E-commerce application based on Magento for the website Quality Materials

A Master’s Thesis

Submitted to the Faculty of the
Escola Tècnica d’Enginyeria de Telecomunicació de
Barcelona
Universitat Politècnica de Catalunya

by

Xavier Casanova Colomé

In partial fulfilment
of the requirements for the degree of
MASTER IN TELECOMMUNICATIONS
ENGINEERING

Advisors: Xavier Castillejo, Juan Luis Gorricho
Barcelona, September 2017
This page intentionally left blank.
Title of the thesis: E-commerce application based on Magento for the website Quality Materials

Author: Xavier Casanova Colomé

Advisors: Xavier Castillejo, Juan Luis Gorricho

Abstract — Magento is one of the most flexible and powerful platforms to develop e-commerce web applications. Furthermore, it is open-source and it has a great community behind working every day to improve it. In this project, a b2b e-commerce web application is to be developed with Magento, following the needs and requirements of the client of Computer Sciences Brand. Even if improvements and refinements still need to be applied, the current results follow the client’s requirements and will allow to start using the site in a not distant future.
ACKNOWLEDGEMENTS

I would like to thank Computer Sciences Brand for the opportunity they gave me. I had the occasion to learn an extensive list of concepts, and to work on a real project, surrounded by colleagues that were always there to solve the questions that eventually arose.

I also want to thank Xavier Castillejo, at Computer Sciences Brand, and Juan Luis Gorricho, at Universitat Politècnica de Catalunya, for the supervision and advice given to me in this project.
# Revision History and Approval Record

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28/08/2017</td>
<td>Document creation</td>
</tr>
<tr>
<td>1</td>
<td>25/09/2017</td>
<td>Document revision 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Written by:</th>
<th>Reviewed and approved by:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date</strong></td>
<td><strong>25/09/2017</strong></td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Xavier Casanova</strong></td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td><strong>Project Author</strong></td>
</tr>
</tbody>
</table>
# Table of contents

Acknowledgements ........................................................................................................... 4  
Revision history and approval record .............................................................................. 5  
Table of contents ............................................................................................................. 6  
List of Figures .................................................................................................................. 8  
1. Introduction ................................................................................................................ 12  
   1.1. Context .................................................................................................................. 12  
   1.2. Requirements and objectives .............................................................................. 13  
      1.2.1. Website design ................................................................................................ 13  
   1.3. Planning ................................................................................................................. 15  
      1.3.1. Gantt diagram .................................................................................................. 15  
      1.3.2. Deviations from the initial plan ....................................................................... 20  
2. State of the art of the technology used or applied in this thesis ................................. 23  
   2.1. Magento ............................................................................................................... 23  
3. System architecture ..................................................................................................... 24  
   3.1. Overview ................................................................................................................. 24  
   3.2. Technologies used by Magento ............................................................................ 26  
      3.2.1. HTML ............................................................................................................. 26  
      3.2.2. CSS & Bootstrap ............................................................................................ 27  
      3.2.3. PHP ................................................................................................................ 28  
      3.2.4. MySQL & SQL .............................................................................................. 29  
   3.3. Magento ................................................................................................................ 30  
      3.3.1. Folder structure ............................................................................................... 30  
      3.3.2. Modules ........................................................................................................... 32  
      3.3.3. Breaking the URL into pieces ........................................................................... 32  
      3.3.4. Database with EAV model ............................................................................. 32  
4. Methodology / project development .......................................................................... 34  
   4.1. Project methodology .............................................................................................. 34  
      4.1.1. Client’s follow-up ............................................................................................. 34
4.1.2. Learning by doing.................................................................34
4.1.3. Development and production environment.................................34
4.2. Tools and resources used in this project........................................35
  4.2.1. Computer specifications......................................................35
  4.2.2. Firefox developer edition..................................................35
  4.2.3. Sublime Text........................................................................35
  4.2.4. Magento’s template path hints..............................................36
  4.2.5. WinScp..............................................................................36
  4.2.6. Online resources.................................................................36
4.3. Project development..................................................................37
  4.3.1. Setting up the environment, installation..................................37
  4.3.2. Website development............................................................40
5. Results.......................................................................................61
6. Budget.......................................................................................71
7. Conclusions and future development............................................72
Bibliography.................................................................................73
Appendices (optional).................................................................76
LIST OF FIGURES

Fig. 1. Design of the ‘landing’ or ‘about us’ page. This was the first design I received to start developing. ................................................. 14

Fig. 2. Index.php, under the root folder of Magento. ................................................. 24

Fig. 3. Each page is made up of different templates put all together. ................................................. 25

Fig. 4. On the left, snippet of registerSuccess.phtml, the template used after the user successes in registering in the platform. On the right, snippet of CSB_customer.xml, which uses the template registerSuccess.phtml. ................................................. 25

Fig. 5. Basic overview of the whole process. ................................................. 26

Fig. 6. General structure of an HTML file document. ................................................. 26

Fig. 7. The page title is shown in the corresponding tab of the web browser. ................................................. 27

Fig. 8. On the left, the css class sectionHeaderUnderline defined in landing.css. On the right, snippet of landingqm.phtml where the CSS class is used to define the style of the HTML element hr. ................................................. 27

Fig. 9. In Magento, head.phtml includes most of the definitions for the head of the HTML document. Here, we define one of the stylesheets to be used as the one from Bootstrap. ................................................. 28

Fig. 10. Snippet of Observer.php, used to control the access to the website. ................................................. 29

Fig. 11. phpMyAdmin allows easily accessing the database by means of the browser. ................................................. 30

Fig. 12. Table eav_entity_type. ................................................. 33

Fig. 13. Table eav_attribute. ................................................. 33

Fig. 14. Firefox developer edition’s code inspector. ................................................. 35

Fig. 15. When path hints is activated, the frontend shows which templates are used in each section of the page. ................................................. 36

Fig. 16. Alias in WAMP. ................................................. 37

Fig. 17. Importing through PHPMyAdmin ................................................. 38

Fig. 18. Connecting to the server with WinSCP via FTP. ................................................. 39

Fig. 19. Content defined in header.phtml. ................................................. 40

Fig. 20. Content defined in footer.phtml. ................................................. 41

Fig. 21. Snippet of CSB_LandingQM.xml. ................................................. 41

Fig. 22. Snippet of local.xml, under the layout directory. ................................................. 42

Fig. 23. Content defined in sliderAndSelector.phtml. ................................................. 42
Fig. 24. Using getSkinUrl to retrieve an image. 42
Fig. 25. Mandatory login: first attempt. 43
Fig. 26. Observer.php controls the access to the page. 43
Fig. 27. Snippet of upgrade-1.0.0-1.0.1.php, responsible for including the new fields for the customer. 44
Fig. 28. Registering page of Quality Materials. 44
Fig. 29. Grid.php is responsible for preparing the view of the customer area in the admin panel. The code on the image adds a new column for the field with id ‘company_name’, which will be displayed as ‘Nombre de la empresa’ (in Spanish). 45
Fig. 30. Customer management section of Magento’s admin page. 45
Fig. 31. In admin’s client management section, the accounts can easily be activated/deactivated. 45
Fig. 32. HTML code for the ‘New account’ e-mail. 46
Fig. 33. Quantity of products ordered in the last orders. 46
Fig. 34. Use of the magic method getTelephone() to retrieve the telephone of the user. 46
Fig. 35. Form for editing the customer’s account information. 47
Fig. 36. CMS page and wysiwyg editor. 48
Fig. 37. Links to the pages with information about Quality Materials. 48
Fig. 38. Image included by means of including some code in Magento’s admin. 48
Fig. 39. The code on the left creates two pieces of HTML and stores them in two PHP variables. The code on the right combines the HTML codes stored in the different PHP variables to finally generate the desired HTML structure. 49
Fig. 40. Rules for storing product images in Magento. Source: https://magento.stackexchange.com/questions/1005/get-original-image-file-from-a-cache-url/1008 50
Fig. 41. Admin’s section for flushing the different caches available in Magento 51
Fig. 42. Quality Material’s placeholder. 51
Fig. 43. Script to delete or disable products. 52
Fig. 44. The search box is only shown after log in. 52
Fig. 45. Definition of the stylesheet to be used in the landing page of Quality Materials. 53
Fig. 46. ‘Send a message’ in the previous webpage. 53
Fig. 47. Defining the admin’s URL in local.xml. .................................................. 54
Fig. 48. Page under maintenance. ................................................................. 54
Fig. 49. Maintenance page of Quality Materials. .......................................... 54
Fig. 50. On the left, product page affected by the Bootstrap’s definitions. On the right, the same page with the styles corrected by the new CSS class. ................................................. 55
Fig. 51. On the left, menu unstructured because of the use of Bootstrap. On the right, menu with the styles corrected by the CSS class. .................................................. 55
Fig. 52. Store views can be enabled or disabled from the admin panel. .............. 56
Fig. 53. Mache_Checkout.csv, under app/local/es_ES, includes translations needed in the checkout page. ................................................................. 56
Fig. 54. CMS pages can also be programmatically disabled or deleted. .............. 57
Fig. 55. An unknown error was shown. ........................................................... 58
Fig. 56. Error report. .................................................................................. 58
Fig. 57. Header. ....................................................................................... 61
Fig. 58. Footer. ....................................................................................... 61
Fig. 59. Landing page. ............................................................................... 61
Fig. 60. Contact us page. .......................................................................... 62
Fig. 61. Login page. ................................................................................ 62
Fig. 62. Page for forgotten password. .......................................................... 62
Fig. 63. New password page. ...................................................................... 63
Fig. 64. Registering page. ......................................................................... 63
Fig. 65. Registering success page. ............................................................... 63
Fig. 66. Shipping conditions. .................................................................... 64
Fig. 67. Menu. ........................................................................................ 64
Fig. 68. Search box. ................................................................................ 64
Fig. 69. Home page. ................................................................................. 64
Fig. 70. Product page. ............................................................................... 65
Fig. 71. My account main page. .................................................................. 65
Fig. 72. My account order history. .............................................................. 66
Fig. 73. My account edit information page. .................................................. 66

10/76
Fig. 74. My account manage addresses page. .................................................. 67
Fig. 75. My account edit address page. ............................................................... 67
Fig. 76. My account print order page. ................................................................. 68
Fig. 77. Shopping cart page. ............................................................................. 68
Fig. 78. Checkout page. .................................................................................. 69
Fig. 79. E-mail sent to the customer before account activation. ....................... 69
Fig. 80. E-mail to the customer after account activation. ............................... 70
Fig. 81 E-mail to the administrator when an account is pending of being activated. 70
1. Introduction

This section introduces the web development project carried out at Computer Sciences Brand S.L. This section starts by presenting the context of the project. After that, the objectives and requirements of the project are stated. Finally, the temporal planning of the project is discussed. The rest of the document is structured as follows: §2 reviews the state of the art of the technologies used in this project. In §3, a description of the whole architecture of the website, from programming languages to database and server management, can be found. §4 specifies the project development methodology used in this project, as well as the project development. The results of the development are exposed in §5. The calculation of this project’s budget can be found in §6, and §7 includes the conclusions and future development for this project.

A word on nomenclature
Throughout all the text, the reader can find the words web, website, site, web application, webpage, or web page. They all refer to the same, the frontend of our page. I also make reference to ‘the client’ and ‘the customer’. The client refers to CSB’s client, the company that asked for the development of Quality Materials. The customer refers to a user of the web page. The customer is also sometimes named the user.

