

Univerza v Ljubljani



Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

**Facultat de Matemàtiques
Universitat de Barcelona**

TOWN OPEN-DATA PORTAL (Erasmus TFG)

Jordi Xaubet Pujadó

Supervisors: Mira Trebar – Simone Balocco

Realitzat a:

- Fakulteta za računalništvo in informatiko.
Univerza v Ljubljani.

- Departament de Matemàtica Aplicada i Anàlisi.
Universitat de Barcelona.

Barcelona, 4 de juliol de 2014

INDEX

- 1. INTRODUCTION.....1**
 - 1.1 Project and motivation.....1
 - 1.2 Summary.....1
 - 1.3 Goals.....1

- 2. PLANIFICATION.....2**
 - 2.1 State of the art.....3
 - 2.1.1 Two large cities.....3
 - 2.1.2 Two small cities.....4
 - 2.1.3 Differences and similarities.....5
 - 2.1.4 Comparison between submit data / register forms.....7
 - 2.2 Requirements and functionalities.....8
 - 2.2.1 Definition of the open-data portal.....8
 - 2.2.2 Definition of the register form of the open-data portal.....8
 - 2.2.3 Definition of the upload / submit data form of the open-data portal.....8
 - 2.2.4 Definition of the add event form of the open-data portal.....8
 - 2.3 Task planning and project phases.....9
 - 2.3.1 Initial phase.....9
 - 2.3.2 Design phase.....10
 - 2.3.3 Development phase.....10
 - 2.3.4 Final phase.....10

- 3. ANALYSIS AND DESIGN.....11**
 - 3.1 Tools and technologies.....11
 - 3.1.1 Programming languages.....11
 - 3.1.1.1 PHP.....11
 - 3.1.1.2 HTML.....11
 - 3.1.1.3 CSS.....12
 - 3.1.1.4 JavaScript.....12
 - 3.1.2 Frameworks and other software tools.....12

3.1.2.1 MySQL.....	12
3.1.2.2 LAMP.....	12
3.1.2.3 Apache.....	13
3.1.2.4 phpMyAdmin.....	13
3.1.2.5 Bootstrap.....	13
3.1.2.6 Symfony2.....	14
3.1.2.7 Sonata Project.....	16
3.1.3 Applications.....	16
3.1.3.1 Linux.....	16
3.1.3.2 Adobe Photoshop CS6.....	16
3.1.3.3 PhpStorm.....	16
3.2 Use cases.....	17
3.2.1 Anonymous user.....	18
3.2.2 Registered user.....	18
3.2.3 Administrator.....	18
3.3 Sitemap.....	19
3.4 Graphic design.....	21
3.5 Database architecture.....	24
4. DEVELOPMENT.....	26
4.1 Installation and first configuration of the framework.....	26
4.2 Install Bootstrap inside Symfony2.....	30
4.3 Creation and configuration of the pages of the portal.....	31
4.4 Configuration of permissions.....	32
4.5 Templates.....	32
4.6 Database and entities.....	33
4.7 Layout of the frontend.....	34
4.8 Security of the application.....	34
4.9 Administration section.....	37
4.10 Move and install this application to another server.....	38
5. CONCLUSIONS AND FUTURE WORK.....	40
6. BIBLIOGRAPHY.....	42

7. ANNEX	44
7.1 Technical manual.....	44
7.1.1 XAMPP (LAMP).....	44
7.1.2 phpMyAdmin.....	46
7.2 User manual.....	47
7.2.1 Anonymous user.....	47
7.2.2 Registered user.....	53
7.3 Administrator manual.....	57

1. INTRODUCTION

1.1 Project and motivation

The main idea of this final project is to create an open-data portal (a public and free repository/catalog) where would be possible to upload all kind of data related with the town like (for example) photos, descriptions of roads (or mountain roads), results of the sports teams, old photos, data of the visitors of the museum, maps, economic data, PDF's of the local newspaper, GPS coordinates of interesting places, etc. etc.

It is believed that it could be an interesting final project topic, because it requires an intense software development, go into detail about web technologies and also because in the near future it could be used as a real application.

1.2 Summary

This project has been carried at the Univerza v Ljubljani (University of Ljubljana - Slovenia). It consists of an open-data portal design and development for any type of information related normally with a town, city, country, government, etc.

Open Data is a global initiative where the idea is that certain data should be freely available to everyone to use, reuse and redistribute as they wish, without restrictions from copyright, patents or other mechanisms of control. The goals are similar to those of other "Open" movements such as open source, open hardware, open content, open access, etc..

The basis of Open Data are transparency, collaboration, participation and generation of economic wealth.

In the following pages, the requirements and objectives of the project are exposed. After that, a task planning and specification of the project phases are showed, and then an explanation of the tools and technologies that have been used for the proper development of the portal are presented. The analysis and design of the site with the main points of the development are explained. Finally, there are exposed the conclusions, future work that could be done on the application and the annex with a technical, user and administrator manual of the portal.

1.3 Goals

The main goal is the analysis, design and implementation of the open-data portal for the management of the public and free repository/catalog.

The main tasks required to achieve this goal are:

- Design of the graphic web interface.
- Design of the database.
- Creation of the login and registration.
- Creation of the interface to upload data and add events.
- Creation of the interface of administration to manage the users, data, events, etc.
- Languages of the portal: English and Catalan

And more technically:

- Look for the most suitable languages to code the website.
- Look for the most suitable type of database.
- Evaluate if use a web framework could be suitable.
- Prepare a suitable hosting to implement and to execute the website.
- Create a user friendly graphic design for the website.

2. PLANIFICATION

2.1 State of the art

There are already a lot of open data portals around the world and to know how are working and their functionalities was considered recommendable to investigate about them. For this reason, was decided to find four existing data portals (two of large cities and two of small cities) and make a comparison to define a list of functionalities.

2.1.1 Two large cities

- **BARCELONA (SPA - Catalonia - 1,620,943 (city limits) / 3,218,071 (Barcelona area)) [2]**

In the menu of the top there are some links with information like:

- “What’s open data” (information of open data)
- “Datasets” (It can be searched the data and also filter by topics, frequency, formats, tags)
- “Dataviewer” (Data displayed in a map with the different districts of the city)
- “Highlights” (Last added, last updates, most downloaded, most viewed)
- “Blog” (News, opened in another tab)
- “Links” (Links related with the City Council of Barcelona and links of public open data initiatives worldwide)
- “eGovernment” (Some digital services of the City Council, opened in another tab)

Furthermore, the portal is divided with the next parts:

- Find open data
 - Search part (with the option of advanced search)
- Community
 - Blog (It explains all the Barcelona's Open Data news)
 - Linkedin group (Connect with professionals from Barcelona interested in open data)
 - Discussion forums (Join the discussion forum about open data)
 - User’s club (Join the Barcelona's Open Data users club and receive information by e-mail)

- FAQ's & help (Questions about this site?)
- Open data catalog
 - Administration (Legislation and justice, Public sector, Treasury)
 - Economy and Business (Employment, Science and technology, Trade)
 - Urban environment (Culture and Leisure, Environment, Sport, Tourism, Transport)
 - Population (Demography, Education, Society and Welfare)
 - Territory (Housing, Town planning and Infrastructures)
- Help us to improve
 - New data (What data is thought should be opened?)
 - I found a bug (If is thought that there are errors in the data)
 - I have an idea (Want to build or have already built an application using open data?)

There are also links to Data using, Accessibility and Sitemap.

- CHICAGO (USA - Chicago - 2,714,856 (city limits) / 9,522,434 (Chicago area)) [3]

In the menu of the top there are some links like:

Home, about, help, developers, terms of use, city of Chicago and a link for Sign up / Sign in.

On the left side, there is a column where it can be chosen the different types of files:

Charts, maps, calendars, filtered views, external datasets, files and documents, forms and APIs.

Also the categories:

Administration & finance, buildings, community & economic development, education and environment & sustainable development.

And finally there is in the center the section of "Search & Browse Datasets and Views" where it can be seen the links to the data (title, category, topics, description, popularity-num of views, type).

2.1.2 Two small cities

- FLORENCE (ITA - Tuscany - 367,796) [4]

Firstly it can be seen a search engine and then a menu similar to the other portals (Home, dataset, information, linked data, statistics, etc.).

Like in the portal of Chicago, there are four sections linked to news, dataset, app and tag cloud.

Then, on the left side, it can be seen a column with the links to each category and in the center,

the links to the data (it can be sorted by type of file and by more recent, more visited and more voted)

Finally, on the bottom, there are different sections like, links for collaborate, tweets with the hashtag #opendatafirenze and links to useful information of the city council of Firenze (Comune di Firenze).

- ASHEVILLE (USA - North Carolina - 83,393 (city limits) / 424,858 (Asheville area)) [5]

This portal has on the top a link for the Register / Login and below this, a search engine for search data.

On the left side, there is a column where it can be selected the category or all data and in the center there are the recent additions and then there is a link for making a submission of data.

The form of register has the next fields: Username, e-mail, password, first name, last name, organization and captcha.

2.1.3 Differences and similarities

- Chicago and Asheville have a system of register/login.
- All the portals have a section (usually a column on the left side) for selecting the different categories of data.
- Usually the categories to choose are the same or similar.
- All the portals have a search engine.
- All the portals have a legal advice or terms of use.
- Usually there is a menu on the top of the website, where different sections can be chosen by the user (Usually these sections are the same or similar).
- Some of the portals have connection with some social networks. The Barcelona's portal has also a blog, FAQ's & help and some options to collaborate (New data, I found a bug, I have an idea).
- Usually the links to the data are in the center of the website and is the biggest section of the website.

BARCELONA	CHICAGO	FLORENCE	ASHEVILLE
Form for uploading docs	Register / Login	Form for uploading docs	Register / Login
“Open data catalog” in the center of the portal	Left column for selecting categories	Left column for selecting categories	Left column for selecting categories
Search engine	Search engine	Search engine	Search engine
Sitemap link	Sitemap link	No sitemap link	No sitemap link
Data using	Privacy policy	Privacy / Legal note	Terms of use
Top menu for selecting sections	No top menu	Top menu for selecting sections	No top menu
Share buttons (social networks)	Share buttons (social networks)	Share buttons (social networks)	No share buttons
Links (only to the categories) of data in the center of the website	Links to data in the center of the website	Links to data in the center of the website	Links to data in the center of the website (only 3 most recent)
No possibility of rating the docs	Possibility of rating the docs	No possibility of rating the docs	Possibility of rating the docs
Possibility of comment the docs	No possibility of rating the docs	Possibility of comment the docs	Possibility of comment the docs

2.1.4 Comparison between submit data / register forms

With an asterisk (*) there are the fields that are obligatory to fill

BARCELONA (Submit Data Form)	CHICAGO (Register Form)	FLORENCE (Submit Data Form)	ASHEVILLE (Register Form)
Type of the consultation *	E-mail *	Name *	Username *
Message *	Display Name (Username) *	E-mail *	E-mail *
Attach File (Button)	Password *	Type of Company *	Password *
Name	Confirm Password *	Title *	Confirm Password *
1st Surname	Captcha	Description	First Name *
2nd Surname	Create my account (Button)	Owner	Last Name *
ID number	I already have an account (Link)	What do you plan to do with the data that you are reporting? *	Organization
E-mail *		Captcha	Captcha

The Barcelona portal has more options like Confirm E-mail (*), Year of Birth (Statistics), District of the City (Statistics), Male / Female (Statistics), Acceptation of Data Protection (Checkbox), Accept (Button) and Remove (Button).

2.2 Requirements and functionalities

- This open-data portal should have three basic views:

- Anonymous user that only wants to navigate in the portal, download documents, check the events, etc.
- Registered user that wants to navigate, download documents, etc. but also can upload documents and add events on the portal.
- Administrator. The portal should have an admin system to control the documents, validate documents, control the size and type of the files, control the events, validate events, control the users, create categories, edit users, edit categories, etc.

- The portal should have a good and usable visual interface to have an easy navigation and to charge correctly documents like, for example, photos, documents or maps.

- A good integration with the basic social networks and buttons of sharing.

- Responsive web design. Good interface in laptops, PC's, tablets, mobiles, etc.

- A good system of uploading of every document for the registered users (always previously validated the documents, events and users).

After all this research, it is possible to make a definition of the main parts of the portal:

2.2.1 Definition of the open-data portal

Register / Login, information about open data, left column to choose categories of data, top menu, links and buttons to share on social networks, central section in the Home to show the links to the data.

2.2.2 Definition of the register form of the open-data portal

E-mail (*), password (*), repeat password (*), first name, last name, register (button).

2.2.3 Definition of the upload / submit data form of the open-data portal

Category (*), attach file (*), title (*), description (*), website related, metadata, upload (button).

2.2.4 Definition of the add event form of the open-data portal

Title (*), organiser (*), date (*), place (*), description (*), link, add (button).

Note: * required fields.

2.3 Task planning and project phases

After the definition of the goals, requirements and functionalities, it can be proceed to make the analysis of the timings of learning, implementation, documentation and testing.

Following the tree model, on the following Gantt diagram (Figure 1) it can be viewed the tasks and subtasks that were realized divided among the phases of the project. As a global task, there is the testing phase, because it is a process that was realised constantly during the development. Furthermore, there are added some milestones of the meetings with the supervisor at the University of Ljubljana, Mira Trebar (normally was every two weeks).

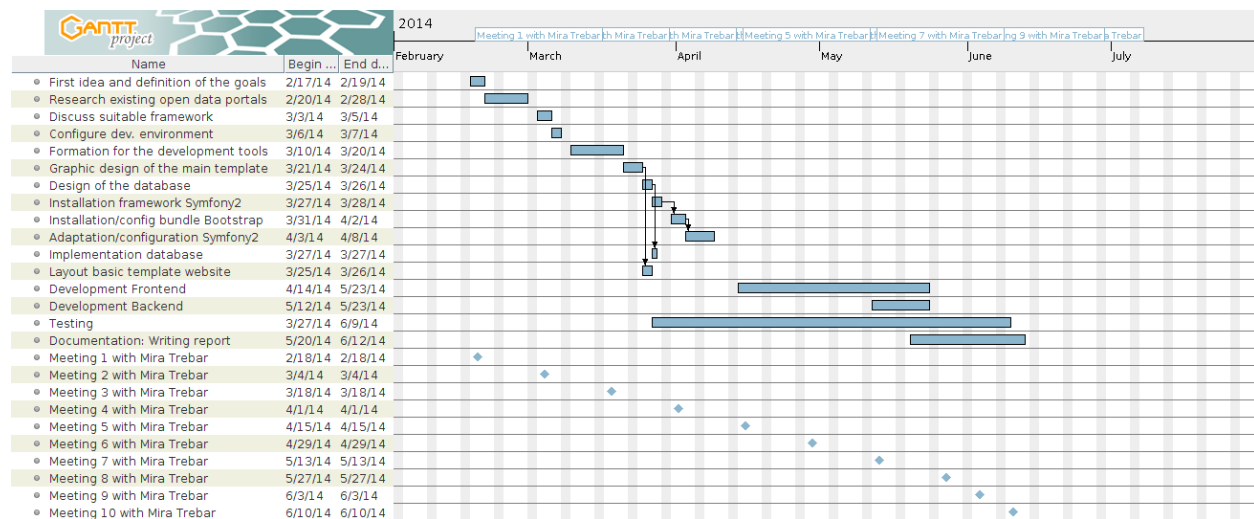


Figure 1. Gantt diagram with the task planning.

In the GANTT diagram there are different phases which is divided the project. In detail, these phases are the following:

2.3.1 Initial phase

Here are defined the first goals of the project and the tools that will be needed to achieve them.

- Creation of the first idea of the project and definition of the goals.
- Research (State of the art) on existing open data portals (And writing of this section in the report).
- Analyse if it is suitable using a Framework or not.
- Prepare the development environment (Local server LAMP, browsers, MySQL Workbench, PhpStorm, etc.)
- Read manuals and learn how to use the chosen tools for the development.

