<td>mar 10/02/09</td>
</tr>
<tr>
<td>7</td>
<td>Implementació</td>
<td>18 días</td>
<td>mar 09/02/09</td>
<td>mar 09/02/09</td>
</tr>
<tr>
<td>8</td>
<td>Test</td>
<td>5 días</td>
<td>mar 16/02/09</td>
<td>mar 16/02/09</td>
</tr>
<tr>
<td>9</td>
<td>Creació gadget e-Calendari</td>
<td>1 dia</td>
<td>mar 07/03/09</td>
<td>mar 07/03/09</td>
</tr>
<tr>
<td>10</td>
<td>Disseny</td>
<td>5 días</td>
<td>mar 11/03/09</td>
<td>mar 11/03/09</td>
</tr>
<tr>
<td>11</td>
<td>Implementació</td>
<td>15 días</td>
<td>mar 11/03/09</td>
<td>mar 11/03/09</td>
</tr>
<tr>
<td>12</td>
<td>Test</td>
<td>2 días</td>
<td>mar 16/03/09</td>
<td>mar 16/03/09</td>
</tr>
<tr>
<td>13</td>
<td>Disseny</td>
<td>5 días</td>
<td>mar 16/03/09</td>
<td>mar 16/03/09</td>
</tr>
<tr>
<td>14</td>
<td>Test</td>
<td>2 días</td>
<td>mar 21/03/09</td>
<td>mar 21/03/09</td>
</tr>
<tr>
<td>15</td>
<td>Memòria</td>
<td>31 díes</td>
<td>mar 05/04/09</td>
<td>mar 05/04/09</td>
</tr>
</tbody>
</table>
The following sections summarize the tasks defined in the gantt chart.

### 6.4.2.1 Importing e-Calendar

This phase is on charge of the development of the gadget and all the mechanisms needed to give support to this application. This part is basically subdivided in: Gadget design, Gadget configuration interface, Gadget descriptor retrieval file. A part from this, another service will be on charge to provide the data that continuously will feed the application. This service will allow changing its data delivery format in order to feed a gadget or conform an activity standard format file. The development of every one of these points was much difficult than expected and it requires a considerable amount of time to be tested. This will change the planning again.

### 6.4.2.2 Gadget system integration

This task consists in the development of a pilot test to see how gadget integration can be useful for e-Catalunya users.

This task was subdivided in two sub-tasks:

1rst) The first subtask focuses on the integration of gadgets in the e-blog tool. This task was developed in the planified time.

2nd) On the other case, a gadget portlet had to be developed in order to offer gadget addition, removal and visualization. This option finally required some more time than necessary not due to implementation problems, but due to my wish to test the environment using the two different server containers.

### 6.4.2.3 Documentation

Documentation includes internal reports, personal notes as well as the final master thesis report. However internal LCFIB documents I provided to my bosses are confidential.
This second stage ended 5 days later than expected. These problems were due to one bug found during the testing phase and the tests done with the own version of shindig. I also had to fix the bug and it took me some time. In the next diagram we can see expressed this delay. If we do the total addition, we see that I finished the implementations 14 days after expected.
Estudia y externaliza que los servicios
ctrl 1
abrir

dia

n

ntrega Documentació externització

3

4

60 días?

6

48 días?

8

17 días?

9

Implem entació

11

Disse

12

Testing

14

15

Final Gantt diagram showing which was the final process.
6.5 Personal Conclusions

Thanks to this project the last six months have been both professionally and personally a very rich experience for me. This master thesis has brought me the opportunity to use many of the skills I have developed last years in FIB to in order to extend its capabilities. While the formation received by the School of Informatics has been most useful in a theoric sense, this working experience has complemented this in a practical sense.

The project has addressed the study about different topics in e-Catalunya and some of them have been finally implemented in a prototype pilot. The development of this project taught me a lot not only in the application development from a technical approach, but in a organizing a work team perspective.