1.1. Context
This project has been carried out at Computer Sciences Brand S.L. (CSB), a software company based in Barcelona. The main target of the project was the development of a new e-commerce channel for CSB’s client. CSB’s client needed an online channel for selling its products exclusively to other companies, not to individuals. This kind of e-commerce websites are better known as b2b, or business to business.

The project itself is based on a previous website named Materials World, which is an e-commerce platform for selling the client’s products and that can be accessed by any kind of customer at http://www.mwmaterialsworld.com/.

The project started in the semester of fall 2016. Some months after the start of the project, the author of this report had available the source code of the previous website. From this moment on, tasks that had to be performed by the author included learning an extensive list of concepts, such as Magento, programming languages such as PHP or HTML, server setting, and others that are described in §3, redeveloping the website, and being in contact with the client for the follow up of the project.

At the start of the project, the author’s knowledge of website developing was very basic, and not enough to start developing under Magento platform. Magento’s learning curve is steep [1], even for someone that has experience in developing with HTML, CSS, PHP or Javascript [2]. That is why, from the beginning, learning and understanding how Magento
works and puts together everything that it needs, was a task to be done. The learning process continues as the project goes on, and the knowledge acquired by the author improves with the time. Magento is very complete and has a great number of functionalities and possibilities to be exploited by developers. However, all these possibilities are not documented in detail in Magento’s official documentation [3], which makes the learning process slower than in other programming environments. Almost everything that is needed in an e-commerce web application has already been taken care of by Magento or one of the third-party developers that are very active in the Magento community (see [https://marketplace.magento.com](https://marketplace.magento.com/)).

1.2. Requirements and objectives

The main objective of this project was the development of an online channel for selling products directly to companies. The list of requirements was as follows:

- The client must be able to control who can access the website.
- The website must feature a welcome or landing page with relevant information about its functioning and an ‘About us’ section.
- There must be a page for each product.
- The website must feature a shopping cart.
- The website must have a series of pages with useful information, such as ‘About us’, ‘F.A.Q.’, ‘Contact us’ ‘Shipping costs’ or ‘Legal notice’.
- A search box must be available for searching products.
- The e-commerce web application must accept different payment methods.
- The website must be only in Spanish.
- The website must have a different look and feel to the previous one on which it is based. This look and feel will be provided by the client.
- There must be a ‘My account page’ which allows to manage the client’s account and to see previous orders.

As stated in the previous section, the website Quality Materials is based on the already existing previous website. Some of these functionalities were already accomplished by the previous website, but they needed to be adapted to the new one. Some other functionalities had to be developed from scratch. Furthermore, the design had to be changed. The project requirements varied slightly as the project advanced.

1.2.1. Website design

Per client’s requirement, the design of the website was to be done by one of their employees. Thus, I was provided with graphical representations of the new website, and I was required
to develop the website according to these images. As an example, Fig. 1 shows the first design that I was provided with (only a part of the image is shown because it is too large to fit properly in this document).

Fig. 1. Design of the ‘landing’ or ‘about us’ page. This was the first design I received to start developing.
1.3. Planning

1.3.1. Gantt diagram

The initial planning was made taking into account that at first I was not working full time on this project.
<table>
<thead>
<tr>
<th>Id</th>
<th>Modo de tarea</th>
<th>Nombre de tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td></td>
<td>Mandatory registering and login</td>
<td>11 días</td>
<td>jue 13/04/17</td>
<td>jue 27/04/17</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>New fields, frontend and admin</td>
<td>6 días</td>
<td>jue 13/04/17</td>
<td>jue 20/04/17</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Customer activation</td>
<td>5 días</td>
<td>vie 21/04/17</td>
<td>jue 27/04/17</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Language of the website</td>
<td>3 días</td>
<td>jul 27/04/17</td>
<td>lun 01/05/17</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Top menu redesign</td>
<td>4 días</td>
<td>mar 02/05/17</td>
<td>vie 05/05/17</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>e-mails</td>
<td>5 días</td>
<td>lun 08/05/17</td>
<td>vie 12/05/17</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>Upload products</td>
<td>2 días</td>
<td>jue 04/05/17</td>
<td>vie 05/05/17</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>Meeting with the client 2</td>
<td>1 día</td>
<td>lun 08/05/17</td>
<td>lun 08/05/17</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>My account page</td>
<td>10 días</td>
<td>mar 09/05/17</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Control panel</td>
<td>10 días</td>
<td>mar 09/05/17</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Account information</td>
<td>10 días</td>
<td>mar 09/05/17</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>My addresses</td>
<td>10 días</td>
<td>mar 09/05/17</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>My orders</td>
<td>10 días</td>
<td>mar 09/05/17</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>Image + button of product result in the search</td>
<td>4 días</td>
<td>mar</td>
<td>vie 26/05/17</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23/05/17</td>
</tr>
<tr>
<td>Id</td>
<td>Modo de tarea</td>
<td>Nombre de tarea</td>
<td>Duración</td>
<td>Comienzo</td>
<td>Fin</td>
</tr>
<tr>
<td>----</td>
<td>---------------</td>
<td>-----------------</td>
<td>----------</td>
<td>----------</td>
<td>-------------------</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>My account page</td>
<td>10 días</td>
<td>mar 09/05</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>Control panel</td>
<td>10 días</td>
<td>mar 09/05</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td>Account information</td>
<td>10 días</td>
<td>mar 09/05</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td>My addresses</td>
<td>10 días</td>
<td>mar 09/05</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td>My orders</td>
<td>10 días</td>
<td>mar 09/05</td>
<td>lun 22/05/17</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>Image + button of product result in the search</td>
<td>4 días</td>
<td>mar</td>
<td>vie 26/05/17 23/05/17</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>Meeting with the client 3</td>
<td>1 día</td>
<td>lun 29/05</td>
<td>lun 29/05/17</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td>Product page</td>
<td>6 días</td>
<td>mar 30/05</td>
<td>mar 06/06/17</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td>Description</td>
<td>6 días</td>
<td>mar 30/05</td>
<td>mar 06/06/17</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td>Price</td>
<td>6 días</td>
<td>mar 30/05</td>
<td>mar 06/06/17</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td>Product configuration</td>
<td>6 días</td>
<td>mar 30/05</td>
<td>mar 06/06/17</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td>Shopping cart</td>
<td>4 días</td>
<td>mié 07/06</td>
<td>lun 12/06/17</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>Checkout page</td>
<td>4 días</td>
<td>mar 13/06</td>
<td>vie 16/06/17</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td>Meeting with the client 4</td>
<td>1 día</td>
<td>lun 19/06</td>
<td>lun 19/06/17</td>
</tr>
</tbody>
</table>

17/76
<table>
<thead>
<tr>
<th>Id</th>
<th>Modo de tarea</th>
<th>Nombre de tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td></td>
<td>Texts + pages</td>
<td>2 días</td>
<td>mar 20/06/17</td>
<td>mié 21/06/17</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>Legal notice</td>
<td>2 días</td>
<td>mar 20/06/17</td>
<td>mié 21/06/17</td>
</tr>
<tr>
<td>51</td>
<td></td>
<td>Privacy policy</td>
<td>2 días</td>
<td>mar 20/06/17</td>
<td>mié 21/06/17</td>
</tr>
<tr>
<td>52</td>
<td></td>
<td>Frequently asked questions</td>
<td>2 días</td>
<td>mar 20/06/17</td>
<td>mié 21/06/17</td>
</tr>
<tr>
<td>53</td>
<td></td>
<td>Terms of use</td>
<td>2 días</td>
<td>mar 20/06/17</td>
<td>mié 21/06/17</td>
</tr>
<tr>
<td>54</td>
<td></td>
<td>Shipping methods</td>
<td>2 días</td>
<td>mar 20/06/17</td>
<td>mié 21/06/17</td>
</tr>
<tr>
<td>55</td>
<td></td>
<td>Shipping and returns terms</td>
<td>2 días</td>
<td>mar 20/06/17</td>
<td>mié 21/06/17</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td>Advanced search</td>
<td>5 días</td>
<td>jue 22/06/17</td>
<td>mié 28/06/17</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td>Mapa web, seo</td>
<td>3 días</td>
<td>jue 29/06/17</td>
<td>lun 03/07/17</td>
</tr>
<tr>
<td>58</td>
<td></td>
<td>Table with last orders</td>
<td>5 días</td>
<td>mar 04/07/17</td>
<td>lun 10/07/17</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td>Meeting with the client 5</td>
<td>1 día</td>
<td>mar 11/07/17</td>
<td>mar 11/07/17</td>
</tr>
<tr>
<td>63</td>
<td></td>
<td>Page styling</td>
<td>15 días</td>
<td>mié 12/07/17</td>
<td>mar 01/08/17</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>Adapting styles</td>
<td>15 días</td>
<td>mié 12/07/17</td>
<td>mar 01/08/17</td>
</tr>
<tr>
<td>65</td>
<td></td>
<td>Removing parts from the previous website</td>
<td>15 días</td>
<td>mié 12/07/17</td>
<td>mar 01/08/17</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td>Page review</td>
<td>5 días</td>
<td>mié 02/08/17</td>
<td>mar 08/08/17</td>
</tr>
<tr>
<td>N°</td>
<td>Modo de tarea</td>
<td>Nombre de tarea</td>
<td>Duración</td>
<td>Comienzo</td>
<td>Fin</td>
</tr>
<tr>
<td>----</td>
<td>--------------</td>
<td>-----------------------------------------------------</td>
<td>----------</td>
<td>-------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td>Meeting with the client 5</td>
<td>1 día</td>
<td>mar 11/07</td>
<td>mar 11/07/17</td>
</tr>
<tr>
<td>62</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td></td>
<td>Page styling</td>
<td>15 días</td>
<td>mié 12/07</td>
<td>mar 01/08/17</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td>Adapting styles</td>
<td>15 días</td>
<td>mié 12/07</td>
<td>mar 01/08/17</td>
</tr>
<tr>
<td>65</td>
<td></td>
<td>Removing parts from the previous website</td>
<td>15 días</td>
<td>mié 12/07/17</td>
<td>mar 01/08/17</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td></td>
<td>Page review</td>
<td>5 días</td>
<td>mié 02/08/17</td>
<td>mar 08/08/17</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td></td>
<td>Final report</td>
<td>10 días</td>
<td>mié 09/08/17</td>
<td>mar 22/08/17</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td>Document</td>
<td>10 días</td>
<td>mié 09/08/17</td>
<td>mar 22/08/17</td>
</tr>
<tr>
<td>71</td>
<td></td>
<td>Power point</td>
<td>10 días?</td>
<td>mié 09/08/17</td>
<td>mar 22/08/17</td>
</tr>
</tbody>
</table>
1.3.2. Deviations from the initial plan.

The initial plan was as stated above. However, several difficulties caused the project development to be different in some ways to the one that was initially planned. This is mostly due to the fact that this project was carried out at a company that had some requirements from a client, and these requirements somehow varied with time. Also, developing under Magento is not a straightforward task, so this made the development slower than if the environment had been another one. Also, the fact that the website is based on a previous one caused sometimes that developments that should be a bit quicker became slower because we combined our code with code that was already done and not documented. Also, the fact that we installed the previous website in a new development environment and we used Bootstrap, made the page not to work properly, and a great number of parts of the page were disorganized or malfunctioning.