2.3.2 Design phase

- Graphic design of the main page (template) of the website, following best practises for the web.
- Design of the database (MySQL).

2.3.3 Development phase

- Installation and first configuration of the framework Symfony2.
- Installation and configuration of the bundle Bootstrap v3.1.1 inside the framework Symfony2.
- Adaptation of the framework to the needs and configuration of the parameters, security, user system, etc.
- Implementation of the database with PhpMyAdmin.
- Layout of the basic template of the website.
- Development of the different pages of the website (Frontend).
- Development of the administration part (Backend).

2.3.4 Final phase

- Testing of the website done during the development by the developer, trying the three existing roles (anonymous user, registered user and administrator).
- Documentation: Writing of the report with all the needed information.

3. ANALYSIS AND DESIGN

After the research and the definition of the goals and requirements, the next step is the phase of analysis and design.

3.1 Tools and technologies

3.1.1 Programming languages

3.1.1.1 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.

The main points to use PHP are:

- It is Open Source.
- This code is executed on the server-side, so it is invisible to the browser and client, and it is secure and reliable.
- It is a robust and flexible system.
- PHP code can be simply embedded into with HTML code, or it can be used in combination with various templating engines and web frameworks.
- It is one of the most popular languages with a very important community, so it is quite easy to find documentation, code examples and solutions to the problems that could be during the development process.

3.1.1.2 HTML

HTML or HyperText Markup Language is the standard markup language used to create web pages. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

It is basic to use HTML on the web development because is the language that finally the browser will interpretate. I use the last version of this language, HTML5, which has some new characteristics, elements and attributes.

3.1.1.3 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language like HTML.

I used CSS because:

- Is easy to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts.
- Improve content accessibility
- Provide more flexibility and control in the specification of presentation characteristics
- Enable multiple pages to share formatting
- Reduce complexity and repetition in the structural content

3.1.1.4 JavaScript

JavaScript (JS) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed.

It allows to make changes and operations without reloading the website, which improve a lot the interactivity with the user and the usability of the website.

I also use jQuery (cross-platform JavaScript library), the most popular JavaScript library in use today, designed to simplify the client-side scripting of HTML. Is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications.

3.1.2 Frameworks and other software tools

3.1.2.1 MySQL

MySQL is the world's second most widely used open-source relational database management system. Is developed as open-source with two types of licence: GNU General Public License and, on the other hand, as under a variety of proprietary agreements.

I decided to use MySQL because:

- It is Open Source.
- It uses SQL language, which is the most standardized language to access a database.
- It exists a lot of information to use correctly this management system.

3.1.2.2 LAMP

The acronym LAMP refers to first letters of the four components of a solution stack, composed entirely of free and open-source software, suitable for building high-availability heavy-duty dynamic web sites, and capable of serving tens of thousands of requests simultaneously. The meaning of the LAMP acronym depends on which specific components are used as part of the actual bundle: There are variants like WAMP (for Windows), but in our case the meaning is Linux, Apache, MySQL, PHP. Also there is incorporated the component PhpMyAdmin.

It has been used for the local development of the project.

It could be done some configurations of security directly on the LAMP server, but is preferable to do it on the Symfony framework, because this way will allow to move easily the application, from one server to another server. The security of the application is explained on the development chapter.

3.1.2.3 Apache

Apache is a web server, the most popular in use. As of June 2013, Apache was estimated to serve 54.2% of all active websites and 53.3% of the top servers across all domains. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Released under the Apache License, Apache is open-source software. A wide variety of features is supported, and many of them are implemented as compiled modules which extend the core functionality of Apache.

3.1.2.4 phpMyAdmin

phpMyAdmin is a free and open source tool written in PHP intended to handle the administration of MySQL with the use of a web browser. It can perform various tasks such as creating, modifying or deleting databases, tables, fields or rows; executing SQL statements; or managing users and permissions all with a web user interface. Is easy and fast to use and there is a lot of documentation.

3.1.2.5 Bootstrap

Bootstrap [10] is a free collection of tools for creating websites and web applications. It contains

HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

After the analysis of some options, I decided to use Bootstrap for the frontend because some reasons:

- Bootstrap is compatible with the latest versions of all major browsers.
- Since version 2.0 it also supports responsive web design. This means the layout of web pages adjusts dynamically, taking into account the characteristics of the device used (desktop, tablet, mobile phone).
- Starting with version 3.0, Bootstrap adopted a mobile first design philosophy, emphasizing responsive design by default.
- Bootstrap is open source and available on GitHub. Developers are encouraged to participate in the project and make their own contributions to the platform.
- Bootstrap is modular and consists essentially of a series of LESS stylesheets that implement the various components of the toolkit. A stylesheet called bootstrap.less includes the components stylesheets. Developers can adapt the Bootstrap file itself, selecting the components they wish to use in their project.
- Bootstrap provides a set of stylesheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements.
- Is easy to customize the code and components of the tool.

3.1.2.6 Symfony2

Symfony [11][12] is a PHP web application framework for MVC (Model–view–controller) applications. Symfony is free software and released under the MIT license.

After the assessment of using or not a framework for the project, I decided to use a framework because it could be a good chance to understand and try how it works a MVC framework.

I chose Symfony2 because some technical but also some personal reasons:

- Personally, there were some people in my personal/professional environment that recommended me to use Symfony2 because there was a lot of documentation and information on the Internet, was a good option to start in the framework development and because was very popular and demanded in the labor market.
- Technical, Symfony2 aims to speed up the creation and maintenance of web applications and to replace repetitive coding tasks.
- Is aimed at building robust applications in an enterprise context, and aims to give

developers full control over the configuration.

- From the directory structure to the foreign libraries, almost everything can be customized.
- Symfony makes heavy use of existing PHP open-source projects as part of the framework, including:
 - Doctrine as Object Relational Mapping layers
 - PDO Database abstraction layer
 - Twig, a templating engine
 - Swift Mailer, an e-mail library
- Is used in several important companies
- There are good manuals and documentation for this framework

Component interactions

In addition to dividing the application into three kinds of components (Figure 2), the model–view–controller design defines the interactions between them.

- A **controller** can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., by scrolling through a document).
- A **model** notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. In some cases an MVC implementation might instead be "passive," so that other components must poll the model for updates rather than being notified.
- A **view** requests information from the model that it needs for generating an output representation to the user.

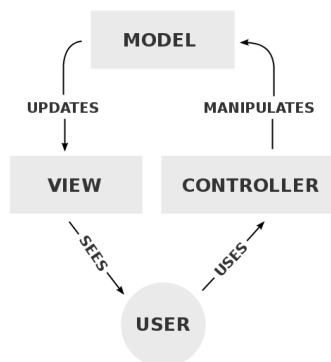


Figure 2. Diagram with the interactions between components of the MVC Framework Symfony2

3.1.2.7 Sonata Project

The Sonata Project [13] is an open source set of bundles which are built on top of Symfony2. It's majorly based on the community work around the globe. In my case I used the AdminBundle to create the administration part of the portal because is easy to integrate and adapt inside Symfony2.

* Bundle: a group of files, PHP classes, methods, etc. For example, in this application, is followed the logic Frontend-Backend, so it has been created a bundle FrontendBundle and another one BackendBundle. In the chapter 4.1 (Installation and first configuration of the framework) is explained in more details.

3.1.3 Applications

3.1.3.1 Linux

Linux is a Unix-like and POSIX-compliant computer operating system assembled under the model of free and open source software development and distribution. In this case the used distribution is Ubuntu.

3.1.3.2 Adobe Photoshop CS6

Adobe Photoshop is a graphics editing program, used in this project to create the graphic design of the portal and to edit some images.

3.1.3.3 PhpStorm

JetBrains PhpStorm is a commercial, cross-platform IDE for PHP. PhpStorm provides an editor for PHP, HTML and JavaScript with on-the-fly code analysis, error prevention and automated refactorings for PHP and JavaScript code. It has a MVC view for Symfony2 which is very useful for this project.

3.2 Use cases

The application will be accessible and managed as anonymous user / registered user (Frontend view) and administrator (Backend view).

3.2.1 Anonymous user

Figure 3 shows the use case of the anonymous user. This is available to the user who is simply a new visitor or already registered user, but is not logged in and can see the pages that are available on the website.

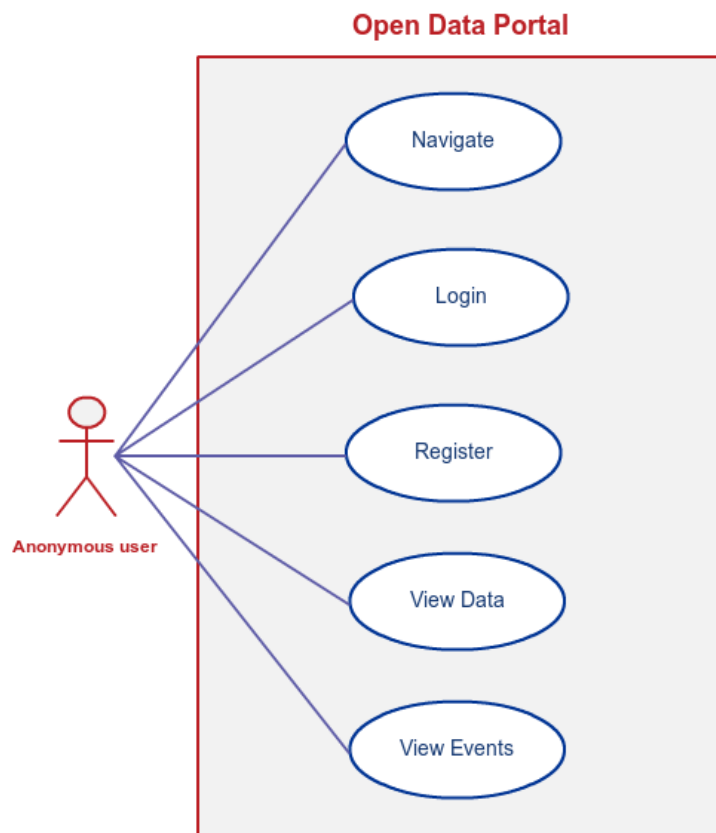


Figure 3. Use case diagram for anonymous user.

3.2.2 Registered user

In this figure (Figure 4), it can be seen the user case of the registered user. This is the case of a user that is registered and already logged in the login page with access to the pages that are exposed on the diagram.

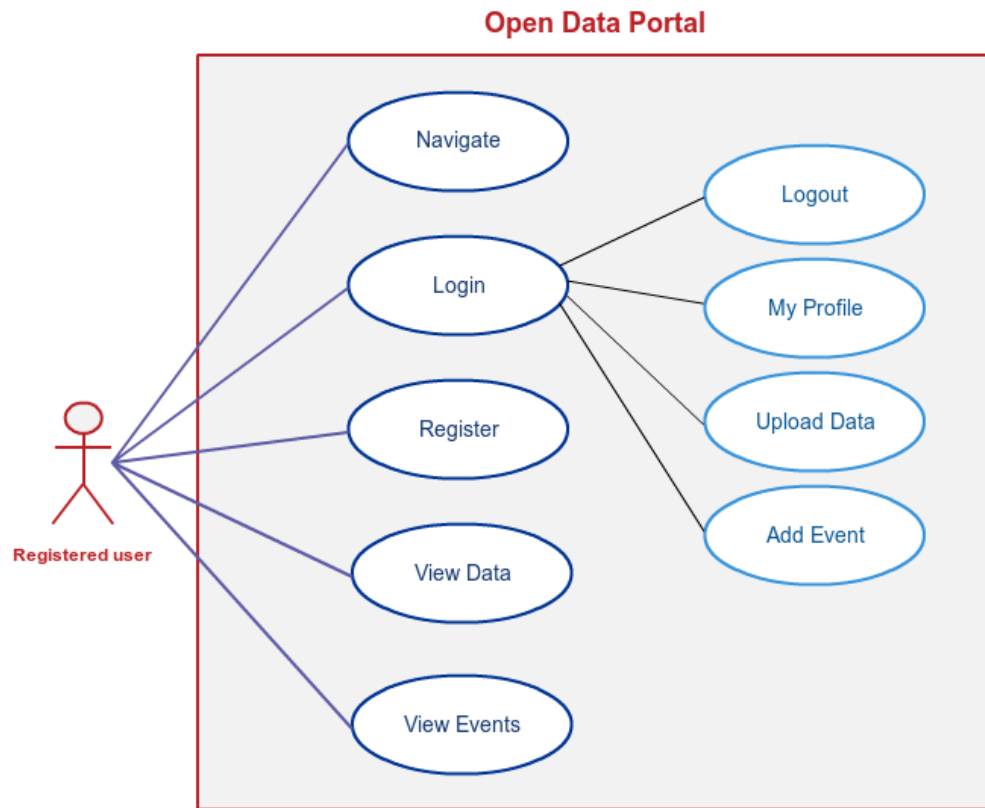


Figure 4. Use case diagram for registered user.

3.2.3 Administrator

Currently there is only one user with role of administrator in the application, but easily can be added more users with this role.

The administrator is who has more cases (Figure 5). He first needs to enter to the administration part (root_url/admin) and then he can do all the operations listed with the cases.

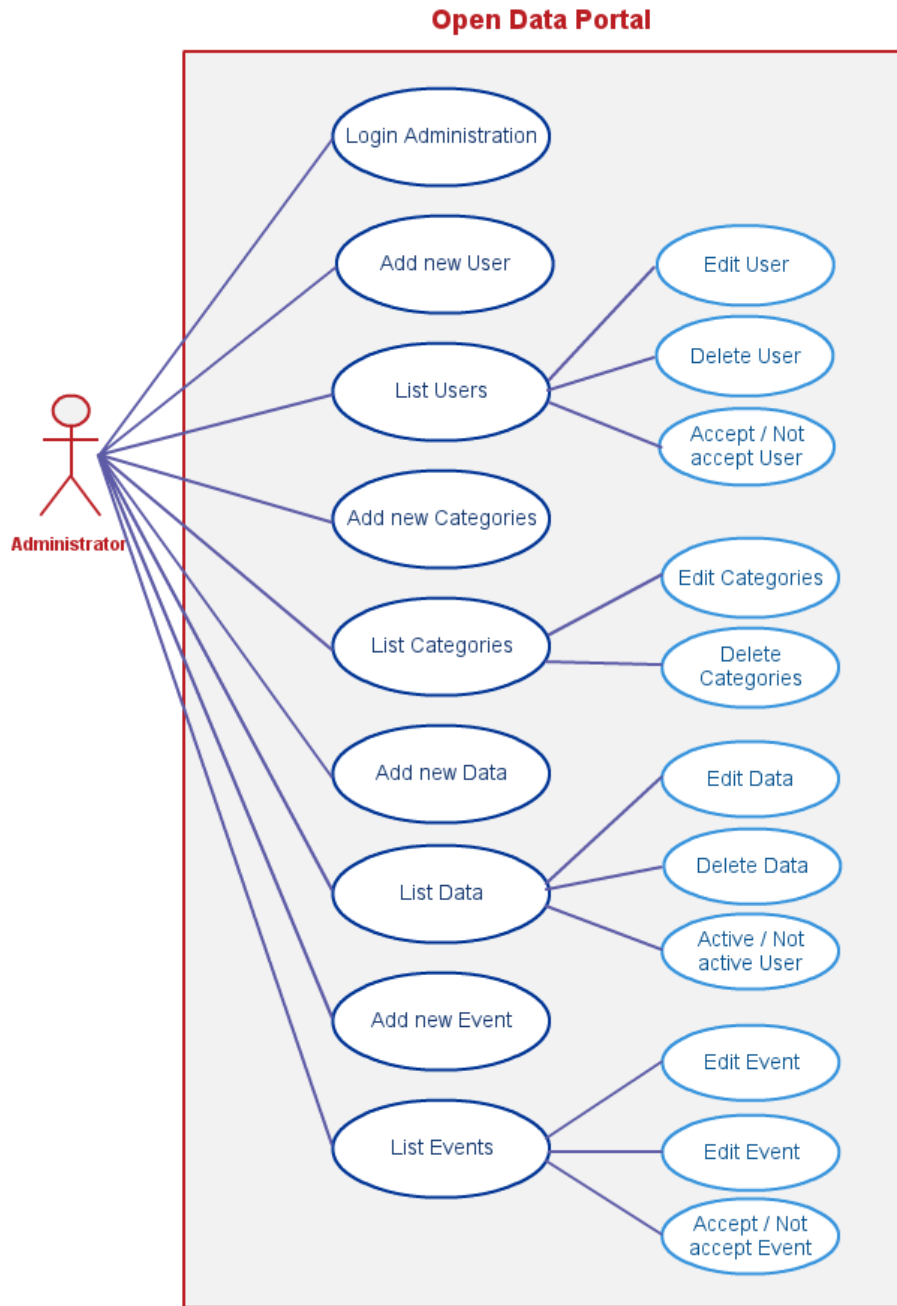


Figure 5. Use case diagram for administrator.

