Dynamic Web Sites

For Erasmus students

And

A Spanish rock band

Ricardo Ibarra Roca
Pablo de la Iglesia Couto
# REPORT INDEX

- **Introduction:**
  - Personal Reasons 3
  - Choosing the most suitable technology 3
  - Initial Goals vs Final Results 5

- **The cryptography and the use of secure hashing** 8

- **Time plan** 12

- **Development:**
  - Intro 13
  - Access Control 14
  - Forum 19
  - Mail server 24
  - Photo album 26
  - Link database 27
  - Language Selector 29
  - Guest Book 30
  - CMS 31
  - Random Sentences Generator 34
  - Web Radio 38
  - Concerts Administrator 40

- **Conclusion** 42
- **References** 43
- **Appendixes**
  - ER-graph 44
  - Time Plan 46
  - List of abbreviations 48
  - Questions at the seminar 49
INTRODUCTION

Personal Reasons

We are two students from Spain and we only have to do in Sweden a final project and some single subjects if we want to end our Bachelor. In Spain you have to do your final project alone but as we are in Erasmus they let us to do some things together. Our project consists in two different Web Sites, we made one part together (the main part), and another separately (the specific part). About the content, Ricardo’s Website is about information for Erasmus students in Sweden and Pablo’s Website is about an amateur rock band from Spain (IllCapo). Ricardo chose this topic because for his personal experience when the Erasmus students are in a foreign country they find a lot of difficulties, because they do not know anything about the country, the city, the university etc... So they need information and he thinks it great to find all this information in a website. And a website like this will have a lot of visitors. Pablo chose this topic because he was hired by a Spanish rock band called IllCapo to build their web site. So he made the website for professional reasons.

Our project consists of Web Sites with access control, forum, photo album, news server by mail, users accounts… We made this part together and it was built by PHP, HTML, CSS and MYSQL. In the other hand Pablo built a Guest book, CMS, Random Sentences, Web Radio, Concerts Administrator, and Ricardo built a Access to a Link database, Language Selector, Video Store.

We chose this project because we want to work in the web design field in our professional future and because we think that Internet is one of fastest growing sectors of the computing world. And it is a very important subject for the entire world (“There are more than 1.1 billion of users right now in Internet” (1)).

Choosing the most suitable technology

The following technology choices were made based on our own experience and current trends in web design for small to medium sized non-commercial web sites. The technology which is used in the website is PHP as the programming language, APACHE as server system, MYSQL as the database system, CSS & Photoshop as the visual solutions to improve the style of the Web Site.

PHP was chosen as the programming language because “more than 20 millions of sites running PHP”, (2) and because it is Open Source, runs stable on a range of operating systems including most flavours of UNIX, Windows and Macs and integrates well with most popular servers including IIS and APACHE and finally because it is very easy to connect to a database like MYSQL. Another point is that the visitors of these web sites don’t need any plug-ins to use the web application since the server translates the results of PHP to HTML. In addition, PHP was chosen because it is a very easy language to learn. It also has many
specific instructions to work with MYSQL which reduces the number of code lines to write. There are a lot of programming languages for building web applications, like ASP, JSP, PERL, COLDFUSSION. One of the advantages of PHP is that it has a lot of functions and there is more documentation about PHP than any other programming language.

**APACHE** was chosen as the server system because it is free of charge and an Open Source server software. “It is the world's most popular HTTP-server since April 1996; as of March 2007 APACHE served 58% of all websites” (3) possibly being the best server around in terms of functionality, efficiency, security and speed. The only problem that we found with APACHE was that it is hard to configure it because you have to edit a complex text file by yourself. There are a lot of other web servers, like Lighttpd, Server, Sun Java System Web Server, IIS. These web servers have some disadvantages for our project: Lighttpd is very fast but it has a lot of restrictions, IIS is not Open Source. One of the reasons to chose APACHE is that it is part of WAMP, MAMP, LAMP solutions.

**MYSQL** was chosen as the database system because it works really well with PHP. And it has become the world's most popular open source database because of its consistent fast performance, high reliability and ease of use. It is used in more than 11 million installations ranging from large corporations to specialized embedded applications. “Not only is MYSQL the world's most popular open source database, it is also become the database of choice for a new generation of applications built on the LAMP server distribution (Linux, APACHE, MYSQL, PHP / Perl / Python.) MYSQL runs on more than 20 platforms including Linux, Windows, OS/X, HP-UX, AIX, Netware, giving you the kind of flexibility that puts you in control.”(4)

**CSS** was chosen because the presentation information of the website can be held in one CSS file, allowing sweeping changes to be propagated with quick changes to this one file. The document code is reduced in size and complexity, since it does not need to contain any presentational markup.(5) The CSS specifications are maintained by the W3C and it is one of the standards for the web.

**Adobe Photoshop** is used to process photographs and generate advanced graphics. No program has the comprehensiveness of features, the global support of the graphic arts and photographic community, and the richness of resources available.(6)

Finally, to develop the website we used different software tools such as Dreamweaver MX (It has a complete library of HTML and PHP functions) and PhpMyAdmin (This software provides us with the most important information about our database for the website and the skill to manage it easily, however, it is recommended to know the SQL language to verify the database results in a separate console window).
Initial Goals vs Final Results

Our overall goal is that our Web Site and each one of its parts have to work correctly. This goal is reached but with change to the detailed goals. The detailed goals are listed in the table 1 & table 2 which also indicates parts that had to be cut from the project. Because at the beginning we planned to do some applications that finally we did not build.