<table>
<thead>
<tr>
<th>ID</th>
<th>Nombre de tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
<th>mar ‘17</th>
<th>apr ‘17</th>
<th>may ‘17</th>
<th>jun ‘17</th>
<th>jul ‘17</th>
<th>ago ‘17</th>
<th>sep ‘17</th>
<th>oct ‘17</th>
<th>nov ‘17</th>
<th>dic ‘17</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Initial planification</td>
<td>7 días</td>
<td>mié 15/03/17</td>
<td>jue 20/03/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Project scope</td>
<td>7 días</td>
<td>mié 15/03/17</td>
<td>jue 20/03/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Functional analysis</td>
<td>7 días</td>
<td>mié 15/03/17</td>
<td>jue 20/03/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Gantt diagram</td>
<td>7 días</td>
<td>mié 15/03/17</td>
<td>jue 20/03/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Recommender system</td>
<td>6 días</td>
<td>vio 29/03/17</td>
<td>vie 1/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Information research</td>
<td>6 días</td>
<td>vio 29/03/17</td>
<td>vie 1/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Features gathering and first scripts</td>
<td>6 días</td>
<td>vio 29/03/17</td>
<td>vie 1/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Meeting with the client</td>
<td>1 día</td>
<td>lun 03/04/17</td>
<td>lun 03/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Upload to live server</td>
<td>5 días</td>
<td>mar 09/04/17</td>
<td>lun 10/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Upload website</td>
<td>5 días</td>
<td>mar 09/04/17</td>
<td>lun 10/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Upload database</td>
<td>5 días</td>
<td>mar 09/04/17</td>
<td>lun 10/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Meeting with the client 1</td>
<td>1 día</td>
<td>mar 11/04/17</td>
<td>mar 11/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>General definition of styles</td>
<td>2 días</td>
<td>mar 12/04/17</td>
<td>mar 13/04/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Learning Magento, programming, databases</td>
<td>20 días</td>
<td>vio 14/04/17</td>
<td>jue 1/05/17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
<td>Duration</td>
<td>Comments</td>
<td>Note</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------------------------------------------</td>
<td>----------</td>
<td>----------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Mandatory, registering and login</td>
<td>3 days</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>New fields, internal and external</td>
<td>3 days</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Container activation</td>
<td>3 days</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Language of the website</td>
<td>3 days</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Top menu redesign</td>
<td>3 days</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>e-mail</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Upload products</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Meeting with the client 1</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Meeting with the client 2</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>My account page</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Contact person</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Account information</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>My addresses</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>My orders</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Image + button of product result in the search</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Product page</td>
<td>1 day</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

21/76
<table>
<thead>
<tr>
<th>Id</th>
<th>Nombre de tarea</th>
<th>Duración</th>
<th>Comienzo</th>
<th>Fin</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>Product page</td>
<td>30 días</td>
<td>mar 05/08/17</td>
<td>mar 15/08/17</td>
</tr>
<tr>
<td>42</td>
<td>Description</td>
<td>30 días</td>
<td>mar 05/08/17</td>
<td>mar 15/08/17</td>
</tr>
<tr>
<td>43</td>
<td>Price</td>
<td>30 días</td>
<td>mar 05/08/17</td>
<td>mar 15/08/17</td>
</tr>
<tr>
<td>44</td>
<td>Product configuration</td>
<td>30 días</td>
<td>mar 05/08/17</td>
<td>mar 15/08/17</td>
</tr>
<tr>
<td>45</td>
<td>Shopping cart</td>
<td>4 días</td>
<td>mar 27/08/17</td>
<td>mar 28/08/17</td>
</tr>
<tr>
<td>46</td>
<td>Checkout page</td>
<td>4 días</td>
<td>mar 22/08/17</td>
<td>mar 23/08/17</td>
</tr>
<tr>
<td>47</td>
<td>Meeting with the client 4</td>
<td>3 días</td>
<td>lun 28/08/17</td>
<td>lun 29/08/17</td>
</tr>
<tr>
<td>48</td>
<td>Tests + pages</td>
<td>2 días</td>
<td>mar 29/08/17</td>
<td>mar 30/08/17</td>
</tr>
<tr>
<td>50</td>
<td>Legal notice</td>
<td>2 días</td>
<td>mar 29/08/17</td>
<td>mar 30/08/17</td>
</tr>
<tr>
<td>52</td>
<td>Privacy policy</td>
<td>2 días</td>
<td>mar 29/08/17</td>
<td>mar 30/08/17</td>
</tr>
<tr>
<td>53</td>
<td>Frequently asked questions</td>
<td>2 días</td>
<td>mar 29/08/17</td>
<td>mar 30/08/17</td>
</tr>
<tr>
<td>54</td>
<td>Terms of use</td>
<td>2 días</td>
<td>mar 29/08/17</td>
<td>mar 30/08/17</td>
</tr>
<tr>
<td>55</td>
<td>Shipping methods</td>
<td>2 días</td>
<td>mar 29/08/17</td>
<td>mar 30/08/17</td>
</tr>
<tr>
<td>56</td>
<td>Shipping and returns terms</td>
<td>2 días</td>
<td>mar 29/08/17</td>
<td>mar 30/08/17</td>
</tr>
<tr>
<td>57</td>
<td>Meeting with the client 5</td>
<td>3 días</td>
<td>jue 31/08/17</td>
<td>jue 01/09/17</td>
</tr>
<tr>
<td>58</td>
<td>Page styling</td>
<td>5 días</td>
<td>vie 01/09/17</td>
<td>vie 02/09/17</td>
</tr>
<tr>
<td>60</td>
<td>Removing parts from the previous website</td>
<td>5 días</td>
<td>vie 01/09/17</td>
<td>jue 07/09/17</td>
</tr>
<tr>
<td>61</td>
<td>Page review</td>
<td>5 días</td>
<td>vie 01/09/17</td>
<td>jue 07/09/17</td>
</tr>
</tbody>
</table>

¡Ya se han terminado todas las tareas! Felicitaciones!
2. STATE OF THE ART OF THE TECHNOLOGY USED OR APPLIED IN THIS THESIS

As explained in §1, this project has been carried out within Computer Sciences Brand, as the work needed for one of its clients. That is why the technologies that are used were defined by the requirements of both companies. This caused that sometimes the technologies used are not in their last version, per client’s requirement, due to the fact that this project is based in a previous one and at the start of the project there were some time restrictions. This project includes the use of several technologies, which are here presented.

2.1. Magento

Quality Materials has been developed with Magento, an open-source platform for e-commerce websites. The last version of Magento is 2.1.8 (to date 12th September, 2017, see https://magento.com/tech-resources/download), but for this project we are using the version 1.7.0.2. The reason for that is that the website is built from another website that already existed, that was using this version of Magento, and the company and its client took the decision of also using the version 1.7.0.2. Using this version, however, ensures a good functioning of the platform, because it has already been widely used and tested, and there is a great support on the Internet (see https://marketplace.magento.com/ or https://magento.stackexchange.com/) to it. There is a great number of articles on the Internet regarding different aspects of Magento, which also made the learning process less difficult to the author.

Magento Open Source, formerly known as Magento Community Edition, is an open-source e-commerce platform written in PHP. The software was originally developed by Varien, Inc, a US private company headquartered in Culver City, California, with assistance from volunteers [4]. According to Magento’s official website, Magento is the most popular commerce platform in the world, with more than 250,000 merchants around the globe [5]. Big companies, such as Ford or Cisco Systems Inc. use Magento as their e-commerce platform [6]. Magento has a global ecosystem of 150,000 developers and a network of more than 300 highly-trained solution partners [5]. The fact of being open source allows the developer to have the full source code at its disposal, making it easier to customize the solution to each specific project. What’s more, Magento features a market place (at https://marketplace.magento.com/) where it is possible to download extensions developed by third-party developers. Some of these extensions are free and others are paid. §3 of this document presents more in detail the insights of Magento and each one of the technologies it uses. Detailed information about all the versions of Magento can be found at https://magento.com/tech-resources/download.
3. System Architecture

One of the author’s main tasks was to understand how the software system works, in order to be able to develop the new website. This section presents the different components that make up all the system, and shows why Magento is a good choice for developing e-commerce web applications. The first subsection gives an overview of the system functioning, and the subsequent subsections describe its most important elements.

3.1. Overview

Magento’s workflow involves a great number of steps and details in order for the webpage to function properly [8]. This section summarizes some of the key steps that take place when the website is accessed, so that the reader that is not familiar with Magento can picture how Magento works. Please, remark that this process is more complex than what is here presented, and that some aspects are not here exposed for the sake of simplicity.

Firstly, the final user enters the desired URL into the browser’s navigation bar, to access the store developed with Magento, in our case, qualitymaterials.es. The Apache server receives the request, and, following the set of rules that determine its behavior, it searches for an index.php in the root folder. If it did not find an index.php, it would look for an index.html, or, otherwise, for the subsequent elements specified in a list appearing in the rules defined in this server. In our case, it finds index.php in the server’s root. This index.php is the file responsible for executing Magento. A snippet of the code can be seen in Fig. 2. Note that Magento’s installation directory is known as Magento’s root folder, and the directory where the server first looks for an index.php file is known as the server’s root directory. In our case, we make them coincide, and that is why we talk about root folder.

```
46 define("MAGENTO_ROOT", getcwd());
47 $compilerConfig = MAGENTO_ROOT . '/includes/config.php';
48 if (file_exists($compilerConfig)) {
49    include $compilerConfig;
50 }
51
52 $magefilename = MAGENTO_ROOT . '/app/Mage.php';
53 $maintenancefile = 'maintenance.flag';
```

Fig. 2. Index.php, under the root folder of Magento.

Magento’s full execution workflow can be found at [8], but here I give a reduced version just for clarification. In order to generate the HTML files that the final user will receive to be shown in his browser, Magento uses what’s known as a layout system. In this layout system, there are a set of template files with extension .phtml that contain HTML and PHP code mixed. The HTML code defines the structure of the website and all its content. The PHP code is, in general (but not always), responsible for retrieving data from the database or images from the folders where they are stored. PHP code is inserted within HTML code to fill the content of the web page. Also,
sometimes HTML code is generated from PHP. A single page in a Magento website is made up of different .phtml templates, as it can be seen in Fig. 3.

![Diagram showing how different templates are used to create a page](image)

Fig. 3. Each page is made up of different templates put all together.

The layout system also makes use of several .xml files, known as layout files, which define how the different templates must be arranged in the page. Fig. 4 shows an example of a template and a layout file.

```xml
<hi class="loginTitle">?php echo $this->__('Create new account'); ?></hi>

<h1 class="loginTitleUnderline registerTitleUnderline">?php echo $this->__('Login'); ?></h1>

<div class="text-center">
   <h2 class="RegisterSuccessTitle marginBottom">?php echo $this->__('Your account registration was successful. You can now log in using your credentials.'); ?></h2>
</div>
```

Fig. 4. On the left, snippet of registerSuccess.phtml, the template used after the user successes in registering in the platform. On the right, snippet of CSB_customer.xml, which uses the template registerSuccess.phtml.

When Magento receives the URL entered by the user, it breaks it down into pieces. These pieces of the URL will determine the specific execution of the web application and which page is shown.

Fig. 5 summarizes the process that Magento follows, in very general terms.
3.2. Technologies used by Magento

This section briefly describes the different technologies that are used by Magento to create the e-commerce web application. In general terms, HTML defines the structure of the web, CSS/Bootstrap defines its style, PHP is used in the backend, and MySQL is used for managing the database.