I have learnt a lot about team work and companies organizational structure. This project has been a very valuable experience on working in large development teams. Communication between team members has proved essential to keep track of the project’s evolution as a whole. Keeping close contact between team members has proven to be important to the project benefit, as it provides flexibility in the team, allowing tasks to be easily shared and mutual help to be effective.

For understanding the design of e-Catalunya it was basic the knowledge I acquired coursing the different subjects in the FIB. Software Engineering II (ESII) helped me to understand the platform infrastructure and the logic representation of the e-Catalunya. On the other hand, computer networks (XC and PXC) or SSI were essential for the understanding of many issues from the topic of my project.

I firmly believe that this master thesis has pushed me towards the enterprise world and help me to put in practice many of the things I learnt during the degree inside a professional environment.

In spit of these considerations it is also good to take into account those things that somehow put on me an additional difficulty when doing the project. I am talking, for example, about the huge quantity of information existing today, and the impossibility to process it all. This unreacheability made me lost many times during the study process, but finally I could find the way to elect those technologies and topics I had to focus on. My guide in that case was completelly the stakeholders and project managers from e-Catalunya. Thanks to my contact
with them I could understand better what they required and so on act and research consequently.
7 BIBLIOGRAPHY

Other e-Catalunya Projects


Ponce Liu, Lucas; “Disseny i implementació del sistema de grups, invitacions i subscripcions a la plataforma eCatalunya”, Projecte fi de carrera-UPC: Professor Josep Casanovas

Visited websites:


OpenSocial: http://www.opensocial.org/

Facebook Developer’s page: http://developers.facebook.com/

OAuth protocol specification: http://oauth.net/


SocialSite: http://incubator.apache.org/shindig/

W3C widget specification: http://www.w3.org/TR/widgets/

Survey Providers:

http://www.hostedsurvey.com/case-study-mathews.html

http://www.keysurvey.com/enterprise/api.jsp

8 Appendix

8.1 OpenSocial Server Container Compliance

To be gadgets-compliant and satisfy standards, a server has to handle three types of requests according to specification description: the Gadget Rendering Request, the Gadget Metadata Request, and the JavaScript Request.

8.1.1 Gadget Rendering Request

It refers to the process of translating gadget XML into “renderizable” code by the browser, typically inside an IFRAME. It concerns the core gadget API provided by gadget container and transparent to the developers.

The input for this request is the following:

- Gadget XML, usually referenced by a URL.
- User gadgets preferences.
- Locale of the end user (information for internationalization gadget application. E.g: time, language, measures …)
- And some other optional features server gadget rendering options (referering to server’s cache usage), or gadget identifiers within gadget server.

To finally get for output:

- HTML, CSS, and JavaScript code.

Process:

1. Fetch the gadget XML.
   i. The server should provide a mechanism to retrieve a gadget descriptor as fast as possible. So it must provide a cache.
   ii. The server should provide mechanisms for balancing its load according to the frequency with which they can reliably deploy updated content.
   iii. The server should provide mechanisms to disable cache-systems which would improve application testing and development tasks.
a. Support should be done via HTTP-based requests.

b. A server denial-of-service protection limiting the number fetches for a specification in a given period of time, should be honored to avoid attacks during this developing early stages.

2. Parse the gadget XML.
   i. The XML file descriptors must conform to the extended gadget specification XSD templates. These templates represent valid format accepted by Google gadget server the descriptor must satisfy.

3. Identify the Locale tag from the XML, and fetch messages according to what it is specified on it.
   i. Locate “locale” element specifying language and/or country.
   ii. Load the message bundle specified in element, if any. Like this gadget will be rendered for a targeted geographical area.