3.3 Sitemap

In the next figure (Figure 6), it can be seen the sitemap of the site. There are four levels of hierarchy and also there are three different colors to know which sections/subsections can be accessed by each kind of user (anonymous user, registered user, administrator).

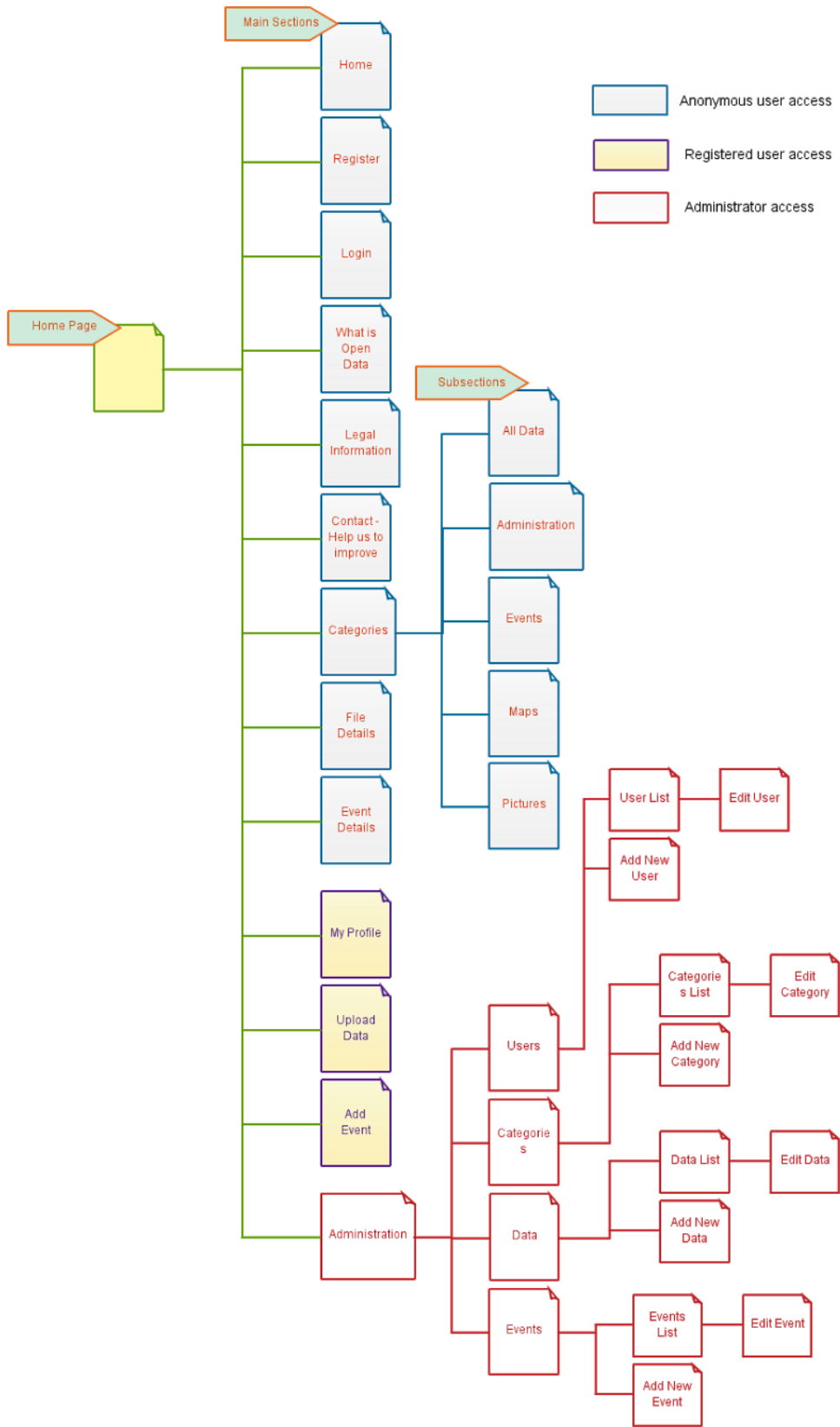


Figure 6. Sitemap of the open data portal.

3.4 Graphic design

Following the principles of design [7][8] and also comparing with other open data portals I made (before start with the development) three proposals of the design of the main page of the website. The first one (Figure 7) is mostly showing the structure of the website. In this one were not important the colors, what was most important was to show the structure of the portal and the different sections.

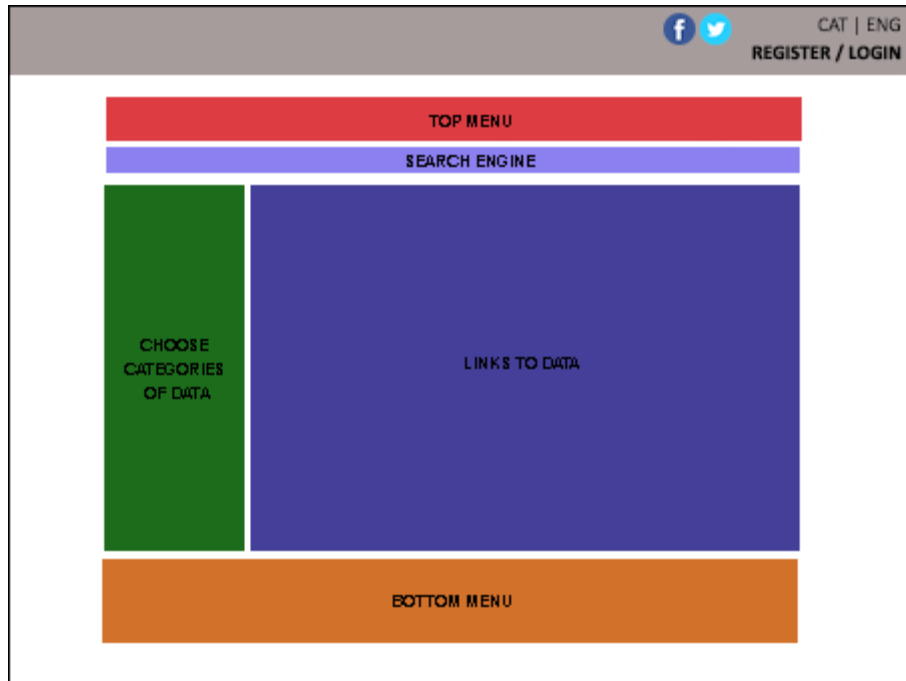


Figure 7. Initial structural 1st graphic design of the planned portal.

In the second graphic design proposal (Figure 8), was made already a more defined version of the page, also following the colors that Bootstrap is normally using.

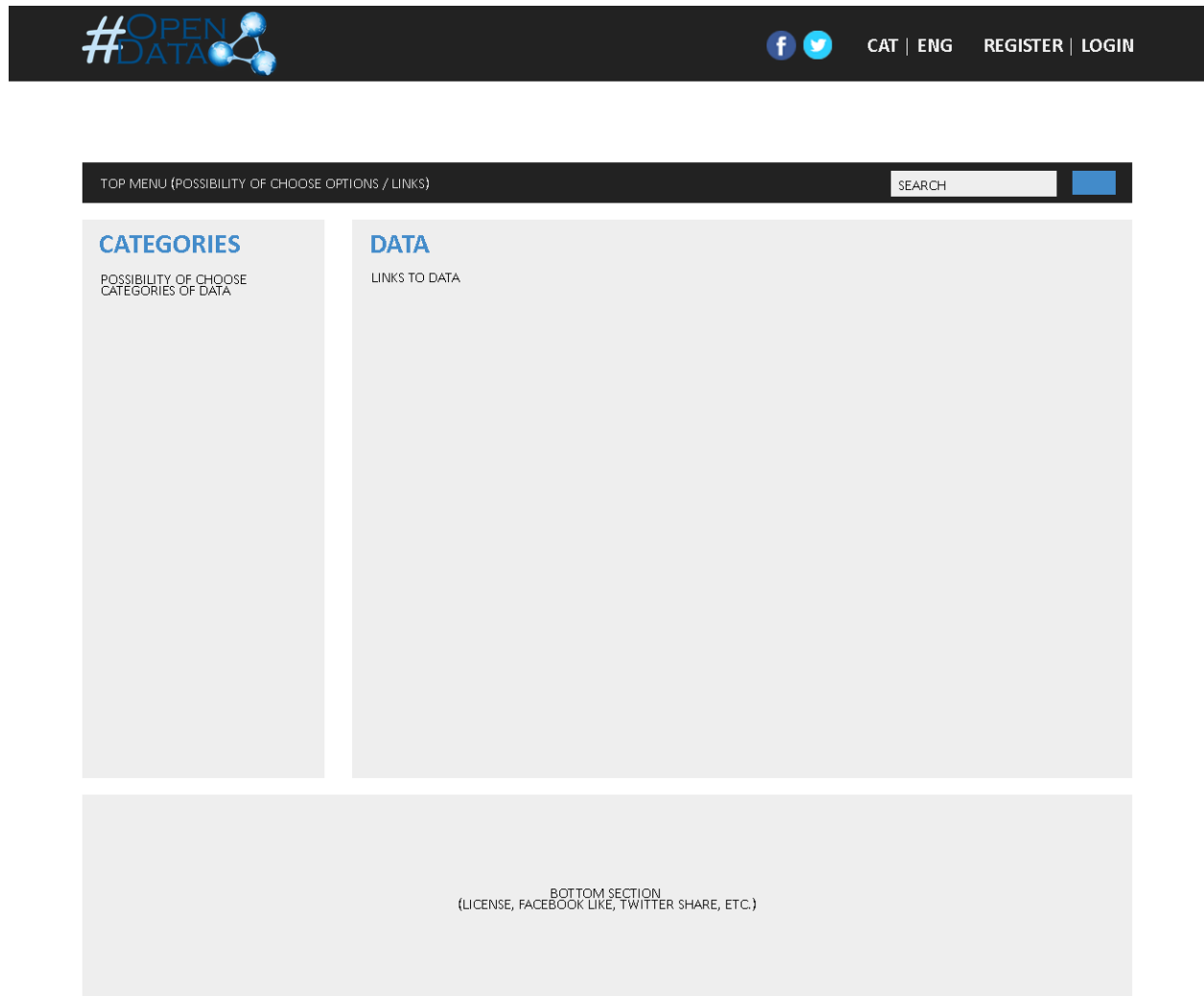


Figure 8. Initial 2nd graphic design of the planned portal.

And finally the third version (Figure 9) is the version closest to the final result of the application, specifying each section of the portal, its content and a proper usability and user-friendliness.

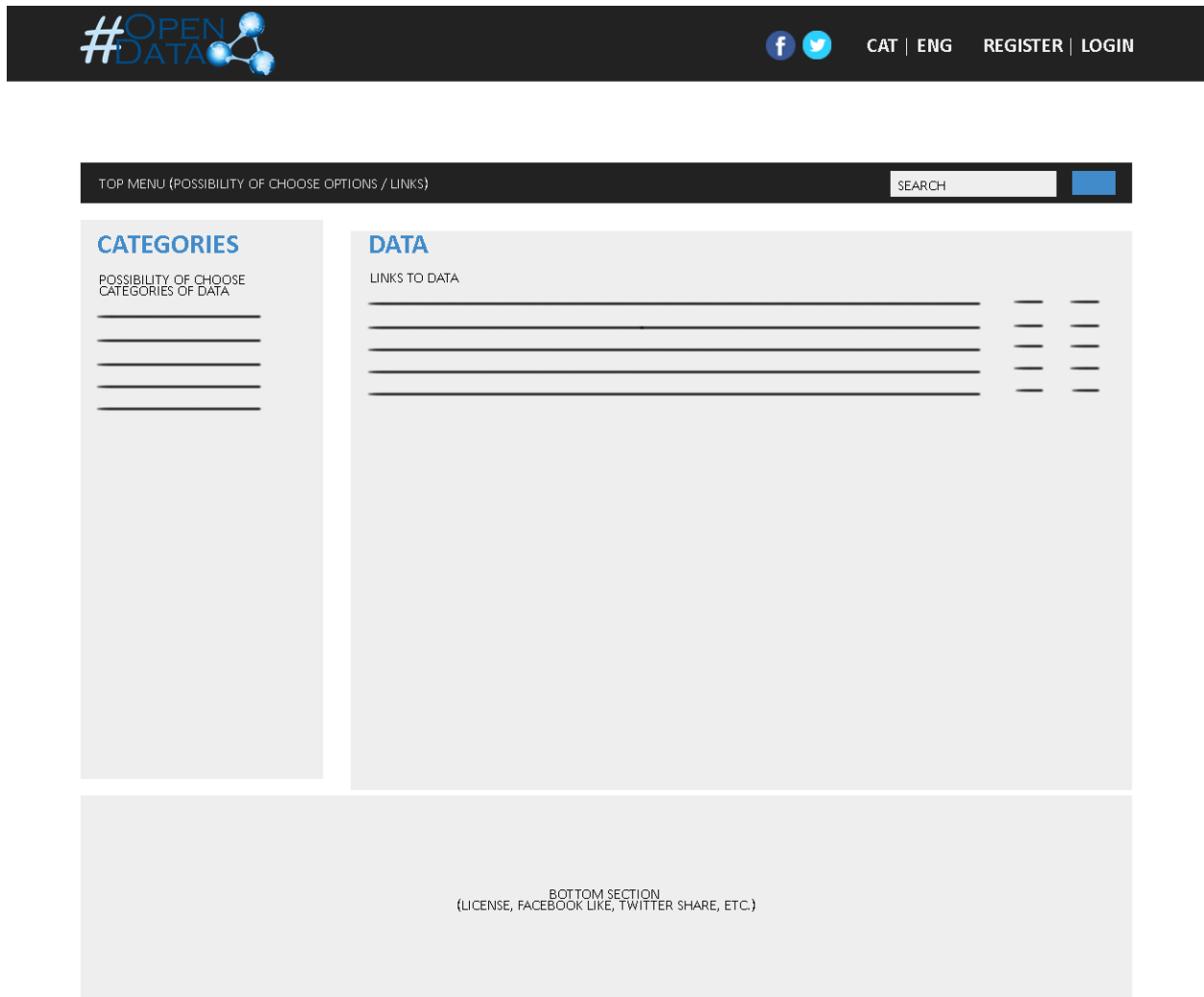


Figure 9. Initial 3rd graphic design of the planned portal.

3.5 Database architecture

The database consists in four tables (Figure 10), where there is saved all the data and information of the categories, users, events and data of the application. There are fields of type varchar (text), int (integer), datetime (to save date+time) and tinyint (boolean).