- Common Applications:

<table>
<thead>
<tr>
<th>Applications</th>
<th>Done?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS CONTROL</td>
<td>YES</td>
</tr>
<tr>
<td>STATISTICS</td>
<td>NO</td>
</tr>
<tr>
<td>FORUM</td>
<td>YES</td>
</tr>
<tr>
<td>VISIT COUNT</td>
<td>NO</td>
</tr>
<tr>
<td>MAILS</td>
<td>YES</td>
</tr>
<tr>
<td>SEARCHER</td>
<td>NO</td>
</tr>
<tr>
<td>PHOTO ALBUM</td>
<td>YES</td>
</tr>
</tbody>
</table>

*Table 1. Common applications planned to do at the beginning.*

- Specific Applications:

<table>
<thead>
<tr>
<th>Applications for IllCapo’s website</th>
<th>Done?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest book</td>
<td>YES</td>
</tr>
<tr>
<td>CMS</td>
<td>YES</td>
</tr>
<tr>
<td>Random Sentences Generator</td>
<td>YES</td>
</tr>
<tr>
<td>Web Radio</td>
<td>YES</td>
</tr>
<tr>
<td>Concerts Administrator</td>
<td>YES</td>
</tr>
</tbody>
</table>

*Table 2. Specific applications planned to do at the beginning.*

We did not build the statistics and the visit count components because first of all you will find applications for these functions in most servers of the world wide web. Secondly they take a lot of time to build and they are not really important for the website. We did not build the search engine because of lack of time.
About the components that are built:

In common:

i. The **access control** is a component that allows the users to enter into the different restricted areas of the website. All their information is saved in a coded database. Registered users can change all their personal information when they decide.

ii. The **forum** has a lot of different parts. It is divided into different sub forums, topics and posts. Each sub forum is managed by a moderator. This moderator decides the rules of the forum and he can edit or delete any topic or post. Each user can delete or edit only their own posts. The posts and topics are ordered by date...

iii. The **mail server** is used to inform users about the most important news or changes of the website. It is built using an SMTP server.

iv. The **photo album** shows a lot of pictures and permits the administrators the possibility to upload and delete old photos from the website. The users can only watch the pictures.

Specific Applications:

**IllCapo’s web site:**

i. The **guest book** is an application that gives visitors the possibility to put a message on a guest web page.

ii. The **CMS** gives the administrators a tool to write, edit, publish unpublished and delete page content of any page stored in the database.

iii. The **random sentences generator** is a small script that changes a text snippet in the main page every time the user clicks a button.

iv. The **web Radio** is an online MP3 player.

v. The **concerts administrator** is a tool to administer a calendar presenting the concerts of the band.
**Erasmus students’ website:**

i. The application for the **link database** that shows different links in response to search request that the user makes.

ii. The **Language Selector** provides the visitor the possibility to change the text language of the whole web at any moment.
“The cryptography and the use of secure hashing”

Intro:

One important part of a website is the security. Nowadays there is a big debate about what is the best option to protect the information of the database. These data have to be protected and in some countries there is a law about this (like in Europe with the Directive 95/46/CE (7)). At the present the best option is to use some cryptographic function. Our website needs this function too.

When the user access to our websites he will give to the website a nickname and a password, these two things have to be stored in the database. But the webmaster has the task to assure that this password will be used only for his owner. For this the webmaster has to use a cryptographic hash function. The hash function will provide an additional security to the password.

The hash functions are based on the compression; they turn some data into a small number which has to be unique for this data. There are some very important hash functions such as MD5, SHA-1 and RIPEMD.

MD5 (Message Digest 5)

Intro

It is the most used hash function.

How it works:

First of all the length of the password have to be enlarge until is exactly 64bit less than a multiple of 512bits. If is not big enough a one will be added followed by as many zeros as is needed.

Secondly 64 bits will be added. These 64 bits represents the original size of the password (without the bits added in the previous process.).

Later four registers of 32bits will be loaded. And with this 128 bits and with the first group of 512bits of our password an 4 groups of 16 operations will be made (64 operations). (8).

The results of all of these will be a word of 128 bits. This word will be divided by 4 registers of 32 bits. Then with this registers and the second group of 512 bits of our password the last process will be made and so on until the end of the password.

Finally we will have a word of 128 bits and this is our final result.
Problems

The problems of MD5 is that could be defeated by means of Brute force attacks, that consists of trying a large number of possibilities until you found the solution. And in MD5 it is proved that a collision of the hash could happen (that happens when two different password have the same hash). (In the 2004-08-17 Xiaoyun Wang, Dengguo Feng, Xuejia Lai y Hongbo Yu they proved that it could be done and the md5crk (9)(10)).

The only way to try to improve this is with password with at least 8 characters and with the method salt. This method consists of adding a string to the password, then this new password has to be coded by MD5. And when the password is decoded the string has to be removed. These solutions will not fix the problems but at least it is more complicated to defeat our system. And at least the dictionary attack will be solved (that kind of attacks is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by searching a large number of possibilities (11).

SHA-1(Secure Hash Algorithm)

Intro

Is the secure hash algorithm required by law for use in certain U. S. Government applications. (12) There are other versions such as SHA-0, SHA-224, SHA-256, SHA-384, and SHA-512.