4. Substitute "hangman variables" into supported gadget specification fields. Hangman variables are of the form __<TYPE>_<ID>__, and they are commonly use in gadget development for programmatically changing gadget content on runtime. At this stage these variables are replaced with right string values. There are four types of hangman variables:
   i. Process message bundle into __MSG_foo__ hangman variables.
      a. The bundle MUST conform to the message bundle XSD.
      b. For each message named foo with value bar, hangman expansion __MSG.foo__ = bar.
   ii. For each provided User Pref with key foo and value bar, hangman expansion _UP_foo__ = bar
   iii. Hangman expansion __MODULE_ID__ = <provided module ID> (gadget identifier).
      a. HTTP-based request servers SHOULD use the URL parameter named mid for this.
   iv. Perform substitution of each hangman variable for those variables concerning BIDI API values. These values reference to the interpretation direction of the content rendered in the gadget. These values depend on the locale writing systems, in different areas in the world can be from right-to-left or left-to-right.
   v. Perform substitutions of each hangman expansion on all fields which get displayed to users.

5. Process gadget features referred as <Require> or <Optional> in the gadget XML.
   i. If an unsupported features is specified in a <Require> block, the server must indicate that it cannot satisfy the rendering request.
   ii. The server must support the following core JavaScript API to use basic data management and features functionalities. (e.g: methods for checking the existence of a library on the server container). (REFERENCIA A LA GADGET METADATA REQUEST).

6. Output gadget content:
   i. For type HTML gadgets the format is the typical skeleton from a website:
      a. Standard HTML header, opening <html> tag and <body> tag. <head> is optional.
      b. Core gadgets JavaScript libraries.
      c. Feature library initialization code, if needed.
d. Processed gadget content, after doing variable substitution and with added JavaScript supported functionalities.

e. A single call to `gadgets.util.runOnLoadHandlers()`. This is used to call the execution of JavaScript after a frame has completely loaded.

f. Standard HTML closing tags.

ii. For type URL gadgets, the provided “url” has to be appended by the following parameters:

- `up_<name>=<value>` for each provided user preference.
- `lang=<language>` and `country=<country>`
- `libs=<jsLib>`, where `<jsLib>` is a comma-separated list of URL path fragments relative to the server's JavaScript request handler path.

  a) These fragments are added to the JS path by the type URL target to load gadget API JavaScript to incorporate its descriptive logic into its execution context.

### 8.1.2 Gadget Metadata Request

This API tells a gadget container how a gadget’s rendering on a page must be done, and gives the server container to support the gadget’s features.

Gadget container’s support is required to satisfy gadget API, due to the browser security model. In other words, some features like `dynamic-height` allows a gadget to request its resizing when being deployed in another context other than that controlled by the same server container. Often, gadgets are rendered on a different domain than their its container and they cannot modify their height.

To fit its content into the displayable iframe within the website where it is deployed, the browser makes an inter-domain procedure call to the iframe container requesting resizing. These calls are done over the core `gadgets.rpc.*` APIs, which uses browser specific transmission methods to pass messages between frames settled on different domains located
in the same browser context (the website containing the gadget). The container needs to receive this request and answers accordingly with the appropriated resized gadget code.

The server module performing this logic is commonly called container-side JavaScript.

**Inputs:**

- Gadget XML, usually referenced by a URL.
- User gadgets preferences.
- Locale of the end user (information for internationalization gadget application. E.g: time, language, measures …)
- And some other optional features server gadget rendering options (referering to server’s cache usage), or gadget identifiers within gadget server.

**Outputs:**

- URL to a Gadget Rendering Request for the requested gadget.
- Reference to container JavaScript needed to satisfy needed gadget API requests.

**Process:**

1. Fetch and after parse gadget XML as previously described in Gadget Rendering Request.
2. Process gadget features, specified as `<Require>` or `<Optional>` in the gadget descriptor.
   - i. Emit an error message if a required feature is not supported.
   - ii. Otherwise, enqueue the container JavaScript to satisfy each of the features required and return them to the client.
3. Construct URL to the corresponding Gadget Rendering Request, generated as follows:
   - i. Initialize URL using a static prefix.
   - ii. Add parameter `url=<url>`, where `<url>` is URL of the requested gadget specification.
   - iii. Add user preferences parameters as described in Gadget Rendering Request 6.
   - iv. Add locale parameters as described in Gadget Rendering Request
4. Return the enqueued container-side JavaScript and Gadget Rendering Request URL.