3.2.1. HTML

HTML [9] is the standard markup language for creating web pages. HTML stands for Hyper Text Markup Language, and it describes the structure of web pages using markup. An HTML page is built upon HTML elements, which are represented by tags. When the browser receives an HTML page, it renders its content by using the tags present in the HTML document.

A tag always must be opened and closed, and is a piece of code of the form `<tag> </tag>`, where tag can take several values, as for example, head, title, body, h1, or p.

An HTML document follows a general structure shown in Fig. 6:

```
<!DOCTYPE html>
<html>
<head>
	<title>Page title</title>
</head>
<body>
	<h1>Heading one is used to emphasize</h1>
	<p>The text goes normally into the tag p.</p>
</body>
</html>
```

Fig. 6. General structure of an HTML file document.
The tag head includes relevant information such as the title of the page, declarations of stylesheets or metadata. The title of the page is displayed by the web browser in the tab where the website is shown, as shown in Fig. 7. The content of the page is included within the tag body.

Fig. 7. The page title is shown in the corresponding tab of the web browser.

3.2.2. CSS & Bootstrap

CSS [10][11], which stands for Cascading Style Sheets, is a language for describing the style to be applied to the HTML elements of a page. CSS controls how HTML elements are displayed. It defines its position, the relationship between different elements, and it allows for the styling of those. The following list a few of the styling aspects that CSS can control:

- Position of an element
- Definition of an element as inline, or block, etc.
- Size of the text.
- Color of the text.
- Font used in the text.
- Background color of an element.
- Size of an image.
- Margins of an element.

There are two ways of defining the styles of an HTML page. It is possible to include CSS definitions within the HTML code, by using the tag ‘<style>’. However, this technique should be avoided always that is possible, at least in complex systems as this one. The alternative to that is making use of .css files, which include the definition of classes that later will be used in the HTML code. To give style to an element, it is only necessary to use the tag ‘class = myCssClass’, and to let the browser know where it can find this class. This can be achieved by including the .css file in the head of the HTML document. In Fig. 8, .sectionHeaderUnderline is a CSS class used in landingqm.phtml.

Fig. 8. On the left, the css class sectionHeaderUnderline defined in landing.css. On the right, snippet of landingqm.phtml where the CSS class is used to define the style of the HTML element hr.

It is worth remarking that more than one stylesheet can be used in the same HTML document, so that classes from different CSS files can be used.
3.2.2.1. Bootstrap

Bootstrap [12][13] (http://getbootstrap.com/css/) is a free and open-source front-end web framework used for designing websites. It contains an extensive list of CSS classes, as well as other components that were not used in this project. One advantage of bootstrap is that most of its classes are designed to be responsive, and therefore in most of the cases the developer does not need to take care of making the element responsive if a Bootstrap CSS class is used.

Using Bootstrap in a web development project is simple. A stylesheet definition must be included in the head section of the HTML document defining the page, as it can be observed in Fig. 9.

```html
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-Nf1BnM2ax3a1Jknhx1OnVhe41J7LMmpQacRrLZFVQhj46szyQ/hy3BXXs98ugV7qz" crossorigin="anonymous">
```

*Fig. 9. In Magento, head.phtml includes most of the definitions for the head of the HTML document. Here, we define one of the stylesheets to be used as the one from Bootstrap.*

3.2.3. PHP

PHP [14][15][16] is a server-side scripting language, and a powerful tool for making dynamic and interactive web pages. PHP is free and widely-used across the world-wide-web.

PHP [14] code may be embedded into HTML or HTML5 markup, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications. The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

PHP files, as the one in Fig. 10, are also known as PHP scripts, and they are executed on the server. PHPer is an acronym for “PHP: Hypertext Preprocessor”. PHP is used, for example, in WordPress [17] and in Facebook [18], showing its capabilities for managing large and complex systems.
A PHP file can contain, in addition of PHP code, code written in HTML, CSS and JavaScript. PHP can accomplish the following tasks:

- Generating dynamic page content
- Creating, opening, reading, writing, deleting, and closing files on the server.
- Collecting data from forms.
- Send and receive cookies.
- Adding, deleting or modifying data in a database.
- Encrypting data.
- Output HTML, images, PDF files, etc.

PHP runs on various platforms, such as Linux, Windows, MAC OS X, Unix, etc., and it is compatible with most of the servers used nowadays, such as Apache. It supports a great number of databases.

3.2.4. MySQL & SQL

MySQL [20] is an open-source relational database management system. SQL [19] is a programming language used for managing data held in relational databases.

The use of SQL commands is not explicitly needed in Magento, because the SQL queries are abstracted under the Magento’s methods, and thus the programmer does not need to code any SQL code nor having any knowledge of this programming language. Notwithstanding, SQL queries can still be done in Magento, if there is a need for a faster implementation of the website.

3.2.4.1. PHPMyAdmin

The database can also be easily accessed via PHPMyAdmin [21], a free and open source administration tool for MySQL. In Fig. 11, a screenshot of phpMyAdmin of Quality Materials shows three rows of the table core_config, one of the tables that Magento defines in the database.
3.3. Magento

Magento is an open-source e-commerce platform, with two available editions (at the start of the project): Community Edition (CE) and Enterprise Edition (EE). For this project, the Community Edition was used. Magento employs different technologies and put them all together in a way that makes it simpler to develop e-commerce web applications than if the developer had to develop everything from scratch. In general terms, Magento uses PHP on the server side (backend), and the client side (frontend) of the application is created using templates that use both HTML and PHP. The styling of the pages shown in the client side is done with CSS. Magento uses a layout system, which in turn uses XML files; the idea of the layout system is that there are certain XML layout files that describe how templates must be organized, and that make it possible to easily design the website just by adding or removing blocks. Magento employs the open-source relational database management system MySQL to manage the database it uses. Some parts of the database are structured following the EAV (Entity Attribute Value) model. Magento features a very complete administration page which is useful both for the developer and for the administrator of the website.

Magento is built on top of Zend framework.

3.3.1. Folder structure

Magento has a set of folders and a database. The folders must be installed in what is known as the root directory of Magento. This root folder must be placed in the www folder of a server (the nomenclature of this folder may vary depending on the server where Magento is stored).

Magento has a defined folder structure [23]. This folder structure must always be respected, because the platform expects to find a given file in a specific place, and it also expects that the name of the folders and the files follow the conventions defined by Magento.

Magento includes several files and folders. In the root directory, there is an index.php file, which is responsible for starting the process required from the moment when the client requests the webpage until it is finally rendered in its screen.
In the root directory, we can find the following folders and files.

- app/code, with almost all the PHP code of the application, which is structured in modules. It has three subfolders:
  - core, with the default Magento modules.
  - community, for modules developed by the Magento community.
  - local, for modules developed by the website developer.
- app/design, with the files related to the design of the webpage. The folders that are within this one are the different design themes available, and the one that is currently used can be selected from the Admin panel. Inside of each theme, we can find the files used by the layout system, which are stored in the subfolders:
  - layout, for the .xml files.
  - template, for the .phtml template files.
- app/etc contains configuration files:
  - local.xml, which defines the database to be used, the data to connect to the database via PhpMyAdmin, and the name of the URL for the admin page.
  - The folder modules/ contains the declarations of the modules, so that Magento knows which modules are available at each moment.
- app/locale, with .csv files used for translating texts and e-mail templates
- In the folder app/, the file Mage.php glues together all the Magento modules, functions and files. This file should be called if Magento is to be executed outside the root folder.

downloader/ is a folder containing the files that allow Magento Connect Manager to work. Magento Connect Manager allows to install third party modules from Magento Connect (https://marketplace.magento.com/), in a very simple way.

- skin/ includes files used by the templates, such as css files or images.
- media/ stores the images uploaded to Magento and the images resized dynamically by Magento.
- var/ holds all the temporary files used by Magento, as for example, the cache or the log files.
- js includes javascript files.
- lib/ contains the code of Zend framework.

As it can be seen, the developer has the full code at its disposal, because Magento CE (Community Edition) is open source. That is why the web-application can be fully customized. The developer can decide to just change the functionality of the application, by just overriding some of the core functionalities or adding new ones, or to just change the design of the application. Or he or she can change both the functionality and the design.
3.3.2. Modules

Magento organizes its code into individual modules [3], based in functionality. A module in Magento has a predefined folder structure, and it includes the following folders: Block, controllers, etc, Helper, Model and sql. Each folder has different responsibilities, and not all of them need to be used for a module to function correctly [25].

3.3.3. Breaking the URL into pieces

Magento’s URL are, in general, clean and intuitive [26]. Briefly explained, when Magento parses a URL it gets separated as follows

http://example.com/frontName/actionControllerName/actionMethod/

The value of ‘frontName’ is defined in the xml file of each module, so all URLs with this frontName will make use what is defined in the module corresponding to the xml file. Further, actionControllerName specifies which controller file is to be used. Normally this file is named IndexController.php, and it is stored under the folder /controllers. Finally, methodName defines the action to be taken, i.e. the function inside the controller file.

As an example, in

baseUrl/customer/account/login/

Magento will consult the global config to find the module to use for the frontName customer (Mage_Customer). Then, it will look for the account Action Controller (Mage_Customer_AccountController). Finally, the method loginAction will be called on the Action Controller.

In this way, it is easy to know what Magento is doing only by taking a glimpse at the URL.

3.3.3.1. Maneres de sobreesciure codi

3.3.4. Datab ase with EAV model

Magento database is structured following the structure entity attribute value (EAV) [27]. In Magento, an EAV database system is built on top of a traditional relational database. In a traditional database, tables have a fixed number of columns. Each ‘entity’ would have the same number of attributes. Instead, in an EAV database, each ‘entity’ being modeled has a different set of attributes. This way of storing data makes sense in an ecommerce platform, because each store will sell different kinds of products and each product will have different attributes. Using a traditional database would cause that the database is very sparse. In Fig. 12, catalog/product is the entity, and its code is stored in entity_tipe_id, which has value 4. If now we take a look at the tableeav_attribute, in Fig. 13, we see which are the available attributes for this entity, for example, name, description, or sku,
<table>
<thead>
<tr>
<th>entity_type_id</th>
<th>entity_type_code</th>
<th>entity_model</th>
<th>attribute_model</th>
<th>entity_table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>customer</td>
<td>customer/customer</td>
<td>customer/attribute</td>
<td>customer/entity</td>
</tr>
<tr>
<td>2</td>
<td>customer_address</td>
<td>customer/address</td>
<td>customer/attribute</td>
<td>customer/address_entity</td>
</tr>
<tr>
<td>3</td>
<td>catalog_category</td>
<td>catalog/category</td>
<td></td>
<td>catalog/category</td>
</tr>
<tr>
<td>4</td>
<td>catalog_product</td>
<td>catalog/product</td>
<td></td>
<td>catalog/product</td>
</tr>
<tr>
<td>5</td>
<td>order</td>
<td>sales/order</td>
<td></td>
<td>sales/order</td>
</tr>
<tr>
<td>6</td>
<td>invoice</td>
<td>sales/order_invoice</td>
<td></td>
<td>sales/invoice</td>
</tr>
<tr>
<td>7</td>
<td>creditmemo</td>
<td>sales/order/creditmemo</td>
<td></td>
<td>sales/creditmemo</td>
</tr>
<tr>
<td>8</td>
<td>shipment</td>
<td>sales/order/shipment</td>
<td></td>
<td>sales/shipment</td>
</tr>
</tbody>
</table>

**Fig. 12. Table eav_entity_type.**

<table>
<thead>
<tr>
<th>attribute_id</th>
<th>eav_entity_type_id</th>
<th>attribute_model</th>
<th>entity_type_id</th>
<th>attribute_code</th>
<th>attribute_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>58</td>
<td>1</td>
<td>available_sort_by</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>1</td>
<td>default_sort_by</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>1</td>
<td>name</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>1</td>
<td>description</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>1</td>
<td>short_description</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>1</td>
<td>sku</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>1</td>
<td>price</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 13. Table eav_attribute.**
4. Methodology / Project Development

In this section, the methodology used for this project is specified, as well as the development performed by the author. It also includes a section specifying all the tools and resources that have been used.