How it works:

It works very similar to the MD5 process but instead of four registers of 32bits it will used 5 registers of 32 bits. And with these 160 bits and with the groups of 512 bits 4 groups of 20 operations will be made (80 operations).

Differences between SHA-1 and MD5

- The solution of MD5 is a string of 128 bits instead of 160 bits in SHA-1.
- The difficulties to have a collision is about of $2^{64}$ in MD5 instead of $2^{80}$ for SHA-1.
MD5 made about 64 operations instead of the 80 operation of SHA-1
- MD5 is faster than SHA-1, for example in a Pentium 266 MHz a program built in C have a promethium of 20Mbits/sec for SHA-1 instead of the 60Mbits/sec for MD5(13).
- The maximum size for the password for SHA-1 has to be smaller than $2^{64}$ bits, in MD5 there is no maximum size.
- MD5 is based on the Little Endian Architecture, SHA-1 is based in the Big-Endian Architecture.

Problems

The problems of SHA-1 are the same as MD5 but in this case is more complicated to defeat it because it has more bits to be decoded(14).

RIPEMD-160

RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest).

Intro

There are several versions such as RIPEMD-128, RIPEMD-160, RIPEMD-256 y RIPEMD-320, the second one is the most used.

How it works:

It works really similar to the MD5 and SHA-1 because is based in MD4.

Differences between RIPEMD-160 and MD5

- The solution of MD5 is a string of 128 bits instead of 160 bits in RIPEMD-160.
- The difficulties to have a collision is about of $2^{64}$ in MD5 instead of $2^{80}$ for RIPEMD-160.
- MD5 made about 64 operations instead of the 80 operation of RIPEMD-160. MD5 is faster than RIPEMD-160, for example in a Pentium 90 Mhz using a 32-bit flat memory model have a promethium of 1013 cycles for RIPEMD-160 instead of the 337 cycles for MD5.(15)

Problems

The problems of RIPEMD-160 are the same for SHA-1 but RIPEMD is slower(9).
Conclusion

We finally used MD5 because it is secure, not so much as the other systems but enough for our project and it is faster and the password will not occupy so much space in the memory and this is very important because the website will have a limited memory.
**Time plan**

At the beginning of the project we made a preliminary timetable but for several reasons we were forced to change it.

**Preliminary timeplan**

<table>
<thead>
<tr>
<th>Week</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Preliminary design of the websites</td>
</tr>
<tr>
<td>2</td>
<td>According of the design of the database</td>
</tr>
<tr>
<td>3</td>
<td>According the method of programing (oop or structured) <em>(Finally we chose the structured one.)</em></td>
</tr>
<tr>
<td>4</td>
<td>Make the login of the Web Site.</td>
</tr>
<tr>
<td>5</td>
<td>Make the forum.</td>
</tr>
<tr>
<td>6</td>
<td>Make the visit count.</td>
</tr>
<tr>
<td>7</td>
<td>Make the statistic system.</td>
</tr>
<tr>
<td>8</td>
<td>Make the mail server</td>
</tr>
<tr>
<td>9</td>
<td>Make the photo album.</td>
</tr>
<tr>
<td>10</td>
<td>Pirify Code</td>
</tr>
<tr>
<td>11</td>
<td>Improve code and new solutions</td>
</tr>
<tr>
<td>12</td>
<td>Prepare the “Preliminary report” and try to solve all the possible problem that will have happened.</td>
</tr>
<tr>
<td>13</td>
<td>Make the Web Site style according CSS.</td>
</tr>
<tr>
<td>14</td>
<td>Try to integrate our Web Site in the Web Accessibility Initiative.</td>
</tr>
<tr>
<td>15</td>
<td>Try to solve all the possible problem that will have happened.</td>
</tr>
<tr>
<td>16</td>
<td>We are going to make the content of the Web Site.</td>
</tr>
<tr>
<td>17</td>
<td>The same of the week 18.</td>
</tr>
<tr>
<td>18</td>
<td>We are going to improve the presentation of the Web Site by Photoshop and maybe Adobe Flash.</td>
</tr>
<tr>
<td>19</td>
<td>Prepare all the documentation.</td>
</tr>
</tbody>
</table>
Development part

First an explanation of how the website works. There is a set of code lines, or documents that we refer to as the “essentials” (see Image 1). These are documents like:

- Connect.php (This document connect the website to the database)
- Viewfunction.php (This document contains all the forms of the website)
- Function.php (This document contains all the common functions for the website)
- List.php (This document calls the other three documents and every other document of the website will call list.php.)
For example if a new PHP document is created for the web sites, the document will have a request to list.php at the beginning of the document. And if inside the document there is a function, the function will be stored in functions.php.

About the database there is a table for the different forums, one for the topics of the forum, one for the posts, one for the links, and another one for the users. All the tables are linked with their own ids. This is more explained in the appendix1.

Here follows a detailed explanation of all the components of the web sites.

- **Access control:**

  At the beginning we built two different components: The Log-In and the User Account. Now we have put these components into one component called access control. Because they are very similar.

  - **Part One: The Log-In:**

    **-Initial objectives:**

    The preliminary access control was supposed to be able to log in the user with a password and register them in the database. Now the access control has been improved so that if the user has forgotten his password he will receive it by e-mail. Furthermore, there is a cryptographic hash function for the password (MD5).