The '**data**' table is the table where the information of the files that are uploaded is saved. This fields are created at the moment that the 'Upload' button of the upload data form is clicked and there are no errors. Immediately, the method of the controller sends the information to the database saving the following fields:

- id: identifies each file and it has the properties 'auto increment' and 'unique'.
- idCategory: is related with the 'id' field of the table 'categories' and is needed to relate the file with its category.
- idUser: is related with the 'id' field of the table 'users' and is needed to relate the file with the user who uploaded the file.
- file: name of the file (while the file is upload to the directory web/uploads/).
- web: link of website that it has been entered in the form.
- type: extension of the file (getting the extension of the file using the php function pathinfo()).
- size: ize of the file in Bytes.
- dateTime: date+time of uploading of the file.
- nameEng: title of the file.
- nameCat: prepared in case of translating the portal to Catalan.
- nameSpa: prepared in case of translating the portal to Spanish.
- message: description of the file.
- metadata: metadata/tags written in the form.
- views: count the number of views of the file (when a user enter to the page of the file it is one more view).
- accepted: 0 if the file is not validated, 1 if is validated by the administrator.

The '**events**' table is the table where the information of the events that are added is saved, with the following fields:

- id: identifies each event and it has the properties 'auto increment' and 'unique'.
- idUser: is related with the 'id' field of the table 'users' and is needed to relate the file with the user who added the event.
- titleEng: title of the file.

- titleCat: prepared in case of translating the portal to Catalan.
- titleSpa: prepared in case of translating the portal to Spanish.
- organiser: organiser of the event.
- date: date of the event.
- place: place of the event.
- link: link of website that it has been entered in the form.
- description: description of the file.
- dateTime: date+time of uploading of the file.
- views: count the number of views of the event (when a user enter to the page of the event it is one more view).
- accepted: 0 if the file is not validated, 1 if is validated by the administrator.

In the table 'categories' simply there is saved the name/title of each category related with its 'id'.

In the table 'users' is where the user information is saved when a new user is registered and it has the following fields:

- id: identifies each user and it has the properties 'auto increment' and 'unique'.
- email: email of the user, is the field that will be used for the login.
- password: password of the user.
- firstName: first name of the user.
- lastName: last name of the user.
- dateTime: date+time of uploading of the file.
- lastConnection: date+time of the last connection of the user.
- note: this a field that can be used by the administrator to write some notes about the user (is only visible in the administration part).
- active: 0 if the file is not validated, 1 if is validated by the administrator.

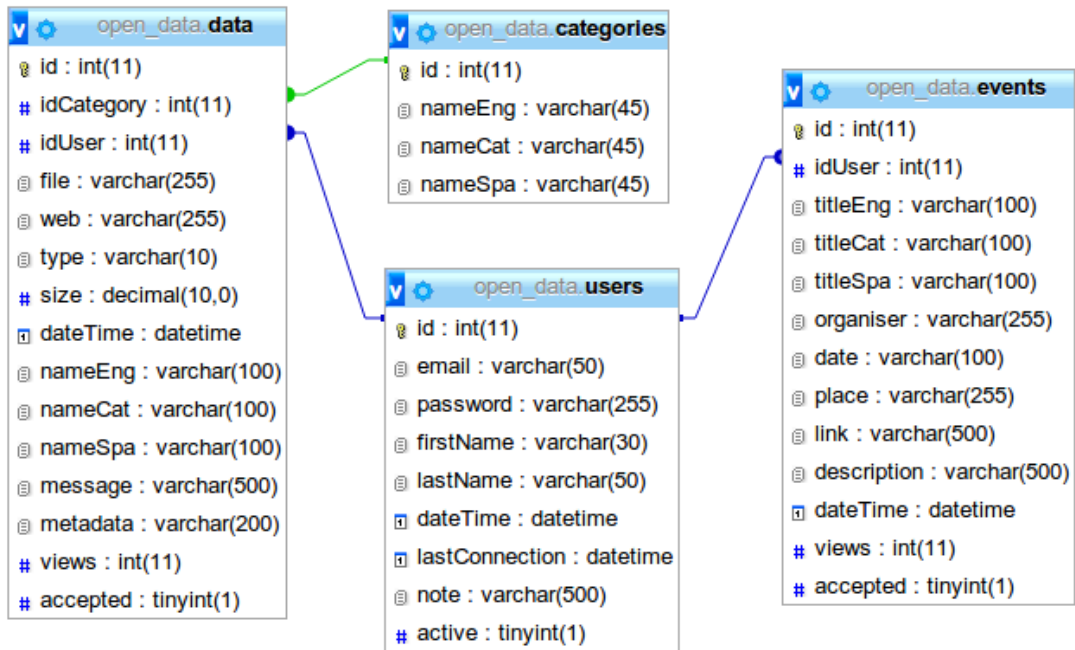


Figure 10. Architecture of the MySQL database of the portal.

4. DEVELOPMENT

In this chapter are explained which are the basics of the process of installation of the application and other interesting details about the development.

4.1 Installation and first configuration of the framework

To install the framework Symfony2, first of all has to be download at the official website (<http://symfony.com/download>). Is preferable, don't choose the file without vendors, because with the vendors version, the packet is already ready to start with the work.

So the name of the packet, should be similar to *Symfony_Standard_Vendors_2.X.XX.tgz* or *Symfony_Standard_Vendors_2.X.XX.zip*.

When the file is downloaded, the next step is to unzip the packet in the folder where is wanted to install the framework (should be inside the /htdocs folder of the LAMP if is wanted to develop it in a local server).

After this steps, it can be checked if the environment satisfies the requirements with the terminal:

```
$ php app/check.php
```

The script check.php will show a list of mandatory requirements and a list of optional checks. It should be sure that, at least, the application is confirming the list of mandatory requirements.

Then already it can be used the following instruction to check if Symfony2 is properly installed.

```
$ php app/console
```

This command must be properly executed, because is mandatory don't have any problem with the installation of the framework.

When the application is properly installed (if not, must be fixed all the problems), the application can be configured accessing to the config.php page of our app (Figure 11). For example:

```
http://localhost/Code/config.php
```

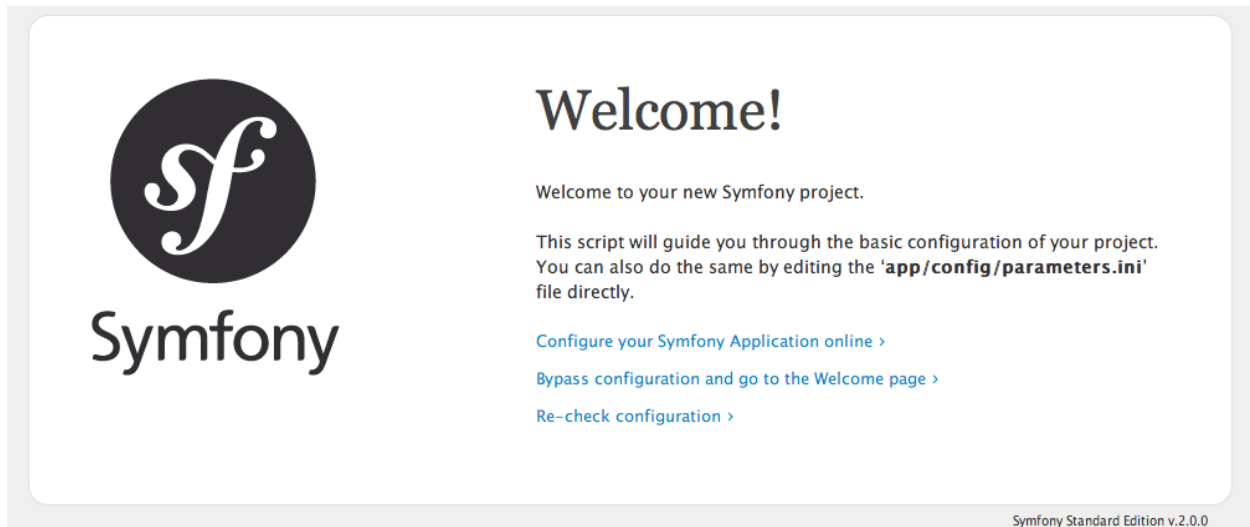



Figure 11. Welcome page of Symfony2 (config.php).

Click on the option “Configure your Symfony Application online”.

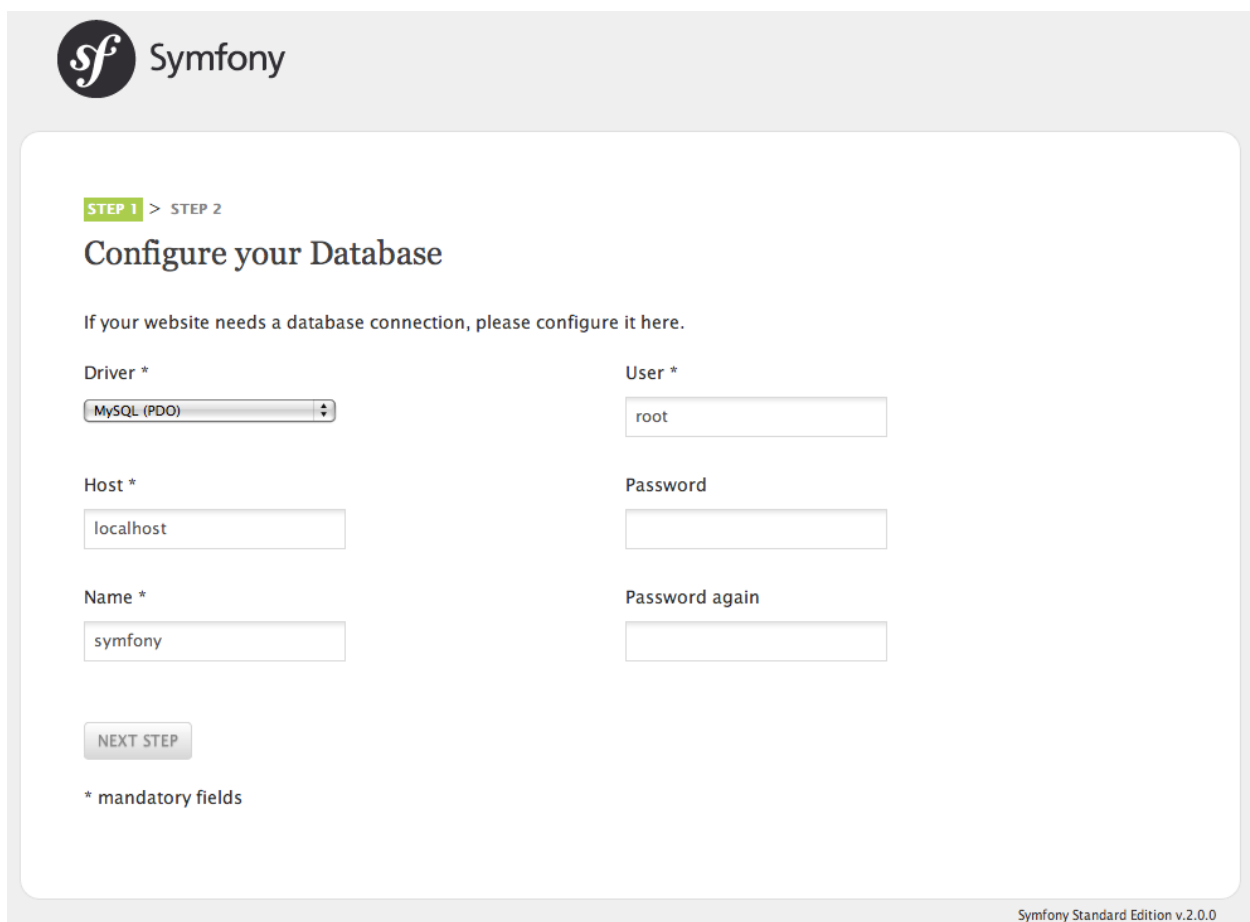


Figure 12. Configuration of the database of the application.

In this step (Figure 12), should be entered the information of the database, that previously has to be created in the server using the phpMyAdmin application (This also can be done manually editing the file `app/config/parameters.yml`).

After the installation, is recommendable to update (or at least check new updates) the framework and libraries. The command is (the instruction is 'install' but is used for install and also update):

```
$ php bin/vendors install
```

When the application is ready, updated and without problems is the moment to start the development. In this framework, the files can be grouped on the way that is thought that follow the same logical or are related, and this is called 'bundle'. For this portal, the most logical structure is to create a `FrontendBundle` and a `BackendBundle`. To make this, should be followed this instructions:

```
$ php app/console generate:bundle
```

The terminal, will make 5 questions:

- *Bundle namespace*: in this case, it will be written 'Site/FrontendBundle'. 'Site' is the folder where there are the bundles, and 'FrontendBundle' the name of the bundle (it has to finish always with the word 'Bundle').
- *Bundle name*: Is the name who will reference the bundle inside the application. It will be written 'FrontendBundle'.
- *Target directory*: It has to be pressed ENTER, by default is `/src`.
- *Configuration format*: It can be chosen YAML, XML, PHP or annotations. It will be chosen 'yml'.
- *Do you want to generate the whole directory structure?*: It has to be answered 'no'.

```
Generating the bundle code: OK
```

```
Checking that the bundle is autoloading: OK
```

```
Confirm automatic update of your Kernel [yes]?
```

```
Confirm automatic update of the Routing [yes]?
```

The answer to these last questions, should be 'yes' and after that, the bundle is properly installed. This means, that the bundle it has been generated, activated, configured properly and it has created the folders structure in the `/src` folder of the framework.

But with these steps, it is not enough to use the bundle. The bundle will be activated in the AppKernel.php file (core of the application Symfony2). It has to be added the following line to the \$bundles array of the method registerBundles().

```
new Site\FrontendBundle\FrontendBundle(),
```

And then, add the route bundle in the routing file (app/config/routing.yml).

```
FrontendBundle:
    resource: "@OfertaBundle/Resources/config/routing.yml"
    prefix: /
```

In this point, the bundle is totally created and activated. Now can be followed the same steps to create the 'BackendBundle' (where will be created the administration section of the portal).

4.2 Install Bootstrap inside Symfony2

As was explained in the chapter of tools and technologies, this project uses Bootstrap for the templates of the frontend [9]. For this reason, Bootstrap is inserted inside the framework Symfony2.

To do this, first the server has to accomplish some requirements/dependencies:

- **PHP 5.3.3** and up
- **Symfony 2.3** and up
- **Twig 1.11** and up
- **AsseticBundle 2.3** and up (optional, but highly recommended)

To install it, will be used Composer (Dependency Manager for PHP - <https://getcomposer.org/>).

In the composer.json file it has to be added the following lines:

```
{
    "require": {
        "braincrafted/bootstrap-bundle": "~2.0"
    }
}
```

And then it has to be launched the instruction, to update the packets:

```
$ php composer.phar update
```

The bundle has to be added also in the AppKernel.php file in the \$bundles array:

```
new Braincrafted\Bundle\BootstrapBundle\BraincraftedBootstrapBundle(),
```

In the project, also it has been inserted the Glyphicons of Bootstrap (customized free icons). To insert it to the folder /web (public files of the website), it has to be launched this instruction:

```
$ php app/console braincrafted:bootstrap:install
```

And again the update of the packets:

```
$ php composer.phar update
```

Now the icon fonts are copied into the web directory and also the css and js needed to work Bootstrap, so it only needs to be linked this files in each page and it can be developed the frontend using Bootstrap.

4.3 Creation and configuration of the pages of the portal

To create pages of the portal, next steps have to be followed:

For example, in the portal is needed a contact page (with a form to contact the administrator/s of the portal). The URL of this page, should be something like <http://localhost/Code/contact>.

First of all, it has to be opened the file of the routing and it has to be added the following code:

```
# app/config/routing.yml
contact:
    pattern: /contact
    defaults: { _controller: FrontendBundle:Default:contact }
```

The 1st line ('contact') configures the name of the route (freely chosen, but has to be unique).

With the 'pattern', is said the regular expression that the URL has to follow.

In 'defaults', 'FrontendBundle' is the name of the bundle where is the controller. 'Default' is the name of the class of the controller and 'contact' is the name of the method (action) that is executed inside this controller. The summary of the notation is: bundle:controller:action

This makes that when the user request a URL with the aspect of /contact, it executes the method contactAction() of the class src/Site/FrontendBundle/Controller/DefaultController.php.

All pages of the portal should be created following these steps.

