_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSIDAD POLITECNICA DE VALENCIA

# DESARROLLO DE UN PORTAL WEB CON TECNOLOGIA APACHE/MYSQL


Autor: **Fabricio Carlos Ibáñez**

Tutor: **José Enrique López Patiño**


Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación.

Curso 2013-14

Valencia, 24 de julio de 2014

**RESUMEN**

Este trabajo final de grado consta en la implementación de la tecnología PHP y MySQL en un portal web desarrollado con NetBeans IDE. La ventaja es que, tanto PHP, MySQL como NetBeans, son open source, con lo cual se evita el sobrecoste de pagar licencias adicionales a la hora de crear una web. En este proyecto he implementado un CMS (sistema de gestión de contenidos), donde el usuario puede poner en su web los objetos que él desee a través de este CMS, de una forma dinámica y rápida, sin tener que acceder al back-end de la web. Esta tecnología es muy usada en la actualidad, ya que permite a usuarios sin conocimientos de bases de datos ni de programación poder tener una web donde sean capaces de auto-gestionar su propio contenido sin necesidad de tener un programador asalariado. También he utilizado una técnica conocida como el “Template”. La ventaja de crear la web en base a un “Template” es que simplemente hay que modificar el contenido interno de la subpágina, es decir, el texto. Además, es una ventaja a la hora de volver a crear una web, ya que puedes usar exactamente el mismo código y sólo modificar el texto.

**RESUM**

Aquest treball final de grau consta en la implementació de la tecnologia PHP i MySQL en un portal web desenvolupat amb amb NetBeans IDE. L'avantatge és que, tant PHP, MySQL com NetBeans, són open source amb la qual cosa s'evita el sobrecost de pagar llicències addicionals a l'hora de crear una web. En aquest projecte he implementat un CMS (sistema de gestió de continguts), on l'usuari pot posar a la seva web els objectes que desitgi fent ús d' aquest CMS, d'una forma dinàmica i ràpida, sense haver d'accedir al back-end del web. Aquesta tecnologia és molt usada en l'actualitat, ja que permet a usuaris sense coneixements de bases de dades ni de programació poder tenir una web on siguin capaços d'auto-gestionar el seu propi contingut sense necessitat de tenir un programador assalariat. També, he utilitzat una tècnica coneguda com el "Template". L'avantatge de crear la web en base a un "Template" és que simplement cal modificar el contingut intern de la subpàgina , és a dir, el text. A més, és una avantatge a l'hora de tornar a crear un web, ja que pots utilitzar exactament el mateix codi i només modificar el text .

**ABSTRACT**

This final degree work consists in the implementation of the PHP and MySQL technology in a web portal developed with NetBeans IDE. The advantage is that PHP, MySQL and NetBeans, are open source with which to pay the extra cost of additional licenses when creating a website is avoided. In this project I have implemented a CMS (content management system), where the user can put the objects that he wants through this CMS dynamically and quickly, without having to access to the back-end of the web. This technology is widely used today because it enables users with no knowledge of databases or programming to have a website where they are able to self-manage their own content without having a salaried programmer. I have also used a technique known as the "Template”. The advantage of creating the web based on a "Template" is that you just have to modify the internal contents of the sub-page, i.e. the text. It is also an advantage when you re-create a website because you can use exactly the same code and just change the text.

# ÍNDICE DE LA MEMORIA

1. INTRODUCCIÓN ..... 5
Modelo-vista-controlador (MVC): ..... 5
Sistema de gestión de contenidos (CMS): ..... 9
NetBeans IDE: ..... 10
Wamp Server: ..... 11
PHP: ..... 12
MySQL: ..... 12
Phpmyadmin: ..... 13
Template Method (patrón de diseño): ..... 14
Conclusión Introducción: ..... 17
2. OBJETIVOS ..... 18
3. METODOLOGÍA ..... 19
4. DESARROLLO Y RESULTADOS ..... 21
Primera parte: Usuario ..... 22
Segunda parte: Administrador ..... 28
5. PLIEGO DE CONDICIONES ..... 35
6. CONCLUSIONES Y PROPUESTA DE TRABAJO FUTURO ..... 36
7. BIBLIOGRAFÍA ..... 37

# ÍNDICE DE FIGURAS

Figura 1. Ejemplo genérico de Modelo Vista Controlador……..……5
Figura 2. Ejemplo del MVC que he utilizado en mi TFG……..……6
Figura 3. Ejemplo de uso de Modelo. /Controller/GameController.php……..……6
Figura 4. Ejemplo de uso de Controlador. /Model/GameModel.php……..……7
Figura 5. Ejemplo de uso de Vista. /Games.php……..…7
Figura 6. Ejemplo de uso de Vista llamando al Controlador. Games.php……..…8
Figura 7. Ejemplo del CMS usado en mi TFG……...9
Figura 8. Ejemplo de creación de proyecto nuevo en NetBeans IDE……..…10
Figura 9. Panel de Control de Wamp Server……11
Figura 10. Estado de los componentes instalados por Wamp Server……..11
Figura 11. Tablas de la base de datos usada en mi TFG “gamedb”……12
Figura 12. Panel de control de phpMyAdmin……14
Figura 13. Template usado en mi TFG. /Template.php……..15
Figura 14. Código de la página home de mi TFG. /index.php……...16
Figura 15. Vista de la portada de mi TFG por un navegador web. /index.php……17
Figura 16. Apartado Home de la web. /index.php……22
Figura 17. Apartado con el catálogo de juegos. /Games.php……23
Figura 18. Apartado con el carrito de la compra. /Cart.php……...24
Figura 19. Mensaje de confirmación al pulsar “Clear Cart”……24
Figura 20. Apartado con la información para preparar el pedido. /billing.php……..…25
Figura 21. Mensaje de confirmación del correo enviado satisfactoriamente……..26
Figura 22. Correo con los datos de facturación……...26
Figura 23. Información de contacto……27
Figura 24. Identificación previa para el acceso al CM……28
Figura 25. Visión del CMS tras identificación……...29
Figura 26. Ejemplo de subir una foto al servidor. /uploadImage.php……...30
Figura 27. Ejemplo de añadir un nuevo juego a la tienda……..31
Figura 28. Tabla con todos los juegos que tenemos en la tienda……32
Figura 29. Tabla con todos los pedidos pendientes de envío……33
Figura 30. Detalles de un pedido específico……34
Figura 31. Plano: carpetas y directorios……35

## 1. INTRODUCCIÓN

En este apartado explicaré y definiré aquellos aspectos claves que me gustaría dejar concretados antes de entrar en el apartado de resultados, con el fin de entender mejor cada concepto.

### Modelo-vista-controlador (MVC):

Para la realización de este trabajo he respetado en medida de lo posible la estructura del MVC. El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

Para ello, MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador. Así, por un lado, define componentes para la representación de la información y por otro, para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento. Podemos observar en la “Figura 1” un ejemplo genérico de un MVC y en la “Figura 2”, un ejemplo de cómo he subdividido en carpetas mi modelo, controlador y vistas.