    In a website it is important that the user can access it by means of a password. The website has to assure the user that his personal information will only be accessed by him. For this reason the website must provide at least the following countermeasures:

    - **Hashed MAC** (Message Authentication Code)s (HMACs). To provide tamper-proofing. There are some possible algorithms such as MD5, RIPE MD SHA-1. In our website we used MD5 because it is the most suitable for our website (Look why in the appendix5.
- Encryption. Encryption turns the clear text data contained in the forms authentication ticket into unintelligible cipher text.
- Session lifetime restrictions. The session variable will only be valid for a limited time in the website.

**How it works:**

First the form is loaded from `viewfunction.php` to the `login.php` (see Image2). Here the user has to fill in the form with his personal data. The user can do two things: register as a new member or just log in.

If the user tries to register as a new user `register.php` will be called. In that document the functions are loaded from `function.php`, a function will check if every input data is correct, if the data is correct, it will be recorded in the database and the user
will be transferred to the `index.php`. If the data is incorrect, a function will send a message with an error description and the user will have to resolve it.

If the user tries to log-in, the `member.php` document will be called. In that document first the functions are loaded by `function.php`, and a function will check if the user appears in our database and if the password is correct. If everything is correct the user will be logged in and remembered by means of session variables. Finally, the user will be transferred to the `index.php` document.

The user can logout if he wants to. If he wants to logout he will press the log-out button and load the document `logout.php` which is responsible for deleting the session variables and to load `index.php`. 
Part Two: The User Account:

-Initial objectives:
The application must be able to show each user the current user information, and it must allow the user to change it. These features are provided together with a possibility for the user to upload a picture and also change it to another image. This option was most complicated to build and gave us many problems before working correctly.

-How it works:

When the user uses this application first profile_form.php will check if the user is logged in. If so the application will show the user information stored in the database.

The user can update some information such as the e-mail address, his picture for the website, and his avatar for the forum. If he wants to update the information, profile_form_new.php will be loaded and the user will have to fill in a form with the new information. This will be sent to profile_new.php by the POST-method.

Profile_new.php is responsible for checking the information and if is correct save it in the database. Afterwards it will be showed in profile_form.php.
the information is not correct, an error will be presented to the user and profile_form_new.php will be re-loaded.
The forum is a dynamic system where users can show their thoughts and opinions on different topics that have been created by them selves. The forum needs some documents to work (see Image4):

**findex.php**: Is the index of the forum and shows the different forums created by the administrator. To show the different topics of each forum we’ve got to click on a forum and it sends the request to **viewforum.php** using the method GET, and a new list of topics of this forum will be showed.

**viewforum.php**: This module shows us the different topics of each forum. In this part a registered user can create a new topic. To do this it is necessary to send the variable `forumid = (int)` which contains the forum id and in return it sends us to the document `compose_post.php`. On the other hand we can also see all the posts of every topic, in this
case we attach the variable $t = (int)$ to the url, which is the identification of the topic.

**viewtopic.php:** It shows us all the posts of every topic ordered by date. Every registered user can respond to a topic post and edit his own posts during the session, however only the administrator has the right to change or delete posts. If the administrator deletes the first post of the topic the rest of that topic’s posts will also be deleted immediately.

The module which is the responsible for deleting posts is the document **transact-affirm.php** called by means of the GET-method. The sentence: “`action=deletepost&id= (post’s id)`” will be sent by the GET-method.

The respond-to and edit actions will transfer control to **compose_post.php** which is the form used to create a new topic, write a response or edit contents of the last post created.

**compose_post.php:** The first thing that this module does is to select which kind of action it is to be performed. For that it takes the value sent by the GET –method(see Image5).

```php
if( isset($forumid) and !isset($topicid) and !isset($reid)) {
    $new_mode = TRUE;
    $titulo = 'Crear Nuevo Tema';
} elseif(isset($forumid) and isset($topicid) and isset($reid)) {
    $response_mode = TRUE;
    $titulo = 'Responder';
} elseif(isset($_GET['a']) and isset($_GET['post'])) {
    $edit_mode = TRUE;
    $titulo = ' Editar Post ';}
```

*Image5. Main code of compose-post.php*

When the fields have been filled in and the submit button has been pressed the information is sent to **transact_post.php** which is
responsible for processing all the information provided by the different modes (create new post or edit old post). The variables that we send using the POST method are the variables from Image5 (see Image6).

```php
$topicid = $_GET['topicid'];
$forumid = $_GET['forumid'];
$idpost = $_GET['post'];
$reid = $_GET['reid'];
$boton = $_GET['boton'];
$subject = '';
$action = '';
$mode = '';
$body = '';
$titulo = '';
```

**Image6. Variables for the POST-method**

**transact-affirm.php:** One of the objectives of this document is to send the necessary information to transact post to delete the post. Using a Javascript script (see Image7).

```javascript
function deletePost(id,redir) {  if (id > 0) {    window.location = "transactpost.php?action=delete&post=" +id;
 } else {
    history.back();
  }
}
```

**Image7. Code of the function deletePost**

*Transact-affirm.php* sends to *transactpost.php* two name-value pairs *action=delete* and *post=(int)* using the GET-method.
transact-post.php: First of all the database has to be connected by means of a function defined in another module conn.php. These are the different cases of execution of the transact engine:

- **Case New Topic:** In this case the engine needs to know if there are any errors such as: empty subject or empty body. If it finds any problems the script redirects processing to compose_post.php with an error message using the GET-method. If there is no error it starts to execute the transaction engine. First the engine creates a new topic in the database with all the fields filled with the default values except for the id, this id will be saved in a variable called $iddeltopic. Then it fills in a new topic with all the values provided in the form in compose_post.php and the field id_topic is set to the value of $iddeltopic. Done with this, the rest of the topic fields that are filled with default values.