4.4 Configuration of permissions

It is also very important to give 777 permissions to the folders app/cache and app/logs. These folders are important for the execution of the app, and is needed to have 777 permissions to work properly.

```
$ sudo chmod 777 -R app/cache
$ sudo chmod 777 -R app/logs
```

Then it is important to clear the cache to keep the application working well. It should be when there are strange errors, when the application is moved to another server, etc.

```
$ php app/console cache:clear --env=dev (for the development environment)
$ php app/console cache:clear --env=prod (for the production environment)
```

Note: In Symfony2 exists two environments, development and production. Development is used during all the process of development and it makes more comprovements and it shows more errors when the execution of the application. The production mode, is used when the development process is finished and it can be opened to the users.

To execute the development environment, only is needed to enter to the URL:

```
http://localhost/Code/web/app_dev.php
```

And for the production environment only:

```
http://localhost/Code/web/app.php or simply http://localhost/Code/web
```

4.5 Templates

Symfony2 uses templates for every page of the application. For example, to create the HTML page of 'contact', it has to be added this line to the contact method:

```
public function contactAction()
```

```
{
    return $this->render('FrontendBundle:Default:contact.html.twig');
}
```

According to this code, the contact template, should be in:

```
src/Site/FrontendBundle/Resources/views/Default/contact.html.twig
```

In this file, is where it will be all the HTML, JavaScript, PHP variables, etc.

Twig is a template engine for PHP: compiles templates down to plain optimized PHP code. With the code reduced at minimum, optimized and very flexible.

It is very important to know that it can not be directly written PHP code because Twig uses his own system (more information can be found at: <http://twig.sensiolabs.org/>).

4.6 Database and entities

Symfony don't manage the information accessing directly to the database. The creation, modification and delete process are made by PHP objects instead of execute SQL sentences.

This is possible because it uses an external libraries called ORM (Object-Relational Mapping). Symfony2 uses the ORM Doctrine2.

To manipulate the information of the database it uses entities. Entities are PHP objects (classes) and generally each table of the database is represented by one entity.

In our case, there are four entities (Categories.php, Data.php, Events.php, Users.php) that are inside the folder /Entity of the FrontendBundle and are responsible of the managing of the information of the database. The manipulation of the information in Doctrine2 is made by a special object called Entity Manager. To get it inside the controller, it has to be written:

```
$em = $this->getDoctrine()->getEntityManager();
```

In our project, first has been created the database using the tool phpMyAdmin and then it is possible to generate automatically the entities using the MySQL database that is already created.

This is done with the line:

```
$ php app/console doctrine:mapping:import FrontendBundle annotation
```

Importing mapping information from "default" entity manager

```
> writing .../FrontendBundle/Entity/Categories.php
> writing .../FrontendBundle/Entity/Data.php
> writing .../FrontendBundle/Entity/Events.php
> writing .../FrontendBundle/Entity/Users.php
```

First parameter of the command ('FrontendBundle') is the name of the bundle where the files of the entities will be saved. Second parameter ('annotation') tells the format to define the

information of each entity.

Then to add the getters and setters of each class, it can be done manually or it can be entered the command:

```
$ php app/console doctrine:generate:entities FrontendBundle
```

4.7 Layout of the frontend

It is very recommendable to separate the HTML/Twig code that is common in all the pages in only one layout file. This will be good for the application because it only will have to be modified one file for the common sections of the other templates, the other templates will inherit that code and will be more easy and flexible.

The layout template defines the blocks that are able to be modified by the other templates. For this, following the syntaxes, are added elements `{% block %}` that can be with the name that is preferred (like footer, header, block, article, etc.) and will include the code of the custom template. For example, if is wanted to inherit the layout in one of the pages, it should be included this line in the page:

```
{% extends '::layout.html.twig' %}
```

And then it can be possible to include the blocks that are wanted to be customized, for example:

```
{% block title %}Contact | Open Data Portal{% endblock %}
```

In this template it is also recommended to insert the links to the .css and .js files that are common in all the pages.

Because it is a template that could be used by all the bundles is recommended to put it in the directory `app/Resources/views/layout.html.twig`

After this basic steps of templates and layouts, it is possible to develop more easily new static pages, dynamic pages, forms, etc.

4.8 Security of the application

Symfony2 is a framework prepared with a strong security. It can be easily configured in the `security.yml` file (`app/config/security.yml`).

For example, this code of the `security.yml` file, is telling to the application that the password of the

login has to match with the password field of the table Users of the database and it has to be codified with the algorithm SHA-512 and encoded in base 64 bits.

```
encoders:  
    Symfony\Component\Security\Core\User\User: { algorithm: sha512, iterations: 0,  
encode_as_base64: true }  
    Site\FrontendBundle\Entity\Users: { algorithm: sha512, iterations: 0, encode_as_base64: true }
```

And for the username field of the login, is it said that the property that will work as a 'username' of the login it will be the 'email' field of the table Users.

```
providers:  
    users:  
    entity: { class: Site\FrontendBundle\Entity\Users, property: email }
```

In the case of the administrators, was decided to use the memory system of the framework instead of a possible Administrators entity, just to try a different system of login. In this case, it is said to the name, password (SHA-512 64 bits) and role of the user.

```
providers:  
    administrators:  
    memory:  
        users:  
        - { name: op_admin, password:  
JOWZthDW4V7BY+d21XS0Hq8l+70S6+SJkqfhyY+9MDzifWc8CnTAdeZtH  
UWph8YiTolgmhyAktvZWagcX9zHg==, roles: [ 'ROLE_ADMIN' ] }
```

In the firewalls section, is written how to work securely every part of the application (There are some comments in the code):

```
firewalls:  
    admin_area:  
    pattern: ^/admin  
    provider: administrators //Only administrators (defined in providers) can login in /admin  
    http_basic: ~  
  
    frontend:
```



```
pattern: ^/
anonymous: ~
form_login:
  login_path: login
  check_path: login_check
  default_target_path: /
  always_use_default_target_path: true
remember_me: //When is checked the checkbox "Remember me" of the login form
  key: opendata2014
  lifetime: 3600
logout:
  path: logout
  target: /user/login
switch_user: true
switch_user: { role: ROLE_ADMIN }
```

And finally, there is the list of paths where there are defined the roles of user that have access to each path:

```
access_control:
- { path: ^/user/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/user/register, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/user/*, roles: ROLE_USER }
- { path: ^/upload/*, roles: ROLE_USER }
- { path: ^/file/edit/*, roles: ROLE_USER }
- { path: ^/admin/*, roles: ROLE_ADMIN, requires_channel: https }
```

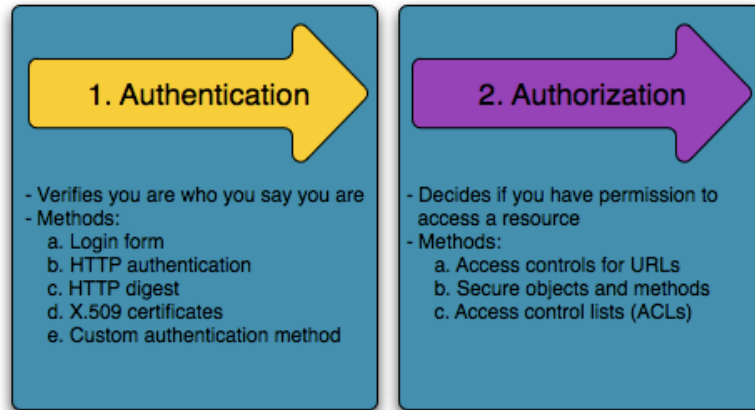


Figure 13. Process of authentication and authorization of an user in Symfony2.

There are more services of security in Symfony2. All of them are explained in detail in the section 'Security' of the manual (<http://symfony.com/doc/current/book/security.html>).

4.9 Administration section

To create the administration section, as it is explained in the chapter of tools and technologies, was used the bundle AdminBundle of the Sonata Project.

To install it, first of all it is recommendable to be installed using Composer:

```
$ php composer.phar require sonata-project/admin-bundle
```

After that, like all the bundles, it has to be added in AppKernel.php:

```
new Sonata\CoreBundle\SonataCoreBundle(),
new Sonata\BlockBundle\SonataBlockBundle(),
new Sonata\jQueryBundle\SonatajQueryBundle(),
new Knp\Bundle\MenuBundle\KnpMenuBundle(),

new Sonata\DoctrineORMAdminBundle\SonataDoctrineORMAdminBundle(),

new Sonata\AdminBundle\SonataAdminBundle(),
```

Then add these lines to the config.yml file:

```
# app/config/config.yml
sonata_block:
    default_contexts: [cms]
    blocks:
```

```
# Enable the SonataAdminBundle block
sonata.admin.block.admin_list:
  contexts: [admin]
```

And finally add these lines to the routing.yml file:

```
# app/config/routing.yml
admin:
  resource: '@SonataAdminBundle/Resources/config/routing/sonata_admin.xml'
  prefix: /admin

_sonata_admin:
  resource: .
  type: sonata_admin
  prefix: /admin
```

With these steps should be the administration section properly installed, and now it can be started the work of create the pages and configuration of the administration (In this link of the documentation is explained in detail how to start and configure this application properly http://sonata-project.org/bundles/admin/2-2/doc/reference/getting_started.html).

4.10 Move and install this application to another server

When the development process of the application is already finished, normally the application is moved to another server (usually from local server to online server).

To do this process should be followed these steps:

1. Clean the cache memory of the application.

```
$ php app/console cache:clear
$ php app/console cache:clear --env=dev
$ php app/console cache:clear --env=prod
```

2. Copy or move all the files of the application to the new server. Normally is done by FTP.
3. Create the database, user and password of the database in the new server.
4. Export the database (normally .sql file) of the old server and import it in the new server (The database of this project can be found in the /database directory of the packet with the code of this application).

5. Edit the file `app/config/parameters.yml` that is located in the new server and write the correct information to connect properly the new database that has been created in the new server.

6. Proper permissions on cache and logs directories.

```
$ sudo chmod 777 -R app/cache
```

```
$ sudo chmod 777 -R app/logs
```

7. In that moment, the application should work properly in the new server.

5. CONCLUSIONS AND FUTURE WORK

There has been a strong learning and development process behind the project, that has improved quite a lot my knowledge of web technologies and of software development.

First of all, it was planned to create this application coding directly the application as was learned in one course of the degree (Computació orientada a la web - Universitat de Barcelona), but then was thought that this project could be a great opportunity to learn new ways of development (that furthermore are very demanded in the labour market of the IT area) and was planned to create this application using a framework, even knowing that in my case was completely beginner using these kind of tools (frameworks).

In accordance with this, as was talked in previous chapters, was started to research which could be the most suitable tools to use to achieve the goals of this project and after that was read information and manuals to know how to use properly and exploit the possibilities of these tools. Finally, is believed that the result (almost all tasks and goals) that were proposed were achieved successfully and the application is ready to be used.

Even that there were some problems or limitations of time:

- First of all, there was a limitation of 20MB to upload files to the application (this was made as a condition in the Data Entity file), but this condition had to be put as a comment because was provoking problems during the process of data edition in the administration part of the portal.
- There was the idea of do the portal in English and then do a translation to the Catalan language, but finally because limitations of time was not possible to translate the portal.
- Is important to mention that during a test of the application by the mentor Mira Trebar, there was a problem to enter to the administration section of the portal. This was caused for one limitation of the free online server where the application was moved to make some tests based in the annex of this report (user and administrator manual). For this reason, is thought that is better not to use this application in free online servers, because normally there are limitations of configuration, security, etc. that can create some operating problems. The suitable servers are local servers or online servers where we have permissions completely to control the whole configuration and security of the server.

As a **future work** there are some additional functionalities that could be interesting to implement on the current application:

- Possibility of rating on the data/events by the registered users.
- Possibility of writing comments on the data/events by the registered users.
- Possibility of editing the uploaded data/event during the period of upload and before the validation of the administrator by the registered user.
- The problem previously mentioned about the limitation of 20MB to upload files is one thing that could be investigated and be fixed in the future.
- The application could be translated to other languages like Catalan or Spanish. In fact, the database has already ready the fields for these languages and only should be modified basically the forms and controllers of the application.

And generally about the future of the application:

- The website could be set for the implementation, in a near future, for my town Capellades, close to Barcelona.
- This application would be used by the citizens and if the portal was working well, even could be implemented and sold to some city halls.
- It would be also possible to create a plan for advertising to get benefits of the portal (according to me, Google AdSense could be the best option).
- To advertise the project, I think that nowadays the best option would be using the social networks (Facebook, Twitter basically).

6. BIBLIOGRAPHY

- [1] The World Bank Group. *Open Data Essentials*. [online]. [Accessed on: February 2014]. Available on: <<http://data.worldbank.org/about/open-government-data-toolkit/knowledge-repository>>
- [2] Ajuntament de Barcelona. *Open Data de l'Ajuntament de Barcelona*. [online]. [Accessed on: February 2014]. Available on: <<http://opendata.bcn.cat/opendata/en>>
- [3] The City of Chicago. *City of Chicago Data Portal*. [online]. [Accessed on: February 2014]. Available on: <<https://data.cityofchicago.org/>>
- [4] Comune di Firenze. *OpenData Comune di Firenze - I dati aperti del Comune di Firenze*. [online]. [Accessed on: February 2014]. Available on: <<http://opendata.comune.fi.it/>>
- [5] The City of Asheville. *OpenDataAsheville - Connecting People With Data*. [online]. [Accessed on: February 2014]. Available on: <<http://opendatacatalog.ashevillenc.gov/>>
- [6] CASTELEYN, Sven; DANIEL, Florian; DOLOG, Peter; MATERA, Maristella. *Engineering Web Applications*. 1st Edition. Berlin: Springer, 2009. ISBN 978-3-540-92200-1.
- [7] Jason Beard. *The Principles of Beautiful Web Design*. [online]. 2nd edition. Canada: Kelly Steele, January 2007, November 2010 [Accessed on: March 2014]. Available on: <<http://it-ebooks.info/read/2626/>>
- [8] Google Developers. *Web fundamentals - Best practices for modern web development*. [online]. [Accessed on: April 2014]. Available on: <<http://developers.google.com/web/fundamentals/>>
- [9] Florian Eckerstorfer. *Braincrafted Bootstrap Bundle*. [online]. [Accessed on: April 2014]. Available on: <<http://bootstrap.braincrafted.com/getting-started.html>>
- [10] Mark Otto; Jacob Thornton. *Bootstrap*. [online]. [Accessed on: April 2014]. Available on: <<http://getbootstrap.com>>
- [11] SensioLabs. *Symfony2*. [online]. [Accessed on: April 2014]. Available on: <<http://symfony.com>>

[12] EGUILUZ, Javier. *Desarrollo web ágil con Symfony2*. 1st Edition. 2012.

[13] Thomas Rabaix. *Sonata Project*. [online]. [Accessed on: May 2014]. Available on: <<http://sonata-project.org/>>

7. ANNEX

7.1 TECHNICAL MANUAL

7.1.1 XAMPP (LAMP)

To develop the application, is needed a local server installed on the computer. This will allow to execute the website as the website was working in an online server. In this manual is guessed that who follows these instructions has, at least, a basic knowledge of Linux OS.

At first, has to be downloaded the pack from tge website <https://www.apachefriends.org/download.html> and has to be used the last version of “XAMPP for Linux” and for 64 bits (...).

Then, with the terminal, the permissions of the installer file have to be changed:

```
$ chmod 755 xampp-linux-*-installer.run
```

Run the installer:

```
$ sudo ./xampp-linux-*-installer.run
```

XAMPP is now installed on the /opt/lampp directory.

To start XAMPP this command has to be called:

```
$ sudo /opt/lampp/lampp start
```

It should appear something like this on the screen:

```
Starting XAMPP 1.8.2...
LAMPP: Starting Apache...
LAMPP: Starting MySQL...
LAMPP started.
Ready. Apache and MySQL are running.
```

To stop XAMPP this command has to be called:

```
$ sudo /opt/lampp/lampp stop
```

Now it should appear something like this on the screen:

```
Stopping XAMPP 1.8.2...
LAMPP: Stopping Apache...
LAMPP: Stopping MySQL...
LAMPP stopped.
```

When XAMPP is started, can be tested that everything is working, typing in the following URL at our favourite web browser:

```
http://localhost
```

Now should be seen the XAMPP start page (Figure 14) containing some links to check the status of the installed software and some small programming examples.

