



INTERDISCIPLINARY PROJECT

Computer & Simulation Engineering
St. Pölten University of Applied Sciences

OptWood database application

Completed under the supervision of
Prof. Thomas Schrefl and Dr. Simon Bance

Completed by
Rubén Bermúdez Pareja
cs090001

St. Pölten, on of 2010

Signature

Declaration

I declare that:

- this thesis is all my own work, that the list of sources and aids is complete and that I have received no other unfair assistance of any kind,
- this thesis has not been assessed either in Austria or abroad before and has not been submitted for any other examination paper.

This piece of work corresponds to the work assessed by the appraiser.

.....
Place, Date

.....
Signature

Abstract

New European standards require a significant improvement of finger-joints (to make them consistent) in wooden lamellas. Industries have to look for new solutions to strengthen finger joints. Among other techniques such as modified glues, the optimization of the finger-joints profile geometry is one means to achieve better connection in wooden constructions.

OptWood aims to develop a computational method to perform profile optimization for finger joint geometries. Currently finger-joints are mostly investigated using experimental methods and commercial finite element packages. These previous work focuses on the characterization of existing finger-joints. In this project advanced mathematical algorithms and simulation models will be developed that allow the optimization of finger-joint geometries taking into account both existing and newly developed European standards and production cost efficiency.

A prerequisite for any optimization is a fast numerical solution technique for the direct problem which can be effectively solved using the fast boundary element method. A short term objective is the development of mathematical methods for optimizing the profile geometry so that the stress distribution is homogeneous. The research project is a joint effort between St. Pölten University of Applied Sciences Research Ltd. and Holzforschung Austria¹ and focuses on combining fast boundary element methods and stochastic optimization techniques to minimize the stress distribution in finger joints. The mid-term objective is the development of software for multi-objective optimization together with industrial partners. Long-term objectives are the use and application of the developed software for engineering parts in wooden constructions.

This joint effort between both institutions requires shared databases, where members can access and contribute new results and parameters about the finger-joint simulations. That's why OptWood web site is necessary. A platform where the new results can be stored well structured and organized. But it should be a secure place where only a restricted number of persons can access and modify the information in it, and this is possible with a typical user names and passwords database. In this web site there is also an area to visualize the information stored.

The shared data should also be accessible to web applications that make quickly simulations over the web. It should allow the OptWood investigators to compare short numerical results quickly without downloads or computationally demanding simulation programs.

All the code developed about OptWood web application is included in Appendix I and Appendix II. They are also all the code is also included in the webapplication.zip file, this file is available in the electronic version of the project.

¹ Research institute and accredited testing and inspection body of the Austrian Society for Wood Research (ÖGH).

List of Figures & Images

Figure 1. Industrial manufacturing process of glulam products.....	9
Figure 2. Stress analysis of the basic finger joint model under a tensional force. Inhomogeneities in the stress distribution can be assessed for different finger joint geometries.....	13
Figure 3. Role of php & MySQL (image: check out [6])	10
Figure 4. Screen shot of the login page.....	16
Figure 5. HTML form for user data input during login (<i>accesso.html</i>).....	17
Figure 6. code of <i>check_input()</i> function.....	19
Figure 7. Users table structure (from userdata database).....	20
Figure 8. Example of the content of a users table row (from userdata database).....	21
Figure 9. Screen shot of the main page <i>MAINoptwood.php</i> (top view).....	22
Figure 10. Screen shot of the main page <i>MAINoptwood.php</i> (bottom view)....	22
Figure 11. Screen shot of the logout page <i>logout.php</i>	23
Figure 12. Screen shot of a login failed in <i>accesso.php</i>	24
Figure 13. Screen shot of the registration page <i>registrar.php</i>	24
Figure 14. Screen shot of a successful registration page <i>registrar.php</i>	25
Figure 15. Screen shot of a failed registration page (some unfilled field).....	25
Figure 16. Screen shot of a registration failed page (wrong password).....	26
Figure 17. Screen shot of a registration failed page (wrong user or e-mail)....	26
Figure 18. Screen shot of wood and glue table structure in phpMyAdmin.....	27
Figure 19. Screen shot of glue variety rows in phpMyAdmin (one row isn't already fill because the user jut added the variety glue name).....	28
Figure 20. Screen shot of wood data example insertion.....	28
Figure 21. Screen shot of successful insertion page <i>insertglue.php</i>	28
Figure 22. Screen shot of glue data example insertion.....	29
Figure 23. Screen shot of successful insertion page <i>insertglue.php</i>	29
Figure 24. Screen shot of the displaying area.....	30
Figure 25. Content of <i>selectdata.js</i> file.....	31
Figure 26. <i>GetHttpObject function</i> code for .js files is case of incompatible browsers.....	32
Figure 27. Screen shot of all the current varieties of glue database displaying, with completed data for the first tow entries.....	33
Figure 28. Screen shot of Polyurethane variety of glue row displaying.....	33

Contents

Declaration.....	3
Abstract.....	4
List of figures.....	5
1 Introduction.....	9
1.1 Optwood project.....	10
1.2 Motivation for the web application.....	11
2 Tools.....	12
2.1 Which technology do we use?.....	12
2.2 Used technologies.....	13
2.3 LAMP Server (Linux+Apache+MySQL+PHP).....	14
2.3.1 phpMyAdmin 3.2.4.....	14
3 Method.....	15
3.1 Processing.....	15
3.2 Login & Registration page.....	17
3.2.1 User session and cookies.....	17
3.2.2 Security preventions.....	18
3.2.2.1 Prevent SQL injections.....	18
3.2.2.2 check_input() function.....	19
3.2.2.3 htmlentities() function.....	19
3.2.2.4 mysql_error() function.....	19
3.2.2.5 \$_POST[] and GET[].....	20
3.2.3 User data storage.....	20
3.2.3.1 userdata database structure.....	20
3.2.4 Login example.....	21
3.2.4.1 Successful login.....	21
3.2.4.2 Failed login.....	23
3.2.5 Registration example.....	24
3.2.5.1 Successful registration.....	24
3.2.5.2 Failed registration.....	25
3.3 Main page.....	26
3.3.1 Wood and glue database structure.....	27
3.3.2 Data insertion.....	28
3.3.3 Data modification.....	29
3.3.4 Displaying of data.....	39
3.3.4.1 AJAX.....	30
3.3.4.2 Whole database displaying.....	32
3.3.4.3 Single row displaying.....	33
4 Conclusions.....	34
4.1 Future work.....	34
4.1.1 How do we add more databases on the website.....	34
4.1.2 Send form with AJAX.....	35
4.1.3 Parameterized SQL queries.....	35
4.1.4 Simulation interface.....	35
4.1.5 Database duplication.....	35
5 Appendix I. Access.....	37

6	Appendix II. Main page.....	45
7	Bibliography.....	56

1 Introduction

Austrian companies are the second largest producers of wooden lamellas and finger jointed timber in Europe with yearly production reaching 1,000,000 cubic meters. Finger joints are a key technology for one-piece construction. Consistency is essential to make wood economically competitive with other materials. New European standard require higher finger joint strength of lamellas. It is an open question in wood industry how the higher strength require by these standard can be met. The OptWood project directly addresses this point by developing mathematical methods for optimizing the geometrical profile of finger-joints.

Meeting the new standards (EN 14080 – Timber structures – Glued laminated timber (glulam, see Figure 1) – Requirements; EN 15497 – Finger-jointed structural timber, [1]) will be essential for wood industry to keep a share in the high end construction market. Currently glulam amounts for 30 to 35 percent in a 1,12 billion EUR market. This market share could be lost if finger-joints remain an optimized. The use of simulation will give a cost benefit as compared to the traditional experimental approach of stress testing. Solutions not reachable with experiments alone may be found through mathematical optimization.

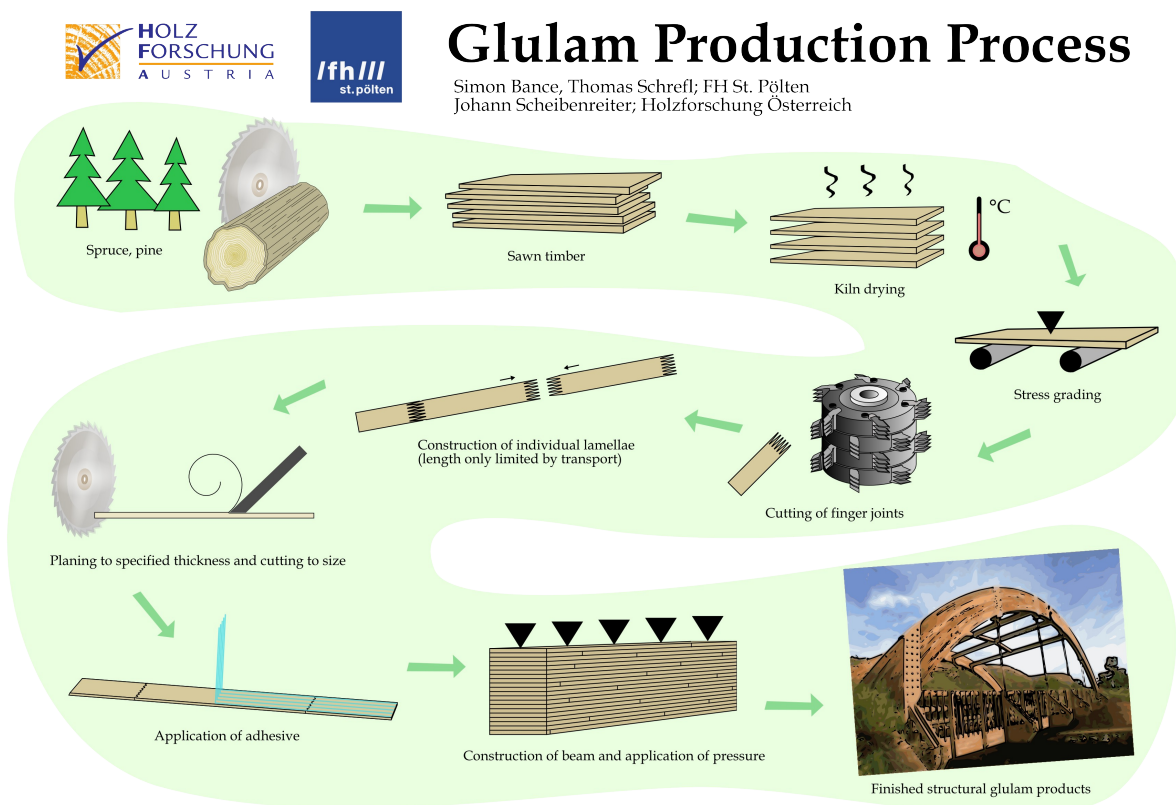


Figure 1: Industrial manufacturing process of glulam products.

1.1 OptWood project

The materials database will store material properties for use in numerical stress analysis simulations. These simulations, using the finite element (FE), check [2] from bibliography, and boundary element (BE) methods, check also [2], aim to improve fingerjoint strength in glulam lamellas by improving the fingerjoint geometry, see Figure 2.

Glulam failure can occur when the fingerjoint is under tension, as the geometry leads to a concentration of mechanical stress at specific locations. By optimising the fingerjoint design it will be possible to homogenise the stress distribution around the fingerjoint, leading to lower peak stresses in the lamella.

The FE/BE simulations, specifically, use a finite element model of a fingerjoint and perform static stress analysis under a specified tensional force at one end. The other end is fixed in position.