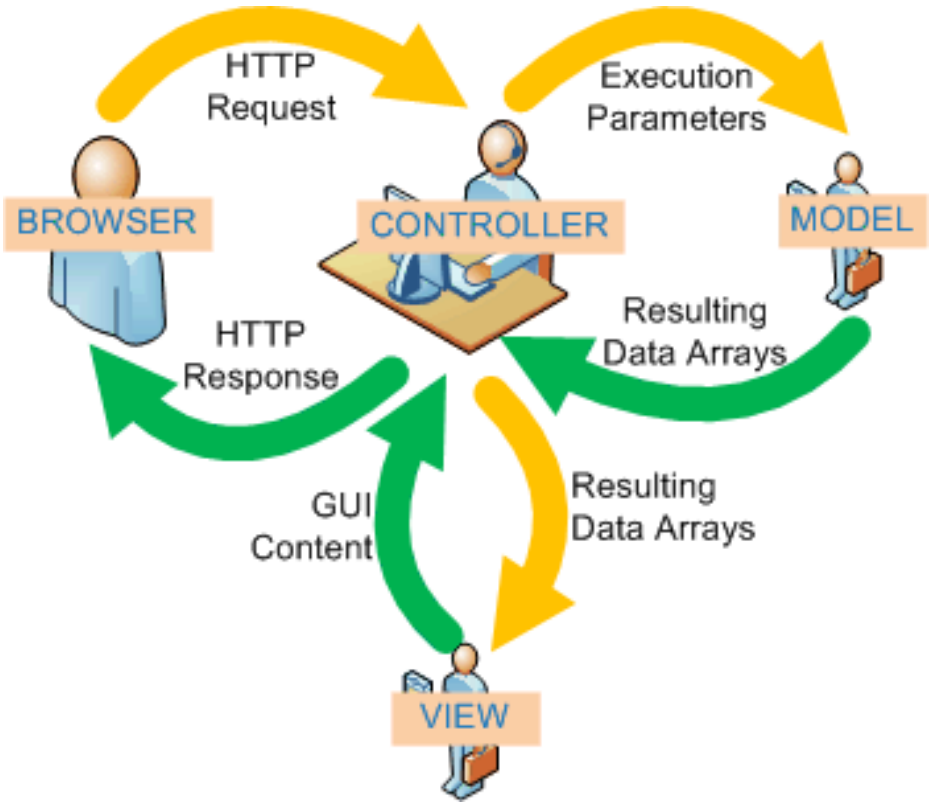



**Figura 1. Ejemplo genérico de Modelo Vista Controlador.**

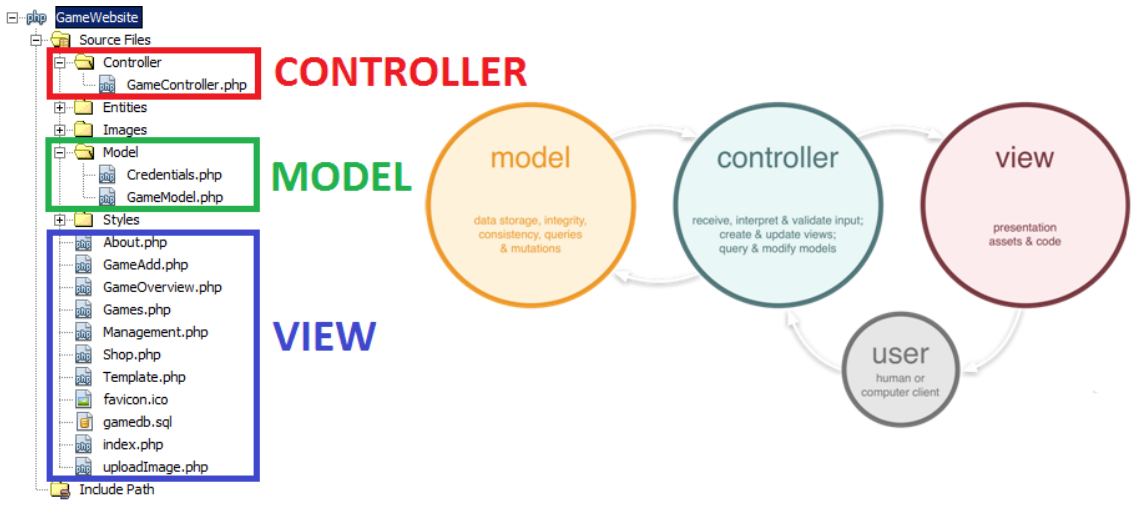



**Figura 2. Ejemplo del MVC que he utilizado en mi TFG.**

De una forma más estándar, podemos definir los componentes de nuestro MVC de la siguiente manera:

El **Modelo**:

Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones. Envía a la **Vista** aquella parte de la información que en cada momento se le solicita para que sea mostrada. Las peticiones de acceso o manipulación de información llegan al **Modelo** a través del **Controlador**. En esta parte, me ocupo de hacer las llamadas a mi base de datos como vemos a continuación:

```php
//Get all game types from the database and return them in an array.
function GetGameTypes() {

    require ('Credentials.php');
    //Open connection and Select database.
    mysql_connect($host, $user, $passwd) or die(mysql_error());
    mysql_select_db($database);
    $result = mysql_query("SELECT DISTINCT type FROM game ORDER BY type") or die(mysql_error());
    $types = array();

    //Get data from database.
    while ($row = mysql_fetch_array($result)) {
        array_push($types, $row[0]);
    }

    //Close connection and return result.
    mysql_close();
    return $types;

}
```

**Figura 3. Ejemplo de uso de Modelo. /Controller/GameController.php**

Observamos en esta imagen como hago una llamada a mi base de datos para obtener todos los diferentes tipos de juegos que hay en mi base de datos “game” donde almaceno los juegos.

El **Controlador**:

Responde a eventos e invoca peticiones al **Modelo** cuando se hace alguna solicitud sobre la información. También, puede enviar comandos a su **Vista** asociada si se solicita un cambio en la forma en que se presenta de **Modelo**, por tanto, se podría decir que el **Controlador** hace de intermediario entre la **Vista** y el **Modelo**. En la "Figura 4" observamos el uso del **Controlador.**

```
function CreateGameDropdownList() {
    $gameModel = new GameModel();
    $result = "<form action = '' method = 'post' width = '200px'>
                Please select a type:
                <select name = 'types' >
                    <option value = '%' >All</option>
                    " . $this->CreateOptionValues($this->GetGameTypes()) .
            "</select>
                 <input type = 'submit' value = 'Search' />
                </form>";

    return $result;
}
```

**Figura 4. Ejemplo de uso de Controlador. /Model/GameModel.php**

Como podemos apreciar, se genera el Dropdown List de HTML gracias a nuestro **Modelo** que llama al **Controlador** para acceder a la base de datos, de forma que éste le devuelve los resultados.

La **Vista**:

Presenta el **Modelo** en un formato que es visiblemente adecuado para interactuar por el usuario. De esta manera, el usuario puede acceder a los contenidos de una manera fácil y sin conocer lo que está ocurriendo en el back-end de la página, ya que sólo ve el HTML que se ha generado y aparece en su navegador. En la "Figura 5" podemos ver un ejemplo de vista que llama a todos los componentes descritos anteriormente:

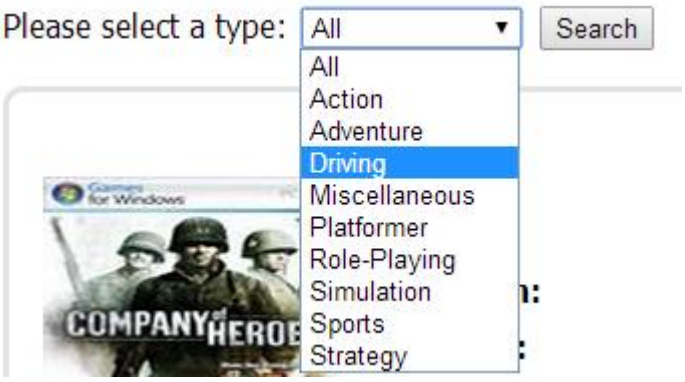



**Figura 5. Ejemplo de uso de Vista. /Games.php**

Como observamos en la “Figura 6”, cuando el usuario accede al apartado “/Games.php” le aparece este Dropdown List, que se ha sido generando porque la **Vista** le ha dicho al **Modelo** que lo generase:

```
if(isset($_POST['types']))
{
    //Fill page with games of the selected type
    $gameTables = $gameController->CreateGameTables($_POST['types']);
}
else
{
    //Page is loaded for the first time, no type selected -> Fetch all types
    $gameTables = $gameController->CreateGameTables('%');
}

//Output page data
$title = 'Game overview';
$content = '<form name="form1">
        <input type="hidden" name="productid" />
        <input type="hidden" name="command" />
        </form>'.
        $gameController->CreateGameDropdownList(). $gameTables;
```

**Figura 6. Ejemplo de uso de Vista llamando al Controlador. Games.php**

En una explicación más detallada de cómo interactúan los componentes entre sí encontramos que:

El usuario interactúa con la web de alguna forma, por ejemplo, accediendo al apartado /Games.php que sería la **Vista**.

El **Controlador** recibe la notificación de la acción solicitada por el usuario enviada por la **Vista.** Ésta le pide que genere un DropDown List con el CreateGameDropdownList() y que también genere las tablas con los juegos con CreateGameTables().

El **Controlador** accede al **Modelo**, actualizándolo, posiblemente modificándolo de una forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). En este caso, es un simple “SELECT” donde accedemos a nuestra base de datos para obtener los juegos y los tipos de los juegos que tenemos en la tienda.

El **Controlador** delega a los objetos de la **Vista** la tarea de desplegar la interfaz de usuario. La **Vista** obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, en este caso, desplegar el DropDown List y generar las tablas con los juegos). El **Modelo** no debe tener conocimiento directo sobre la **Vista**.

## Sistema de gestión de contenidos (CMS):

El gestor de contenido es la parte de mi proyecto donde permite a los administradores crear, editar, gestionar y publicar contenido. El gestor de contenidos interactúa con la base de datos para generar nuevos objetos, modificar los existentes y/o eliminarlos. De esta forma, el administrador puede poner nuevos juegos en venta, cancelar pedidos realizados, mirar los juegos que tiene en su tienda, subir caratulas para sus juegos. En la “Figura 7” podemos ver mejor nuestro CMS:

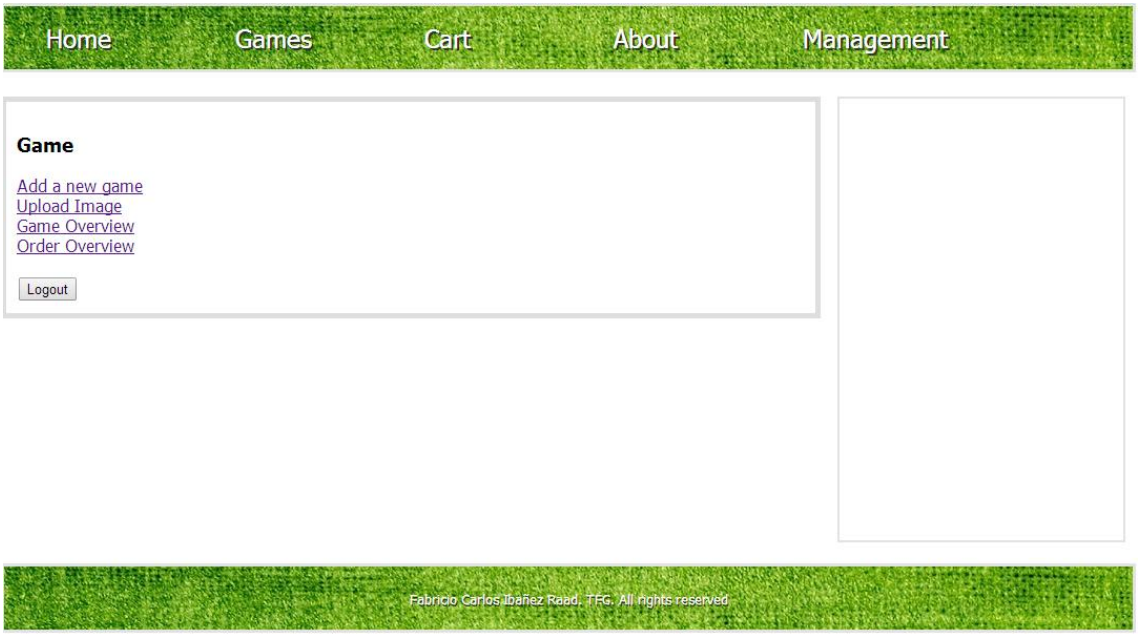



**Figura 7. Ejemplo del CMS usado en mi TFG.**

Aquí observamos como con el CMS que he implantado podemos añadir un juego nuevo, subir una imagen a nuestra web, ver todos los juegos que disponemos y ver los pedidos pendientes.