4.1. Project methodology

4.1.1. Client’s follow-up

As it is usual in every software project, at the beginning of our project, I had to understand what the need of the client was. In this case, the client needed to set up an online channel for selling products to other companies, but not to individuals. After defining the main objective of the project, the requirements had to be specified.

After that, we set periodic meetings with the client to show them how the project advanced. In the meetings, we were provided with the graphic design of the website, we reviewed the development of the website, the problems that eventually arose, and the following steps to take.

4.1.2. Learning by doing

At the start of the project, the author’s knowledge on web development was scarce. Magento is a very complex system which implies the use of a great range of technologies, as exposed in previous sections. Most of these technologies were not so well-known by the author before starting the project, and they needed to be learnt and understood. It was an ambitious task, and firstly, it implied reading several tutorials about programming languages, server setting, etc. But the range of technologies is so wide, that the most effective way of learning was by learning-by-doing [28], which means that, instead of having a theoretical knowledge about something specific, the learning of the knowledge is achieved at the same time than the development is performed. Thus, at first it is a slow process, since everything is new and needs to be understood and learnt at the same time. The first developments are less elaborated, but this is not a big problem, because after the development phase starts, the learning process becomes increasingly faster. Developing in Magento ranges from something so simple as modifying a setting in the administrator page to developing complex PHP modules. With this methodology, the simple things are learnt first, and as the project goes on, more advanced things can be performed by the learner.

4.1.3. Development and production environment

Since this was a website development project, the usual procedures for developing websites were applied. One important point when developing a website is the environment where this is done. The development is performed in the so-called development environment, which is nothing more than the computer where the development is done, properly set up
for each specific case. Once the development is completed, an FTP client (in our case, WinSCP, see §4.2.5) is used to upload the website to a live server, which is also known as the production environment. If there are changes to be applied to the website, they are to be done under the development environment, and they are only uploaded to the production environment once the developer verifies that everything works correctly and is well configured.

4.2. Tools and resources used in this project

A web development project implies the use of several technologies and platforms. For each one of the tasks to be accomplished, there is often more than one tool available. This section lists the ones that were used in our case.

4.2.1. Computer specifications

The development was done in a computer of the brand Acer, with Windows 10 64 bits, processor x64 Intel® Core™ i5-4210U CPU @ 1.70GHz – 2.40GHz, and RAM of 8 GB.

4.2.2. Firefox developer edition

Firefox Developer edition is a browser from Mozilla which makes it easier to develop websites. With Firefox Developer, the developer is able to easily inspect all the elements of the HTML page and its related CSS classes, in an enhanced way. Firefox Developer edition features a great number of tools specific for website developers which are of great help when developing and debugging the website. The page inspector can be observed in Fig. 14. Note that on the left side of the inspector the HTML code of the page is shown, whereas in the right side the CSS rules can be seen.

![Firefox developer edition's code inspector.](image)

There are two basic plugins which are very useful and have been used:

- Updating the page and clearing Firefox’s cache in a single button.
- Measuring distances in pixels.

4.2.3. Sublime Text

There are several code editors, each one with different advantages. Sublime Text is a text editor for code, markup and prose. Its interface its very simple, and that is one of its best
aspects. It can be fully customized, which makes it a powerful tool. It has several features that are very useful for writing code in a more efficient way, such as auto-completion, code-coloring or writing in different lines at the same time.

4.2.4. Magento’s template path hints

Magento’s administrator features a very powerful tool for developers which, if enabled, shows in the frontend hints about the templates used across the page, as shown in Fig. 15. However, for security reasons, this should not be used in the production environment, but only in the development environment.

This may seem trivial, but it can be extremely useful in some situations, since Magento file and folder structure is so extensive that sometimes it is difficult to know which template is the one that is being used in a specific place.

![Magento’s template path hints]

Fig. 15. When path hints is activated, the frontend shows which templates are used in each section of the page.

4.2.5. WinScp

An FTP client is the most efficient way to transfer files to the server. The alternative would be using the cPanel provided by the hosting provider, and accessible by means of the browser. However, the latter is not a good option, since the interface is not optimized for managing and uploading files and folders to the server. The only time when cPanel needs to be used is when creating, modifying, reading or deleting .htaccess files, because they cannot be seen from the FTP client.

In our case, we use WinScp, which stands for Windows Secure Copy, and is a free and open source FTP client for Microsoft Windows. When accessing the server for uploading files via FTP, the user needs to introduce the IP and the password for the FTP connection, which should be provided by the host provider.

4.2.6. Online resources

Magento’s official documentation features some useful tutorials that introduce the developer to Magento. However, they are not enough to understand all the capabilities of Magento, and it is often necessary to search elsewhere to find information about different topics. Among an extensive list, the pages with the most useful information are:

• http://magento.stackexchange.com/ and https://stackoverflow.com/ are the best forums for posting questions about Magento.
• http://alanstorm.com which contains some useful tutorials.

4.3. Project development

4.3.1. Setting up the environment, installation

4.3.1.1. Installing a server

As in every website development project, the first step after the requirements gathering is the setting of a development environment. At the start of the project, we were provided with the source code of the previous website. This would be the starting point from which our website’s development would start. So, firstly we had to install the source code in our computer. It became then necessary to install a server in the development computer. We chose WAMP server, which stands for Windows, Apache, MySQL and PHP, and is a variant of LAMP [29], a solution stack suitable for building web applications. In WAMP, which is stored under the folder wamp64, there is a folder named www, where the PHP scripts must be stored. Thus, the websites are also stored inside www, each one in a separate directory. To be able to access the website, in WAMP it is necessary to create what’s known as an alias, as shown in Fig. 16.

![Fig. 16. Alias in WAMP.](image)

Which means that when accessing the URL 127.0.0.1/qualitymaterials or localhost/qualitymaterials, WAMP must look inside the C:/wamp64/www/qualitymaterials folder and look for the file index.php, or the subsequent elements defined in the list specified by the rules of the server.

In order to start developing, I needed to install the Magento store in the development computer. Firstly, all the files and folders needed to be copied into the appropriate folder. However, this was not enough for the store to work properly. The database needed to be imported.
4.3.1.2. Importing the database

As explained in the previous sections, Magento makes use of a database with a defined structure. The file we were provided with was big, and thus importing it was not a straightforward task. The file we were provided with was an .sql, which contains a series of SQL queries, weighting about 1.8 Gigabytes. I first tried to import it through PHPMyAdmin. In the panel, there is an option called ‘Import’ which allows to browse for an sql file and import it, as shown in Fig. 17.

![Importing through PHPMyAdmin](image)

The first issue that I faced was that the file was larger than the maximum allowed size. In order to solve this problem, I changed this size in the file php.ini, which defines the PHP settings for the server. The code for doing so was as follows:

```ini
upload_max_filesize = 2000M
post_max_size = 2001M
max_execution_time = 1800
```

After that, PHPMyAdmin was ready to import the file. I realized then that the import achieved the maximum allowed execution time, so I also had to change it with the code:

```ini
php_admin_value max_execution_time 21600
```

Now the import time was under the maximum allowed by PHP. The last issue that I faced was related to SQL. The import required deleting the code ‘ROW_FORMAT=FIXED’ from the file to be imported. Once the corresponding lines were deleted, the database could be properly imported.

Magento needs information for accessing the database. In the folder app/etc there is a file named local.xml which defines:

- The name of the host, in this case, localhost.
- The user name of PHPMyAdmin.
- The password for this user.
- The name of the database.
- The URL for the Admin panel.
This way, Magento has all the necessary information to access the database. After defining the corresponding values in local.xml, everything was ready to start developing the website under the development environment.

4.3.1.3. Uploading the application to a live web server.

Once the development is finished, the website is ready to be uploaded to a live server. The first step is to own a domain and a hosting, that can be purchased to any hosting provider. The easiest way to upload the website is using the file transfer protocol (FTP), by means of one of the several FTP clients available in the market. In our case, we made use of WinScp, which allows the upload of files via FTP in an easy and intuitive way. Before starting a FTP session in WinSCP, the user must specify the protocol to be used (FTP, in our case), the IP to which WinSCP must connect to, the port, the user, and the password, as shows Fig. 18. After that, the application easily allows to drag and drop files from the local computer to the live server. The first upload took several hours because of the connection speed. However, once the first upload is done, the next ones take considerably fewer time, because only the files that are changed need to be uploaded. The reason why more than one uploads are necessary is because the website is in continuous development and the client wanted to see its evolution. Nevertheless doing it this way was not a problem because the final user couldn’t see the development process, because the user accounts were not active while the development was taking place. The users could indeed see the pages available before login, which were effectively uploaded in the first upload.

![WinSCP](image)

*Fig. 18. Connecting to the server with WinSCP via FTP.*

WinSCP allowed us to upload all the files and folders to the server. Yet, the database still had to be uploaded. The process for uploading the database to the server was slightly more complex than in the case of the development environment. Firstly, the database under localhost must be exported, creating an .sql file. Then, in the server, a new database needs to be created (if there was an old version, it is previously deleted). Using PuTTY (http://www.putty.org/), I established an SSH connection with the server, and logged in using the credentials for FTP. Then, via PuTTY, I established a tunnel. In another command line terminal, I navigated to the folder where the .sql file was stored and executed
the command line `mysql -u UserName -p Password database_name < .sql_file`. Once the upload finished, I specified the values of local.xml, under app/etc/ to fit the new specifications for this server. The last step to be performed is changed the URLs. In the database, in the table core_config_data, these values can be found in the rows whose path are web/secure/base_url and web/unsecure/base_url. It is important remembering to delete the content of var/cache/

After this process, the website is ready to be accessed via a standard web browser.

4.3.1.4. Error 403

One day, the production environment showed a 403 forbidden access error in a section of Magento’s admin, namely the Admin -> Catalog -> Manage categories section. I did not know why this error was appearing, because I did not change anything in the code or the server settings. This error was solved when I contacted the hosting provider. They answered us back, explaining that they had included an .htaccess with a code to avoid this error to appear. The code itself is not included here for security reasons.

4.3.2. Website development

These sections describe the development that I performed. I only include here the major changes and new functionalities and designs to the website. In order not to make the document too long I do not include an exhaustive list of all the code that I wrote and all the actions I performed, but I summarize the most important ones.

4.3.2.1. Page layout

In Magento, all pages follow a page layout. A page layout is a .phtml template which defines the structure of a webpage. It generally includes parts of the webpage such as the header, the footer, or the content, among others. The big advantage of using page layouts is that the sections that are repeated in all the pages, such as the header or the footer, don’t need to be coded and stored in different files but in the same file.

Most of the pages of Quality Materials use the page layout 1column.phtml, which includes for all the pages the same header and footer.