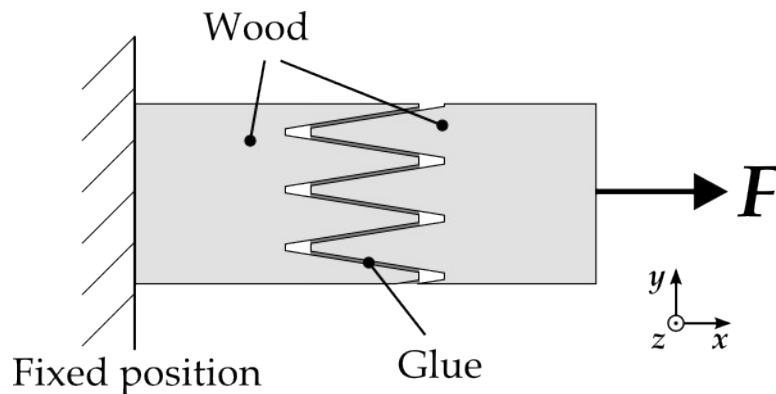


Figure 2: Stress analysis of the basic finger joint model under a tensional force. Inhomogeneities in the stress distribution can be assessed for different finger joint geometries.

Material properties are fed into the simulation to provide mechanical information on the elastic (Young's) and shear moduli, which are related to the elasticity along relevant axes. The Poisson ratio describes how the material expands or contracts under external forces.

The OptWood project has three main aims:

- It shall demonstrate the advantages of modern mathematical techniques for solving industry related, economically relevant questions.
- It will establish a curriculum within the Master Program in Simulation that is relevant for local industrial partners.
- It will trigger new collaborative research projects for the further development of simulation software, joint teaching and training of students, and collaborative FTEI projects.

1.2 Motivation for web application

It is expected that the simulations will create much useful data for the improvement of the finger-joint profiles. St. Pölten University of Applied Sciences and Holzforschung Austria need to share the research information somehow, and the best way to do that is using some shared online database.

A fast and secure way to share that data is with a website where is possible to visualize and modify the data, but accessible only to authorized users. It is essential to have restricted access to this website. That's why it is necessary to have a login website with preventions against hacking. For example, the configuration of the configuration file of the server.

The web application will be database-driven and should be scalable in order to ultimately allow simulations to be controlled in the browser².

2 A **web browser** is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An *information resource* is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content. Hyperlinks present in resources enable users to easily navigate their browsers to related resources. [3]

2 Tools

We choose a very standard set of tools to build our database-driven application.

2.1 Which technology do we use?

HTML (Hypertext Markup Language) is the backbone of the worldwide web, check [4].

CSS (Cascading Style Sheets) is a style language that defines layout of HTML documents, and provide a convenient way to define the appearance of a web page, separating content from styling and appearance, check [4] from bibliography.

For our database we choose MySQL [5]. MySQL is widely used as a relational database management system (RDBMS) and powers many content heavy applications on the weben 1547.

Not only is MySQL the world's most popular open source database, it's also become the database of choice for a new generation of applications built on the LAMP (check [5]) stack (Linux, Apache, MySQL, PHP / Perl / Python.) MySQL runs on more than 20 platforms including Linux, Windows, Mac OS, Solaris, HP-UX, IBM AIX, giving the kind of flexibility that puts users in control.

If it is required to embed dynamic text into a web page, PHP is extremely useful for integrating web pages with databases and that's what is required for the web application that is presented in this report.

Using PHP scripting and MySQL enables programmers to create portable applications that run on just about any computer, regardless of operating system (Figure 3). PHP has thousands of programming functions to facilitate almost any task.

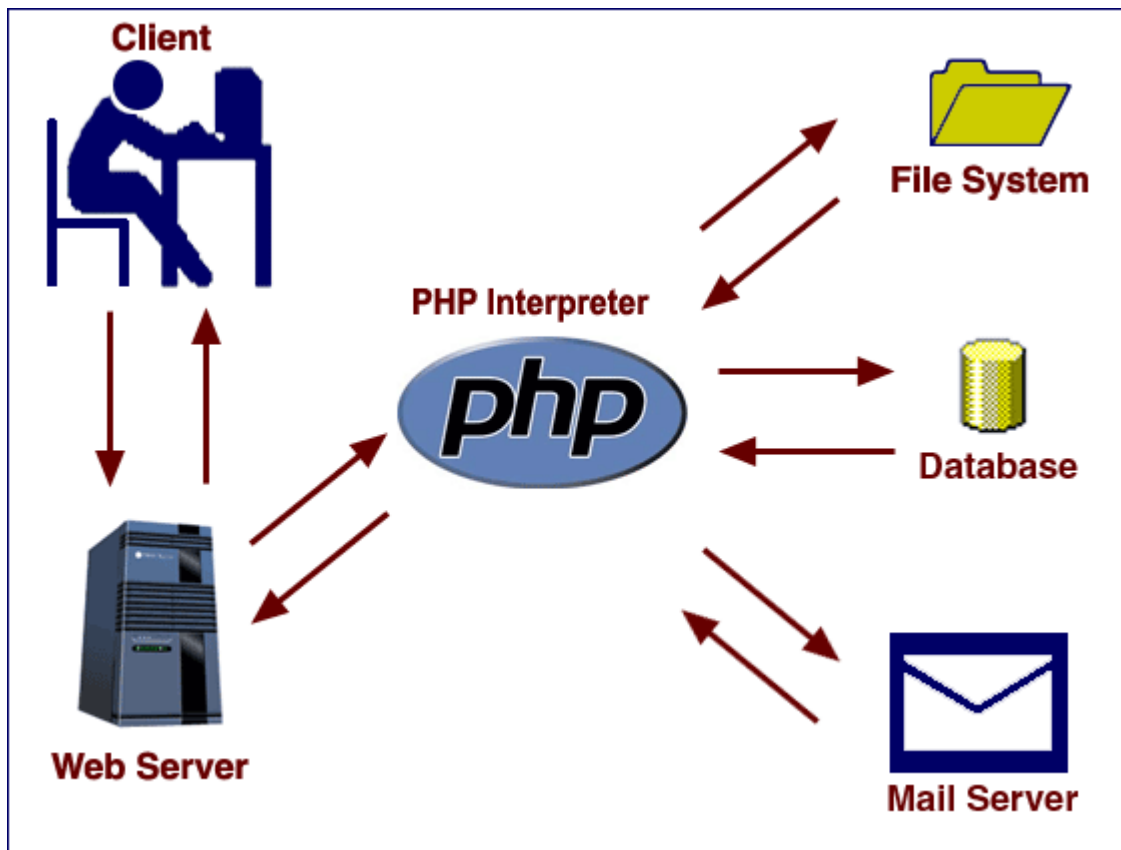


Figure 3. Role of php & MySQL (image: check out [6])

Modern web users have come to expect sites that do not require a page reload with every data request from a server. For that purpose there is a group of interrelated web development techniques used on the client-side (due to the using of the JavaScript) to create interactive web applications. AJAX (shorthand for asynchronous JavaScript and XML) allows web applications to retrieve data from the server asynchronously in the background without interfering with the display and behaviour of the existing page.

2.2 Used technologies

In the above section we have introduced a list of technologies:

- PHP
- MYSQL
- HTML
- CSS
- AJAX

2.3 LAMP Server (Linux+Apache+MySQL+PHP)

LAMP (check [5]) is an open-source web development platform, also called a Web stack, that uses Linux as the operating system, Apache as the Web server, MySQL as the RDBMS and PHP as the object-oriented scripting language. Perl or Python is often substituted for PHP. An alternative version is available for Microsoft Windows called XAMPP (X meaning cross-platform, Apache HTTP server, MySQL, PHP, Perl), check [7] from bibliography.

2.3.1 phpMyAdmin 3.2.4

To handle a database in the period of development is easier to see and modify it graphically. For this LAMP includes phpMyAdmin, which run on a LAMP server.

For more information check out [8] and from the bibliography.

3 Method

We approach the development in the following order:

1. Website and database design.
 1. Wood database: contains a table with varieties of wood
 2. Glue database: contains a table with varieties of glue
2. Installation of LAMP.
 1. Installation PHP5
 2. Installation APACHE
 3. Installation MySQL
3. Development of the main page PHP backend.
4. Create a login/registration page to access the application.
5. Insert the security preventions (SQL injections and malicious code).
6. Testing and improvement.

3.1 Processing

All the pages of the web application on this project include the *standard.html* file and *optwood.css* which determine the graphical style of all the pages. It is important as well to plan the structure of the databases that we will handle: wood, glue and user data. We will connect to them to send and receive information, like the login data for example.

All the website files must be in the LAMP home directory, in order to execute queries.

The first page that a user sees is the login page *login.php* (Figure 4) where they can login with their user name and password on the form in *login.html* (included on *login.php*). If it is written correctly the session will be initialized. The sessions work by creating a unique id (UID) for each visitor and stored variables based on this UID. The UID is either stored in a cookie³ or is propagated in the URL⁴.

3 A **cookie**, also known as a **web cookie**, **browser cookie**, and **HTTP cookie**, is a text string stored by a user's web browser. A cookie consists of one or more name-value pairs containing bits of information, which may be encrypted for information privacy and data security purposes. [9]

4 In computing, a **Uniform Resource Locator (URL)** is a Uniform Resource Identifier (URI) that specifies where an identified resource is available and the mechanism for retrieving it. In popular usage and in many technical documents and verbal discussions it is often incorrectly used as a synonym for URI. The best-known example of a URL is the "address" of a web page on the World Wide Web, e.g. <http://www.example.com>. [10]

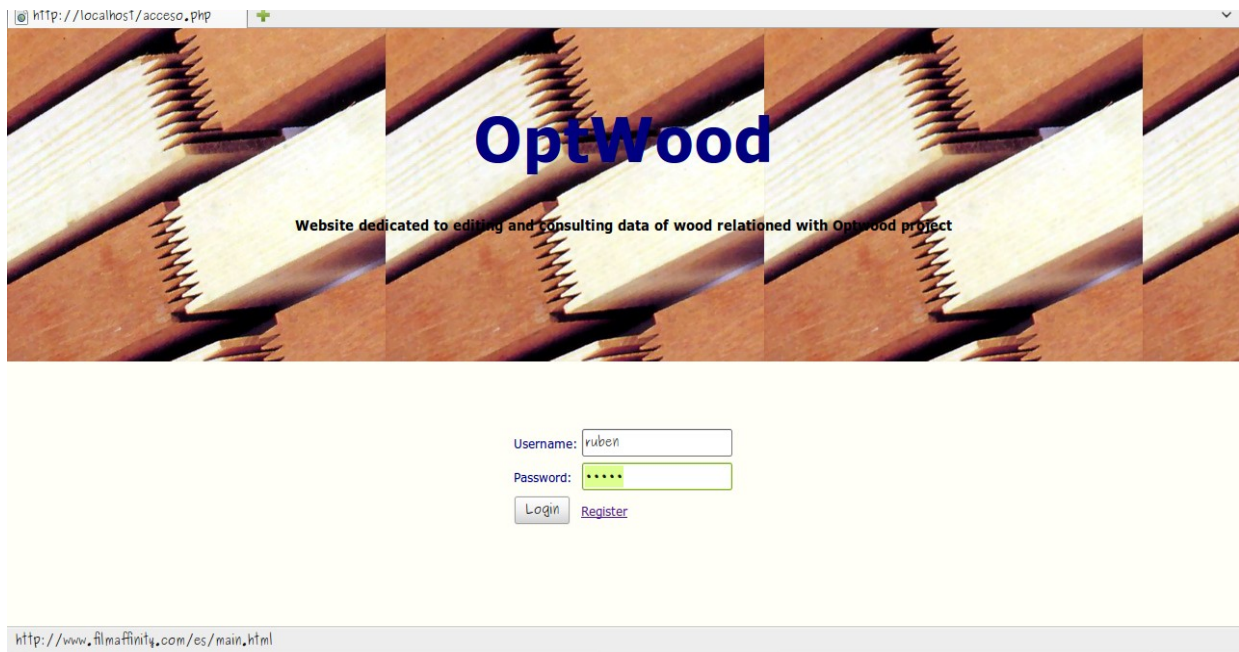


Figure 4: Screen shot of the login page

If the information written by the user in the login form doesn't match with a correct user name and password stored on the database, it will show an error message.