- Funcionamiento:

Cuando el administrador accede a /Management.php y se identifica adecuadamente podrá tener acceso al CMS. A través de éste podrá acceder a la base de datos a través del mismo navegador de una forma fácil y segura.

- Ventajas

El gestor de contenidos facilita el acceso a la publicación de contenidos a un rango mayor de usuarios. Permite que, sin conocimientos de programación ni diseño, cualquier usuario pueda añadir contenido en el portal web.

Los costes de gestión de la información son mucho menores, ya que no se necesita un programador ni diseñador para añadir el contenido. El diseño ya ha sido previamente definido y creado en el momento de creación de la web e implementación del CMS para que no tenga que ser modificado posteriormente.

La actualización, backup y reestructuración del portal son mucho más sencillas al tenerlo todos en una base de datos estructurada en el servidor.

❖ Desventajas

Al ser un sistema ya creado previamente no es tan escalable y moldeable, con lo cual se pierde flexibilidad.

## NetBeans IDE:

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. También posee ayuda y facilidades para la programación en PHP y es por ello que me decanté en utilizar este IDE.

Además, NetBeans es un proyecto open-source, con lo cual no es necesario pagar licencias para hacer proyectos en él. NetBeans está dividido en módulos y por ello, sólo he instalado aquellos que eran necesarios para mi proyecto, es decir, los módulos de PHP y HTML5/CSS3. Esto me permite trabajar con un IDE muy ligero donde sólo tengo aquello que necesito.

Como podemos observar en la "Figura 8", cuando creo un nuevo proyecto sólo me ofrece la posibilidad de crearlo con los módulos que tengo ya instalados.

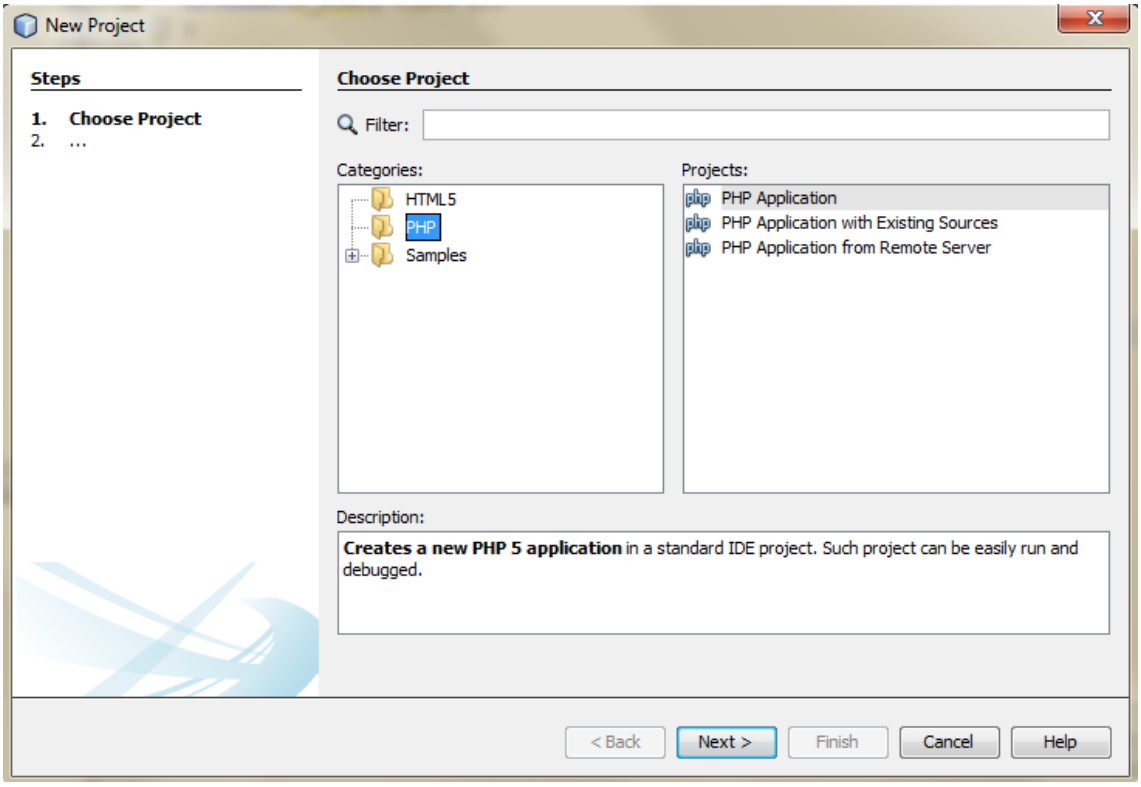



**Figura 8. Ejemplo de creación de proyecto nuevo en NetBeans IDE.**

## Wamp Server:

Para poder crear la web, probarla, testearla, hacía falta tener instalado en tu localhost un entorno que te permitiese hacer todo lo explicado anteriormente. Wamp server es el encargado de permitirlo.

Así, Wamp server es un programa que instala todo lo necesario para poder usar PHP sobre un servidor Apache y con bases de datos MySQL. Lo mejor de este producto es que gracias a él y con una simple instalación tienes instalado tu servidor Apache, PHP5 y MySQL en tu sistema operativo. Además Wamp Server ofrece un panel de configuración donde se puede añadir los addons que se necesiten. En la "Figura 9" podemos ver el panel de control, que todo funciona correctamente y añadir addons, en caso necesario.


**Figura 9. Panel de Control de Wamp Server**

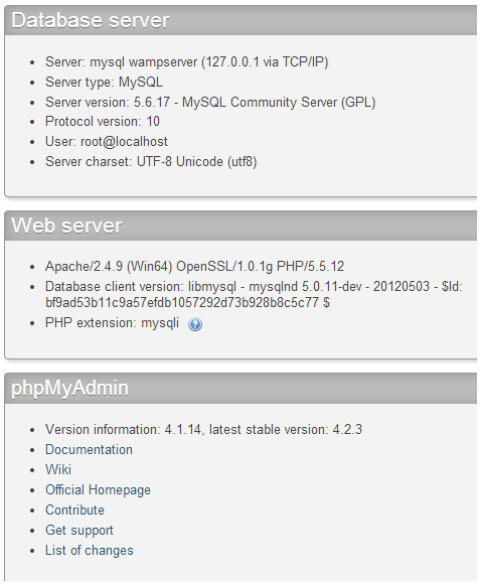



**Figura 10. Estado de los componentes instalados por Wamp Server.**

En la "Figura 10" podemos observar el estado de los componentes que tenemos instalados en nuestro sistema.

### PHP:

PHP es un lenguaje de programación de uso general de código del lado del servidor, originalmente diseñado para el desarrollo web de contenido dinámico. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. Además, PHP es un lenguaje de programación de código abierto.

- Características:

Está orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. Es considerado un lenguaje fácil de aprender y gracias a ello, he podido hacer un proyecto de principio a final sin la ayuda física de terceros, sencillamente apoyándome en tutoriales y libros.

El código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.

Tiene capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destacando su conectividad con MySQL.

Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

PHP permite trabajar de una forma muy sencilla, ya que se acopla muy bien a la arquitectura MVC, la cual ha sido utilizada en mi TFG.

### MySQL:

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario. Es software libre, con lo cual no es necesario gastar dinero para hacer uso de estas bases de datos.

MySQL es utilizado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr y YouTube.

- Características:

MySQL utiliza la mayoría del lenguaje SQL. Tiene una gran disponibilidad en cuanto a plataformas y sistemas. Permite la replicación de sus bases de datos. La conectividad con la base de datos es segura. Tiene transacciones y claves foráneas. Se pueden indexar campos de texto.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Ésta puede ser desde mi sencilla lista de juegos o el complejo volumen de información de una red corporativa.

En la “Figura 11” podemos observar todas las tablas que he utilizado en mi TFG pertenecientes a la base de datos “gamedb” donde, más adelante, se pueden apreciar las relaciones existentes entre dichas tablas:

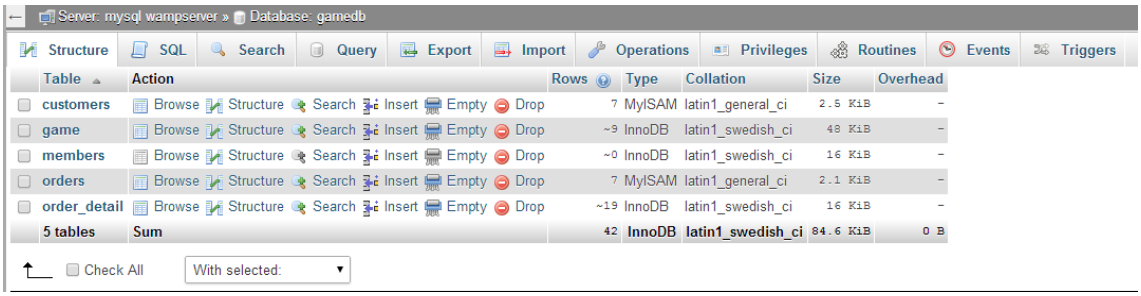



**Figura 11. Tablas de la base de datos usada en mi TFG "gamedb".**

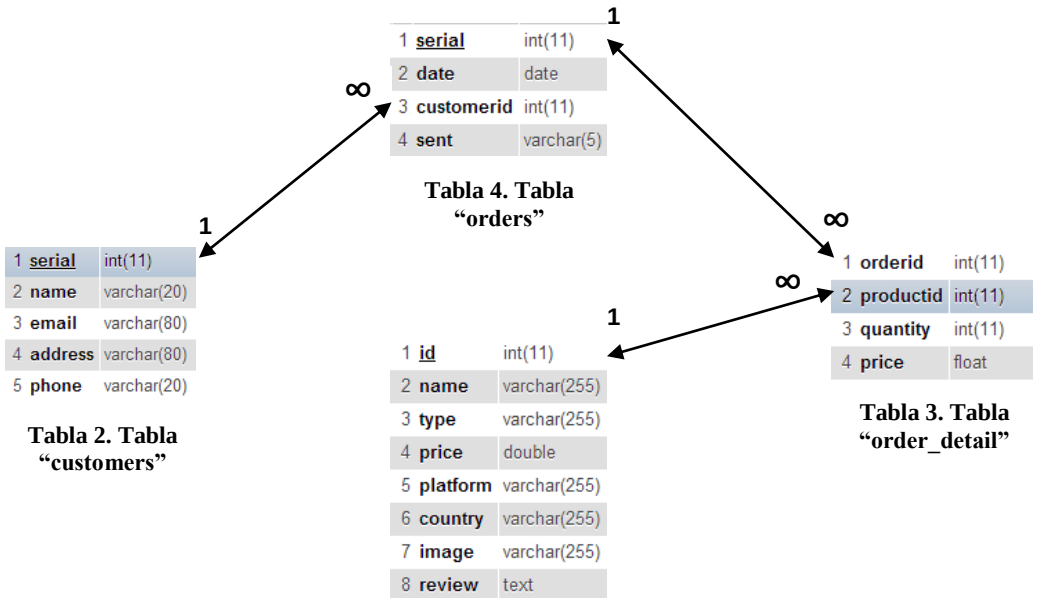



**Tabla 4. Tabla "orders"**

**Tabla 2. Tabla "customers"**

**Tabla 3. Tabla "order_detail"**

**Tabla 1. Tabla "game"**

| | | |
|---|---|---|
| 1 | id | int(11) |
| 2 | username | varchar(50) |
| 3 | password | varchar(255) |
| 4 | firstname | varchar(30) |
| 5 | lastname | varchar(30) |
| 6 | activated | enum('0', '1') |
| 7 | randSalt | int(255) |

**Tabla 5. Tabla "members"**

## Phpmyadmin:

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando Internet. Actualmente, puede crear y eliminar bases de datos; así como crear, eliminar y alterar tablas; borrar, editar y añadir campos; ejecutar cualquier sentencia SQL; administrar claves en campos; administrar privilegios; exportar datos en varios formatos. También, es software libre.

Como podemos apreciar, en todo momento se han utilizado herramientas gratuitas. El panel de control para acceder a él es: “http://localhost/phpmyadmin/”. Como observamos en la “Figura 12”, Phpmyadmin nos permite todas las funcionalidades que cualquier sistema de gestión de bases de datos nos permitiría, como mirar nuestras bases de datos, el estado de las mismas, insertar comandos SQL y demás.

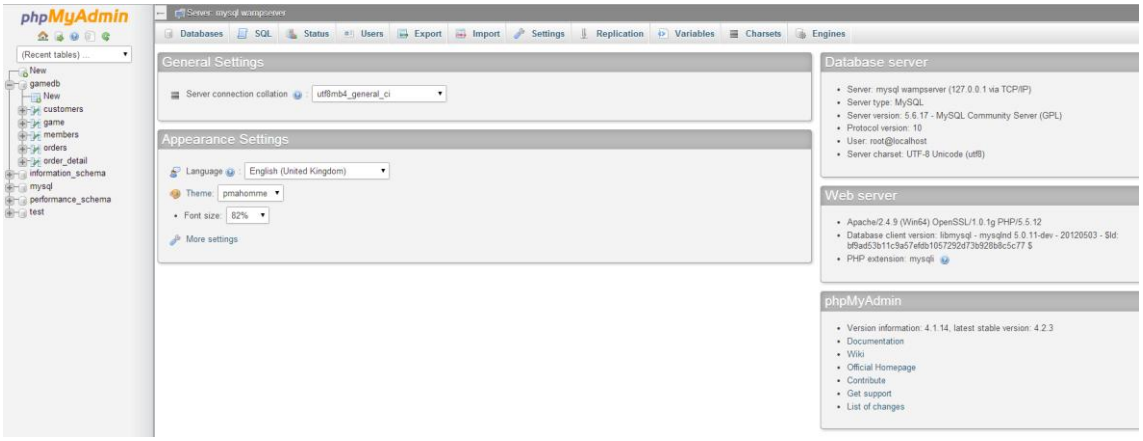



**Figura 12. Panel de control de phpMyAdmin**

## Template Method (patrón de diseño):

El “Template Method” se utiliza para, desde una plantilla creada, poder generar las páginas que pertenezcan a la misma web de una forma mucho más rápido y que tengan similar aspecto. Un ejemplo sería mantener las barras de navegación y barras laterales en cada página de la web, siendo el contenido de cada una diferente.

Los beneficios de este sistema son, en primer lugar, para el programador, ya que no tiene que generar cada vez una página nueva y, en segundo lugar, para el usuario, ya que mientras se mueve por la web sólo tiene que descargar el contenido principal que cambia entre una página y otra de la web, además de que los tiempos de carga son mucho menores.

En mi patrón de diseño he creado todo lo que lleva una página como pueden ser las definiciones dentro de la cabecera como el tipo de contenido, el favicon, título y el link a los estilos. También, los marcos de la página web con su barra de navegación y sus respectivos estilos.

En la “Figura 13” podemos observar mi “Template”, donde se muestra todo lo indicado anteriormente. Como vemos, sólo hay dos variables que no están definidas, las cuales serán definidas en las diferentes páginas que usen nuestro “Template”. Aquí es donde guardaremos nuestro contenido y lo que hará que las páginas sean diferentes entre sí.

```php
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="shortcut icon" href="favicon.ico" >
        <title><?php echo $title; ?></title>
        <link rel="stylesheet" type="text/css" href="Styles/Stylesheet.css" />
    </head>
    <body>
        <div id="wrapper">

            <div id="banner">
            </div>

            <nav id="navigation">
                <ul id="nav">
                    <li><a href="index.php">Home</a></li>
                    <li><a href="Games.php">Games</a></li>
                    <li><a href="Cart.php">Cart</a></li>
                    <li><a href="About.php">About</a></li>
                    <li><a href="Management.php">Management</a></li>
                </ul>
            </nav>

            <div id="content_area">
                <?php echo $content; ?>
            </div>

            <div id="sidebar">

            </div>

            <footer>
                <p>Fabricio Carlos Ibañez Raad. TFG. All rights reserved</p>
            </footer>
        </div>
    </body>
</html>
```

**Figura 13. Template usado en mi TFG. /Template.php**

Para poder hacer uso de este patrón de diseño, tenemos que incluir en cada página que queremos que muestre nuestro “Template” la instrucción de PHP “include ‘Template.php’”.

Como podemos observar en la “Figura 13”, que corresponde al apartado /index.php de mi proyecto, incluyo la instrucción antes mencionada para que aparezca la página de la forma que muestra la “Figura 14”. Esto es posible ya que las dos únicas variables que modifico en todo momentos son las variables $title y $content.

La variable $title será la encargada de dar el título apropiado a cada página de la web y en la variable $contenido, guardaré todo el contenido que quiero que muestre como si fuera contenido HTML. De esta forma, sólo modificando la variable $content en cada página soy capaz de generar una nueva página con contenido diferente.

```php
<?php
$title = "Home";
$content = '
        <img src="Images/game1.jpg" class="imgLeft" />
        <h3>The games taught us the game</h3>
        <p>
            We have learnt to draw experience from all the variety of sources, from
            business and beyond. As video-game enthusiasts we know how to cooperate,
            enjoy and succeed in a competitive environment. And, most importantly, we
            know how to share success with our Customers and Partners..We have learnt to draw experien
            business and beyond. As video-game enthusiasts we know how to cooperate,
            enjoy and succeed in a competitive environment. And, most importantly, we
            know how to share success with our Customers and Partners..
        </p>
        <img src="Images/game2.jpg" class="imgRight" />
        <h3>Mission</h3>
        <p>
            Etsitgamers.com is mainly an innovative platform that offers comfortable and
            swift access to wide range of products such as software activation
            licenses for Steam, Origin, Xbox Live cards, PSN codes and time-cards
            for online games in unique prices. The company is occupied with digital
            distribution . Etsitgamers.com is also a game supplier. Thanks to the implementation
            of the latest technologies, we have the highest level of security and
            availability on the European market of digital sales platforms to our
            credit. Our mission is to provide the best digital game licenses andretail service to all
            Etsitgamers.com is mainly an innovative platform that offers comfortable and
            swift access to wide range of products such as software activation licenses for Steam, Ori
            for online games in unique prices. The company is occupied with digital
            distribution . Etsitgamers.com is also a game supplier. Thanks to the implementation
            of the latest technologies, we have the highest level of security and
            availability on the European market of digital sales platforms to our
            credit. Our mission is to provide the best digital game licenses and
            retail service to all our Partners and Customers.
        </p>
        <img src="Images/game3.jpg" class="imgLeft" />
        <h3>History</h3>
        <p>
            EtsitGamers is at the forefront of providing quality service at an affordable rate
            for gamers from all walks of life and across the whole spectrum of game platforms. We
            adopt leading technologies of world class standards, to push the boundaries of monetizing
            while pioneering new concepts and products. Having been in the industry for over 10 years,
            we continue to push the envelope in the industry by developing and fostering strong
            partnerships with reputable game publishers and developers. OffGamers also facilitates
            the gaming industry by innovating effective game card top up systems and enhancing our
            user's experience.
        </p>';
include 'Template.php';
?>
```

**Figura 14. Código de la página home de mi TFG. /index.php**


**Figura 15. Vista de la portada de mi TFG por un navegador web. /index.php**

Tal y como podemos observar, la "Figura 15" muestra la portada de mi trabajo utilizando un navegador web. /index.php.

A continuación, tras conocer todos los apartados de este TFG, podremos entender más fácilmente de donde surgen los resultados y cómo se han generado, visualizándolos en el apartado 4: Desarrollo y resultados de mi trabajo.

## 2. OBJETIVOS

Los objetivos generales de este trabajo se centran en realizar un portal de contenido web dinámico, en el que se utilice en el back-end programación PHP y MySQL. De esta forma, se pretende implementar estas tecnologías en las webs que se utilizan en el día a día.