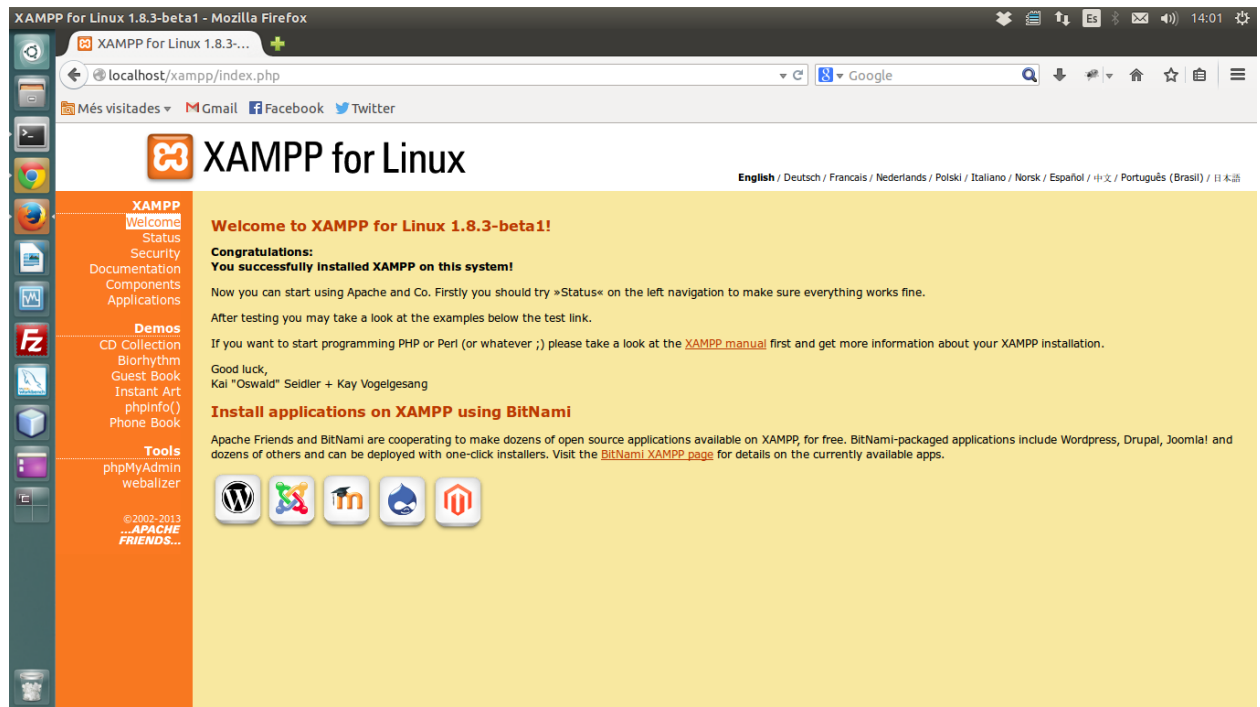


Figure 14. XAMPP for Linux working successfully.

When XAMPP is working, can be inserted the files in the directory 'htdocs' (/opt/lampp/htdocs/). In this folder, is where the server can show and execute the files of the application.

To execute the project, should be typed in the following URL:

```
http://localhost/%name_file_or_name_folder%
```

7.1.2 phpMyAdmin

Furthermore, there is the tool phpMyAdmin (Figure 15), to manage our databases. Has to be typed the following URL:

<http://localhost/phpmyadmin>

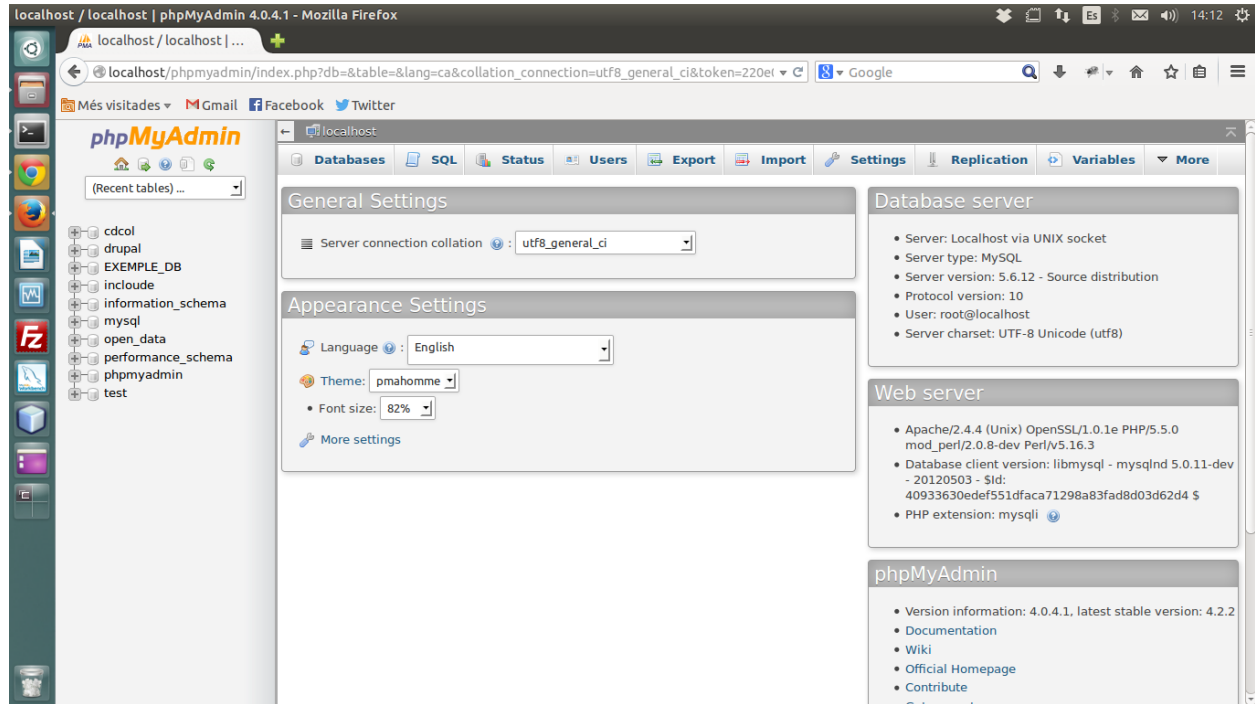


Figure 15. phpMyAdmin database management system.

7.2 USER MANUAL

In this section, is explained how to navigate and use the portal in the case that the user is a visitor (anonymous user) without login and then if the user is a registered user, already logged on the portal.

7.2.1 Anonymous user

As a visitor (without login) when is being entered in the portal, the homepage is like can be seen in the next figure (Figure 16).

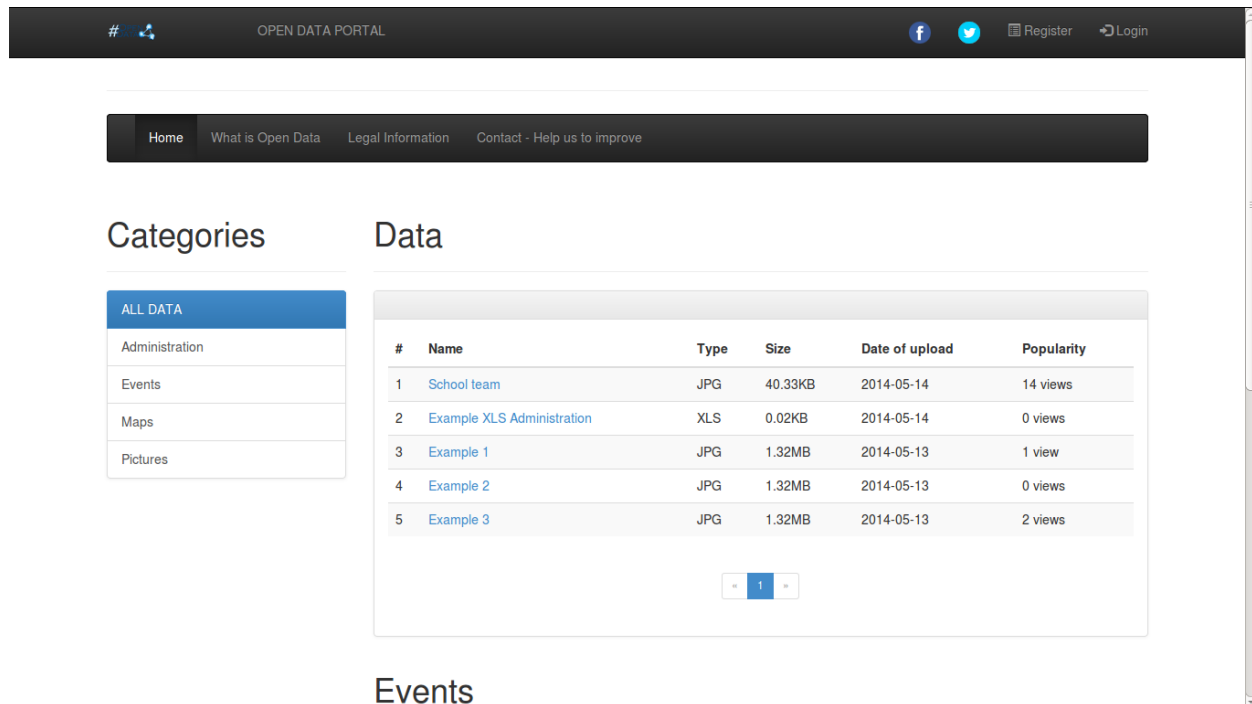


Figure 16. Homepage of the portal (Anonymous user view).

There are the options of “Register” and “Login” and also the top menu with the links to the different pages. On the left side, can be seen the list of categories (can be chosen “All Data” if is wanted to show all the data or can be chosen one of the categories) and in the center of the portal there is a list of data with some basic information and the link to every file/event page.

When is chosen a category (different of the button “All Data”), it appears only the data that are related with that category (Without the list of “Events”). The “Events” category works as a “special” category, so when is chosen this category, the other data disappears and only it appears in the center of the homepage, the list of events (Figure 17).

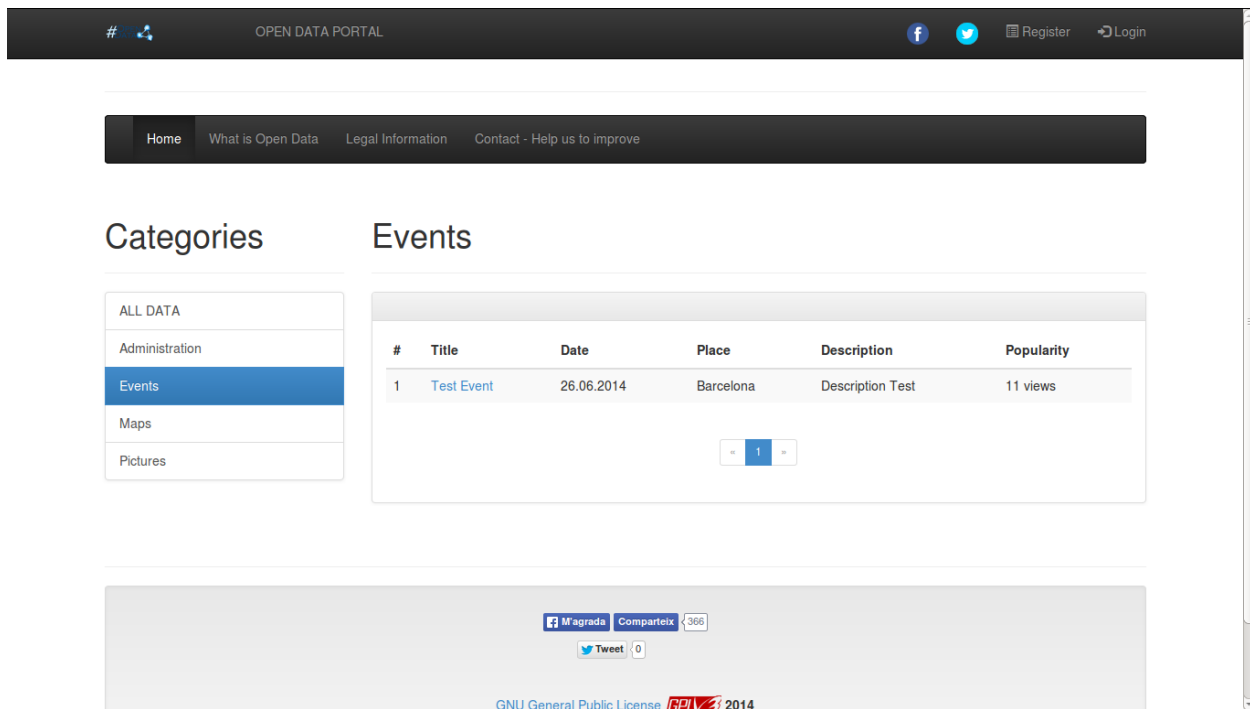


Figure 17. Homepage - “Events” category (Anonymous user view).

Furthermore, will be able to go to any of the pages that are linked on the top menu, like “What is Open Data” (Basic information about Open Data), “Legal Information” (Important legal and license terms that would be written if is decided to put this portal online in the future) and “Contact - Help us to improve” where is a contact form (Figure 18). The user can contact the administrator about any subject and also is invited to send some advices or recommendations about the portal. These will be send by email to the administrator of the portal.

The image shows a web browser window displaying the 'Contact form' on the 'OPEN DATA PORTAL'. The page has a dark header with the portal name and social media icons. A navigation bar below the header contains links for 'Home', 'What is Open Data', 'Legal Information', and 'Contact - Help us to improve'. The contact form itself consists of four input fields: 'Full name', 'Email address', 'Subject', and 'Message'. A blue 'Send' button is located at the bottom of the form. Below the form, a small orange warning box states 'All form fields are required.' The browser's scrollbar is visible on the right side of the page.

Figure 18. Contact form of the portal.

On the homepage, if is clicked the link of any of the files of the data section, will be seen the details of the selected file (Figure 19). If the text of the field 'Link' is clicked, a new tab will be opened with the content of the file. Probably if is an image or a video will be directly opened and if is another type of file probably it will be opened a dialog for downloading the file, but this depends on the configuration of the browser.

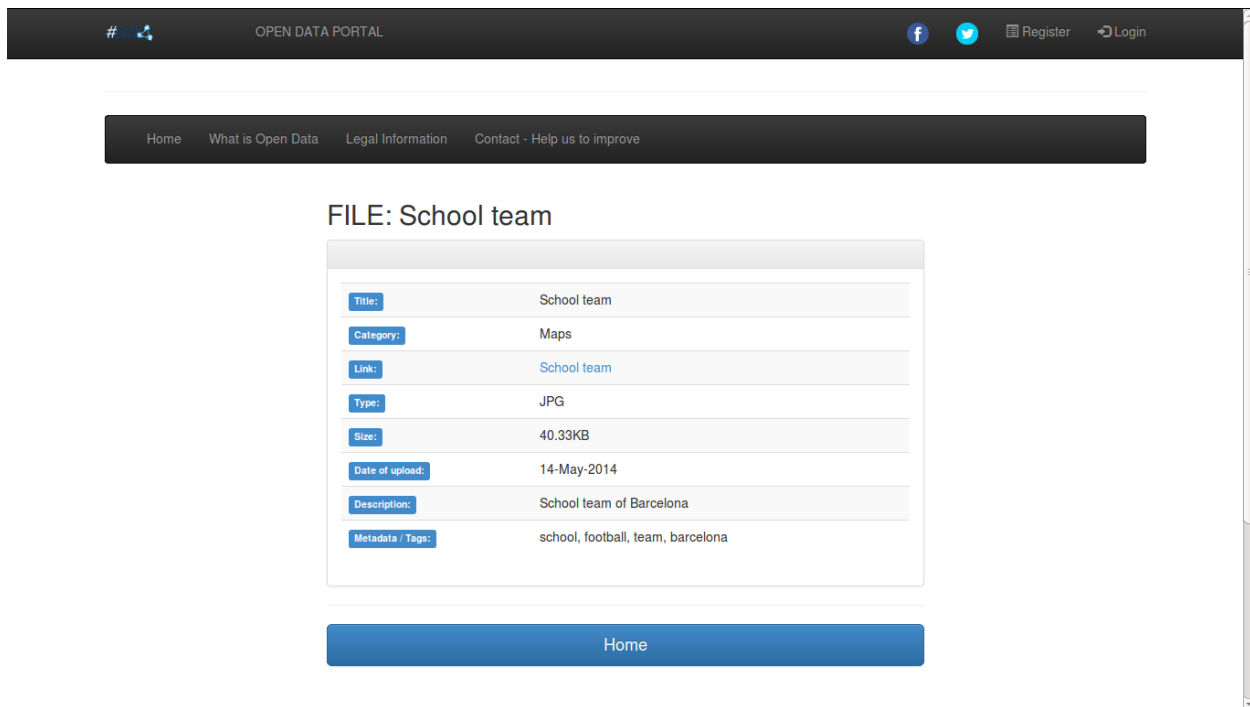


Figure 19. Page of data/file details of the portal.