It is also possible to access the registration page through the link to *registrar.php*. This includes *registrar.html*, file which has a registration form with four fields:

- new user name
- password (which must be with more than 4 characters)
- repeat the password
- and the e-mail

It is also possible to go back to the login page through the link “Login”, which take you to *login.php* again.

Once we have achieved a successful login we access *MAINoptwood.php* where there are two forms to fill, one for the wood database and the other for glue. There are two kinds of visualizations of those tables: on it is possible to choose the whole database with two radio buttons; on the other one it is possible to choose specific data, that we can select with an option list. This will be demonstrated in section 3.3.

Some of the form fields are essential for the database entry, so we make it mandatory that they are filled. When we click the “Add” button it guides us to *insertwood.php* or *insertglue.php* where the insertion data code is processed and displays a notification telling us information about the insertion data result, next to a “back” button that take us to the main page again.

When the user want to end the session there is a “Logout” link in the bottom of the main page. This link take us to *logout.php* file where the session is destroyed and a link is shown that take us back to the login page.

3.2 Login & Registration page

Figure 3 shows the HTML form included on the *acceso.php* page that the links to the registration page:

```
<html>
<body>
<!-- the form is built in a table with 3 rows <tr> and 2 columns
<td> -->
<form name='input' action='' method='post'>
<table align="center">
<tr>
  <td>Username:  </td>
  <td> <input type='text' name='username' size='20' > </td>
</tr>
<tr>
  <td>Password:  </td>
  <td> <input type='password' name='password' size='20' > </td>
</tr>
<tr>
  <td><input type='submit' value='Login' ></td>
  <td><a href='registrar.php'> Register </a></td>
</tr>
</table>
</form>

</body>
</html>
```

Figure 5. HTML form for user data input during login (*acceso.html*)

On pages where a user can enter data it is important to have security preventions against malicious code and guarantee that the user data is safe.

3.2.1 User session and cookies.

The first point is the use of sessions and cookies, because with them we guarantee that the user data is not stored for a long time in the browser. That would be dangerous because it could be used by a malicious user or hacker over an insecure network.

The session starts when we call the *session_start()* function, which must be used before any kind of information is sent to the browser. Any query won't be executed until the input variable of the form code is initialized. When the login

is successful the `$_SESSION[]` variable is initialized with the user data in *accesso.php*.

Cookies are useful to store the session information. In our case we only want to keep the user name.

3.2.2 Security preventions.

One important step is to verify that the directive *safe_mode* is ON and *register_globals* is OFF in the file *php.ini* (which normally is in the current working directory of PHP, in the path designated by the environment variable `PHPRC`, and in the path that was defined during compilation). Also we have to unset the `777` permission for the folders once we have uploaded the website to the server because `777` permission allows everyone to read, write or execute the important configuration files and databases.

The PHP safe mode is an attempt to solve the shared-server security problem [5]. It is architecturally incorrect to try to solve this problem at the PHP level, but since the alternatives at the web server and OS levels aren't very realistic, many people, especially ISP's, use safe mode. When *safe_mode* is ON, PHP verifies if the owner of the current script coincides with the owner of the file, which will be processed by a function file.

When *register_globals* is OFF the variables from the form aren't show in the URL and it is impossible to change the variable in the URL.

We can check the path of *php.ini* or the status of all those directives with *phpinfo()* function. It outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

The security preventions taken are:

3.2.2.1 Prevent SQL injections

SQL injection [11] refers to the act of someone inserting a MySQL statement to be run on your database without your knowledge. Injection usually occurs when you ask a user for an input, like their name, and instead of a name they give you a MySQL statement that you will unknowingly run on your database.

PHP has a specially-made function to prevent these attacks; *mysql_real_escape_string()*. This function escapes special characters in a string for use in an SQL statement. The following characters are affected:

- `\x00`

- \n
- \r
- \
- '
- "
- \x1a

For any kind of login/registration query in the database it will be necessary to use this function. Check out [12] , [13] and [14] from the bibliography where there is examples of the processing of this function.

3.2.2.2 check_input() function

```
function check_input($value)
{
// Stripslashes
if (get_magic_quotes_gpc())//Returns 0 if magic_quotes_gpc is off, 1
otherwise.
{
$value = stripslashes($value);//removes backslashes added by the
addslashes() function.
}
// Quote if not a number
if (!is_numeric($value))
{
$value = "'" . mysql_real_escape_string($value) . "'";
}
return $value;
}
```

Figure 6: code of check_input() function

This function does two things: it removes backslashes, This is done already by *mysql_real_escape_string()*, but they could appear again if some string is escaped by error twice. But at the same time, with the second *if* , it checks if the string wasn't encrypted (with *md5()* or *sha1()* function) thanks to the function *is_numeric()*, because if they're encrypted they will never be a numerical string.

3.2.2.3 htmlentities() function

This function converts characters to HTML entities and is just protocol security prevention. It used always before the login/registration queries, check this function in [14] from the bibliograpy.

3.2.2.4 mysql_error() function.

The *mysql_error()* function returns detailed information when an error occurs in the PHP code. This is a great help during development, but can also aid someone with malicious intent by providing clues regarding security weaknesses.

Therefore, it is better to avoid this function after the development stage.

3.2.2.5 \$_POST[] and \$_GET[]

There are a member of methods for taking information from HTML forms [4]. To catch the data from the different pages of our website is useful the \$_GET method, but it has limits on the amount of information to send (max. 100 characters). That's why in login and registration pages we use \$_POST method because the information will be encrypted and adapted to HTML standard and the strings generated could overload the 100 characters. This method is also invisible to others.

3.2.3 User data storage

It is necessary to encrypt the password of each user because because if a hacker were to gain access to the user database it would be impossible to know the original password, it is impossible to decrypt them.

Is not useful to encrypt the name or e-mail address because md5 hashing is irreversible and we would not be able to reobtain this useful information.

There are two functions: *md5()* and *sha1()*. They are very similar but there are studies that prove that is more difficult to crack *sha1()* function. Check out [15] in bibliography.

3.2.3.1 *userdata* database structure

User data is stored within a table called *users*. Figure 7 gives a screen shot from phpMyAdmin of the users table, doing with the properties of each field:

	Field	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	id	int(20)			No	None	auto_increment
<input type="checkbox"/>	user	varchar(20)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	password	varchar(40)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	email	varchar(45)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	date	date			No	None	

Figure 7: Users table structure (from *userdata* database)

None of the data fields can be Null, so if some field in the registration form is empty any data could be written in the database. That's why the table is filled with none by default. Also all data fields have the default collation⁵. Anyway, we

⁵ Collation refers to the character set used to store data in text fields and is necessary to provide support for all of the many written languages of the world. MySQL 4.1 added the ability to override the default system collation at the

check that all fields in the new user registration form are filled.

The Id field is the identification number for each row and its integer type attribute has maximum 20 digits. This provides a very big number of rows useful since OptWood is big project with many collaborators. The attribute AUTO_INCREMENT is used to create an unique id number for each new row. This attribute, actually, is not useful in the code of this project because is more interesting for multiple row insertions. But it is good to have this attribute activated for futures work on this project because it helps to have it in mind.

The user type attribute will be a string of 20 character (varchar(20) enough for a user name) and e-mail will be 45 characters in case of a very long e-mail.



The screenshot shows a table with columns: id, user, password, email, and date. The password column has a tooltip that says "The length is 40 cause the password is stored in sha1() format". Below the table, there are controls for "Check All / Uncheck All With selected:" and icons for edit, delete, and refresh.

	id	user	password	email	date
<input type="checkbox"/>	1	ruben	c33367701511b4f6020ec61ded352059	rbpareja@gmail.com	2010-06-08

Figure 8: Example of the content of a users table row (from userdata database)

The format of the date stored is the default format returned by the `date()` PHP function, which is YY-MM-DD where YY=year, MM=month and DD=day.

In Figure 8 we can see that the content of the password field is a string of 40 characters, returned by the `sha1()` function, which always returns an array with the same length independently of the original password. That's why the password type attribute is `varchar(40)`.

3.2.4 Login example

To access the login page it is necessary to enter in the browser the URL of our website, in this case: `http://localhost/acceso.php`. When the website is uploaded to some server it is necessary to write the URL of `acceso.php` on the server.

There are three possible outcomes: successful login, failed login and registration.

3.2.4.1 Successful login

The login page redirect to the main page without any kind of notification (Figure 9 and Figure 10).

database, table, and field level.

The default collation in MySQL, `latin1_swedish_ci`, works fine for English because English contains no special characters such as accents. phpMyAdmin allows you to edit the collations of your database, tables, and fields if you desire to support languages other than or in addition to English. [16]

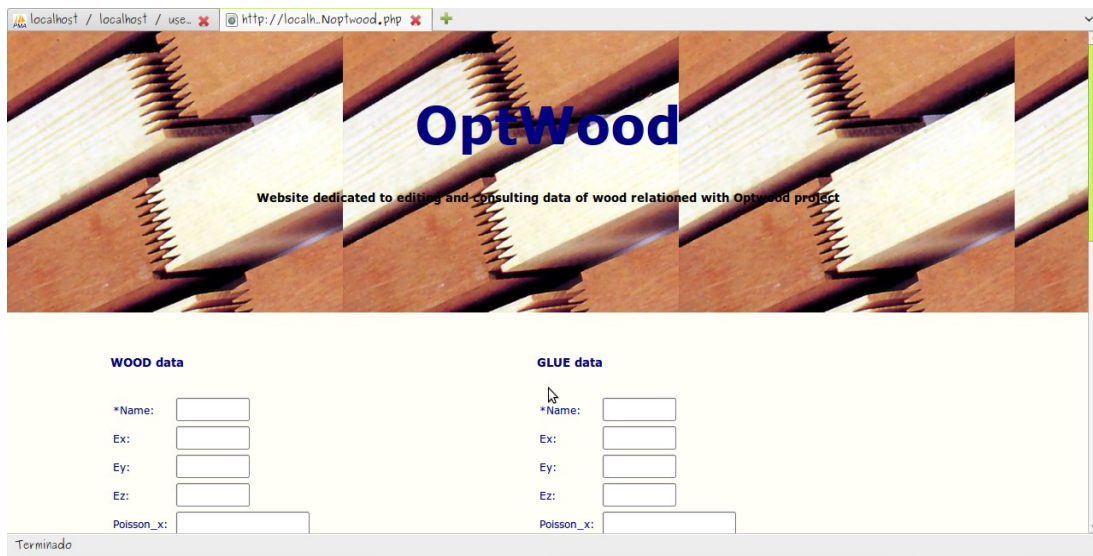


Figure 9: Screen shot of the main page MAINoptwood.php (top view)