Como objetivos específicos, me he centrado en:

- Añadir un front-end usando HTML5 y CSS3.
- Hacer uso de la tecnología “Template”.
- Implementar un “CMS”.
- Hacer uso de la arquitectura “MVC”.

## 3. METODOLOGÍA

### Fase 1: Encontrar un Tema. Julio-Agosto 2013. 2 meses.

Realizar este trabajo fue una propuesta personal que le hice al profesor José Enrique López Patiño. El verano de 2013 estuve trabajando en Estambul en una empresa que vendía software para bancos. Este software siempre basaba sus proyectos en el patrón de arquitectura MVC y por eso, me sentí muy atraído por esta arquitectura. Al estar trabajando en Java y con bases de datos Oracle me interesó aprender una tecnología que no conocía para enriquecer mis conocimientos en programación. Por estos motivos, decidí realizar una web con tecnología PHP y MySQL porque son las tecnologías más usadas actualmente en la red y son open source.

### Fase 2: Buscar al tutor. Septiembre 2013. 2-3 días.

Quise como tutor a José Enrique López Patiño porque es un profesor con buenas habilidades en programación que, además, sabe transmitir muy bien. Asimismo, hemos visto con él bases de datos MySQL, por lo que al volver de mis prácticas en septiembre le propuse este trabajo, el cual le pareció interesante y me adjudicó como trabajo fin de grado.

### Fase 3: Buscar un tema para la web. Octubre 2013-Diciembre 2013. 3 meses.

Una vez tuve el proyecto adjudicado, necesitaba encontrar un tema para la web. Durante 3 meses, estuve buscando información y contactando con varias personas que habían hecho proyectos en PHP. Concluí que para lo que más se usa actualmente PHP y MySQL es el e-commerce. Al ser un amante de los vídeojuegos decidí hacer una tienda online para la venta de éstos.

### Fase 4: Elección de IDE. Diciembre 2013. 1 Mes.

Para realizar el proyecto estuve mirando varios IDEs con los que trabajar en PHP e incluso en la utilización de algún Framework. Tras debatirlo con varios compañeros, me dijeron que lo mejor era utilizar NetBeans con el Zend o Symphony como Framework, pero que se necesitaría tiempo para saber usarlo. Por ello, decidí usar NetBeans sin ningún Framework para, en un futuro, poder utilizar alguno de estos dos con más disponibilidad de tiempo.

### Fase 5: Aprender PHP. Enero 2014- Mayo 2014. 5 meses.

Al no tener apenas conocimientos de PHP, tuve que empezar por aprender a saber cómo se utiliza. Es por ello que, en esta fase, invertí mucho tiempo para saber cómo se implementa la tecnología PHP. Así, empecé aprendiendo los conocimientos básicos de PHP en vídeos tutoriales de PHP Academy [12] y cuando ya tuve una cierta preparación para utilizar PHP y MySQL, fui a W3Schools para profundizar más sobre comandos en PHP [13] y en MySQL [14]. Finalmente, para poner en práctica todo lo aprendido, realicé todos los ejercicios que propone CodeAcamedy [15]. Una vez finalizados, tuve una idea más elaborada de la sintaxis de PHP, lo cual me permitió aprender de tutoriales más avanzados, los cuales me ayudaron a resolver las dudas que me surgieron en la elaboración de mi proyecto web.

**Fase 6: Desarrollo de la web. Mayo 2014- Junio 2014. 2 meses.**

Esta fase junto la fase 5 fueron las más extensas y laboriosas, en cuanto a tiempo y dedicación para el desarrollo de la web. Es en esta fase donde encontramos el grosso del proyecto, es decir, lo que realmente lo define.

Tras la fase previa de preparación y entendimiento para crearla, empecé a hacer cuadros y estilos para obtener de una forma más visual un punto por dónde empezar a trabajar. Esta parte es el front-end de mi proyecto donde empecé a usar HTML5 y CSS3 y, de esta forma, crear un "Template". Tras realizar la parte visual, me dediqué a programar en PHP y MySQL para hacer el back-end de la web.

**Fase 7: Testear, pulir detalles y hacer la memoria. Junio-Julio 2014. 1 mes y medio.**

Cuando la web ya funcionaba y no había ninguno tipo de error aparente, realicé un testeo intensivo para probar todos los módulos de la web y comprobar qué no hubieran errores. Una vez comprobado qué el correo, bases de datos y funciones iban correctamente sin provocar errores, depuré el código.

## 4. DESARROLLO Y RESULTADOS

En este apartado expondré cada una de las partes de mi web y su funcionalidad. Principalmente, la web está dividida en dos partes: la parte que verá el usuario que desee comprar un juego y la parte que verá el administrador, el cual gestionará mediante el CMS todos los objetos de su web.

Asimismo, he realizado unos vídeos sencillos, más visuales, de una duración de 5 minutos cada uno, donde explico el funcionamiento de la web de una forma más rápida y fácil de entender. Recomiendo, así, ver tanto los vídeos como la memoria de trabajo. Los enlaces a estos vídeos se encuentran en la bibliografía de esta memoria [16] [17].

## Primera parte: Usuario

En esta parte simularé un ejemplo de uso de un usuario que desea comprar un juego, desde que se conecta a la web hasta que acaba recibiendo el mensaje para pagar su producto.

- Página inicial: /index.php.

Lo primero que verá el usuario al conectarse a la web será el "/index.php". La "figura 16" así lo muestra. Se puede observar la barra de navegación y el contenido inicial de la página web, con una sencilla explicación de la misma. Suponemos que el usuario desea ir al apartado de "Games" para realizar la compra de algún juego.


**Figura 16. Apartado Home de la web. /index.php**

❖ Página con el catálogo de juegos: /Games.php

Esta página se iniciará mostrando todo el catálogo, teniendo seleccionado el atributo “All” del DropDownList que vemos en la “Figura 17”. Como podemos comprobar aquí el usuario podrá ver la colección de juegos organizados por el tipo que él seleccione. Simulamos el ejemplo que el usuario simplemente desee comprar el juego “Fallout 3” y presione el botón “Add to Cart”.


**Figura 17. Apartado con el catálogo de juegos. /Games.php**

❖ Página con el carrito de la compra: /Cart.php

En la “Figura 18” podemos observar nuestro carrito de la compra. Si apretamos en “Continue shopping” nos devolverá al catálogo de juegos y podremos seguir añadiendo juegos a nuestro carrito de la compra. Si apretamos “Clear Cart” nos vaciará TODO el carrito y si apretamos “Remove” borrará el juego al que corresponda. A estos dos comandos les he añadido un script en JavaScript para que cuando los pulsemos nos salte un mensaje de confirmación si realmente deseamos vaciar el carrito o borrar el juego de nuestro carrito como vemos en la “Figura 19”. Si modificamos el número de “Quantity” y pulsamos en “Update Cart”, aumentaremos el número de unidades al número que hayamos escrito y, por lo tanto, también variará el precio. Finalmente, si queremos comprar y pagar pulsamos “Place Order” y es lo que hacemos.

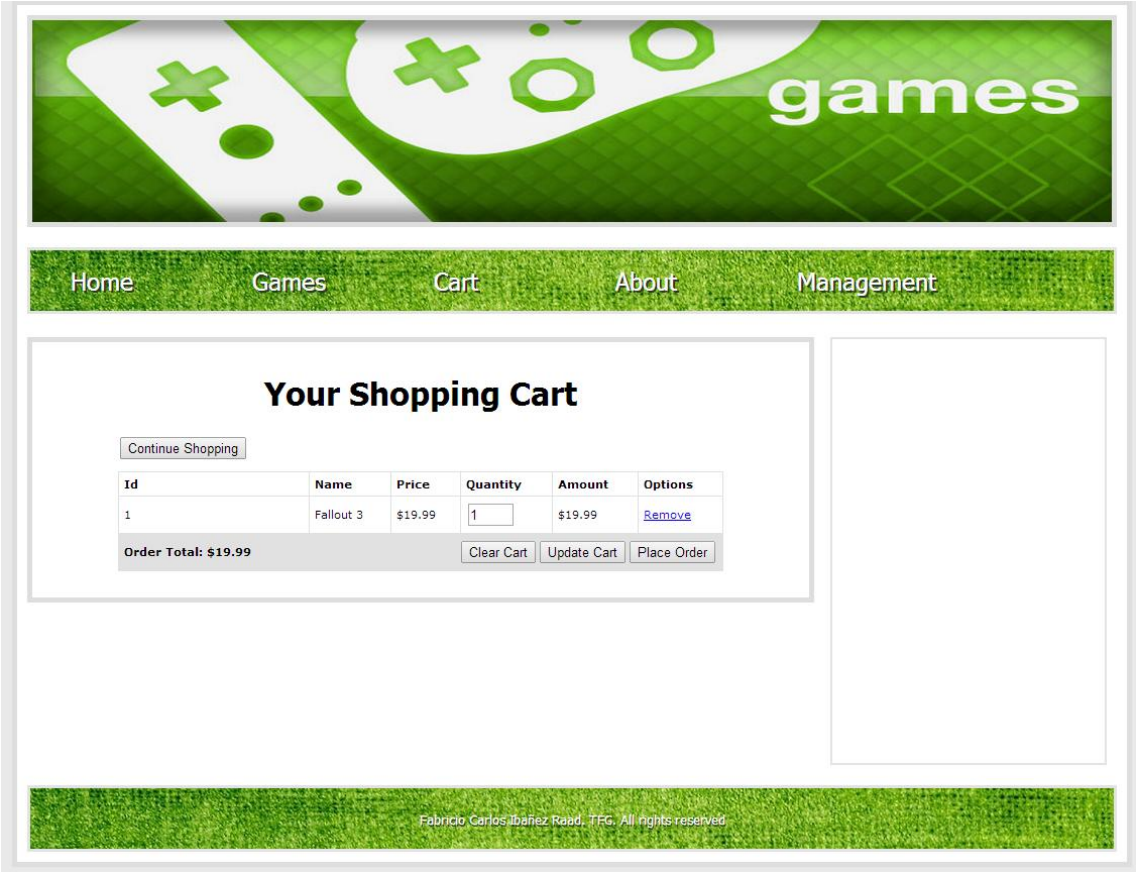



**Figura 18. Apartado con el carrito de la compra. /Cart.php**

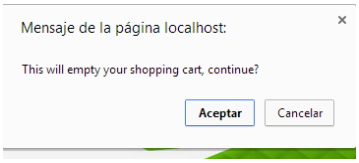



**Figura 19. Mensaje de confirmación al pulsar “Clear Cart”**

❖ Página con el apartado para preparar el pedido: /billing.php

Vemos en la “Figura 20” que nos solicita nuestro nombre, dirección, email y teléfono. El usuario rellanará los campos para que el pedido pueda llegar correctamente a la dirección que nos facilite. También, el email, ya que es necesario para enviarle en correo de confirmación de compra con los datos de facturación. Cuando hayamos rellenado todo correctamente pulsaremos el botón “Place Order”. En este ejemplo utilizaré una cuenta temporal que crearé en 10minutemail.com para verificar que el correo haya llegado correctamente. De esta forma, rellenaremos los campos con:

Your Name: Fabricio; Address: C/Tarongers, ETSIT; Email: a5642477@drdrb.net; Phone: 6666666.