- **Case Response:** As in the previous case it has to be checked if there are any errors. If no errors are found, a new post will be inserted into the database with the information sent by the POST-method e.g: $forumid, $topicid and $_SESSION['user_id'] to know which is the main topic. Then the topic will be updated with the date of the last post in one field of the database, this is necessary to do for order it later.

- **Case Edit:** We test if there are any errors. If not, the post will be updated with the new values entered in the form in compose post and this topic will be updated with a new value for last date.

- **Case Delete:** Before deleting the post we need to know if it is the main post of the topic. If it is, all the post will be deleted of this topic and, of course the topic will be deleted of the database.

Problems that we had: We came across a severe problem while programming the forum. We started out without a topic table and instead organized the topics in a temporary table. This produced a disorder in the layout of the topics in viewforum.php. This forced us to create a topic table in the database and rewrite all the code of compose-post.php and transact-post.php.
Other features: After finishing the forum we decided to insert smileys in the forum, as in common forums on the Internet. We preferred to create our own smiles using Photoshop.
The Mail Server component has the function to create a new password for a user who has forgotten his password. This randomly generated password is sent to the user by e-mail. The mail server needs some documents to work (see Image8):

- **nuevomail.php**: This module presents a simple form with just one field, the e-mail address of the user. This information is sent to **generapassword.php** using the POST-method.

- **generapassword.php**: The engine first tries to identify if there is a registered user with this e-mail address. If indeed there is, the function `GeneraPassword()` creates a new password (see Image9) randomly.
The engine updates the user’s password and encodes it using the PHP function `md5()` creating a hashed version of the password. It then sends the e-mail using the PHP function `mail($to, $subject, $message, $headers);` An external SMTP server configured for the website is needed for this function.
**Photo album:**

- **Initial objectives:**

  At the beginning we tried to make a document which shows all the pictures of the database. We realized that was not enough so we tried to improve it. Finally we found a script on the web. We found that we can do more things like upload, edit and delete pictures online. So we changed the application for the new one and we adapted it to our web.

- **How it works:**

  First *photo.php* loads the pictures of the database, and then checks the level of the user if the user is an administrator he can upload pictures, if he is the webmaster he can upload, edit and delete pictures. The document that shows the pictures is *galeria.php*, the document that contains the application for the upload, edit and delete of the pictures is *indexalbum.php*.

  *Headeralbum.php* contains the different forms for *indexalbum.php*. 

*Image10.Description of the photo album performance*
Config.php contains all the paths for the pictures, all the paths for the directory which contains the pictures.

Upload.class.php contains the basic functions.

Funciones.inc.php contains some functions for indexalbum.php

- **Link database:**

  - **Initial objectives:**
    The website with information for Erasmus students in Sweden has to show a lot of links about different aspects of Sweden. These links have to be grouped and the list of links will be dynamic. This created a problem for us: to see how we solved it. Look at how it works.

  - **How it works:**
    The solution was simple, a database was built only for links and there will be some fields like (the bold text is the important part):
    - Id (different for each one)
    - url (the web address)
    - score (the users can vote the link)
    - votes (the total of votes)
    - intro (little description)
    - name (The name)
    - type (If is 0 means for everything, if is 1 means for hotels, if is 2 is for museums, ....)
    - type2 (if is 0 is for all Sweden, if is 1 is for Gotheborg etc)

    So if the user wants to know the links for Halmstad he only has to search for links with type=0 type2=2, so we only have to make one specific loop for each document and now he can save new links in the database at any time and we do not have to upload all the documents.
- **Language selector:**

**Initial objectives:**
This website will be used for people from every part of Europe so it is very important to put all the information not only in English. But it is not intelligent to build a website for every language so we create an application with the skill of change the language at any document.

The first version of the application could change the language but always the web site was redirect to the index, so we tried to improve it and finally we solved it. Now the document is the same if the user changes the language except in the forum because there is a forum in English and another in Spanish so you are redirected to the correct forum’s index.

And if we want to add a language we only have to create a document not more.

**How it works:**

![Diagram](Image11.png) *Description of the language control performance*

All the documents have the possibility to change the language, the user can press a button to define the language. If the user press the button, it will call the respective document *(english.php, spanish.php, otherlanguage.php)*.
spanish.php... this document will set this language as the
language for the web by means of session variables.

When you open a document, this document will call to the
document idioma_x.php (where x will be the language that we
set before, such as es for Spanish, en for English...).

If there isn’t a language set, English will be appearing.
**Guest Book:**

Image12. Description of the access control performance

The Guest Book provides to the users and visitor the chance to write what they want. The Guest Book needs some documents to work (see Image12):

**libro.php:** This module shows us all the commentaries that visitors and users have made. It has a link to drive us to **libroredactar.php**, this module is a formular to be filled with the information.

**libroredactar.php:** It’s just a formular which sends the information to **transactlibro.php** using the POST method. If the one write is a user logged in the name field will appearance automatically.