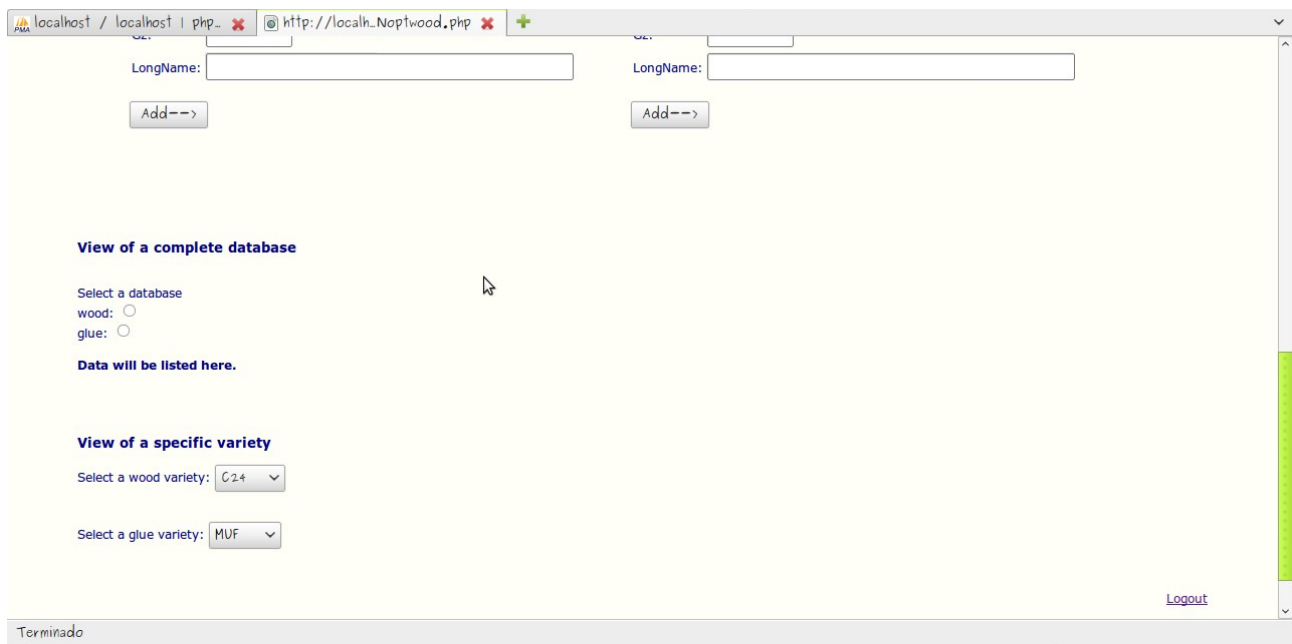


Figure 10: Screen shot of the main page MAINoptwood.php (bottom view)

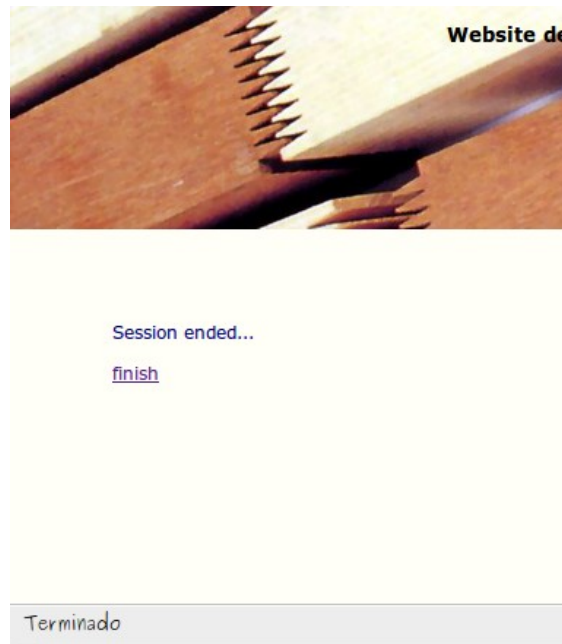
If the user doesn't click on the logout link at the bottom of the page and tries to access login page the URL `http://localhost/acceso.php` will redirect to `http://localhost/MAINoptwood.php`. That's done checking if the variable `$_SESSION[]` is still set, here is the code that allows it:

```
if(isset($_SESSION["k_username"]))  
    echo '<SCRIPT LANGUAGE="javascript">
```

```
location.href = "MAINoptwood.php";  
</SCRIPT>';
```

To see the whole code check out Appendix I. Access in *acceso.php* file.

When the users log out they see the *logout.php* page with the message (Figure 11). The session is destroyed with *session_destroy()* function, otherwise the user data would stay long time stored in the browser and this is security risk. There is a link that take us to *login.php*.



*Figure 11: Screen shot of the logout page
logout.php*

3.2.4.2 Failed login

In Figure 12 a notification message is shown.

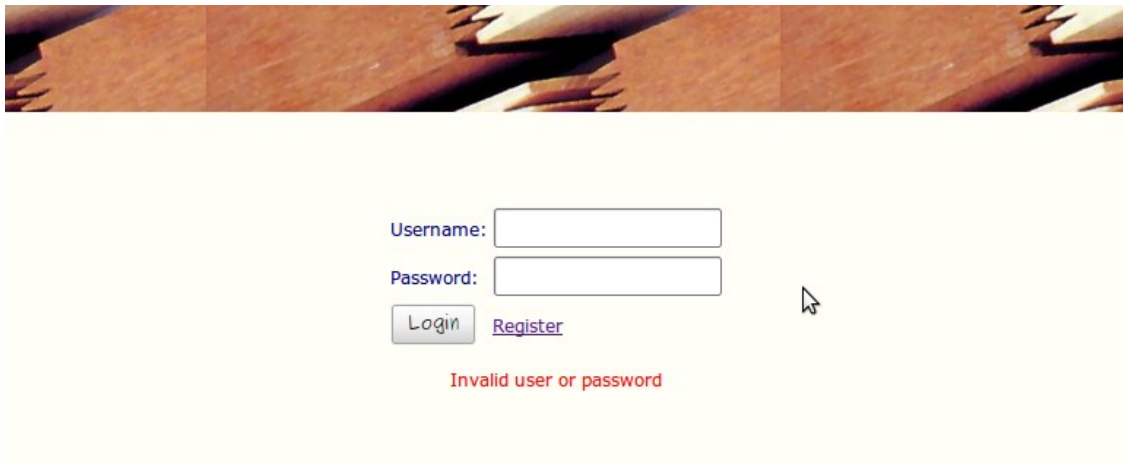


Figure 12: Screen shot of a failed login in *acceso.php*

3.2.5 Registration example

On the registration page (Figure 13) there are three possible outcomes from user form submission: successful registration, failed registration and redirection to the login page (a link that redirects to *acceso.php*).

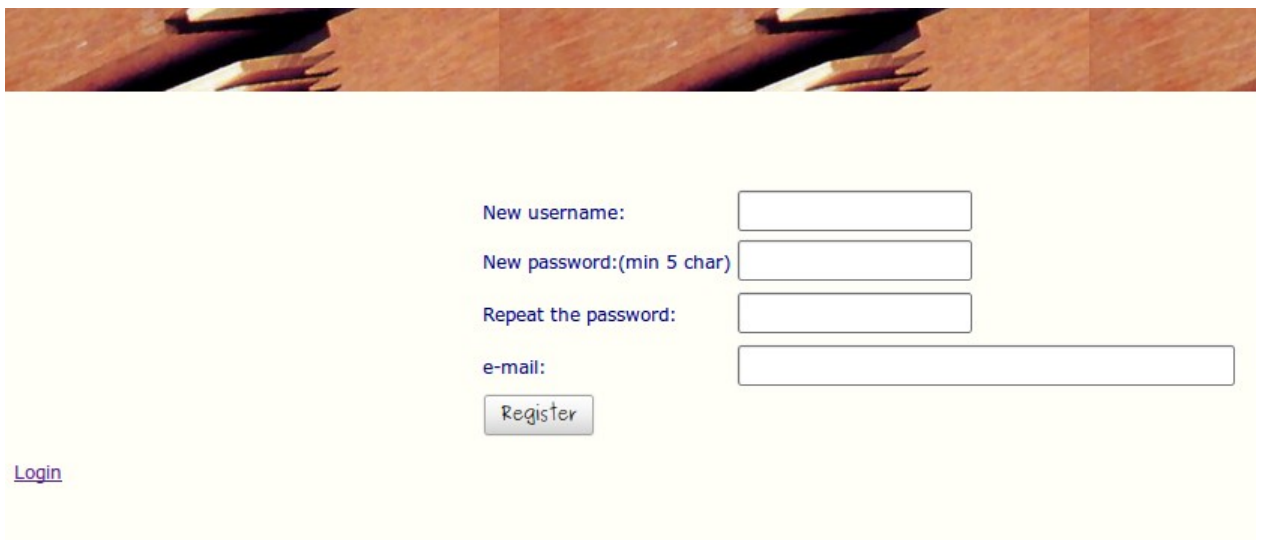


Figure 13: Screen shot of the registration page *registrar.php*

3.2.5.1 Successful registration

The registration is successful if the user name or e-mail aren't already stored in the user database because otherwise a duplicated e-mail or user name would create conflicts in any kind query; or if the password is more than 5 characters (we force the user to create a password with minimum length because longer

passwords are more secure). In any case a message will be shown (Figure 14).

In Appendix I. “Access” we can see the *if...else* condition which makes possible the successful login.

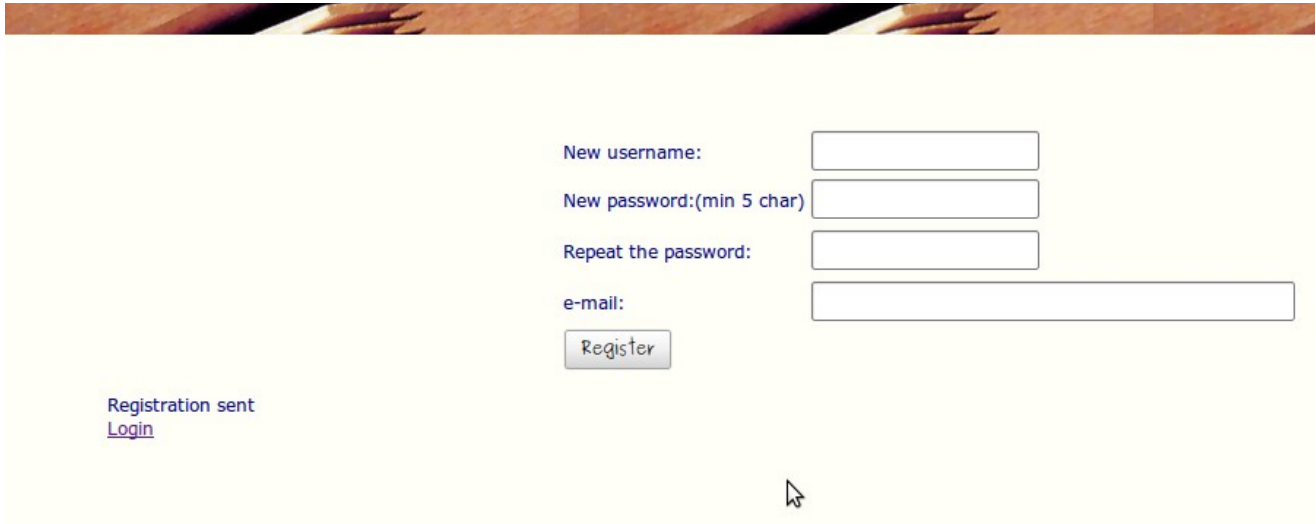


Figure 14: Screen shot of a successful registration page registrar.php

3.2.5.2 Failed registration

There are a number of reasons for a registration failure to be returned to the user

- If some field haven't been filled or the passwords don't match a message will be shown like in Figure 15.