**Sanitized Gadget**

Still under development this API will incorporate soon sanitization of the gadget code. This will be done by rewriting HTML, CSS, and JavaScript maintaining syntactically unaltered the content and preventing unsafe DOM or cookie access to the gadget content running inside the target iframe.
8.1.3  *JavaScript Request*

All gadget core APIs are provided by the server in order to conform to the Gadgets API Reference. Gadget developers should not have to request any of these features, as the container should provide them all automatically. Obviously other functionality and feature-specific libraries will have to be requested to get benefit of its capabilities.

8.2 *Shindig server side Components*

Shindig is composed of 3 main components giving the following functionalities: Persistent Data Loading Mechanism, Gadget Rendering Infrastructure and OpenSocial server side implementation. In addition to that a REST Implementation is also provided to attend REST requests.

![Diagram of Shindig components](image)

The four components are accessed using HTTP or HTTPS by means of a servlet. OpenSocial Server side components and Persistent Data Access Layer share a common Servlet, while Gadget Rendering and REST handler components have a separate Servlet entry point for carrying out their purposes.
This is a class model showing what shindig is composed of. There are 9 servlets or multiple points of entry.

**JsonRpcServlet or GadgetDataServlet:** This class serves JSON requests for OpenSocial API. It loads data from a storage system and convert it into DataObjects.

**DataServiceServlet:** Serves REST calls.

**GadgetRenderingServlet:** Is the one that converts Gadget Xml into executable end-user code.

**Proxy Servlet:** Get the remote content for “url” gadgets and converts it into JSON and sends it back.

**JsServlet:** Used for Loading JS libraries from external sources. It allows loading core.js and rpc.

**RpcServlet:** Used to handle RPC metadata requests (e.g: configurable calls for gadget rendering). Returns Gadget and its specification in the JsonFormat for all the gadgets.

**MakeRequestServlet:** Serves MakeRequest.

**ConcatProxyServlet:** concatenates responses of several proxied HTTP responses.

**OAuthCallBackServlet:** Call back servlets for Oauth calls.
8.2.1 Persistent Data Loading Mechanism

This layer is in charge of loading the stored social data like for example: person details, friends list, activities associated with a person.

The data requested is loaded into container’s memory in a previous step. Later on OpenSocialData Handler populates the Open Social objects with information coming from the social data storage system, and finally it performs queries over them. To do this, matching between OpenSocial API methods and the queries to the core social network database the GadgetDataServlet provides an entry point for it.
8.2.2 **Gadget Rendering Infrastructure**

This component concerns about both client side and server side components. It is responsible for generation of Gadget HTML from the Gadget XML source file descriptor.

**Client Side Components**

The browser will use the gadgets.js provided by the server container in order to display the gadgets. This library will call the GadgetRenderingServlet.

**Server Side Components**

The client will perform HttpServletRequests which will be attended by the GadgetRenderingServlet. It will convert XML into HTML and a GadgetSpec object will be created. The next step will be populating it with the appropriate views, modulePrefs and userPrefs. Finally, it creates a Gadget object and then sends it to the browser.

The scheme below explains the sequence in more detail on how HTML is generated from the Gadget XML and sent back to the browser.

Collaboration diagram for rendering the Gadget – HTML from the Gadget XML

Collaboration flow on how the Gadget XML gets converted into HTML and by the GadgetRenderingServlet
8.2.3 OpenSocial Server Side Implementation

Client Side Components

On the client side, the browser must be able to use the OpenSocial for retrieving information from the platform by requesting it from the server. All the Java script files within these are added or referenced to the HTML generated.

Server Side Components

All the OpenSocial calls are received by GadgetDataServlet and passed on to the OpenSocialDataHandler. This handler will be called according to the type of request for data retrieval. The available types of request are the following:

FETCH_PEOPLE: to fetch Person data

FETCH_PERSON_APP_DATA: to get the data attributes for a person

UPDATE_PERSON_APP_DATA: update attributes of a person

FETCH_ACTIVITIES: To get List of activities for a Particular Person

CREATE_ACTIVITY: To create Activity associated with a Person

The request FETCH_PEOPLE is chosen as an example and goes through the flow in the figure below. Default People service is a wrapper on the platform’s system persistent mechanism.
The service loads, parses and stores the data of and from Person, Friends, Activities in Memory. Depending on the nature of the call data is retrieved in memory (6 to 14) and converted into JSON object (14) and sent back (15) to the client side for presentation.