**transactlibro.php:** In case the user tries to send a commentary, first it has to be checked if there are any error like no body or no author, in any of these cases or both **libroredactar.php** is showed with an error message. If there is no error it starts to execute the script. This new information will be inserted in the database. Then the user will be transferred **libro.php**.

**Problems that we had:** We had no problem with this part.
**CMS:**

The CMS is a dynamic system which provides to the administrators of the web the change of write, edit, publish, unpublished and delete news. The CMS needs some documents to work (see Image13):

- **compose.php:** Is a simple form with just two fields. One for the title of the news and the other for the body. However this part of the website is only visible for the administrators. Because of that there is a filter that is the responsible to redirect the normal users to an error page is they try to see this form. In the form the information will be sent to the transact_article.php using POST-method. The visible variables such as: `body` and `title` of the news and also the variable `action` will be sent if the users tries to create a new one. Also invisible variables such as `idnew` and `idauthor` will be sent if the user tries to edit news. If the administrator goes to compose.php to edit news, the module will look for all the information about it. First the module will look for the id of (previously sent by reviewarticle.php) by means of
POST-method. The module will show the entire body and title that the administrator wants to change.

**transact_article.php:** Is an invisible PHP script that is the responsible to process all the information about the news.

**Case Send a News:** In this case the engine needs to know if there are any errors such as: empty title or empty body. If it finds any problems the script redirects processing to `compose.php` with an error message using the GET- method. If there is no error it starts to execute the engine. It will insert a new field in the table news in the database filling the fields with the variable previously sent by `compose.php` and the field `id_author` is will be filled with the variable `$_SESSION['user_id']`. Then we redirect to `cpanel.php`.

**Case Save Changes:** As in the previous case it will check if there’s an error. If there’s no error it will update the field of the data base with the variables of `compose.php`.

**Case Publish:** After checking for errors the engine updates the values of this new with the values:

```
"UPDATE news " .
"SET publish=1, publish_date=" .date("Y-m-d H:i:s",time()) . " " .
"WHERE id=" .$_POST[id_new];
```

*Image14. Description of case Publish*

**Case DishPublish:** The engine updates the values of this newsitem with the values:

```
"UPDATE news " .
"SET publish =0, publish_publi='0000-00-00 00:00:00' " .
"WHERE id=" .$_POST[id_new];
```

*Image15. Description of case Publish*
**Case Delete:** The engine delete the new of the database:

**cpanel.php:** This module shows to the administrators the published and unpublished news. Each new has a link to `reviewarticle.php`, each one are referenced with the variable `idnew=(int)` using the GET method.

**reviewarticle.php:** In this part the administrator can edit, publish, unpublish and delete the news. Publish and unpublish send the order to `transact_article.php` with the necessary variables using the POST method. If the administrator clicks in DELETE button a warning message will be shown by `transact-affirm.php` telling him that the newsitem will be delete by the engine. If the administrator accepts, the newsitem will be deleted by transact-article. If the administrator press edit he will go to `compose.php`, then `reviewarticle.php` will send the id of the new using the POST method.

**viewarticle.php:** Show to the users the entire new

To show the news the users have to go to `index2.php` and the main part of every new will be show. If the user clicks on the link *read more* he will go to `viewarticle.php` to show the newsitem referenced by the variable `new` using GET method.
Random Sentences Generator:

The aim of this application is to show in the title of the web a random sentence done by a famous artist. The random sentences generator needs some documents to work (see Image15):

**header.php**: the title and the sentence are right here. In this part the module calls a defined function named...
mostrarfrasealeatoria() that returns the sentences. mostrarfrasealeatoria() is in outputfunctions.php