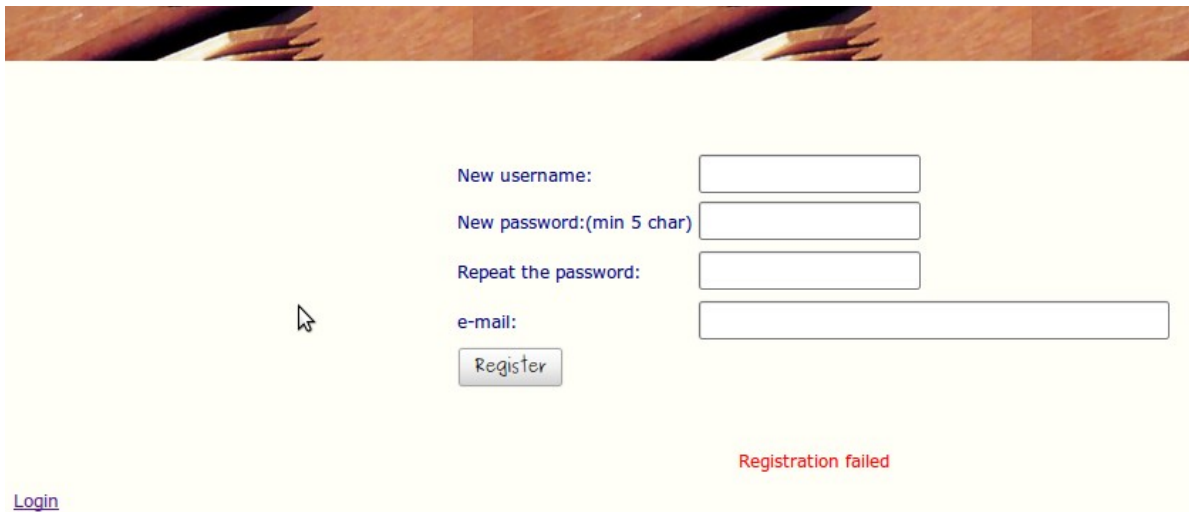
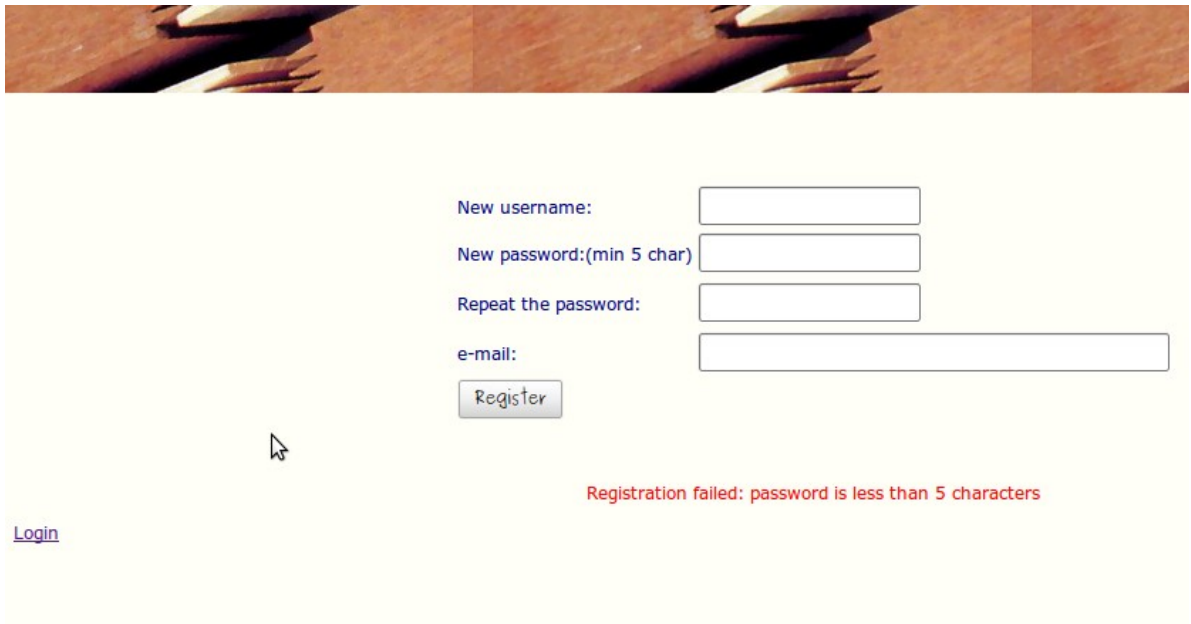


Figure 15: Screen shot of a failed registration page (some unfilled field)

- If the password is less than 5 characters (Figure 16). In the *registrar.php* code we check that the password string introduced is long enough with an

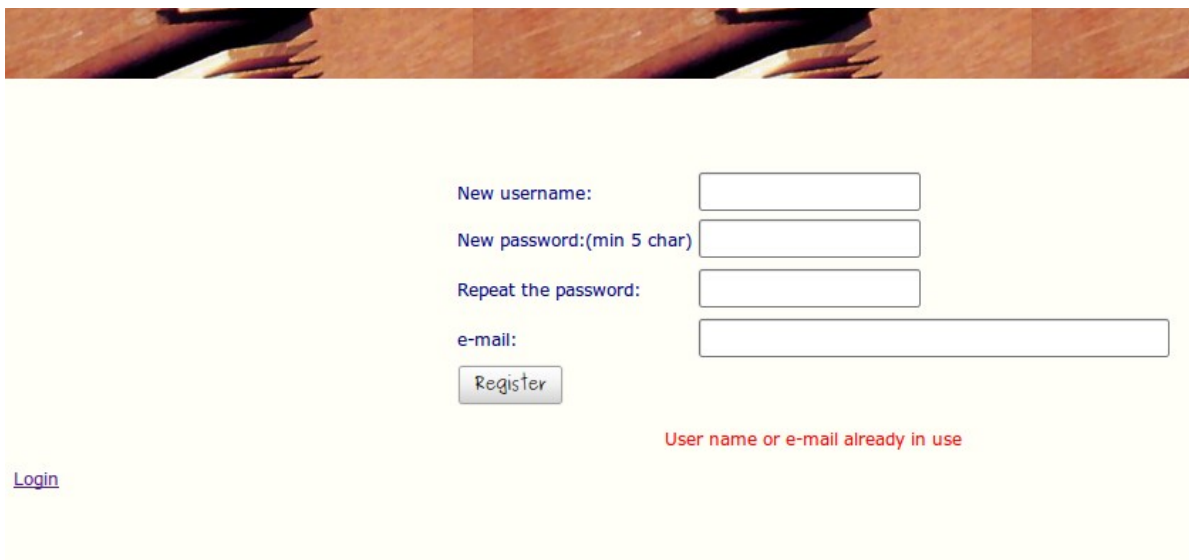
if...else condition before passing it through the sha1 hash algorithm.



The screenshot shows a registration form with the following fields: "New username:", "New password:(min 5 char)", "Repeat the password:", and "e-mail:". A "Register" button is located below the "e-mail" field. A red error message at the bottom right reads "Registration failed: password is less than 5 characters". A "Login" link is visible in the bottom left corner.

Figure 16: Screen shot of a registration failed page (wrong password)

· If the e-mail or user name already exists (Figure 17).



The screenshot shows the same registration form as in Figure 16. A red error message at the bottom right reads "User name or e-mail already in use". The "Login" link is also present in the bottom left corner.

Figure 17: Screen shot of a registration failed page (wrong user or e-mail)

3.3 Main page

This page allows access to the *glue* and *wood* material properties databases. It provides an authenticated user with a method for inserting new material information, and viewing or modifying existing data.

3.3.1 Wood and glue database structure

Each database has only one table called *Varieties* and they have exactly the same structure. There are two separated databases because it is possible to add new tables with properties of wood or glue depending on the future work of OptWood. In Figure 18 it is shown the properties of each field and in Figure 19 some example rows:

	Field	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	id	int(5)			No	None	auto_increment
<input type="checkbox"/>	name	varchar(20)	latin1_swedish_ci		No	None	
<input type="checkbox"/>	Ex	float			Yes	NULL	
<input type="checkbox"/>	Ey	float			Yes	NULL	
<input type="checkbox"/>	Ez	float			Yes	NULL	
<input type="checkbox"/>	Poisson_x	float			Yes	NULL	
<input type="checkbox"/>	Poisson_y	float			Yes	NULL	
<input type="checkbox"/>	Poisson_z	float			Yes	NULL	
<input type="checkbox"/>	Gx	float			Yes	NULL	
<input type="checkbox"/>	Gy	float			Yes	NULL	
<input type="checkbox"/>	Gz	float			Yes	NULL	
<input type="checkbox"/>	LongName	varchar(50)	latin1_swedish_ci		Yes	NULL	

Figure 18: Screen shot of wood and glue table structure in phpMyAdmin

All the fields have the default collation attribute.

The Id field can be an integer variable with 5 digits, enough for the number of materials that will be ever stored in the system. The rest of the properties are the same, as was explained in section 3.2.3.1.

The Name field is a 20 character string, enough for the acronyms that are used for the glue and wood varieties. But this field can't be null by default because the name is essential for any kind of query in the database.

The LongName is set to a 50 character string to accommodate the material's full name. This field and those of the material properties can be set to null because the databases can be completed at a later date.

The numerical fields (Ex, Ey, ...) are float type because normally they are numbers that require.

	id	name	Ex	Ey	Ez	Poisson_x	Poisson_y	Poisson_z	Gx	Gy	Gz	LongName
<	1	MUF	6300	6300	6300	0.34	0.34	0.34	2400	2400	2400	Melamine-urea-formaldehyde
<	2	PUR	470	470	470	0.3	0.3	0.3	180	180	180	Polyurethane
<	3	PRF	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 19: Screen shot of glue variety rows in phpMyAdmin (one row isn't already fill because the user jut added the variety glue name)

3.3.2 Data insertion (Figure 20)

If the name of the variety doesn't exist in the table, a new row is inserted in the table. Otherwise the row matched will be filled with the information written in the different fields of the form. The insertion result will be notified with message (Figure 21). The query (which is in Appendix II. Main page, in *insertwood.php* and *insertglue.php*) below allow the insertion of data:

```
$result = mysql_query("INSERT INTO Varieties (`id`, `name`, `Ex`, `Ey`, `Ez`, `Poisson_x`, `Poisson_y`, `Poisson_z`, `Gx`, `Gy`, `Gz`, `LongName`) VALUES ($Nrows, '$name', '$Ex', '$Ey', '$Ez', '$Poisson_x', '$Poisson_y', '$Poisson_z', '$Gx', '$Gy', '$Gz', '$LongName')", $con);
```

Figure 20: Screen shot of wood data example insertion



Figure 21: Screen shot of failed insertion page *insertwood.php*

3.3.3 Data modification (Figure 22)

The query below (which is in *insertglue.php* and *insertwood.php* files, in Appendix II). Main page allow the modification of data in case the name introduced already exist in the table:

```
$result = mysql_query("UPDATE Varieties SET Ex = '$Ex',  
Ey = '$Ey', Ez = '$Ez', Poisson_x = '$Poisson_x',  
Poisson_y = '$Poisson_y', Poisson_z = '$Poisson_z', Gx =  
'$Gx', Gy = '$Gy', Gz = '$Gz', Gz = '$Gz', LongName =  
'$LongName' WHERE name = '$name'");
```

GLUE data

*Name:

Ex:

Ey:

Ez:

Poisson_x:

Poisson_y:

Poisson_z:

Gx:

Gy:

Gz:

LongName:

Figure 22: Screen shot of glue data example insertion

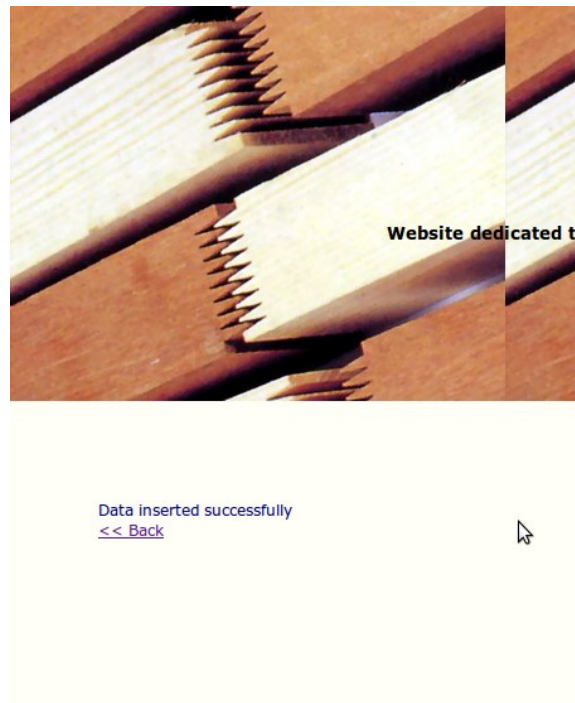


Figure 23: Screen shot of successful insertion page *insertglue.php*

The message shown in Figure 23 means that the query above have been realized successfully.