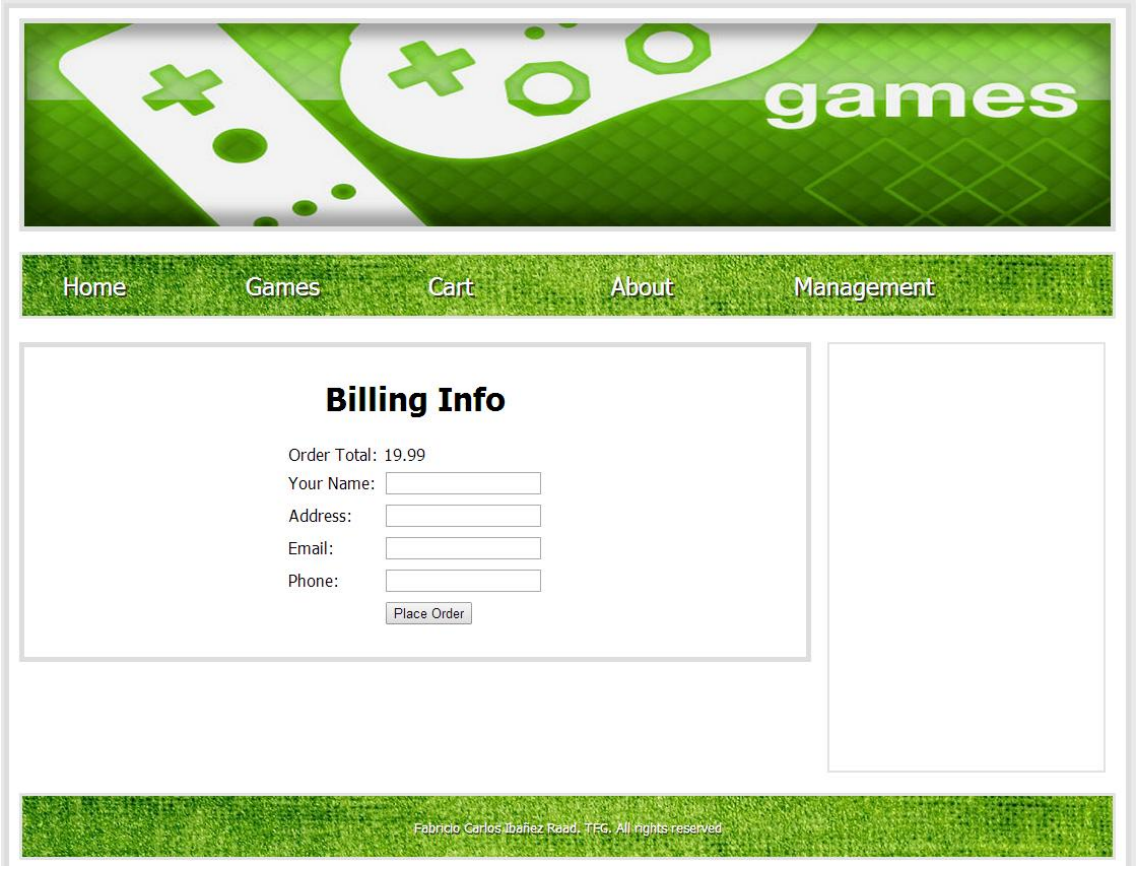



**Figura 20. Apartado con la información para preparar el pedido. /billing.php**

❖ Página con el mensaje de confirmación del correo

Si todo se ha realizado correctamente nos mostrará un mensaje de confirmación como el que podemos ver en la “Figura 21”.

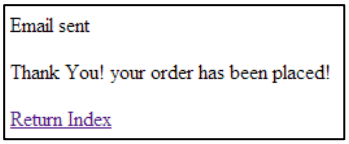



**Figura 21. Mensaje de confirmación del correo enviado satisfactoriamente.**

Si entramos al correo que acabamos de recibir podemos comprobar como el mensaje de confirmación se ha generado como podemos comprobar en la “Figura 21”. En este momento, envío un correo para que el usuario que acaba de hacer el pedido envíe el dinero a “FAKEACCOUNT”, con el asunto del ingreso bancario “8”, que corresponde al id del usuario, que necesitaremos más adelante para hacer el envío del pedido, cuando comprobemos que nos ha pagado.

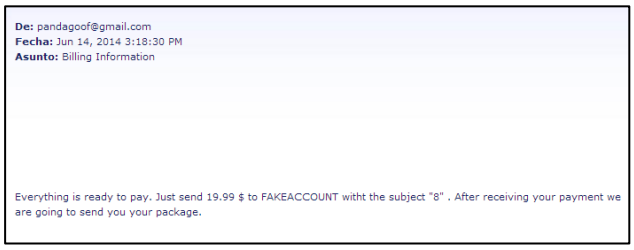



**Figura 22. Correo con los datos de facturación.**

Tras estos pasos, el usuario habrá realizado el pedido y solamente le quedará ingresar el dinero, en este caso 19.99$, para recibirlo.

También, he añadido la pestaña “About” donde podemos consultar información sobre la empresa, tal y como podemos observar en la “Figura 23”.

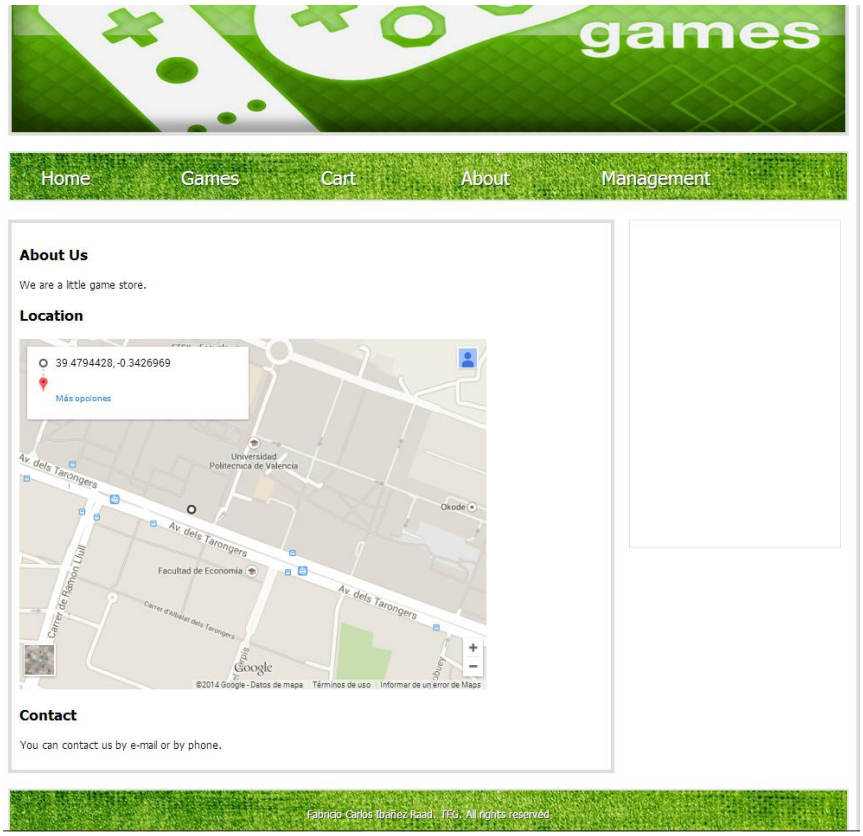



**Figura 23. Información de contacto.**

Una vez finalizados estos pasos, podemos dar por terminada la información del usuario. Asimismo, he realizado un vídeo más explicativo para representar este proceso de una manera más sencilla y visualmente más atractivo [16].

## Segunda parte: Administrador

En esta parte, simularé un ejemplo de uso de un administrador que desea añadir un nuevo juego a su tienda o que desea ver el listado de sus juegos y que, finalmente, quiere realizar el envío de un pedido que acaba de ser pagado.

- Página de acceso al CMS: /Management.php

Para poder acceder al CMS debemos ser identificados propiamente como vemos en la “Figura 24”. Una vez nos identifiquemos propiamente y pulsemos el botón “Login”, accederemos al CMS.


**Figura 24. Identificación previa para el acceso al CMS**

❖ Página con el CMS tras identificación: /Management.php

De esta forma, podemos acceder al CMS como comprobarmos en la “Figura 25”. En nuestro CMS podemos añadir un nuevo juego pulsando en “Add a new game”, subir una imagen que necesitaremos para ponerle carátula a nuestro juego en “Upload Image”, ver todos los juegos que tenemos en nuestra tienda y borrarlos o actualizarlos en “Game Overview” o, finalmente, ver la información sobre pedidos que aún no han sido enviados en “Order Overview”. Como lo que deseamos es introducir un juego, primero nos vamos a “Upload Image” para subir la imagen que usaremos como carátula de nuestro nuevo juego.