Collaboration diagram tracing call to view friends from client side javascript to server side components and back

8.2.4 REST Implementation
**Client Side container**

Shindig Client side has a Restful container extending OpenSocial container and is the component which forward all the OpenSocial API calls to the REST endpoints. This means that the Endpoints serve both direct HTTP calls as well as calls from the OpenSocial APIs. Restful Container uses gadget.io package which opens an XmlHttpRequest object and then sends the data to the server. The REST endpoints are supported by the service DataServiceServlet.

The server container provides by means of REST endpoints four types of data: people, activities, application data and groups.

![Diagram](image.png)

*Restful Client container to make Client calls to REST Endpoints for OpenSocial Javascript API.*

---

8.3 **Shindig integration in e-Catalunya infrastructure**

In order to do the server container integration with e-Catalunya some mechanism needed to integrate code on the container. Google Guice is an open source software framework for Java platform. It supplies external dependencies (in this case the specific classes to implement social information in e-Catalunya) to a software component by using specific annotations to configure Java objects.
I have carried out a study about what should be done in case a server container to support gadgets wants to be added to e-Catalunya infrastructure. In this study I have found some pros and cons which I will explain in the following paragraphs.

e-Catalunya is nowadays running under 1.4.2 java version. The platform is deployed on a Apache Tomcat server which is working with the same java version. A future migration to the 1.5 version is forecasted. However, as is known, some problems cannot be fixed and it is running under the previous version. As we can notice this version doesn’t match the requirements for Shindig extension. This supposes an inconvenience, as we must set up another internet application server working with Shindig java version to support the server container.

By doing this we must play with Apache server workers which allow splitting the required functionalities into two servers. It must redirect those requests concerning e-Catalunya towards the Tomcat running the 1.4.2 version and the others requests for gadget or OpenSocial data services to the other Tomcat running Apache Shindig. This matching is done by url request pattern matching and this will solve the problem of java version. Once we fix this problem, the next step should be downloading the Shindig installation source code available on the OpenSocial official webpage and buildint the project.

This is done, Shindig defines a language specific through which a social network's backend can be connected to the OpenSocial Service Provider Interface (SPI), which is part of Shindig. It allows an OpenSocial application to access to e-Catalunya data. The SPI implements: Retrieving people information, Storing and retrieving activities, Storing and retrieving persistent data and messaging capabilities.

The purpose of a Service Provider Interface (SPI) is to make it easy for new websites to become an OpenSocial container. With that in mind, the current SPI only requires a few files’ changes. In SPI there are the services which are normally interfaces or abstract classes defining the services, and service providers which are the concrete implementations of those services.

This interface between OpenSocial Shindig classes and those from e-Catalunya should be programmed. In order words, OpenSocial core relies on the data extracted from a social network database, and the matching between OpenSocial data model objects and those from e-Catalunya should be done. So some opportune changes to Shindig code must be done.

OpenSocial data requests are provided by Shindig through three main services feeding data:
PersonService
ActivityService
AppDataService

8.3.1 Defining an own data service implementation

1. Make an implementation for all the methods to retrieve data in the three interfaces. This must be done in org.apache.shindig.social.spi.ShindigIntegratorPersonService.

   This implementation looks similar to the sample implementation provided with Shindig. The big difference is that the db instance is an instance of a class that holds the data in memory. Special attention must be put when matching OpenSocial class names with those from our platform. For example in our case e-Catalunya definition for people is user.

   The mechanism or methods to create person instance must also be defined in order to make the data persistent. This should be done by modifying ShindigIntegratorPersistenceAdapter class.