3.3.4 Displaying of data

The displaying of data (Figure 24) could be very annoying if each time that we choose on database or search some specific data the whole page has to be reloaded. The problem is solved with AJAX. In the main page there is an area where the data will be shown.

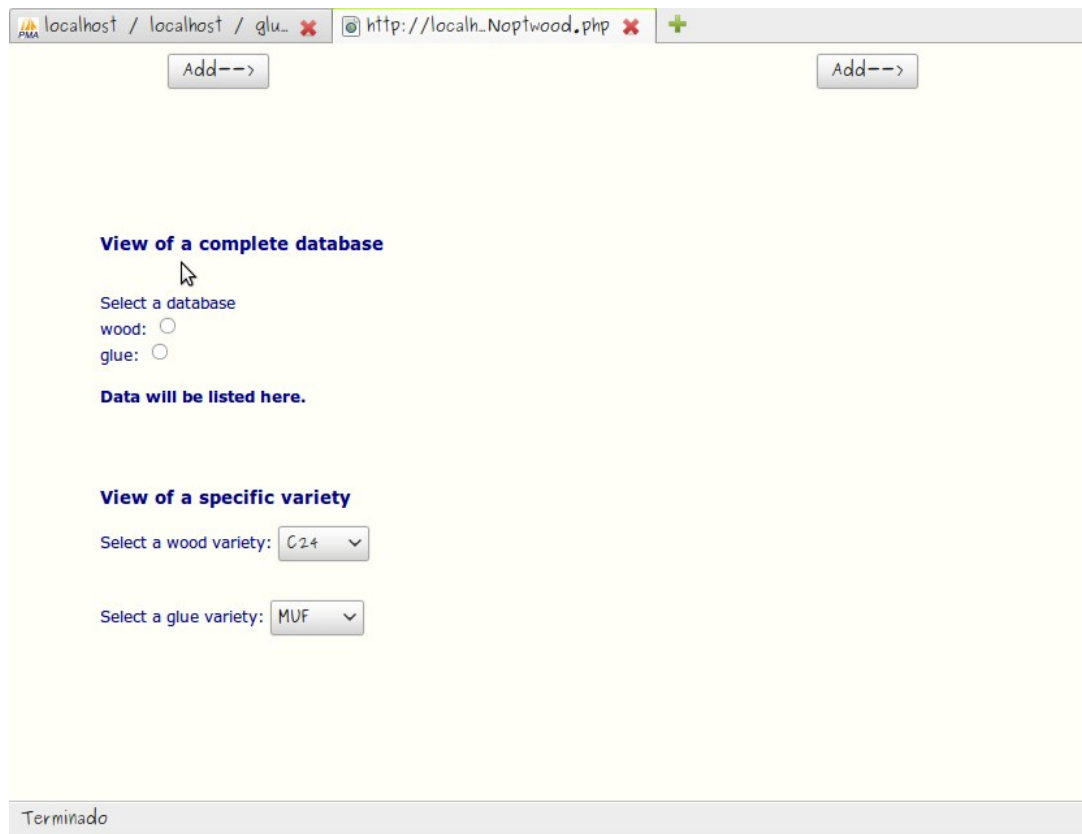


Figure 24: Screen shot of the displaying area

3.3.4.1 AJAX

AJAX (also referred to as "remote scripting") stands for "Asynchronous JavaScript And XML", and refers to a method of programming that incorporates the following technologies to send and receive data between the browser and server:

- JavaScript
- XML
- HTML
- CSS

This technology is used in the javascript files *selectdata.js* and *selectrow.js* (both codes are like in Figure 25). They have the same structure. The first one is used for the complete visualization with the function *showData()*, and the second one is used for the specific visualization with the functions *showRowWood()* and *showRowGlue()*.

```
var xmlhttp;

function showData(str)
{
xmlhttp=GetXmlHttpRequest();
if (xmlhttp==null)
{
alert ("Browser does not support HTTP Request");
return;
}
var url="getdata.php";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlhttp.onreadystatechange=stateChanged;
xmlhttp.open ("GET", url, true);
xmlhttp.send(null);
}

function stateChanged()
{
if (xmlhttp.readyState==4)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}

function GetXmlHttpRequest()
{
if (window.XMLHttpRequest)
{
// code for IE7+, Firefox, Chrome, Opera, Safari
return new XMLHttpRequest();
}
if (window.ActiveXObject)
{
// code for IE6, IE5
return new ActiveXObject("Microsoft.XMLHTTP");
}
return null;
}
```

Figure 25: Content of selectdata.js file

This JavaScript code does the following:

The *showData* function uses the *open()* method to initialise the connection. The *send()* method can be used for sending extra data but we don't have any to send (hence "null").

The *stateChanged* function uses the *readyState* variable of the XMLHttpRequest object to determine when the server has been contacted and the data retrieved. It then populates the "txtHint" div with the data retrieved by the *getdata.php* page.

It does this by using the *responseText* method.

The *GetHttpRequest* function creates an XMLHttpRequest object. It is compatible with a selection of modern browsers (IE7, Firefox, Chrome, Opera and Safari). For compatibility with older browsers we must use a different code in *GetHttpRequest* function with a couple of tries using *try* and *catch* like in Figure 26. Microsoft IE creates XMLHttpRequest as an ActiveX object while most other browsers do it the other way. Some don't support the XMLHttpRequest object.

```
function GetXmlHttpRequest()
{
    try
    {
        var oRequester = new XMLHttpRequest();
        oRequester.onload=handler
        oRequester.onerror=handler
        return oRequester
    }
    catch (error)
    {
        try
        {
            var oRequester = new ActiveXObject("Microsoft.XMLHTTP");
            oRequester.onreadystatechange=handler
            return oRequester
        }
        catch (error)
        {
            return false;
        }
    }
}
```

Figure 26: *GetHttpRequest* function code for .js files in case of incompatible browsers

3.3.4.2. Whole database displaying

Id	name	Ex	Ey	Ez	Poisson_x	Poisson_y	Poisson_z	Gx	Gy	Gz	LongName
1	MUF	6300	6300	6300	0.34	0.34	0.34	2400	2400	2400	Melamine-urea-formaldehyde
2	PUR	470	470	470	0.3	0.3	0.3	180	180	180	Polyurethane
3	PRF										

Current database -> glue

View of a specific variety

Select a wood variety: C24 ▾

Select a glue variety: MUF ▾

Figure 27: Screen shot of all the current varieties of glue database displaying, with completed data for the first tow entries

3.3.4.3. Single row displaying

View of a complete database

Select a database

wood:

glue:

Id	name	Ex	Ey	Ez	Poisson_x	Poisson_y	Poisson_z	Gx	Gy	Gz	LongName
2	PUR	470	470	470	0.3	0.3	0.3	180	180	180	Polyurethane

Current database -> glue

View of a specific variety

Select a wood variety: C24 ▾

Select a glue variety: PUR ▾

Figure 28: Screen shot of Polyurethane variety of glue row displaying

The result of using javascripts like *selectdata.js* and *selectrow.js* is that we can see displays of the databases like in Figure 27 or Figure 28 and it is not necessary a reload of the whole web page.

4 Conclusions

Our website has simple styling and is focussed on well-built content and the PHP backend. The simplicity of the web software development allows us to expand, include with other technologies or reorganize the elements in the web.

The program is modular, meaning that each page and each functionality are separated in the file structure and they work together to achieve the final result.

The design stage starts when we know what OptWood is looking for in lamella profiles. So, the first step was the design of the databases in a fast way with phpMyAdmin. These databases are created graphically with this application but if the code is needed it is in *glue.sql*, *wood.sql* and *userdata.sql* which are included in Appendix III. SQL. These files can be imported to phpMyAdmin and it's not necessary to build again the databases structure if the web site is moved to a different server.

Next the software development stage starts. We start with the main page, which handles glue and wood databases through two forms made in html. PHP provides many MySQL functions to manipulate the databases. The following step is the visualization area where it is necessary to link the main php page with the javascript and the insertion php files.

The last part of the software development is the login/ logout/ registration pages, where we include all the security preventions mentioned in the sections above. These security measures should be completed with the configuration file of the server where the web will be uploaded.

I consider this project to be the first stage of a big and solid web site where it would be possible to do any kind of modification to the material databases and simulations. For example, expansion of the different databases and the adding of new code files, which combine with the existing files, would make possible a fast simulation submission.

4.1 Future work

Here we list some suggestions for a better web site:

4.1.1 How do we add more databases on the website

It is possible to expand the system by introducing further databases and tables. Information such as simulation results could be stored and shared in this way.

Creation of tables can be performed using a PHP/MySQL script or, as before, using phpMyAdmin.

The code in the main PHP pages and HTML forms would need to be modified to accommodate new databases, and this can easily be performed at a later date. The

javascript *selectrow.js* would need to be modified to select the new database.

4.1.2 Send form with AJAX

The “Add” button action could be improved using AJAX, so that it wouldn't be necessary to guide us to an empty web page just to show a notification with information about the insertion data result, or to reload the whole page. To add this new functionality it is faster to use XAJAX.

XAJAX is an open source PHP class library implementation of AJAX that allows developers to create web-based AJAX applications using HTML, CSS, JavaScript, and PHP. Applications developed with XAJAX can asynchronously call server-side PHP functions and update content without reloading the page.

For more information check [17] from bibliography.

4.1.3 Parameterized SQL queries PDO (PHP Data Objects)

Is recommend using PDO (PHP Data Objects) to run parameterized SQL queries. Not only does this protect against SQL injection, it also speeds up queries. This is the only different between using *mysql_real_escape_string()* function. But with PDO the code is more abstract, simple and modulated.

For more information about PDO check out [13] and [14] from the bibliography.

4.1.4 Simulation interface

It would be desirable for the web interface to allow users to configure and run stress analysis simulations on the finger joints that are being investigated as part of the OptWood project. These simulations could be realized by a javascript file which open a new window where it's possible to choose the data of each variety of wood and glue.

4.1.5 Database duplication

It is necessary to store an official duplicate database for each database in a very restricted file in the server, where the security of the information is guaranteed because any user can modify the information once registered. This information could be accessed only by the administrator or the responsible person of that information.

It is an easy and quick security measure because it is achieved only by changing the permission of one folder in the server, but it is a very important security prevention.