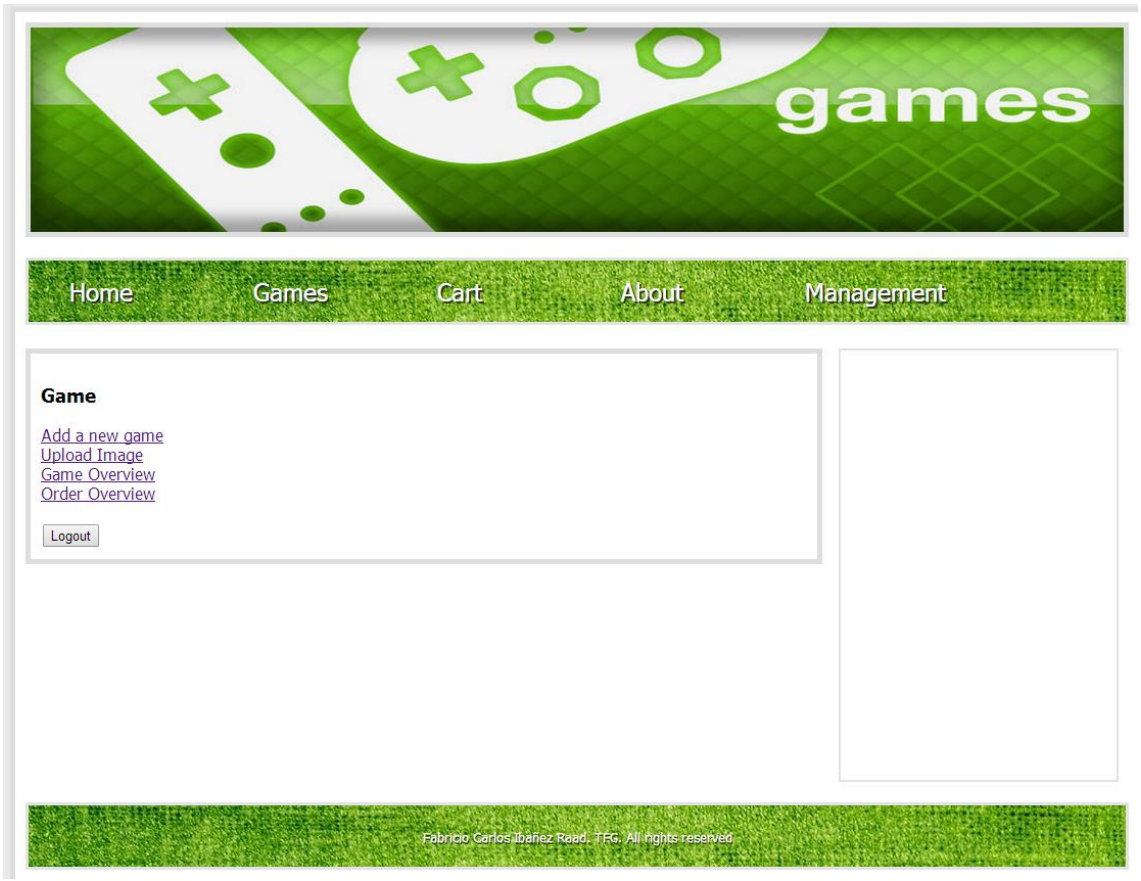



**Figura 25. Visión del CMS tras identificación.**

❖ Página para subir una imagen: /uploadImage.php

Simplemente pulsamos en “Seleccionar archivo” y buscamos en nuestro ordenador la carátula que queremos subir. Tras esto, pulsamos el botón “submit” y tendremos la foto en nuestro servidor web.


**Figura 26. Ejemplo de subir una foto al servidor. /uploadImage.php**

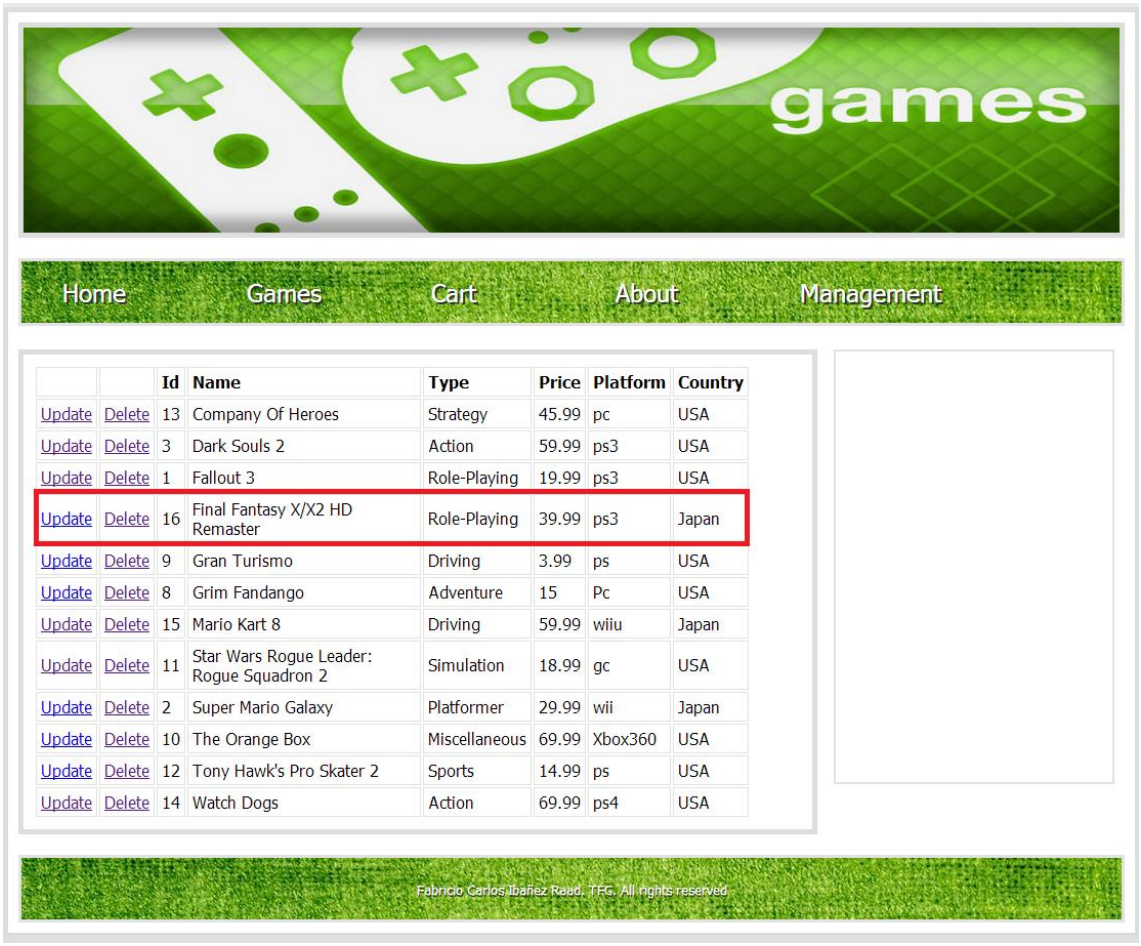
❖ Página para añadir un nuevo juego: /GameAdd.php

En este momento, podemos ya añadir el juego nuevo a nuestra tienda tal y como vemos en la “Figura 27”. Una vez rellenamos los campos clicamos “Submit”. El DropDownList que aparece para seleccionar el tipo de juego es el mismo que se genera cuando queremos seleccionar el tipo de juego en el catálogo de juegos. Lo siguiente que haremos tras pulsar “Submit” será entrar en el “Game Overview” de la “Figura 25” para ver los juegos que tenemos.


**Figura 27. Ejemplo de añadir un nuevo juego a la tienda.**

- Página que nos muestra todos los juegos que tenemos en la tienda: /GameOverview.php

En la “Figura 28” podemos ver como nuestro juego que acabamos de añadir ya aparece en la base de datos. Desde esta página podemos borrar aquellos juegos que ya no tengamos en tienda pulsando “Delete”. Si pulsamos este botón nos pedirá la confirmación de borrar, ya que he implementado un script con JavaScript para que solicite confirmación de borrado. Si lo que queremos es actualizar un juego como, por ejemplo, cambiar la foto, variar el precio y demás, podremos hacerlo pulsando el botón “Update”.




| | | Id | Name | Type | Price | Platform | Country |
|---|---|---|---|---|---|---|---|
| Update | Delete | 13 | Company Of Heroes | Strategy | 45.99 | pc | USA |
| Update | Delete | 3 | Dark Souls 2 | Action | 59.99 | ps3 | USA |
| Update | Delete | 1 | Fallout 3 | Role-Playing | 19.99 | ps3 | USA |
| Update | Delete | 16 | Final Fantasy X/X2 HD Remaster | Role-Playing | 39.99 | ps3 | Japan |
| Update | Delete | 9 | Gran Turismo | Driving | 3.99 | ps | USA |
| Update | Delete | 8 | Grim Fandango | Adventure | 15 | Pc | USA |
| Update | Delete | 15 | Mario Kart 8 | Driving | 59.99 | wiiu | Japan |
| Update | Delete | 11 | Star Wars Rogue Leader: Rogue Squadron 2 | Simulation | 18.99 | gc | USA |
| Update | Delete | 2 | Super Mario Galaxy | Platformer | 29.99 | wii | Japan |
| Update | Delete | 10 | The Orange Box | Miscellaneous | 69.99 | Xbox360 | USA |
| Update | Delete | 12 | Tony Hawk's Pro Skater 2 | Sports | 14.99 | ps | USA |
| Update | Delete | 14 | Watch Dogs | Action | 69.99 | ps4 | USA |

**Figura 28. Tabla con todos los juegos que tenemos en la tienda.**

❖ Página que muestra todos los pedidos pendientes: /OrderOverview.php

En la “Figura 29” aparecen todos los pedidos que aún no han sido enviados. Para diferenciar estos pedidos en la “Tabla 4. Orders” he añadido un atributo “Sent” para saber que pedidos han sido enviados y cuáles no. De esta forma, sólo muestro aquellos que están pendientes por enviar. Por ejemplo, acabamos de recibir de nuestro banco un ingreso de 19.99$ con el asunto “8”. Para comprobar que el ingreso haya sido correcto pulsamos el enlace “View” para acceder a los detalles del pedido.