2. Registering these classes on Shindig to be used in our platform’s OpenSocial layer. To do the registration it is required to use Google Guice module.

   Guice bound implementation classes to be bounded programmatically into an interface, being injected into constructors, methods or fields using an @Inject annotation.

   org.apache.shindig.social.core.config.ShindigIntegratorGuiceModule.class binds the Shindig services to the new implementations. So they will be called instead of the sample coming with Shindig. If the registration of classes has been done into a new module it’s important to call it when starting the application.

3. Register the Guice module in the web.xml. In this file, the param guice-modules has to be changed to reference the new implementation of the module.
There are several web.xml files in the shindig installation.

web.gadgets.xml: just the gadget rendering server

web.social.xml: just the social data / RESTful server

web.full.xml: both servers in a single container

4. Rebuild Shindig by using Apache Maven. It has to be run with a clean install and if everything goes well Shindig OpenSocial data model and e-Catalunya's should coincide.

8.3.2 Creating a backend database for OpenSocial data

1. The first step to be done is setting up a MySQL database schema for OpenSocial.

   Setup 'opensocial' schema. The creation of a new database is required. A new user opensocial with password opensocial must be also created.

2. After setting up the OpenSocial data core database, the classes supporting the specification have to be loaded. The installation of SQL objects can be done by means of opensocial_mysql.sql script provided by the implementation.

8.3.3 Using extended SQL Java classes within Shindig to manage the database

Once the database is set up with OpenSocial data, it is time to enable the container within Shindig for use it instead of working directly with the static XML file set Shindig uses as default like database. With this purpose, a set of custom Java classes have been added to Shindig for providing a simple way to connect to MySQL instead of reading the XML file.

The classes have been added are: those implementing the OpenSocial interfaces: SQLActivitiesService, SQLDataService, SQLPeopleService, implementing GadgetDataHandler
like **SQLDataHandler** (which differ the different requests into the different interfaces) and classes called by SQLDataHandler to extract state from JDBC source is **SQLDataLayer**.

In order to add the new SQL implementation classes to Shindig, they must be downloaded to a new package folder in `java/gadgets/src/main/java/org/apache/shindig/social/samplecontainer` called `org.apache.shindig.social.samplecontainer.sql`.

**Editing the existing Shindig code to use the new SQL classes**

1. Apart from these new classes, some modifications must be done to `<shindig>/java/social-api/src/main/java/org/apache/shindig/social/SocialApiGuiceModule` to invoke them and like this binding the SQL implementations.

   Finally some edition to the existing Shindig code must be done to the new SQL classes. For example edit the existing SocialApiGuiceModule file to invoke these new classes.

   The new datahandler (SQLDataHandler) will also have to be invoked instead of the old one (OpenSocialDataHandler). The StateFileHandler handler has to be removed as it won’t be used anymore.

2. Import statements must be changed in the modified SocialApiGuiceModule.java file to now refer to these new samplecontainer/sql/SQL*.java classes instead of the old ones.

3. A MySQL dependency must be added to to the social-api project’s “`<shindig_home>/java/social-api`” pom.xml file. This will pull down the necessary MySQL JDBC jar file and package it with Shindig.

4. Finally, if needed, edit the JDBC connection information in the class file SQLDataLayer.java.

5. Once these changes are applied, compilation and code packaging will be done by issuing the maven build commands again.

After following these steps, Shindig should be providing social information through the different endpoints from OpenSocial layer. The server should also be ready to start gadget rendering.
Setting up a new server to run Shindig and providing OpenSocial functionalities, as we have seen, is a laborious job which means changing many classes and configurations files. The implementation of these services may look straightforward, but the difficulty of the matter lies in getting familiarized with Shindig architecture.
8.4 Glossary & Basic Technological Issues

**ical Format:** iCalendar is an activity file format for transmission.

that allows internet users to send meeting requests and tasks to other internet users, via email, or sharing files. Its extension is .ics, .ifb or .ical. It allows storing scheduled activities in a standard format to be read by programs such as email clients or calendar applications.