Appendix I. Access

```
<?php
setcookie("user", "", time()+3600);//here the cookie is set
session_start();//is necessary to call this funtion at the
begining of all the
        to access session data*/

include("standard.html");
include("acceso.html");

function check_input($value)
{
// Stripslashes
if (get_magic_quotes_gpc())//Returns 0 if magic_quotes_gpc is
off, 1 otherwise.
    {
        $value = stripslashes($value);//removes backslashes added by
the addslashes() function.
    }
// Quote if not a number
if (!is_numeric($value))
    {
        $value = "'" . mysql_real_escape_string($value) . "'";
    }
return $value;
}

////////////////////////////////////

$con = mysql_connect("localhost", "root", "hola");

if (!$con)
    {
        die('Could not connect: ');
    }

mysql_select_db("userdata", $con);

/* IMPORTANT: in the login and register pages we use $_POST
cause
it has no limits on the amount of information to send.*/

//Variables from login page
$user = $_POST["username"];
$password = $_POST["password"];
```

acceso.php

```
if(isset($user) || isset($pwd))/*if fiel username is filled it
will proceed to query and security prevention*/
{
    $user = htmlentities($user, ENT_QUOTES);/*converts
characters to HTML entities*/
    $pwd = htmlentities($pwd, ENT_QUOTES);/*Encoded double and
single quotes*/

    $pwd = sha1($pwd);

    // Make a safe SQL
    $user = check_input($user);
    $pwd = check_input($pwd);
    $result = mysql_query("SELECT COUNT(*) AS matches FROM
users WHERE user = $user AND password= $pwd") or die ("Query
failed");
    $row = mysql_fetch_array($result);
    if($row['matches'] == 0)
    {
        //In case the user doesn't exist in the database this
message is showed
        echo "<p style='color:red' align='center'> Invalid user or
password </p>";
    }
    else
    {
        /*initialization of $_SESSION and $_COOKIE variables
when
the login is successful*/
        $_SESSION["k_username"] = $user;
        $_COOKIE["user"] = $_SESSION["k_username"];

        //this script redirects to MAINoptwood.php in case of
successful login
        echo '<SCRIPT LANGUAGE="javascript">
location.href = "MAINoptwood.php";
</SCRIPT>';
    }

    mysql_free_result($result);
}
if(isset($_SESSION["k_username"]))/* If the user doesn't logout
the login page will always redirect to the main page */
    echo '<SCRIPT LANGUAGE="javascript">
location.href = "MAINoptwood.php";
</SCRIPT>';

mysql_close($con);
?>
```

acceso.php

```
<html>
<head>
<link rel='stylesheet' type='text/css' href='optwood.css' />
</head>
<body>
<h1>OptWood</h1>
<h2>Website dedicated to editing and consulting data of wood
related with Optwood project </h2>
</br></br></br></br></br></br></br></br></br></br></br>
</body>
</html>
```

standard.html

```
<html>
<body>

<form name='input' action='' method='post'>
<table align="center">
<tr>
<td>Username: </td>
<td><input type='text' name='username' size='20' > </td>
</tr>
<tr>
<td>Password: </td>
<td><input type='password' name='password' size='20' > </td>
</tr>
<tr>
<td><input type='submit' value='Login' ></td>
<td><a href='registrar.php'> Register </a></td>
</tr>
</table>

</form>

</body>
</html>
```

accesso.html

```
<?php
session_start();
include("standard.html");
include("registrar.html");

function check_input($value)
{
    // Stripslashes
    if (get_magic_quotes_gpc())
    {
        $value = stripslashes($value);
    }
    // Quote if not a number
    if (!is_numeric($value))
    {
        $value = "'" . mysql_real_escape_string($value) . "'";
    }
    return $value;
}

function spamcheck($field)
{
    //filter_var() sanitizes the e-mail
    //address using FILTER_SANITIZE_EMAIL
    $field=filter_var($field, FILTER_SANITIZE_EMAIL);

    //filter_var() validates the e-mail
    //address using FILTER_VALIDATE_EMAIL
    if(filter_var($field, FILTER_VALIDATE_EMAIL))
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

$con = mysql_connect("localhost", "root", "hola");

if (!$con)
{
    die('Could not connect: ' );
}

mysql_select_db("userdata", $con);

//Variables from register page
$user = $_POST["newusername"];
$password1 = $_POST["newpassword"];
$password2 = $_POST["newpasswordR"];
$email = $_POST["email"];
```

registrar.php


```
//we check always if the variables from the form are set or not
if (isset($user))
    //We proceed to check out the new registration
    if($email == '' || $pwd1 == '' || $pwd2 == '' || $pwd1 !=
$pwd2)
        echo "</br> <p style='color:red' align='center'>
Registration failed </p>";
    else
        {
            if(strlen($pwd1) < 5)
                echo "</br> <p style='color:red' align='center'>
Registration failed: password is less than 5 characters </p>";
            else
                {
                    // Make a safe SQL
                    check_input($user);
                    check_input($email);

                    //Check out if user or e-mail exist already
                    $checkuser = mysql_query("SELECT * FROM users WHERE
user='$user'");
                    $username_exist = mysql_num_rows($checkuser);
                    $checkemail = mysql_query("SELECT * FROM users WHERE
email='$email'");
                    $email_exist = mysql_num_rows($checkemail);
                    if($email_exist>0 || $username_exist>0)
                        echo "<p style='color:red' align='center'>
User name or e-mail already in use </p>";
                    else
                        {
                            $user = htmlentities($user, ENT_QUOTES);
                            $email = htmlentities($email, ENT_QUOTES);
                            $pwd1 = htmlentities($pwd1,
ENT_QUOTES); /*Encoded double and single quotes*/
                            $pwd1 = sha1($pwd1);

                            check_input($pwd1);

                            $Nrows = mysql_num_rows(mysql_query("SELECT *
FROM users"));
                            $Nrows++;
                            $date = date("Y-m-d");
                            //We proceed to the registration
                            mysql_query("INSERT INTO users (`id`,`user`,`
password`,`email`,`date`) VALUES ('$Nrows','$user',
'$pwd1','$email','$date')", $con);
                            echo "Registration sent</br>";
                        }
                }
        }
}
```

registrar.php

```
/*should be nice that the registration data
would be sent
by e-amil to the user and the
administrator*/
}

mysql_free_result($checkuser);
}
}

echo "<a href='acceso.php'> Login </a>"; //link that redirects to
acceso.php

mysql_close($con);

?>
```

registrar.php

```
body
{
font-size:75%;
font-family:verdana,arial,'sans serif';
background-image:url('FingerJoint.jpg');
background-repeat:repeat-x;
background-color:#FFFFFF8;
color:#000080;
margin:70px;
}

h1 {font-size:500%;text-align:center;}
h2 {font-size:110%;text-align:center;color:black}
h3 {font-size:110%;}

th {background-color:#ADD8E6;}

table, td
{
font-size:100%;
font-family:verdana,arial,'sans serif';
}

ul {list-style:circle;}
ol {list-style:upper-roman;}

a:link {color:#000080;}
a:hover {color:red;}
```

optwood.css



FingerJoint.jpg

```
<html>
<body>

<form name='input' action='' method='post'>
<table align="center">
<tr>
  <td>New username:      </td>
  <td> <input type='text' name='newusername' size='20' > </td>
</tr>
<tr>
  <td>New password: (min 5 char)      </td>
  <td> <input type='password' name='newpassword' description=' '
size='20' > </td>
</tr>
<tr>
<tr>
  <td>Repeat the password:      </td>
  <td> <input type='password' name='newpasswordR' size='20' >
</td>
</tr>
<tr>
<tr>
  <td>e-mail:      </td>
  <td> <input type='text' name='email' size='45' > </td>
</tr>
<tr>
  <td><input type='submit' value='Register' ></td>
</tr>
</table>

</form>

</body>
</html>
```

registrar.html

Appendix II. Main page

```
<?php
session_start();
include("MAINoptwood.html");

echo "<html><body>";
$con = mysql_connect("localhost","root","hola");

if (!$con) die('Could not connect: ');

mysql_select_db("wood", $con);
$result = mysql_query("SELECT * FROM Varieties");

$numrows = mysql_num_rows($result);

// Execute query
mysql_query($sql,$con);

/* This form allows to choose a wood varieties and the for loop
   displays the specific row */
echo "<form>
Select a wood variety:
<select name='Varieties' onchange='showRowWood(this.value)'>";

for($i=1; $i<=$numrows; $i++)
{
    echo "<option value='$i'>";
    $result = mysql_query("SELECT * FROM Varieties WHERE id=
'$i' ");
    $row = mysql_fetch_array($result);
    echo $row['name'];
    echo "</option>";
}

echo "</select>
</form>
<br />";

mysql_select_db("glue", $con);
$result = mysql_query("SELECT * FROM Varieties");

$numrows = mysql_num_rows($result);

// Execute query
mysql_query($sql,$con);
/* this form is does the same but for glue varieties */
echo "<form>
Select a glue variety:
<select name='Varieties' onchange='showRowGlue(this.value)'>";
```

MAINoptwood.php

```
for($i=1; $i<=$Nrows; $i++)
{
    echo "<option value='$i'>";
    $result = mysql_query("SELECT * FROM Varieties WHERE id=
'$i' ");
    $row = mysql_fetch_array($result);
    echo $row['name'];
    echo "</option>";
}

echo "</select>
</form>
<br />";

mysql_select_db("glue", $con);
$result = mysql_query("SELECT * FROM Varieties");

$Nrows = mysql_num_rows($result);

// Execute query
mysql_query($sql,$con);

echo "<form>
Select a glue variety:
<select name='Varieties' onchange='showRowGlue(this.value)'>";

for($i=1; $i<=$Nrows; $i++)
{
    echo "<option value='$i'>";
    $result = mysql_query("SELECT * FROM Varieties WHERE id=
'$i' ");
    $row = mysql_fetch_array($result);
    echo $row['name'];
    echo "</option>";
}

echo "</select>
</form>
<br />";
echo "</body>";
echo "</html>";
//Check out if the session started already to show the logout
link
if (isset($_SESSION['k_username']))
{
    echo "<p align='right'><a
href='logout.php'>Logout</a></p>";
}

mysql_free_result($result);
mysql_close($con);
?>
```

MAINoptwood.php

```
<html>
<head>
<link rel='stylesheet' type='text/css' href='optwood.css' />
<script type='text/javascript' src='selectdata.js'>
</script>
<script type='text/javascript' src='selectrow.js'>
</script>
</head>

<body>
<h1>OptWood</h1>
<h2>Website dedicated to editing and consulting data of wood
related with Optwood project </h2>
</br></br></br></br></br></br></br></br>

<table cellspacing="50">
<td>

<h3>WOOD data</h3>
</br>
<!-- The size fo each field is 10 because styling reasons -->
<form name='input' action='insertwood.php' method='get'>
<table>
<tr>
<td>*Name:      </td>
<td> <input type='text' name='Name' size='10' > </td>
</tr>
<tr>
<td>Ex:      </td>
<td> <input type='text' name='Ex' size='10' > </td>
</tr>
<tr>
<td>Ey:      </td>
<td> <input type='text' name='Ey' size='10' > </td>
</tr><tr>
<td>Ez:      </td>
<td> <input type='text' name='Ez' size='10' > </td>
</tr><tr>
<td>Poisson_x:</td>
<td> <input type='text' name='Poisson_x' > </td>
</tr><tr>
<td>Poisson_y:</td>
<td> <input type='text' name='Poisson_y' > </td>
</tr><tr>
<td>Poisson_z:</td>
<td> <input type='text' name='Poisson_z' size='10'></br></td>
</tr></tr><tr>
<td>Gx:      </td>
<td> <input type='text' name='Gx' size='10' > </td>
```

MAINoptwood.html

```
</tr><tr>
  <td>Gy:  </td>
  <td> <input type='text' name='Gy' size='10' > </td>
</tr><tr>
  <td>Gz:  </td>
  <td> <input type='text' name='Gz' size='10' > </td>
</tr><tr>
  <td>LongName:</td>
  <td> <input type='text' name='LongName' size='50'></br> </td>
</tr>
</table>
</br>
<input type='submit' value='Add-->' >
</form>

</td>

<td>

<h3>GLUE data</h3>
</br>

<form name='input' action='insertglue.php' method='get'>
<table>
<tr>
  <td>*Name:      </td>
  <td> <input type='text' name='Name' size='10' > </td>
</tr>
<tr>
  <td>Ex:  </td>
  <td> <input type='text' name='Ex' size='10' > </td>
</tr>
<tr>
  <td>Ey:  </td>
  <td> <input type='text' name='Ey' size='10' > </td>
</tr><tr>
  <td>Ez:  </td>
  <td> <input type='text' name='Ez' size='10' > </td>
</tr><tr>
  <td>Poisson_x:</td>
  <td> <input type='text' name='Poisson_x' > </td>
</tr><tr>
  <td>Poisson_y:</td>
  <td> <input type='text' name='Poisson_y' > </td>
</tr>
</tr><tr>
  <td>Poisson_z:</td>
  <td> <input type='text' name='Poisson_z' size='10'></br></td>
</tr><tr>
  <td>Gx:  </td>
  <td> <input type='text' name='Gx' size='10' > </td>
</tr><tr>
```