outputfunctions.php: This module contains the function mostrarfrasealeatoria(). This function look for the maximum id of sentences saved in the database. Then select a random number between 1 and the maximum id. This number will be the id of the sentence that will be show in the title, however first it has to be checked if there is a sentence with this id. If there is no sentence the function repeat execution until it finds a sentence. Then it calls another function named mostrarfrase(), and it returns the sentence to the title.

```php
function mostrarfrasealeatoria(){
  global $conn;
  $sql = "SELECT MAX(id) as total FROM frases;";
  $result = MYSQL_query($sql,$conn) or die("No se pueden acceder a las frases");
  $row = MYSQL_fetch_array($result);
  $id = mt_rand(1, $row['total']);
  $sql = "SELECT * FROM frases WHERE id = ".$id."; 
  $result = MYSQL_query($sql,$conn) or die("No se pueden acceder a la frase en cuestion");
  $row = MYSQL_fetch_array($result);
  if($row['id'] == NULL){
    mostrarfrasealeatoria();
  }
  mostrarfrase($id);
}

function mostrarfrase($id){
  global $conn;
  $sql = "SELECT * FROM frases WHERE id = ".$id.";
  $result = MYSQL_query($sql,$conn) or die("No se pueden acceder a la frase en cuestion");
  $row = MYSQL_fetch_array($result);
  echo "<id="frase">".".$row['frase']."</id>";
  <em><strong>
There is an option to the administrators to manage the Sentences. This part is invisible for the standard users.

**frases.php:** show all the sentences saved in the database. The administrator can create a new sentence, edit or delete any sentences. In case to delete a sentence the module will send the variables `action=deletefrase&id=(int)` using the GET method to `transact-affirm` to show a message warning of the risks to delete a sentence, after confirm the deleting of the sentences the information is send to `transact-frase.php` (by JavaScript) using GET method.

**compose-frase.php:** It is a simple form with just 2 fields. One for the sentences and the other for the author of this. However this part of the site is only visible for the administrator. Because of that it has a filter that is the responsible to redirect the normal users to an error page is they try to see this form. In the form it will send to the `transact_article.php` using POST method visible variables such as: `author` and `sentence` and the action to send a new item. Also it will send invisible variable `idfrase` in case it tries to edit a new item. If the users returns to `compose-frase.php` to edit any new item, the module will look for all the information about the new item, defined by the id of the new previously sent by `frases.php` using GET method, in the database and show the entire sentence and author that would be change by the administrator.

**transact-frase.php:** Is invisible PHP script that is the engine that process all the information about sentences

**Case Send Sentence:** In this case the engine needs to know if there are any errors such as: empty author or empty body. If it finds any problems the script redirects processing to `compose-frase.php` with an error message using the GET- method. If there is no error it starts to execute the engine. It will insert a new field in the table filling the fields with the variable previously sent by `compose-frase.php` and the
field \textit{id \_author} is will be filled with the variable \$_SESSION[\textquote{user \_id}]. Then we redirect to \textit{cpanel.php}.

\textbf{Case Edit Sentence:} As in the previous case it will check if there’s an error. If there’s no error it will update the field of the database with the variables of \textit{compose-frase.php}.

\textbf{Case Delete Sentence:} The engine deletes the sentence in the database

\textbf{Problems we had:} A problem has been found during the development of this part. The problem was that the random number that is the id of a sentence that had previously existed, wasn’t in the database. This happened because when a row is deleted in MYSQL the id is never used again. It has been solved by looking first if there is any sentence with the random id, in case there isn’t the process will be repeated again.
**Web Radio:**

The Web Radio provides to the visitors the possibility to listen the music that the band plays. We use a free MP3 player call **XSPF Web Music Player** (17) which is really easy to use and configure using an XML file. If we click in section Radio Ill Capo a new window of the explorer appears with the player.

**radio.html:** Is the hosting of the MP3 player. This is inserted by:

```html
<object type="application/x-shockwave-flash"
</object>
```

**lista.xspf:** Is a XML template which contains the location and other parameters.

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<playlist version="0" xmlns = "http://xspf.org/ns/0/">
    <trackList>   <track>
        <location>http://www.illcapo.com/audio/desde la luz directo.mp3</location>
        <image>http://www.illcapo.com/audio/desdelaluz.jpg</image>
        <annotation>Ill Capo - Desde La Luz (directo)</annotation>
        <info></info>
    </track>   <track>
        <location>http://www.illcapo.com/audio/intro directo.mp3</location>
        <image>http://www.illcapo.com/audio/intro.jpg</image>
        <annotation>Ill Capo - Intro (directo)</annotation>
        <info></info></track>
    </trackList>
</playlist>
```

---

**Image17.Code of mostrarfrasealeatoria()**

**Image18.Code of mostrarfrasealeatoria()**
Problems that we had: First we tried to make the radio with HTML, we used the tag embed, bgsound and object but the result was improper. So finally we decided to use flash but the problem was that we did not know anything of flash programming. So we decided to use an existent script. The other problem that we found was that if a user click in any part of the page the MP3 was reset to the beginning of the playlist. We solved this putting the MP3 player into a new emerging window.
This application provides to the administrators of the web site the chance to insert new concerts in a calendar. The past concerts are deleted automatically; by the other hand we can insert a notification day for the users to mail them a notification about the concert this day automatically too. The concerts administrator needs some documents to work (see Image11):

- **actuaciones.php**: Shows us all the concerts that are programmed into the database ordered by date. Also it checks if there is any past concert in the database to be deleted.

- **agenda.php**: It’s divided in two parts. The first one is a form to insert a new concert. However this part of the site is only visible for the administrator. Because of that it has a filter that is the responsible to redirect the normal users to an error page if they try to see this form. In the form it will send all the information to the `transact-concierto.php` using POST method visible variables such as: `day`, `day_of_new`, `time`, `place`, `address`, `ticket_price`, `capacity`, and `comentary` of the concert and the `action` to send a new. Also it sends the invisible variable called `id_concierto`, case the administrator
tries to edit a concert. If the administrator returns to agenda.php to edit any new item, the module will look for all the information about the concert, defined by the id of the concert previously sent by transact-concierto.php using GET method. The other part shows the rest of the concerts and if the administrator wants to edit them.

**transact-concierto.php:** Is an invisible PHP script that is the responsible to process the concerts in the database.

**Case New Concert:** In case the administrator tries to send a new concert, first the module has to check if there is any error, in any of these cases or both of them it returns to agenda.php with an error message. If there is no error it starts to execute the script to insert a new field in the conciertos table in the database filling the fields with the variable previously sent by agenda.php. Then it redirects to actuaciones.php.

**Case Save Changes:** As in the previous case it checks if there’s an error. If there is no error it updates the field of the data base with the variables of agenda.php

**index.php:** There is a default execution in the index module. This execution tries to look if there’s any concert in this day that need to be notified consulting the database for the same day. If indeed there is one, a mail message is automatically send to all the registers users who wish it. Then this concert is updated in the database to not be notified anymore.
Conclusion

Finally the project has got less things than the preliminary project suggested. But the applications of the websites have more options and they work better.