And the same procediment when is clicked any event. The page will show the details of the selected event (Figure 20).

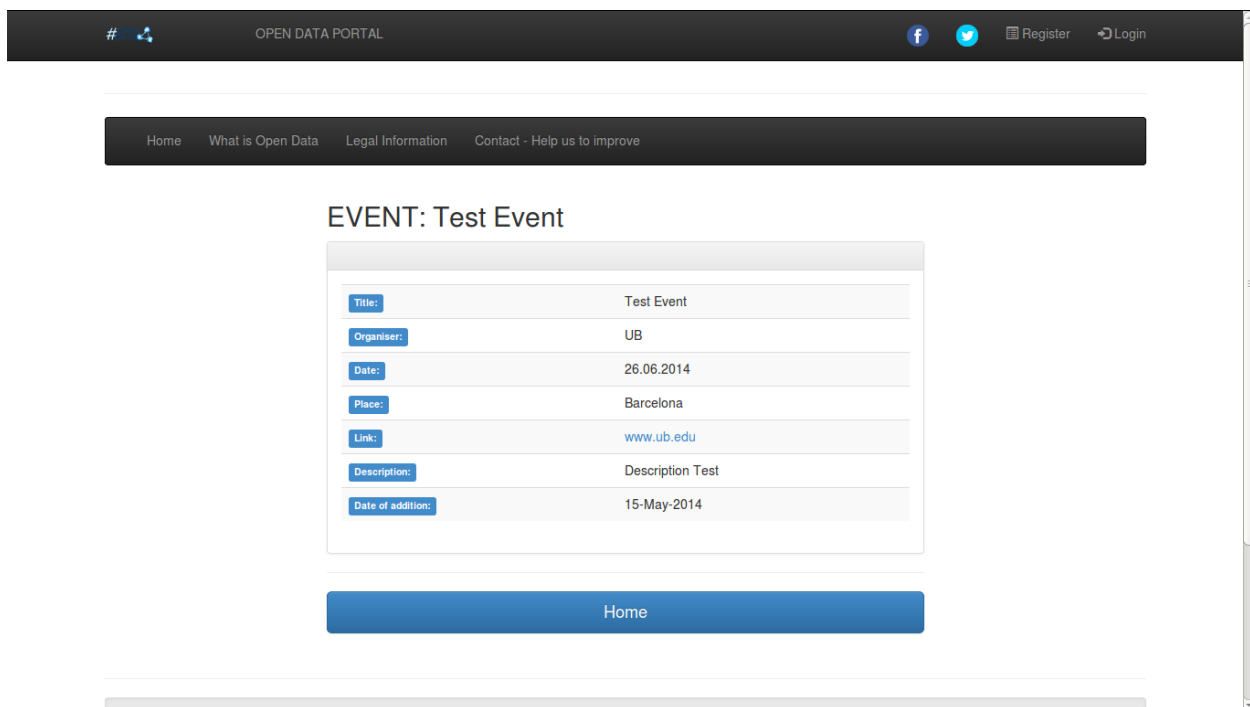


Figure 20. Page of event details of the portal.

Now, if is clicked the link of “Register” it appears the form with the different fields that should be filled in (Figure 21). In this form all the fields are mandatory, so there is an orange alert message which is alerting of that rule.

If is put an invalid address, an invalid password or the two password fields are not the exactly the same, the navigator will show alerts warning about the problem (These messages will be shown in the language of browser configuration).

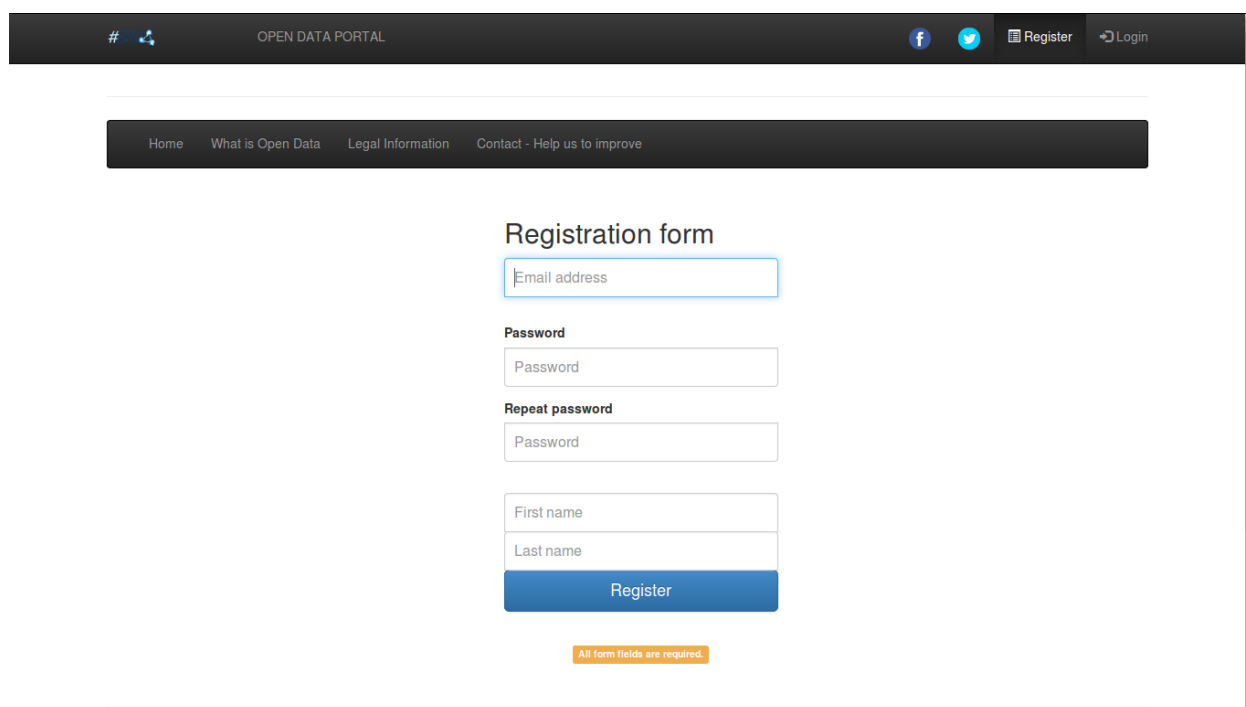
The image shows a web browser window displaying the registration form of an 'OPEN DATA PORTAL'. The page has a dark header with social media icons and 'Register' and 'Login' links. Below the header is a navigation bar with links for 'Home', 'What is Open Data', 'Legal Information', and 'Contact - Help us to improve'. The main content area is titled 'Registration form' and contains several input fields: 'Email address', 'Password', 'Repeat password', 'First name', and 'Last name'. A blue 'Register' button is positioned below the 'Last name' field. At the bottom of the form, there is an orange alert message that reads 'All form fields are required.'.

Figure 21. Registration form of the portal.

When is finished the registration will be received an email saying that has been successfully registered, but before login, the administrator has to accept the user (also the administrator receives an email that there is a new user registered and has to be accepted or not). This will be explained in details in the next chapter.

When the user is validated by the administrator, already can login on the portal (Figure 22). If are entered bad credentials, the user will be alerted of the problem.

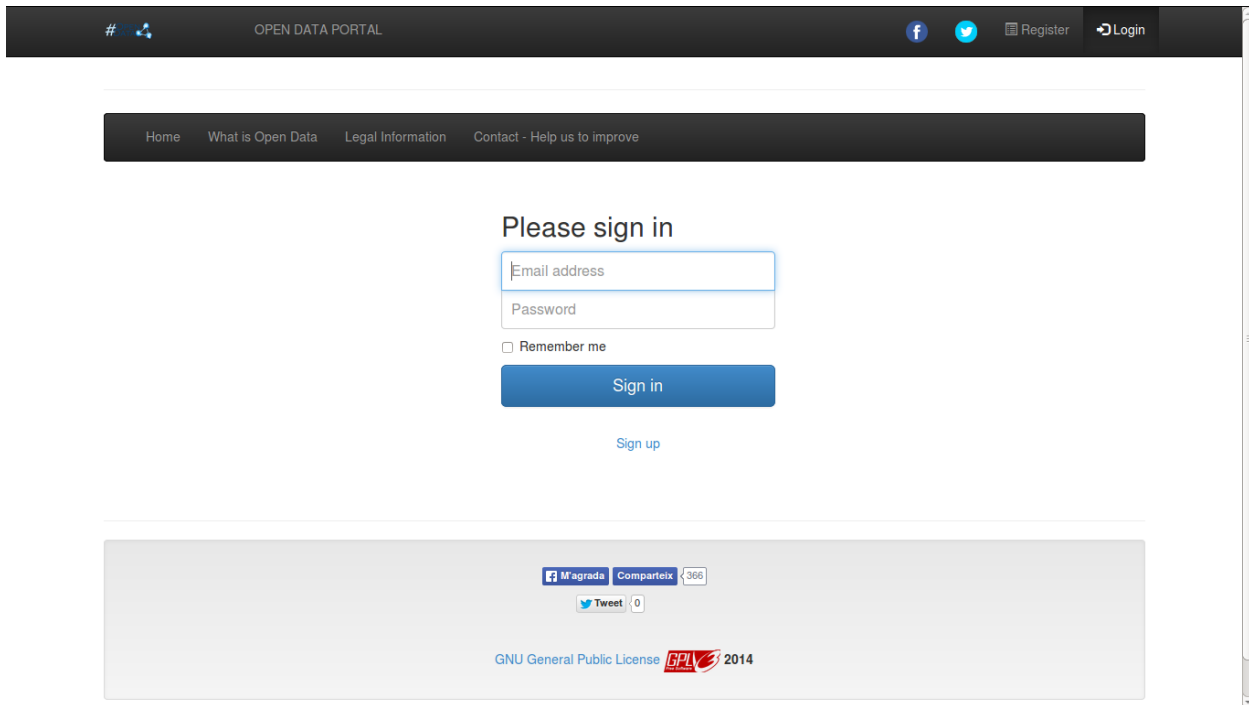


Figure 22. Login form of the portal.

7.2.2 Registered user

If the login procediment has been successful, the user will be moved to the homepage again, but at this time there will be some different things. As it can be seen (Figure 23), now it has appeared an “Upload Data” button. Furthermore, now there isn’t the button of “Register” and “Login” and instead of this, there is a “Logout” button and a dropdown button with the “Name + Surname” of the user that is logged in that moment and was entered at the moment of the registration.

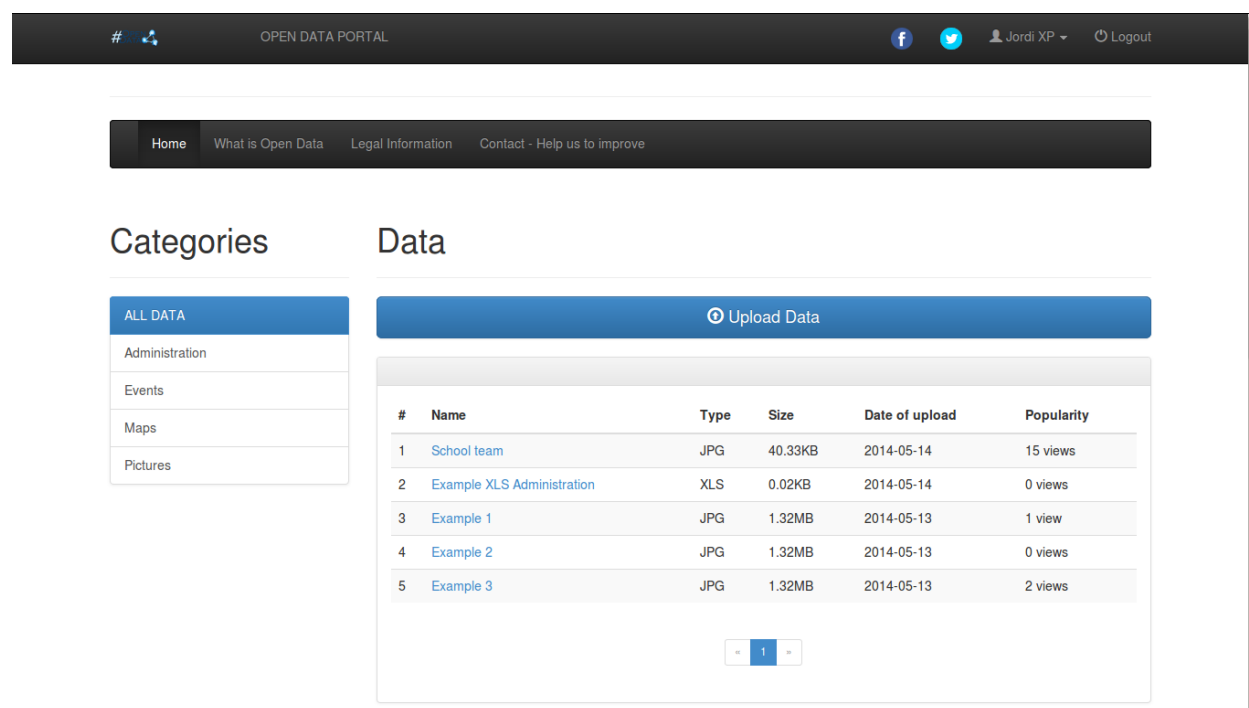


Figure 23. Homepage (Registered user view).

Also it appears a button “Add Event” (Figure 24) and if is clicked the name + username button, it is shown the list with the two buttons to upload data and add events, and also a link “My Profile” where can be edited the password and the information that was entered when the registration was done.