**Google GUICE:** It is a software framework which provides external dependencies to a software component. It is used to configure Java objects. Guice allows implementation classes to be programatically bound to an interface, then injected into constructors, methods or fields using an @Inject annotation.

**OpenSocket:** OpenSocket is a service to convert your Open Social gadgets into Facebook apps.

**SocialSite:** Is an open source project for building Widgets and webServices. This software allows easy deployment of a social network in a website. It implements th OpenSocial Layer and offer some extra functionalities.

**Gadget Server o Server Container:** It is a on charge of performing the rendering of the gadget.

**OAuth:** An open protocol to allow secure API authorization in a simple and standard method from desktop and web applications.

**OpenSocial:** Set of common APIs for web-based social network applications (social gadgets). Developed by Google along with other social network
partners. Applications (gadgets), that implement OpenSocial APIs will be interoperable with any social network system that supports them.

**Apache Shindig:** Opensource project to provide reference implementation for the OpenSocial standard, capable of rendering OpenSocial gadgets in a client browser.

**Service provider:** A service provider is an entity that provides services to other entities. Usually this refers to a business that provides subscription or web. (e.g: Yahoo offer his users access to their e-mail data.)

### 8.4.1 Gadget

Gadgets are designed to offer exportation services to any website desiring to import remote applications or content. They give to developers the chance to write applications once and run them everywhere they are embedded in, without concerning about which target web environments will contain them. This is the main strength of this technology.

This technology is suitable for both developers who require a mechanism to improve their personal web experiences, and bigger projects like social networks and companies. The latter uses gadgets to improve syndication mechanisms and integrate third party social functionalities. For instance: gadgets taking advantage of the information provided by services in the same domain (Ajax calls), or social information from that social network domain.

Gadgets are mini-applications that can serve almost any purpose (e.g. mail readers, weather reports, slide shows, search, games, etc.). Some gadgets integrate with other Windows Live services, including Mail, Search, and Favorites.

Users can create multiple site tabs and customize each with different feeds, gadgets, layouts, and color schemes.
8.4.2 **Data Management**

8.4.2.1 **Hibernate**

Hibernate is a persistence engine for Java, capable of maintaining persistence of a given Object domain of Plain Old Java Objects (POJOs), by means of mapping and storing its information into a database. Hibernate is useful for easing mapping object oriented data into a relational database.

It offers the mechanisms to map how an object and its associations towards others must be stored, updated or removed from a relational database. Once this configuration is cleared, objects can be loaded from and stored into the database without writing further “object to relational” mapping code.

*Hibernate acts as a database interface for applications providing persistence to Java objects*
8.4.3 Communications

8.4.3.1 HTTP

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, TCP/IP routing based, stateless protocol which can be used for many tasks beyond its hypertext usage. For instance it is used for name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP is built over the TCP/IP control and routing protocols.

8.4.3.2 REST

REST Architecture concerns a set of network principles which outline how resources are defined and addressed. It is often used to describe any simple interface which transmits data from one domain to another over HTTP without an additional messaging layer like SOAP, or session tracking.

It operates in terms of resources and operations on them. This protocol is defined on top of the HTTP protocol, and uses the standard HTTP methods to retrieve and change server state. With REST, the object and the verb are separated. A fixed set of verbs or methods exists (GET, POST, PUT, etc), and they may be applicable over an infinite set of internet resources. It is called resource to the path of the object within the schema on which the action is desired to take place.

No single data representation is ideal for every client. Thus the server offers a generic interface that allows building the final result output in the desired output format, by just writing the protocol once. This is achieved thanks to the “one to one” mapping XML and JSON formats have, while the Atom format by which it was first designed, is defined separately.
8.4.3.3  **RPC**

It is a communication technology that allows performing remote calls to procedures hosted in another domain transparently to the user. In short, the programmer would write essentially the same code whether the subroutine is local to the executing program, or remote.

RPC is an alternative to the RESTful API specification. It attempts to support the same data and operations as the RESTful API does, but in a form that is easier to manage with JSON data format. Any sequence of RPCs should be convertible to a sequence of RESTful calls automatically and maintain the same semantics.