There are more things which we want to include in the future, things like a: search engine, a FLASH introduction, forms built by AJAX, and also try to integrate the website in the Web Accessibility Initiative(18).
References:


2 http://safari.oreilly.com/0596101015 (access: 2007-05-09)


13 Chapter 15 pag 687 http://www.criptored.upm.es/guiateoria/gt_m001a_en.htm (access: 2007-05-20)


18 http://www.w3.org (access: 2007-05-19)

19 http://recaptcha.net (access: 2007-05-29)
APPENDIX1

ER Graph:
**Description:**

The tables *conciertos, libro, frases* and *link_info* do not have any relationship with the other tables, they just keep information *(see Image20).*

However *galeria_categorias* and *galeria_imagenes* have a relationship n:1 because in a *galeria_categorias* could be more than one *galeria_imagenes*. Also one *galeria_categoria* could have another *galeria_categoria* in its own.

The rest of the database has as main table users. Every relationship is 1:n or 1:1. For example: one post could be in only one topic and forum, however one forum could have more than a one topic or post.
## Time Plan

<table>
<thead>
<tr>
<th>Week</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Preliminary design of the web sites</td>
</tr>
<tr>
<td>2</td>
<td>According of the design of the database</td>
</tr>
</tbody>
</table>
| 3    | According the method of programing (oop or structured)  
(Finally we chose the structured one.) |
| 4    | Make the login of the Web Site. |
| 5    | Make the forum. |
| 6    | Redesign the forum  
This is because was built in an improper way. |
| 7    | Make the user account and the specific applications  
We decided not to do the statistic system because most of the hosting has his own statistics system |
| 8    | Make the mail server |
| 9    | Make the photo album. |
| 10   | Pirify Code |
| 11   | Improve code and new solutions |
| 12   | Prepare the “Preliminär slutrapport” and try to solve all the possible problem that will have happened. |
| 13   | Make the Web Site style according CSS. |
| 14   | Make more specific applications  
We did not integrate in a Web Accessibility Initiate because we had time problems with the timetable |
| 15   | Try to solve all the possible problem that will have happened. |
| 16   | Make the content of the Web Site. |
| 17   | The same of the week 18. |
| 18 | We improve the presentation of the Web Site by Photoshop and maybe Adobe Flash. Make more specific applications |
| 19 | Make more specific applications. Prepare all the documentation. |

Image21.Actual Time Plan
APPENDIX3

List of abbreviations:

**AIX:** Advanced Interactive eXecutive. It is an operating system by IBM

**ASP:** Active Server Pages. It is a dynamic language for web applications

**CMS:** Control Management System. A web application to manage news

**CSS:** Cascading Style Sheet. It is a style sheet language used to describe the presentation of a document written in a markup language

**HP-UX:** Hewlett Packard UniX. It is Hewlett-Packard's proprietary implementation of the Unix.

**HMAC:** keyed-hash message authentication code. It is a type of message authentication code (MAC) calculated using a cryptograghic hash function in combination with a secret key.

**HTML:** HyperText Markup Language. It is the predominant markup language for the creation of web pages

**HTTP:** HyperText Transfer Protocol. It is a method used to transfer or convey information on the World Wide Web

**JSP:** Java Server Pages. It is a dynamic language for web applications

**LAMP:** Linux + Apache + MYSQL + PHP. It refers to a solution stack of software programs, commonly open-source programs, used together to run dynamic Web sites or servers.

**MAC:** Message Authentication Code. It is a short piece of information used to authenticate a message

**MAMP:** Mac + Apache + MYSQL + PHP. It refers to the Apple Macintosh operating system, Mac OS X, and a set of free software programs commonly used together to run dynamic Web sites or servers

**MD5:** Message-Digest Algorithm 5. It is a widely used cryptographic hash function with a 128-bit hash value.

**PERL:** Practical Extraction and Report Language. It is a dynamic language for web applications

**PHP:** PHP Hypertext Pre-processor. It is a dynamic language for web applications

**RIPEMD:** RACE Integrity Primitives Evaluation Message Digest. It is a 160-bit message digest algorithm

**SHA1:** Secure Hash Algorithm 1. It is a hash function

**W3C:** World Wide Web Consortium. It is the main international standards organization for the World Wide Web (W3)

**WAMP:** Windows + Apache + MYSQL + PHP. It refers to a solution stack of software programs, commonly open-source programs, used together to run dynamic Web sites or servers.
APPENDIX5

“Questions at the seminar”

1. Do you have any protection against scripts/program that generates fake use accounts?

Not before but now the web site is integrated in the reC60aptcha project (19), so now the website has a program who is able to know if the user is a human or a computer by means of colorful images with distorted text at the bottom of Web registration forms.

2. When a user uploads a picture, does an administrator have to approve the picture or can the user upload whatever picture he/she wants?

Now only the administrators are the only ones who have this skill.

3. The user passwords are stored encrypted, is any other information stored encrypted?

The email address is not encrypted but the user can choose if it is public or not.

4. When a user wants to create an account, is there any control for a valid email address? For instance, does the server send a email with a verification link the user have to click on to activate the account?

For the first one the web site use the stuff of the question1 and about the activate account this option was not implemented because nowadays this is not useful because is easy to create a temporal email address.