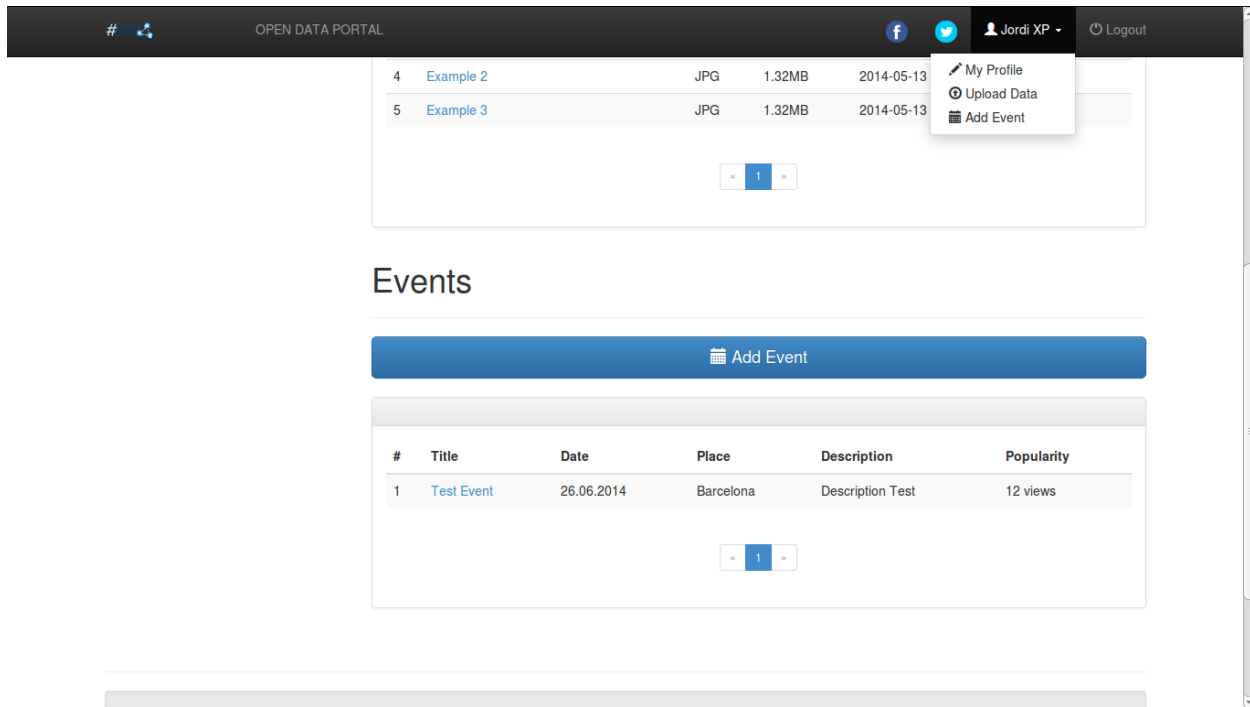


Figure 24. Homepage - "Events" category (Registered user view).

If is clicked the button "Upload Data" will be shown the form to upload data (Figure 25). The mandatory fields are marked with an asterisk (*) and if the mouse is over one of the icons with the question mark (?), will be received important information about the field.

The second field, is the field which will upload the file (data). This field appears automatically in the language which the user has the browser configured.

The screenshot shows the 'Upload file form' interface. At the top, there is a dark navigation bar with the text 'OPEN DATA PORTAL' and user information 'Jordi XP' and 'Logout'. The form itself is centered and contains the following elements:

- A dropdown menu labeled 'Choose the category *' with a downward arrow.
- A button labeled 'Navega...' followed by the text 'No s'ha seleccionat cap fitxer'.
- A text input field labeled 'Title of the file *' with a question mark icon to its right.
- A larger text input field labeled 'Description of the file *' with a question mark icon to its right.
- A text input field labeled 'Website relationated'.
- A text input field labeled 'Metadata / Tags' with a question mark icon to its right.
- A blue 'Upload' button.
- A small orange box at the bottom with the text '* Required fields'.

Figure 25. Upload data/file form of the portal.

All file formats are allowed on the portal but the application was thought to upload files of maximum 20MB. As was explained in detail in the chapter of 'Conclusions and future work', there was a problem that made that finally was not possible to put a limitation for the size of the files. When a new file is uploaded, this file will not appear on the homepage list of data. Before that, has to be checked and validated by the administrator (the administrator will receive an email warning about the new file to be checked and validated and the user who uploaded the file will receive an email telling that the file was successfully uploaded but first has to be validated by the administrator).

And finally, when the button "Add Event" is clicked is shown an event form, where can be entered the information of an event (Figure 26). It also has to be validated by the administrator, following the same procediment that was mentioned in the case of uploading data.

OPEN DATA PORTAL f Jordi XP Logout

Home What is Open Data Legal Information Contact - Help us to improve

Add event form

Title of the event *
Organiser of the event *
Date of the event *
Place of the event *
Description of the event *
Link of the event

Add

* Required fields.

Figure 26. Add event form of the portal.

7.3 ADMINISTRATOR MANUAL

In this section, is explained how to use the administration section of the portal.

First of all, if is wanted to enter to the administration section, when the user is at the homepage of the portal should be added the text `/admin` to the URL.

After that, it will appear a dialog to enter the user and password of the administrator (Figure 27). The credentials for this administrator are described in the file README.md that is in the packet with the code of this application.

Also it could happens that it should add an exception to the browser of the user, because this section requires HTTPS (secure connection).

(These messages will be shown with the language that the user has the browser configured).

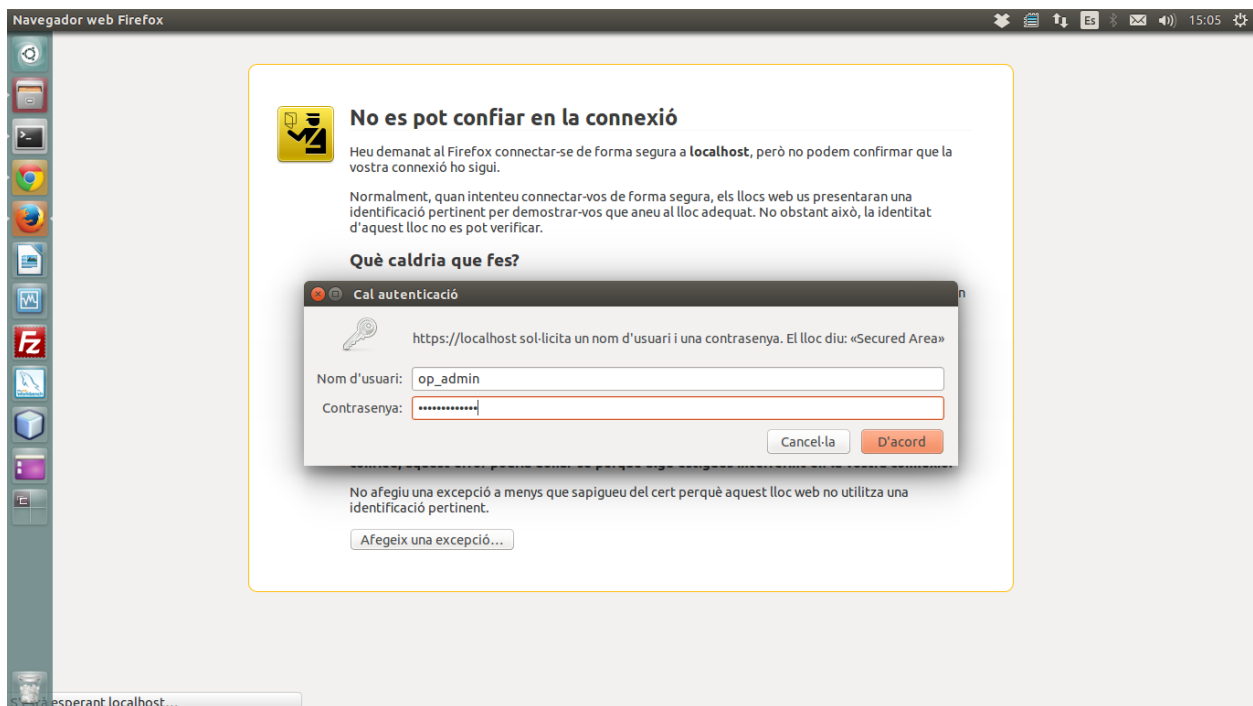


Figure 27. Login dialog of the administration section.

If our credentials are right, will be see the homepage of the administration section. Figure 28 shows the links to each part of the portal/database that is possible to be modified (Users, Categories, Data and Events).

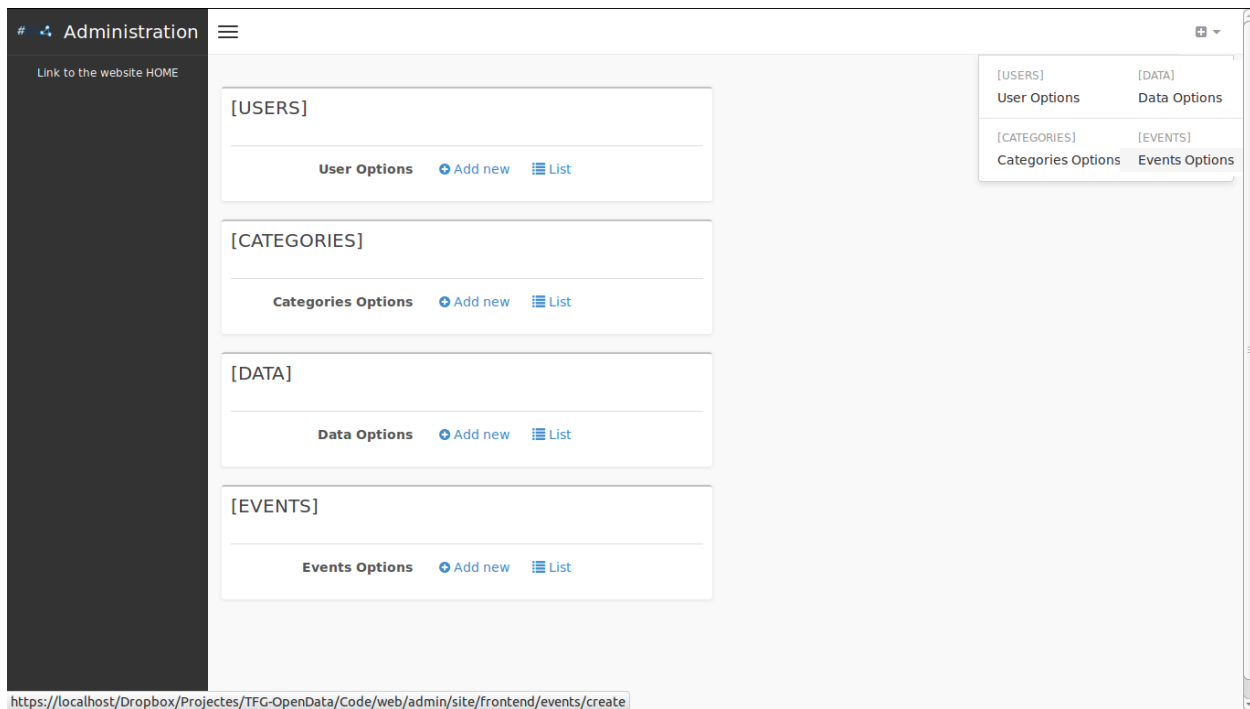


Figure 28. Homepage of the administration section of the portal.

When is clicked on the link “List” of one of the sections can be seen the list of items with the information of each item (Figure 29).

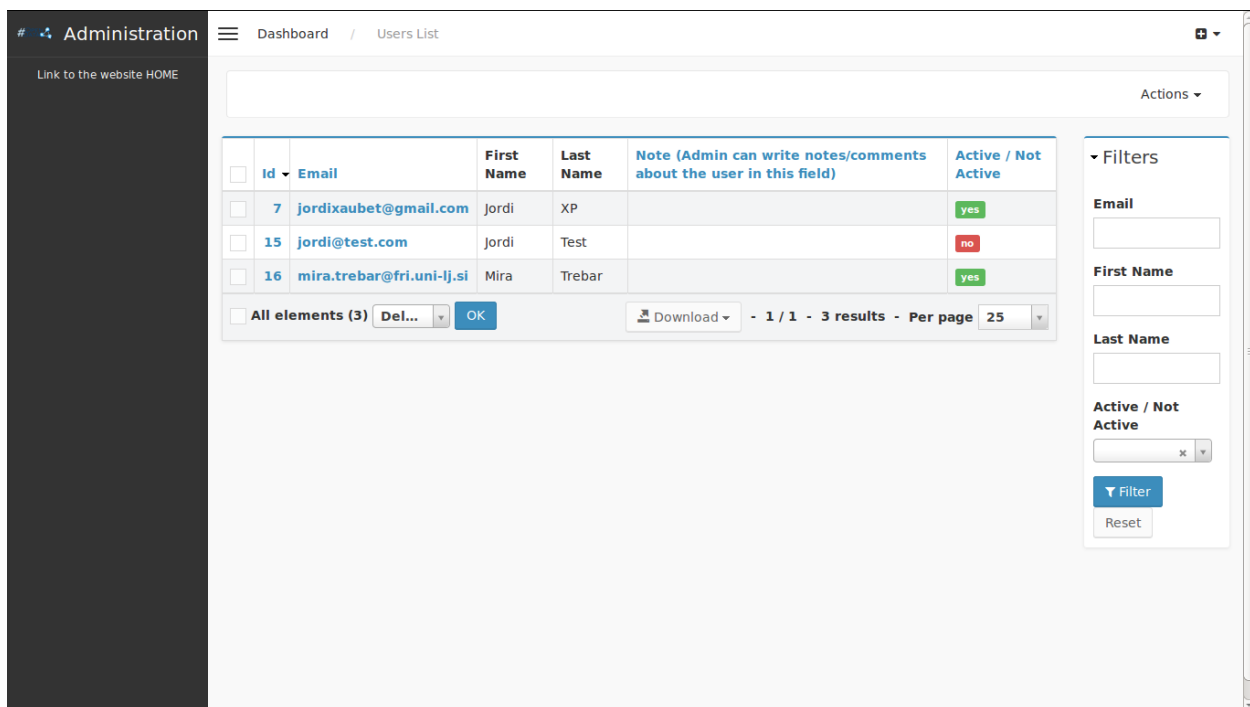


Figure 29. List of users of the portal.

Also it can be created new items easily (Figure 30), clicking the button “Add new”.

The screenshot shows a web application interface for managing categories. The top navigation bar includes 'Administration', 'Dashboard', and 'Categories List'. A sidebar on the left contains a 'Link to the website HOME' button. The main content area features a 'Create' form with a text input field for 'Name Category *'. Below the form are three buttons: 'Create', 'Create and return to list', and 'Create and add another'. An 'Actions' dropdown menu is visible in the top right corner.

Figure 30. Page to create new categories.

When there is the list of items, it can be clicked on the Id field of the list (or some other linked fields of the item) and the page is moved to a page where it can be edited the field (Figure 31). In that page, there is also the possibility of activate or deactivate items (“Active” checkbox).

The screenshot shows a web application interface for editing a user. The top navigation bar includes 'Administration', 'Dashboard', 'Users List', and 'Mira Trebar'. A sidebar on the left contains a 'Link to the website HOME' button. The main content area features a 'User Options' form with the following fields: 'Email *' (mira.trebar@fri.uni-lj.si), 'Password (Has to be codified with algorithm SHA-512 base 64) *' (KjB/VSlgcr1MeFLgm9FkChS4IBOgc7B7N5qtZJYxbRTuMFWGgKa0QNJAp2JvFbOR87P9e39T4UDz7IaF2fcJlQ==), 'Firstname *' (Mira), and 'Lastname *' (Trebar). There is also a 'Note' text area and an 'Active' checkbox which is checked. At the bottom, there are three buttons: 'Update', 'Update and close', and 'Delete'.

Figure 31. Edit user page.

When there is the list of items, there is also the possibility of selecting items and delete them (Figure 32). On the other hand, is recommendable don't delete items if is possible the edition of the item.

The screenshot shows a web application interface with a dark sidebar on the left containing the text 'Administration' and 'Link to the website HOME'. The main content area has a breadcrumb 'Dashboard / Events List' and an 'Actions' dropdown. A table with the following columns is displayed: 'Id', 'User', 'Title', 'Organiser', 'Date', 'Place', 'Description', 'Link', and 'Accepted / Not Accepted'. The table contains one row with the following data: '4', '7', 'Test Event', 'UB', '26.06.2014', 'Barcelona', 'Description Test', 'www.ub.edu', and 'yes'. Below the table, there is a summary bar showing 'All elements (1)', a 'Del...' dropdown menu with 'Delete' selected, an 'OK' button, a 'Download' button, and pagination information '- 1 / 1 - 1 result - Per page 25'. On the right side, there is a 'Filters' panel with input fields for 'Id User', 'Title', 'Organiser', 'Date', and 'Place', a dropdown for 'Accepted / Not Accepted', and 'Filter' and 'Reset' buttons.

Figure 32. User list page - Possibility of deleting users.