MAINoptwood.html

```
<td>Gy: </td>
<td> <input type='text' name='Gy' size='10' > </td>
</tr><tr>
<td>Gz: </td>
<td> <input type='text' name='Gz' size='10' > </td>
</tr><tr>
<td>LongName:</td>
<td> <input type='text' name='LongName' size='50'></br> </td>
</tr>
</table>
</br>
<input type='submit' value='Add-->' >
</form>

</td>
</table>

</br></br>

<h3>View of a complete database</h3>
</br>Select a database</br>
<form name='input' action='' method='get'>
wood:
<input type='radio' name='database' value='wood'
onchange='showData(this.value)' />
</br>
Glue:
<!-- onchange takes the return of the function in the
selectdata.js file -->
<input type='radio' name='database' value='glue'
onchange='showData(this.value)' />
<br /><br />
<div id='txtHint'><b>Data will be listed here.</b></div>
</br></br></br><h3>View of a specific variety</h3>

</body>
</html>
```

MAINoptwood.html


```
var xmlhttp;

function showData(str)
{
xmlhttp=GetXmlHttpRequest();
if (xmlhttp==null)
    {
    alert ("Browser does not support HTTP Request");
    return;
    }
var url="getdata.php";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlhttp.onreadystatechange=stateChanged;
xmlhttp.open ("GET",url,true);
xmlhttp.send(null);
}

function stateChanged()
{
if (xmlhttp.readyState==4)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}

function GetXmlHttpRequest()
{
if (window.XMLHttpRequest)
    {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    return new XMLHttpRequest();
    }
if (window.ActiveXObject)
    {
    // code for IE6, IE5
    return new ActiveXObject("Microsoft.XMLHTTP");
    }
return null;
}
```

selectdata.js

```
var xmlhttp;

function showRowWood(str)
{
xmlhttp=GetXmlHttpRequest();
if (xmlhttp==null)
    {
    alert ("Browser does not support HTTP Request");
    return;
    }
var url="getrowWood.php";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlhttp.onreadystatechange=stateChanged;
xmlhttp.open ("GET", url, true);
xmlhttp.send (null);
}

function showRowGlue(str)
{
xmlhttp=GetXmlHttpRequest();
if (xmlhttp==null)
    {
    alert ("Browser does not support HTTP Request");
    return;
    }
var url="getrowGlue.php";
url=url+"?q="+str;
url=url+"&sid="+Math.random();
xmlhttp.onreadystatechange=stateChanged;
xmlhttp.open ("GET", url, true);
xmlhttp.send (null);
}

function stateChanged()
{
if (xmlhttp.readyState==4)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}

function GetXmlHttpRequest()
{
if (window.XMLHttpRequest)
    {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    return new XMLHttpRequest();
    }
}
```

selectrow.js

```
if (window.ActiveXObject)
{
  // code for IE6, IE5
  return new ActiveXObject("Microsoft.XMLHTTP");
}
return null;
}
```

selectrow.js

```
<?php
include("standard.html");

$name = $_GET["Name"];
$Ex = $_GET["Ex"];
$Ey = $_GET["Ey"];
$Ez = $_GET["Ez"];
$Poisson_x = $_GET["Poisson_x"];
$Poisson_y = $_GET["Poisson_y"];
$Poisson_z = $_GET["Poisson_z"];
$Gx = $_GET["Gx"];
$Gy = $_GET["Gy"];
$Gz = $_GET["Gz"];
$LongName = $_GET["LongName"];

if($name == "") echo "Error: field Name must be filled.";
else
{
    $con = mysql_connect("localhost","root","hola");

    if (!$con) die('Could not connect: ');

    mysql_select_db("wood", $con);

    /* This query count the name of rows with the value
    introduced in the field name
    which is stored in the variable $name. This query also
    creates a new column
    called filas where the number of row found is stored */

    $result = mysql_query("SELECT COUNT(*) AS filas FROM
Varieties WHERE name = '$name'", $con);
    $row = mysql_fetch_array($result);
    $Nrows = mysql_num_rows(mysql_query("SELECT * FROM
Varieties"));

    /* If the number of rows found is 0 it means that the name
of wood variety is new and
    a new row is inserted in the table with the statement
INSERT TO... */
    if($row['filas'] == 0)
    {
        $Nrows++;
        $result = mysql_query("INSERT INTO Varieties (`id`,
`name`, `Ex`, `Ey`, `Ez`, `Poisson_x`, `Poisson_y`, `Poisson_z`,
`Gx`, `Gy`, `Gz`, `LongName`) VALUES ($Nrows, '$name', '$Ex',
'$Ey', '$Ez', '$Poisson_x', '$Poisson_y', '$Poisson_z', '$Gx',
'$Gy', '$Gz', '$LongName')", $con);
    }
}
```

insertwood.php

```
else
{
/* Otherwise, if the number of rows found isn't 0 it means
that the user is
modifying a determinated variety of wood. With the
statement UPDATE is
possible to modify the data in a row. */
$result = mysql_query("UPDATE Varieties SET Ex = '$Ex', Ey
= '$Ey', Ez = '$Ez', Poisson_x = '$Poisson_x', Poisson_y =
'$Poisson_y', Poisson_z = '$Poisson_z', Gx = '$Gx', Gy = '$Gy',
Gz = '$Gz', Gz = '$Gz', LongName = '$LongName' WHERE name =
'$name'");

}

if($result == FALSE) echo "An error has ocurred inserting
the data";
else echo"</br>Data inserted successfully";

mysql_close($con);
}

echo "<a href='MAINoptwood.php'></br> << Back </a>";

?>
```

insertwood.php

```
<?php
include("standard.html");

$name = $_GET["Name"];
$Ex = $_GET["Ex"];
$Ey = $_GET["Ey"];
$Ez = $_GET["Ez"];
$Poisson_x = $_GET["Poisson_x"];
$Poisson_y = $_GET["Poisson_y"];
$Poisson_z = $_GET["Poisson_z"];
$Gx = $_GET["Gx"];
$Gy = $_GET["Gy"];
$Gz = $_GET["Gz"];
$LongName = $_GET["LongName"];

if($name == "") echo "Error: field Name must be filled.";
else
{
    $con = mysql_connect("localhost","root","hola");

    if (!$con) die('Could not connect: ');

    mysql_select_db("glue", $con);

    /* This query count the name of rows with the value
introduced in the field name
    which is stored in the variable $name. This query also
creates a new column
    called filas where the number of row found is stored */
    $result = mysql_query("SELECT COUNT(*) AS filas FROM
Varieties WHERE name = '$name'", $con);
    $row = mysql_fetch_array($result);
    $Nrows = mysql_num_rows(mysql_query("SELECT * FROM
Varieties"));

    /* If the number of rows found is 0 it means that the name
of glue variety is new and
    a new row is inserted in the table with the statement
INSERT TO... */
    if($row['filas'] == 0)
    {
        $Nrows++;
        $result = mysql_query("INSERT INTO Varieties (`id`,
`name`, `Ex`, `Ey`, `Ez`, `Poisson_x`, `Poisson_y`, `Poisson_z`,
`Gx`, `Gy`, `Gz`, `LongName`) VALUES ($Nrows, '$name', '$Ex',
'$Ey', '$Ez', '$Poisson_x', '$Poisson_y', '$Poisson_z', '$Gx',
'$Gy', '$Gz', '$LongName')", $con);
    }
}
```

insertglue.php

```
else
{
/* Otherwise, if the number of rows found isn't 0 it means
that the user is
modifying a determinated variety of glue. With the
statement UPDATE is
possible to modify the data in a row. */
$result = mysql_query("UPDATE Varieties SET Ex = '$Ex', Ey
= '$Ey', Ez = '$Ez', Poisson_x = '$Poisson_x', Poisson_y =
'$Poisson_y', Poisson_z = '$Poisson_z', Gx = '$Gx', Gy = '$Gy',
Gz = '$Gz', Gz = '$Gz', LongName = '$LongName' WHERE name =
'$name'");

}

if($result == FALSE) echo "An error has ocurred inserting
the data";
else echo"</br>Data inserted successfully";

mysql_close($con);
}

echo "<a href='MAINoptwood.php'></br> << Back </a>";

?>
```

insertglue.php

```
<?php
session_start();
include("standard.html");
//Delete the session
session_destroy();
echo "Session ended... <p><a href='acceso.php'>finish</a></p>";

?>
```

logout.php

Bibliography

- [1] Katzengruber, R. and Jeitler, G. and Brandner, R. and Schickhofer, G. *Tensile proof loading to assure quality of finger-jointed structural timber*. Proceedings of WCTE (CD-ROM), 9th World Conference on Timber Engineering, 2006, p6-10.
- [2] Hunter, P. And Pulan, A.: "FEM/BEM notes", Department of Engineering Science, University of Auckland, 1998 <http://www.esc.auckland.ac.nz/Academic/Texts/FEM-BEM-notes.html>
- [3] Jacobs, Ian; Walsh, Norman (15 December 2004). "URI/Resource Relationships". *Architecture of the World Wide Web, Volume One*. World Wide Web Consortium. <http://www.w3.org/TR/webarch/#id-resources>. Retrieved June 2010
- [4] Elisabeth Freeman, Eric Freeman. *Head first HTML with CSS & XHTML*. Sebastopol, CA: O'Reilly, 2009.
- [5] Jason Gerner, Elizabeth Naramore, Morgan Owens. *Professional LAMP: Linux, Apache, MySQL, and PHP Web development*. Wiley Pub., 2006
- [6] PHP Dasar <http://meta14.wordpress.com/2009/12/14/php-dasar/> (accessed on 2010-06)
- [7] Kevin Potts, Robert Sable, Nathan Smith, Mary Fredborg, Cody Lindley. *Textpattern solutions: PHP-based content management made easy*. Apress, 2007
- [8] Marc Delisle. *Mastering Phpmyadmin for Effective Mysql Management*. Ed. Packt Publishing, 2006
- [9] Wikimedia Foundation, Inc. http://en.wikipedia.org/wiki/HTTP_cookie (accessed on June 2010)
- [10] Wikimedia Foundation, Inc. http://en.wikipedia.org/wiki/Url#cite_note-0 (accessed on June 2010)
- [11] Justin Clake. *SQL Injection Attacks and Defense*. Syngress Publishing, 2009
- [12] w3schools (2010) <http://www.w3schools.com/> *Tutorials* (accessed on 2010-06)
- [13] The PHP Group (2010) <http://www.php.net/> *Documentation* (accessed on 2010-06)
- [14] Rasmus Lerdorf, Kevin Tatroe, Peter MacIntyre. *Programming PHP* O'Reilly Media, Inc., 2006
- [15] Wang, X. and Yu, *How to break MD5 and other hash functions*. Advances in Cryptology - EUROCRYPT 2005, p19-35.

- [16] Oregon State University. <http://oregonstate.edu/cws/docs/collation> (accessed on June 2010)
- [17] Jared White & J. Max Wilson (2005-2007) <http://xajaxproject.org/> *Docs & Tutorials* (accessed on 2010-06)