The header of the page, in Fig. 19, is defined in header.phtml, and the footer of the page is defined in footer.phtml, and can be seen in Fig. 20. Both files are stored in template/page/html.

Fig. 19. Content defined in header.phtml.

40/76
Fig. 20. Content defined in footer.phtml.

It is important to remark that all the pages that make us of 1column.phtml will follow the same structure. This is a very useful feature, since there are parts of the page that are the same for all the different pages, and in this way, it is only necessary to define that the template to be used is 1column.phtml. Other options include 2column-left.phtml or 2column-right.phtml.

4.3.2.2. Landing page

When Quality Materials is firstly accessed, a landing page is presented to the user. The landing page had to include information about Quality Materials, as well as links to the registering page. The design of the page was provided by the client.

The development of the landing page was done by means of a new module and its corresponding layout. The module is simple; it only includes two files. In IndexController.php, the layout is loaded and rendered, and config.xml defines the module. Under app/etc/modules, CSB_LandingQM.xml declares the module so that Magento is aware of its existence. This file is also responsible for defining the URL that will be used to render the page by making use of the tag <frontName>, as shown in Fig. 21.

Fig. 21. Snippet of CSB_LandingQM.xml.

The layout responsible for defining the structure of the page is local.xml, under the layout folder, in the app/design/frontend/ directory. As an example, local.xml is presented in Fig. 22 to make it clear how different templates can be combined in Magento to make a single page. Observe that we first define the page layout (see §4.3.2.1) of the page to be used, 1column.phtml, then we add a stylesheet only to be used in this page, and finally we specify the two template files to be used in the content of this page. landingqm.phtml defines all the content that can be seen in the landing page of Quality Materials. For this page, in addition of the standard HTML and PHP, I also used Bootstrap. sliderAndSelector.phtml defines the content shown in Fig. 23.
sliderAndSelector.phtml makes use of an extension named banners, by VIAN, that was already installed on the previous store. I learnt how to create and add a new banner programmatically, although it can also be done via admin. The template firstly includes this banner and after that, it includes what we named selector, which are the two purple boxes that can be seen in Fig. 23. Content defined in sliderAndSelector.phtml. Depending on the displayed page, the one on the left or the one on the right are given more weight in terms of color.

The images that need to be displayed in Magento are retrieved using a Magento built-in function, called getSkinUrl, which points to the folder skin, that contains images and CSS that the page uses. The code in Fig. 24 shows the use of this function.

The resulting landing page can be found in §6.

4.3.2.3. Mandatory registering and login

An important requirement for the webpage was that it should only be accessible to business, and our client needs to be able to control who is able to access the webpage, i.e., it must be able to activate or deactivate a customer’s account. When a customer first accesses the webpage, it is required to register, and after doing so, it is told that his account is now registered but not active yet, and that it will be activated soon by members of Quality Materials.

The first attempt to solve this problem was the use of an extension to prevent the user to use the webpage before login. The solution was not elegant and the design was very poor.
Additionally, the required design changed. All of these reasons made us try different options for accomplishing our purpose.

My second attempt was adding the login via a .xml layout file to the home page. I could then show the login in the home page, and, using the `<remove>` tag in the corresponding .xml layout file, remove it after the clients’ login. This is shown in Fig. 25. However, some parts of the webpage could still be accessed even before login, so this solution was not good, it needed to be refined.

![Fig. 25. Mandatory login: first attempt.](image)

The final solution was using an Observer. This is a clean, elegant and robust solution. The way this was done was by creating a new module in the local code pool. In the module, the file Observer.php, in Fig. 26, controls which pages can be accessed before the customer logs in. If necessary, more pages can be added by simply including them in the code.

```php

//Check if any customer is logged in or not, or if he is clicking the contacts page
if(!Mage::getSingleton('customer/session')->isLoggedIn() && !in_array($request->getModuleName(), array('landingqm', 'landingqm/'))) { Mage::getSingleton('core/url')->setRedirect($controller, $action, $ident); exit; }
```

![Fig. 26. Observer.php controls the access to the page.](image)

The observer continuously verifies if the user is logged in or not, and in case it is not, it allows to access only certain pages. In this case, if the user tries to access a page that is not allowed, the observer will redirect him to the page /landingqm, explained in §4.3.2.2. Otherwise, in case the user is logged in, the observer simply allows him to browse through all the pages.

Additionally to the mandatory registering, the design of the login/registering page had to be changed. Different templates are involved in the login/registering panel, such as login.phtml, forgotpassword.phtml, registerSuccess.phtml and register.phtml.

The new design included new fields that were not present in the previous website, so I decided to create a new module. The new module is in fact overwriting Magento’s default registering process. In this new module, the file config.xml defines the overriding of Magento’s default module, the changes to be done in the Admin -> Customer area, and
the version of the module, which is used to let Magento know that it needs to execute the install/upgrade script under the /sql folder. Indeed, the /sql folder includes install/upgrade scripts, which define new attributes that Magento has to define and include in the database. This is the usual way of updating or adding fields in Magento, and an example can be seen in Fig. 27.

![Code snippet](https://example.com/code_snippet.png)

**Fig. 27. Snippet of upgrade-1.0.0-1.0.1.php, responsible for including the new fields for the customer.**

As a result, the registering page design is as shown in Fig. 28.

![Registering Page](https://example.com/register_page.png)

**Fig. 28. Registering page of Quality Materials.**

Another relevant change induced by the addition of the new fields to the customer was in the Admin panel of Magento. In the Admin panel, there is a section that summarizes the customers’ information in a table, showing the value of their different fields. To show the new fields, some coding needs to be done. Indeed, under Customer/Block/Adminhtml/Customer, the file Grid.php oversees the addition of this new fields to Magento’s Admin area. shows the code for adding one of the fields to the admin panel. Grid.php is 134 lines long, and thus I only include a snippet of the code here. The resulting view of the admin’s customer management section is presented in Fig. 30. Recall that the fields from ‘Nombre de la empresa’ until ‘Teléfono’ are all new fields added in the way explained above.
4.3.2.4. Customer activation

Another important requirement was that the client must be able to decide which customers are allowed to access the webpage. Technically, this means that there must be a way to activate or deactivate customers’ accounts. This is easily accomplished by installing a free extension from Vinai (https://github.com/Vinai/customer-activation). After this extension is installed, the customer management section of the admin includes a column indicating if the account is active or not, as presented in. It also includes a means to activate the account and, if the administrator wishes, to prevent the customer via e-mail that the account has been activated, as it can be observed in Fig. 31.

There are three e-mails involved in this process, and the three of them need to be designed and coded. In Magento, the code of the e-mails can be included in Admin -> System -> Transactional e-mails, as Fig. 32 shows. Here we can find different templates to be used in different situations. In our case, an e-mail is sent to the administrator of the page each time a client is registered (this can be deactivated if needed). An email is also sent to the customer before and after his account is active. The code of the e-mails is HTML and it is defined in the mentioned admin’s section.
4.3.2.5. My account page

Quality Materials features a ‘My account’ page that can be accessed at the URL baseUrl/customer/account, where baseUrl, in our case is qualitymaterials.es/index.php/es. In ‘My account’, a new table was included per clients’ requirement. The idea is that the customer should be able to see the quantity of all the products that he ordered in the lasts orders, as it can be observed in Fig. 33. The coding here was interesting because it involved different things. Firstly, the magic method (see §4.3.2.15) getOrders() is used here. There was also the need of grouping the quantity of ordered product for each order, and finally, to present this result in a table in HTML. This table is created using loops and combining HTML created from PHP. The coding here must be accurate; all tags must be properly closed, which is something that the developer might forget if not being cautious because here the HTML is being outputted from PHP.

![Fig. 33. Quantity of products ordered in the lasts orders.](image)

The page must also include a section with the account information and a link for editing it, and a section with the address information and a link for editing it. The page for editing the account information was the one that implied the most relevant changes, because the customer had new fields. This page uses a combination of templates, and the most important of them is edit.phtml, under template/customer/form. In order to display the information in the new fields, several magic methods (see §4.3.2.15), such as getTelephone(), are used as shown in Fig. 34.

```php
<input type="text" name="telephone" value="<?php echo htmlspecialchars($this->getAddress()->getTelephone()); ?>"/>
```

![Fig. 34. Use of the magic method getTelephone() to retrieve the telephone of the user.](image)
Bootstrap was also used here, making the design more robust. The content of the edit form is responsive because it is done with Bootstrap, but the whole page is not responsive because it was not made using Bootstrap. This is a major change that by the moment is not done, and that is left for future development. Fig. 35 presents the final version of the form for editing the account information, which can be accessed at baseUrl/customer/account/edit/.

![Edit form](image)

*Fig. 35. Form for editing the customer’s account information.*

Bootstrap was also used in other forms: create a new address (/customer/address/new) and edit an address (/customer/address/edit/id/-number-).

4.3.2.6. Pages with information about Quality Materials

As in most ecommerce webpages, Quality Materials has an ‘About us’ page, as well as a ‘Legal notice’ page, a ‘Terms of use’ page and a page with information about the delivery and the shipping. In Magento there is a special kind of pages known as CMS pages that can be easily created from the Admin -> CMS -> Pages section. The page ‘About us’ was made as explained in §4.3.2.2, and the other three were made as CMS pages.

In the admin, the title and the URL of the page can be defined. The user also needs to define which store views will be able to display this page, as well as if the page is active or not. This section also features a wysiwyg editor, as shown in Fig. 36 which allows to add text to the page in an easy way. The editor allows to modify the HTML code generated, which is useful if small changes are to be performed to the page.
These pages can be accessed at their corresponding URL (for example, baseUrl/aviso-legal) and by clicking the links that can be found in the footer of the page, Fig. 37.

4.3.2.7. Including a block via admin

In Magento, the admin’s CMS -> Pages section allows to edit the HTML code of the page within an editor. In one case, I used this option to include a block to show an image at the end of the page. The way of proceeding is as follows: the code

```php
{{block type="core/template" after="banners.home.row.2"
           name="reposicionesExpress.content" template="CSB/reposicionesExpress/template.phtml" }}
```

had to be included in the corresponding place of the HTML code. This piece of code includes the file template.phtml that I defined under the template/CSB/reposicionesExpress. The code of this file simply includes an image with some styling applied to it to make it fit properly in the corresponding place. The result can be observed in Fig. 38.

4.3.2.8. Categories and products

Magento classifies the items that are sold in the store in groups called categories. Each store must have a defined root category. The root category is the parent category for all the categories that the developer or administrator need to create, and it is not displayed on the frontend, even if Magento requires it to be defined.
In Quality Materials, new product categories are defined. I added the categories manually via Admin -> Manage categories, because the number of them was small, and in this case it was faster. It can also be done programatically if there is a big number of categories to be created. Quality Materials must have, by the moment, ten categories (the names are in Spanish):

- Metacrilato
- Policarbonato
- Poliestireno rígido
- PVC
- Polipropileno celular
- Caúcho / polietileno
- Cartón
- Espuma de polietileno
- Porexpan
- Espuma filtrante

These categories must be accessible from a menu shown at the top of the page, and the design must adapt to the one provided by the client, and thus the extension that was being used in the previous website (https://www.magentocommerce.com/magento-connect/responsive-custom-menu.html) needed to be used for the menu. However, the fact of using this extension made it more difficult to change the menu, because I was not its original developer and the documentation on how to use it is not very extensive. The extension itself is made up of several files, but it is remarkable to present here just a snippet of the code, because I spent a considerable time trying to understand how the extension was coded, and I consider that even if I did not code this by myself, it helped me to learn something that otherwise I might have not learnt. The code in question can be seen in Fig. 39.