**Figura 29. Tabla con todos los pedidos pendientes de envío.**

❖ Página con los detalles de un pedido: /OrderView.php

Como vemos en la “Figura 30”, el ingreso de 19.99$ se corresponde al precio que el usuario nos tenía que pagar. Ahora sólo nos falta enviarle el producto que el usuario solicitó a la dirección que nos facilitó. Tras esto, pulsamos el botón “Sent” para que quede registrado como que el envío ya ha sido realizado. Cuando hayamos pulsado el botón ya dejará de aparecer en el apartado de la “Figura 29”, ya que el valor de “Sent” se habrá actualizado a “Yes” y en esa tabla sólo se muestran aquellos que estén pendientes, es decir, con “Sent = No”.

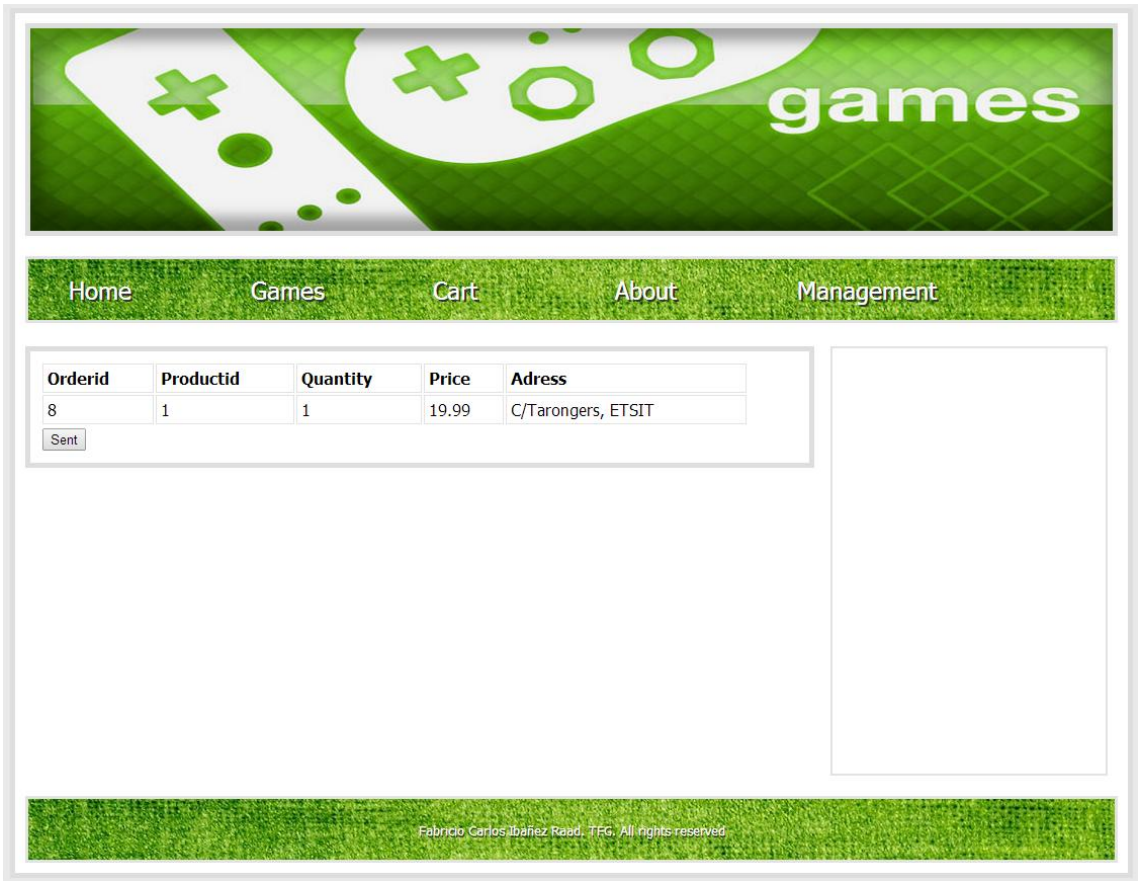



**Figura 30. Detalles de un pedido específico.**

Tras esto puedo dar como concluido cómo se gestiona el CMS.

## 5. PLIEGO DE CONDICIONES

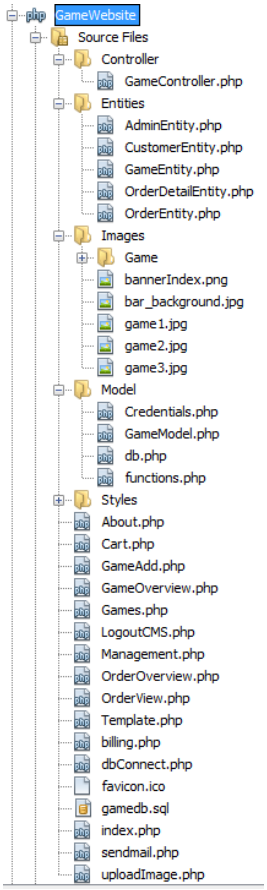



**Figura 31. Plano: carpetas y directorios**

En la “Figura 31” observamos la organización de las carpetas y directorios de que he usado en mi TFG.

Al tratarse de una web hace falta un dominio para alojarla. En mi caso he buscado uno de los dominios más baratos y que puedan soportar PHP+MySQL(PHPmyAdmin). He elegido 1and1.com.

Lo idóneo es elegir el servicio “Basic”, ya que nuestra tienda no es muy grande y nos permite hasta 100GB de almacenamiento y un máximo de 20 tablas en la base de datos. Además, ofrece soporte para PHP 5.5, MySQL 5.0 y cuentas de correo, con lo cual podríamos implementar todo.

Esto tendría un coste mensual de:

4.99€/mes durante primer año y a partir de éste, el precio aumentaría a 6.99€/mes.

Sencillamente, deberíamos alojar la web aquí, donde ya funcionaría y podría ser perfectamente utilizada.

## 6. CONCLUSIONES Y PROPUESTAS DE TRABAJO FUTURO

El trabajo ha sido una experiencia muy enriquecedora para fortalecer mis conocimientos en la arquitectura MVC. Además, afrontar un proyecto de programación desde cero hasta tener un producto acabo ha sido realmente provechoso, ya que todas las dudas y problemas que me han ido surgiendo a la hora de realizar el proyecto han hecho que mis conocimientos de programación hayan mejorado notablemente.

Personalmente, pienso que todo este proceso de formación puede ayudarme mucho a la hora de trabajar en equipos donde se lleven a cabo proyectos más grandes y ambiciosos, ya que conozco muy bien cada una de las partes del proyecto, aunque sea a menor escala.

De hecho, estoy trabajando en SAP donde modelo, diseño y administro constantemente bases de datos y haber interactuado de esta manera con las bases de datos en mi TFG me ha ayudado mucho a poder tener todos los conceptos sobre las bases de datos muy integrados.

Por otro lado, me hubiera gustado mejorar mi arquitectura MVC, ya que en algunas partes como es el “Login” no he podido migrarlo todo a MVC por falta de tiempo.

Por último, las propuestas que añadiría a mi TFG serían las siguientes:

- ✓ Mejoras en la seguridad. Poder hacer un sistema de autentificación y sesiones totalmente seguro y más robusto frente a ataques. Ampliar mi experiencia en la seguridad de PHP para hacer frente a cualquier tipo de ataque como puede ser el “Code Injection”.
- ✓ Permitir la creación de usuarios. Los usuarios que deseen crearse un perfil, para añadir juegos a favoritos, ver sus pedidos realizados…
- ✓ Añadir formas de pago. Poder pagar con paypal, tarjeta bancaria…
- ✓ Añadir comentarios de usuario a los juegos. Permitir que los juegos puedan tener también opiniones sobre los usuarios.

## 7. BIBLIOGRAFÍA


[1] Wikipedia. “La arquitectura Modelo Vista Controlador”. http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[2] Trygve Reenskaug and James O. Coplien. “The DCI Architecture: A New Vision of Object-Oriented Programming”. http://www.artima.com/articles/dci_vision.html

[3] Steve Burbeck, Ph.D. “Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)”. http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

[4] Wikipedia. “Sistema de gestión de contenidos”. http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_contenidos

[5] Wikipedia. “NetBeans IDE”. http://es.wikipedia.org/wiki/NetBeans

[6] Desarrollo Web. “Que es Wamp Server”. http://www.desarrolloweb.com/wiki/wamp-server.html

[7] Sourceforge. “What is Wamp Server”. http://sourceforge.net/p/wampserver/wiki/Home/

[8] Wikipedia. “PHP: Lenguaje de programación”. http://es.wikipedia.org/wiki/PHP

[9] Wikipedia. “MySQL”. http://es.wikipedia.org/wiki/MySQL

[10] Wikipedia. “phpMyAdmin”. http://es.wikipedia.org/wiki/PhpMyAdmin

[11] Wikipedia. “Web template system”. http://en.wikipedia.org/wiki/Web_template_system

[12] Php Academy. “PHP and MySQL Introduction”. https://www.youtube.com/watch?v=BEbKji_pSZM&list=PLfdtiltiRHWEbLm0ErHe7HgEOVIO26R_o

[13] W3schools. “PHP Basics”. http://www.w3schools.com/PHP/

[14] W3schools. “MySQL Basics”. http://www.w3schools.com/sql/default.asp

[15] CodeAcademy. “How to code PHP”. http://www.codecademy.com/tracks/php

[16] Fabricio Carlos Ibañez Raad. “Parte del cliente”. http://youtu.be/u3ZMAAc0fCE

[17] Fabricio Carlos Ibañez Raad. “Parte del administrador”. http://youtu.be/rtgZ8FxQZ7g