8.4.4  **Web based interface**

8.4.4.1  **Portlets**

Portlets are pluggable user interface components that are managed and displayed in a web portal. Portlets produce fragments of mark-up code that are aggregated into a portal page. A portal page is displayed as a collection of non-overlapping portlet windows, each displaying a portlet. Portlet applications include email, weather reports, discussion forums, and news.

Portlet standards are intended to enable software developers to create portlets that can be plugged in any portal supporting the standards.

Portlets are used to provide a web interface for the platform’s different functionalities, being managed by the eXo Platform’s portlet container, which in turn delivers the web-contents through Apache + Tomcat servers.
Here we can appreciate three different portlets hosted on the same page.

8.4.4.2 Velocity

Velocity is a template engine allowing easy generation of files of whatever type, in our case HTML, by means of using templates which are filled with values retrieved from Java object instances. The engine also supports some basic conditional branching and looping instructions while processing the templates.

Velocity has been used in the platform to split the presentation layer from the domain layer:

Portlets at the e-Catalunya platform use velocity to build their HTML responses, rather than writing HTML by themselves. In this way, style and format of any page can be modified as simply as modifying the corresponding velocity template.

Velocity template for HTML generation:

```html
<HTML>
<HEAD>
</HEAD>
<BODY>
  Hello $customer.Name!
  Here is a list of your current subscriptions:<br/>
  <table>
    #foreach ($subscription in $subscriptions )
      #if ($customer.isSubscribed($subscription) )
        <tr>
          <td>
            $subscription.Name
          </td>
        </tr>
      #end
    #end
  </table>
</BODY>
</HTML>
```
8.4.4.3  **Servlets**

The Java Servlet API allows a software developer to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets are the Java equivalent to non-Java dynamic Web content technologies such as CGI. Servlets can maintain state across many server transactions by using HTTP cookies, session variables or URL rewriting.

8.4.5  **Digital signature**

**Process interacting Actors**

Because understanding of OAuth protocol is tied to digital signature understanding, the second will be defined in terms that will help the comprehension of the first. Some terms deserve to be introduced before definition:

- The **Service Provider** (SP) is the term used to describe the website or web-service where the restricted resources are located. E.g: photo sharing site where users keep albums, an online bank service or any other service where ‘user’s private stuff’ is kept.
- The **users** have ‘stuff’ they don’t want to make public on the Service Provider, but they do want to share it with another site or they want to have access personally.
- **Consumer** is the name for the application or physical person trying to access the User’s resources. This can be a website, a desktop program, a mobile device ... The Consumer is the one to who the resources access is granted. In OAuth terms Consumer is the entity writing code to interact with the Service Provider.
- **Protected Resources**: the ‘stuff’ OAuth protects and allows access to. This data can be: photos, documents, contacts, activities...

Tokens are used instead of user credentials to access resources, when the user has already been authenticated. They are random unique strings, hard to guess, and paired with a Secret to protect the Token from being abused.
A digital signature guarantees that a digital message or document has not been modified, demonstrating its authenticity. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, and that it was not altered while being transferred. Digital signatures are commonly used for setting up a secure connection to a Web site and verifying the integrity of files transmitted.

Digital signatures employ a type of asymmetric cryptography. For messages sent through an insecure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender. Digital signatures also provide non-repudiation, understanding that the signer cannot successfully claim that they did not sign a message, while also claiming their private key remains secret. Moreover, some non-repudiation schemes offer a time stamp for the digital signature, so that even if the private key is exposed, the signature is not valid anymore.

The asymmetric key-pairs work in the following way. Each side, the consumer and Service Provider, uses one key to sign the request and another key to verify the request. The keys, which we refer as private key, for the consumer and public key, for the Service Provider, must match, and only the right pair can successfully sign and verify the request. The advantage of using asymmetric shared secrets is that even the Service Provider does not have access to the Consumer's Private Key, which reduces the possibility of the secret being cracked.