![Code Snippet](image)

*Fig. 39. The code on the left creates two pieces of HTML and stores them in two PHP variables. The code on the right combines the HTML codes stored in the different PHP variables to finally generate the desired HTML structure.*

The whole process for creating the menu involves the use of PHP, JavaScript and HTML, and it is considerably difficult. That is why trying to modify the extension is not a trivial task, because the menu that is finally rendered does not depend only on one template, as it happens with other parts of the website and Magento.

49/76
One problem that I encountered is that the use of Bootstrap broke the structure of the menu, and it took me a long time to find out that this was indeed the causing of the problem. More information on this problem can be found at §4.3.2.14.

**Product images**

In localhost, all the images are correctly shown in Quality Materials. However, after the upload to qualitymaterials.es some images did not appear. I made a research and tried different things which are here presented.

The first attempt was to upload again the images. The images in question are stored under media/catalog/product. Inside this folder, the images are stored in function of its name [30]. The two first characters of the name are taken into account and define the folders where an image will be stored. For example, policarbonato-celular will be stored under media/catalog/product/p/o, as well as all the other images whose name starts by ‘po’. I uploaded again the images of one of the products, deleting the corresponding folders in the server and uploading them again from localhost. After that, the images did still not show.

![Image](https://example.com/image.png)

*Fig. 40. Rules for storing product images in Magento. Source: https://magento.stackexchange.com/questions/1005/get-original-image-file-from-a-cache-url/1008#1008*

Fig. 40 is taken from an answer of the user Ben Lessani – Sonassi from Stack Overflow Magento’s forum, and it quickly shows how Magento stores its product images.

I learnt then that Magento has a cache for the images of the products. This cache can be found at media/catalog/product/cache. From the code inspector, I also could see that the page was taking the images from this folder. Thus, I tried reuploading the folder. Again, I had no success. I also tried to flush the cache from Magento’s admin panel. Magento allows to flush all its caches with only some clicks, as shown in Fig. 41. After that, the images did still not show.
After some research, I found that a cause for this problem can be that the folder in the server do not have the appropriate read/write permissions. The idea is that servers define some permissions for each one of the folders of the website. Changing the permissions of the media folder and other folders did neither result in the images showing correctly on the webpage.

When Magento cannot find an image, it shows instead an image known as placeholder. As Fig. 42 shows, I changed the placeholder of Quality Materials to adapt it to the new site, in contrast with the old placeholder in the previous website.

Finally, and because the number of products that will be included in Quality Materials will be considerably smaller than in the previous website, I decided to upload the product images manually. Learning how exactly Magento manages product images is something that should be taken care of in the future. However, the lack of documentation sometimes makes this process slow.

**Script to remove/disable products**

In Quality Materials, a great number of the products that were included in the previous website are not sold anymore. The previous website has thousands of products stored, and thus, manually removing/disabling them is not a feasible process. Instead, programmatically removing/disabling them would be the most appropriate way of doing so. For this purpose, I created a script to remove or disable a list of given products. The functions shown in Fig. 43 should be used in case that there is a need of removing or disabling a great number of products.
Fig. 4.3. Script to delete or disable products.

4.3.2.9. Hiding menu and search box before log in

The menu used in Quality Materials is created by a third-party extension (https://www.magentocommerce.com/magento-connect/responsive-custom-menu.html). In order not to show the menu before the user is logged in, the sentence

```php
<?php if(Mage::getSingleton('customer/session')->isLoggedIn()): ?>
```

is used. This piece of PHP code calls a function included in Magento named isLoggedin(), which returns true in case that the user is logged in or false otherwise.

The search box is also created by an extension by Mirasvit (https://www.magentocommerce.com/magento-connect/search-ultimate-sphinx-search.html), and the same procedure is used to show it only after login. Shows the piece of code in head.phtml where this is defined.

```php
<?php if(Mage::getSingleton('customer/session')->isLoggedIn()): echo $this->getChildHtml('topSearch'); ?>
```

Fig. 4.4. The search box is only shown after log in.

4.3.2.10. Specific CSS for some pages

In Magento, it is easy to apply the styles defined in a CSS stylesheet to a specific page. To achieve this behavior, in the .xml layout file of the corresponding page, I defined the stylesheets to be used, as for example in the case of the landing page. The tag `<action method = "addCss"><stylesheet>` must be included inside the tag `<reference name ="
“head”>, and then the stylesheet is to be defined. A snippet of this coding is shown in Fig. 45.

```xml
<reference name="head">
  <action method="addCss"><stylesheet>css/csb/landing.css</stylesheet></action>
</reference>
```

*Fig. 45. Definition of the stylesheet to be used in the landing page of Quality Materials.*

4.3.2.11. Removing Zopim’s chat

The previous page features a chat from Zopim (https://www.zopim.com/), as shown in Fig. 46. In Quality Materials, this chat had to be removed. There was no way of removing it from the Admin, because there was not a section to manage the extension.

*Fig. 46. ‘Send a message’ in the previous webpage.*

After some search, I found that the chat was included in the page as a definition in the HTML head. In head.phtml, under template/page/html, there is a line of code like this: `<?php echo $this->getIncludes() ?>`, which is retrieving information from the configuration in Magento’s admin panel. Indeed, it is retrieving the information that the developer can specify under Admin -> System -> Configuration -> General -> Design -> Html Head. After removing this line of code, the chat did not show anymore in the page. However, the aforementioned admin’s section was empty, and thus the content of the chat could not be retrieved from that place. However, writing the line of code back caused the chat to show again on the page. After a bit of search, I found that in the database, in the table core_config_data, the row with path design/head/includes had indeed the definition of the chat from Zopim. Deleting this definition finally resulted in the chat not showing on all pages of Quality Materials.

4.3.2.12. Admin’s URL

In Magento, by default the admin URL is baseUrl/admin. baseUrl is the base URL of the store, in our case qualitymaterials.es/index.php/es. It is advisable to change this URL to a value that cannot be easily guessed by a potential attacker to the site. For security reasons, the specific URL that was chosen is not here included. The URL can easily be changed using the tag frontName in local.xml, under app/etc/local.xml, as it is presented in Fig. 47.
4.3.2.13. Page under maintenance

A problem that I sometimes encountered while developing was that the page entered in ‘maintenance mode’. At first it seemed as if this problem disappeared for no apparent reason. This was the very first problem I encountered when installing the webpage in the development environment and it can be seen in Fig. 48.

![Page under maintenance](image)

Fig. 48. Page under maintenance.

After I observed this behavior several times, and after a bit of research, I found out that Magento creates a file named maintenance.flag under certain conditions, such as when there are some kind of errors, and this forces the webpage to enter the ‘maintenance mode’. Then, I also changed the page in order to adapt it to Quality Materials. The final appearance is presented in Fig. 49.

![Quality Materials](image)

Fig. 49. Maintenance page of Quality Materials.

4.3.2.14. Page structure broken by Bootstrap

The version of Magento that we are using, 1.7.0.2 (per client’s requirement) does not use Bootstrap by default. In the new pages that we created we decided to use Bootstrap because it is a powerful and flexible tool for designing web pages. However, in order to use Bootstrap (see §3.2.2), a definition of a stylesheet must be included in the head. This makes that all the pages are affected by the styles defined in Bootstrap. Bootstrap defines a great number of CSS classes, which affect a good number of elements in the page. Sometimes the styles defined by Bootstrap collide with the styles defined by the developer, and normally the ones from Bootstrap prevail over the others, because they normally have more weight. After
starting using Bootstrap, some of the pages’ structure appeared to be disorganized, but at first I did not realize why this was happening. After different attempts that did not show any effect, I discovered that Bootstrap defines the CSS property box-sizing with the value border box, and this was messing up the page. A new class definition was then included in several templates used, for example, in the product page, in order to get the expected page structure back. The old and the new behavior can be observed in Fig. 50. This problem caused the development to take longer, because the behavior introduced by Bootstrap was not so straightforward to be found.

Fig. 50. On the left, product page affected by the Bootstrap’s definitions. On the right, the same page with the styles corrected by the new CSS class.

Bootstrap also caused the menu to display unstructured. The same solution than in the case of the product page was applied and the problem was solved. In this case, it took me longer to discover what was causing the error, because the menu of Quality Materials is created by a third-party extension (https://www.magentocommerce.com/magento-connect/responsive-custom-menu.html).

Fig. 51. On the left, menu unstructured because of the use of Bootstrap. On the right, menu with the styles corrected by the new CSS class.

4.3.2.15. Magento’s magic methods

In Magento there is a special kind of functions known as magic methods [31]. The idea is that if a function called in the code cannot be found by Magento, the latter uses then the _call() method from the Varien_Object class to perform the action. Varien_Object is the class from which all the objects in Magento inherit from. The method _call() will then be
able to manipulate the protected \$_data array within the object. There are four options for defining a magic method: get, set, uns, has. For example, getOrders() would return the information contained in \$_data['orders']; setOrders() would set the information contained in \$_data['orders']; unsOrders() would unset the information contained in \$_data['orders']; and hasOrders() would return a boolean indicating whether or not there is a key named ‘orders’ in the array \$_data.

4.3.2.16. Language of the website

The previous website is in three languages. Quality Materials will be only in Spanish. This was simple to achieve thanks to how Magento builds store views. Indeed, for the same store, different views can be created, each one with a different language. The only thing that needed to be done is disabling the store views in English (EN) and in Catalan (CA). This can easily be achieved through Magento’s admin page, under System -> Manage stores, as presented in Fig. 52.

![Fig. 52. Store views can be enabled or disabled from the admin panel.](image)

Having the same page in different languages implies that translations need to be done. Instead of repeating the same code three times with the text translated, Magento uses some special functions that are responsible for the translations. The translations themselves are stored in files with extension .csv that are stored under app/locale/(language of choice, in our case es_ES). An example can be seen in Fig. 53. There are several files, each one corresponding to a different section of the page.

![Fig. 53. Macho_Checkout.csv, under app/local/es_ES, includes translations needed in the checkout page.](image)

Each file has several lines, with two strings separated by commas. The first string is in English, and the second in the desired language, in our case, Spanish. Most of the translations are already done in the default Magento installation, in different languages. However, new translations can be added to the files. I had to add several translations for different sections of the webpage. The advantage of using this way for translating strings is that the code is the same for all the stores, and adding new translations does not require
to modify the code but to add a line in the corresponding .csv file. It is important to remark that before a new translation is displayed in the website, Magento’s general cache needs to be deleted. Magento’s cache is stored under var/cache.

4.3.2.17. Removing CMS pages

Several CMS pages used in the previous website are no longer needed in Quality Materials, and thus they needed to be removed in Quality Materials. This can be done through Magento admin panel, under CMS -> Pages. However, if the number of pages is big, this can be a long process.

As it is usual in Magento, everything that can be done through the admin panel can also be done programmatically, i.e. by coding it in PHP. In order to make the process faster and reusable in the future, I created two functions, one for disabling CMS pages and another for deleting them. Fig. 54 shows a snippet of the code.

![Snippet of code](image)

_Fig. 54. CMS pages can also be programmatically disabled or deleted._

4.3.2.18. Default/home page

The homepage in Quality Materials, after the customer has logged in, is slightly different from the one in the previous website, and its structure is the same. We were provided with the design of the page.

From the previous website, the administration panel features a tab to manage banners. A banner is nothing more than an image that is placed in a specific place of a page. This is useful for the client and the administrator of the site, because it allows them to change images manually, in a fast way and without the need for contacting the developer, and thus providing them with flexibility. On the homepage, several banners are used. However, they appear in the same place than they were in the previous website, so the only change was in the images. Some of the banners needed to be removed, which was also a simple task via Magento’s admin panel.

The structure of the boxes where products are shown had its structure broken, so I had to fix it. Also, there were style changes to be applied.

Another important change to the home page was including a simple block to add the image show at the end. This is explained in §4.3.2.7.-
The home page’s final design can be found in §5.

4.3.2.19. Styling of the page

A considerable amount of the work done consisted in styling the web page. This styling is done with CSS, as already explained, combining different stylesheets depending on the page that is to be rendered. The results of the style changes or the new parts that were added to the website can be found in §5.

4.3.2.20. Error reports

After one error on the development environment, I had to reinstall the whole server and the magento store. When trying to access the frontend or the admin page, the only page that showed was the one in Fig. 55. I spent some days searching what could be the possible cause for that.

![Fig. 55. An unknown error was shown.](image)

Firstly I thought that the store had entered the maintenance mode, but no maintenance.flag file was found in the folder. After the research, I found that under var/report Magento stores a file reporting errors. An example of that can be seen in Fig. 56. As it can be observed, a specific table was not found in the database. After importing the table again, and importing other tables that were also missing, I could finally access the store and the admin.

![Fig. 56. Error report.](image)

4.3.2.21. Recommending products

Due to time restrictions and change in the requirements from client, the development of a recommender system is not finished by the moment. I started doing some research, but the main idea now is to finish a first version that is simple. The development of several parts of the website took longer than forecasted, and this caused that I could not go on with the
development of a recommender system. However, in this section I explain the ideas that should be used in the development of this first version.

The idea is that a customer should see on the display a list of products that he may be likely to buy, but that he did not look for intentionally. There are several kinds of recommender systems, depending on which information they use to base the recommendation. However, in this simple version, we will focus on a few things.

The information that we have available from the previous website, and that we can use to develop the recommender system in this simple version is:

- For a given customer, all the products that he ever bought.
- For a given customer, all the products that he ever viewed.
- The number of times that each product in the store was bought or viewed.

From this information, we can try to find similarities between customers and from there we can try to do recommendations. The idea here is that we first see which are the products that a customer bought. Then, we need to find similar customers. A similar customer is a customer who bought the same products, or a set of products similar to the one of our customer. Then, from all the customers that are similar, we can try to find which product they have in common that our customer did not buy yet. If we are able to find one, we can recommend it to our customer.

We can also recommend popular products, no matter which is the customer that we are recommending them to. If a product is the most popular in a store, we can recommend it to any customer. We can combine this recommendation with the other kind of recommendation, more targeted to the profile of our user.

The first products to take into account, both for our customer and for all the similar customers, are the ones whose bought quantity is greater. If the customer bought more quantity of a product we can assume that this product is important to him. This is a very simple assumption that needs to be refined, but keep in mind that by now we are only trying to face the problem and obtain a first version of a recommender.

The products a customer viewed and the number of times he viewed them is assumed in our case as purchased products, for simplicity. This is also another assumption that is done now, for the sake of simplicity.

The recommender system is under development, but here I present the set of actions in pseudo-code of this first version. Bear in mind that this pseudo-code is done taking into account that we are using Magento, and we need to work accordingly to it.

1*/Obtain the number of times each product was purchased*/

2*/Obtain the number of times each product was viewed*/
3*/Merge both values into an array and select the n (e.g., n=2) most popular ones. Recommend this n products*/

4*/For a given customer, obtain the products he bought and the quantity*/

5*/For all other customers, obtain the products they bought and the quantity*/

6*/For the given customer, find the m (e.g., m=5) most bought products*/

7*/Find the set of customers that bought the same m products than this customer, or a list of similar products*/

8*/Find which products were bought by these customers that were not bought by our customer*/

9*/Recommend the found products in common*/

Note that I already developed some scripts to retrieve the features of each product, which will be used to see if two products are similar.
5. Results

In this section, I present screenshots for the pages in which I have done some development.

Fig. 57. Header.

Fig. 58. Footer.

Fig. 59. Landing page.
Fig. 60. Contact us page.

Fig. 61. Login page.

Fig. 62. Page for forgotten password.
NUEVA CONTRASEÑA

Introduzca la nueva contraseña

* Nueva contraseña

* Vuelva a introducir la nueva contraseña

Cambiar contraseña

* Campos obligatorios

SOBRE NOSOTROS

ATENCIÓN AL CLIENTE

CREAR UNA CUENTA

Registrarse como PROFESIONAL, para beneficiarse de nuevos productos adicionales y de las mejores condiciones de precios. Antes de comenzar, será necesario que se active su cuenta y valide la verificación de la misma.

Datos personales

Nombre

Apellido

Dirección

País

Email

Teléfono

Información sobre la empresa

Nombre de la empresa

Puede ver información relacionada con perfiles de miembros de mi perfil pada

Comprar

Crear cuenta

* Campos obligatorios

CREAR UNA CUENTA

¡PERFECTO, YA SE HA ENVÍADO SU SOLICITUD DE ALTA!

En breve recibiremos su solicitud y activaremos su cuenta de QUALITY MATERIALS.

Si desea contactarnos con nosotros para resolver cualquier duda.

SOBRE NOSOTROS

Quiénes somos

Aviso legal

Condiciones de uso

Política de reembolso

ATENCIÓN AL CLIENTE

Contacta

SEGURIDAD

100% pago seguro

VISA

PayPal

QUALITY MATERIALS

Fig. 63. New password page.

Fig. 64. Registering page.

Fig. 65. Registering success page.

63/76
Normas de envío y devolución

Enviados
Los gastos de envío no están incluidos en los productos y son mostrados antes de realizar la compra. En caso de devolución se ha de pagar al remitente en función de una tarifa equitativa y previamente esta la devolución se realiza a cuenta del comprador, el desplazamiento se realizará a la dirección del Almacén y Reembolso.

En el momento de recibir el envío, se debe a comentar las faltas para la devolución en concreto. En estos casos, los gastos del envío son los siguientes:
- Precio y Reembolso: 4,50 euros.
- Las Hotes y Santa Fe: 6,50 euros.
- Europa los países que aparecen la hora de efectuar la compra: 12 euros.

Para el resto de detalle, se recomienda ponerse en contacto con nosotros en info@qualitatematerials.es

Plazos de entrega
A partir del momento de recepción de pedidos, el plazo de entrega es de 2 a 3 días laborables, en función del modelo solicitado, de la cantidad y de la región o zona principal. QUALITY MATERIALS SL, empresa que gestiona este servicio, no es responsable de retrasos que se puedan dar en la entrega y los retrasos que pudieran presentar se deben valorar en función de las circunstancias de cada paso.

Reclamaciones y devoluciones

Fig. 66. Shipping conditions.

Fig. 67. Menu.

La más amplia gama de materiales para los profesionales

Fig. 68. Search box.

Fig. 69. Home page.
Fig. 71. Mi panel de control.
Fig. 72. My account order history.

Fig. 73. My account edit information page.
Fig. 74. My account manage addresses page.

Fig. 75. My account edit address page.
Pedidos #100016361
Fecha del Pedido: 6 de septiembre de 2017

Dirección de envío
Monsenor Miquelito
avd. A. Conesa, 12345
España
T: 123456789

Dirección de facturación
Monsenor Miquelito
avd. A. Conesa, 12345
España
T: 123456789

Método de envío
Tarifa - Nacional

Método de pago
Transferencia bancaria

Artículos pedidos
<table>
<thead>
<tr>
<th>Nombre del Producto</th>
<th>Código</th>
<th>Precio</th>
<th>Cantidad</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tubo de resina clara transparente</td>
<td>123456789</td>
<td>1.65 €</td>
<td>1</td>
<td>1.65 €</td>
</tr>
<tr>
<td>Diámetro 3,5 x 6 mm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subtotal 1.65 €
Manipulación y Envío 0,90 €
Impuestos 0,20 €
Importe total 2,75 €

Fig. 76. My account print order page.

Fig. 77. Shopping cart page.

68/76
Fig. 78. Checkout page.

Fig. 79. E-mail sent to the customer before account activation.
Fig. 80. E-mail to the customer after account activation.

Fig. 81 E-mail to the administrator when an account is pending of being activated.
6. **Budget**

In this project, no hardware has been needed. It is a software-based project, and thus, the total economic cost of the project should be measured in terms of the hours spent by each one of the project members. Below is the breakdown of hours spent by each one, as well as the total cost of the project. The total cost of the project would be about 13,100.00€ approximately.

<table>
<thead>
<tr>
<th>Project member</th>
<th>Number of hours</th>
<th>Hourly cost* (€/hour)</th>
<th>Total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xavier Casanova (project author)</td>
<td>900</td>
<td>9</td>
<td>8,100.00 €</td>
</tr>
<tr>
<td>Xavier Castillejo (project tutor)</td>
<td>70</td>
<td>60</td>
<td>4,200.00 €</td>
</tr>
<tr>
<td>Juan Luis Gorricho (project cotutor)</td>
<td>20</td>
<td>40</td>
<td>800.00 €</td>
</tr>
</tbody>
</table>

*the hourly cost is an estimation.*
7. Conclusions and Future Development

Quality Materials will be an online channel for selling products to business, i.e., a b2b site. The main functionalities of the page are already in place, and most of the designs have been taken into account when developing the new website. The site features a landing page that explains in a simple but concise way what the page consists of. It has a mandatory login and offers the possibility of registering to new companies that would like to use the page. The administrator of the site is responsible for activating new accounts. After a user is logged in, it can fully use the page, browsing through all the products and buying them. It can also control its orders in the ‘my account’ section.

Developing, managing and administrating this website implies taking care of the development environment, the production environment, the connection to the server, contacting the hosting provider, and coordinating the following steps with the client.

The website is not fully finished, but the number of steps that need to be performed before it is fully functional is small. If needed, the site could be accessible to all the customers with activated account in few months, or even weeks. One important step that was not required by the moment was securing the website with SSL, which I would strongly recommend to do. Other minor steps would be to double check with the client if the last developments done are what they expect.

From a personal side, I have learned a great number of concepts, ranging from a simple line of HTML code to uploading a database to a server or understanding and using a complex system as Magento is. I have also had the opportunity to collaborate in a web development project in a business environment, and thus, to understand how different a project can be in a company than in an academic domain.
Bibliography


Appendices (optional)

The code of my project includes about 40 thousand files. Adding it as an appendix would not be useful. Instead, I will try to summarize which are the kind of files I have modified. I cannot list all of them because they also are a big number of files.

- Under app/code/local/CSB I have added some modules, namely:
  - Admincustomergrid
  - Customer
  - ForceLogin
  - LandingQM
- Under app/code/community and app/code/core, I modified some of the files of different modules. The most important one was the top menu, which is an extension, as explained in this document.
- Under app/design/frontend, under the different themes, I modified several tempaltes. Some of them include registerSuccess.phtml, landingqm.phtml, sliderAndSelector.phtml, dashboard.phtml, navigation.phtml, form.phtml, and several more.
- Under skin, I included and modified css files, and stored images.
- Under media, I stored images.
- Under app/etc, in local.xml I defined the most important settings of the store.