

**PROYECTO DE CICLO DE G.S. DE
DESARROLLO DE APLICACIONES WEB**

ARGAMING



**ANTONIO RODRIGUEZ
GONZALEZ
CURSO 2019/2020**

Indice

RESUMEN DEL PROYECTO3
INTRODUCCIÓN.....	.3
OBJETIVO.....	.3
DOMINIO DEL PROBLEMA4
HIPÓTESIS DE TRABAJO5
MATERIALES8
DIAGRAMA DE COMPONENTES.....	.10
CASOS DE USO.....	.10
WIREFRAMES15
CODEIGNITER21
BASES DE DATOS.....	.29
DESARROLLO DEL PROYECTO33
MODELOS/MODELS33
Funciones del BackEndModel35
Funciones del FrontEndModel36
CONTROLADORES/CONTROLLERS37
Funciones del UserController.....	.37
Funciones del AdminController38
Funciones de LoginController39
Funciones de EmailController40
Funciones del FormController.....	.43
Otras funciones a destacar.....	.47
VISTAS52
RESULTADOS54
REFERENCIAS - Webgrafía61
CONCLUSIONES64
AGRADECIMIENTOS64
ANEXOS:65
ANEXO I65
Entorno de trabajo65
ANEXO II71
Narrativa de casos de uso71
ANEXO III76
Configuración del repositorio con github76
ANEXO IV79
Errores y problemas durante el proyecto.....	.79

Título del proyecto: ARGAMING.

Información adicional: aplicación web con información y un pequeño foro para usuarios interesados en hablar sobre videojuegos.

Autor: Antonio Rodríguez González.

Ciclo formativo: Desarrollo de Aplicaciones Web (DAW).

Centro: IES Virgen del Carmen (Jaén).

Tutor del proyecto: Rafael García Cabrera.

RESUMEN DEL PROYECTO

Mi aplicación gira entorno a la creación de un lugar donde los usuarios interesados pueden ingresar e interactuar entre ellos, ya sea comentando o respondiendo otros comentarios sobre el inmenso mundo de los videojuegos. No pretende ser la web donde se albergue toda la información posible sobre los videojuegos, y las respectivas consolas que existen, pues sería una labor simplemente inabordable para un proyecto como este. Aunque sí intenta proporcionar información sobre las últimas novedades y permitir a sus usuarios crear contenido relacionado sobre la temática “gaming”.

INTRODUCCIÓN

Mi idea será crear una web que recoja los datos de un api sobre diversos videojuegos (de diversas temáticas, más antiguos, más nuevos, más o menos famosos, etc.) y permita a los usuarios que accedan a esta información, ya sea a través de un login o registro.

El nombre de la web vendría de combinar las iniciales y gaming: [Antonio Rodríguez González + Gaming](#), fusionado con la ‘g’ de gaming.

OBJETIVO

Conseguir una página informativa/blog, que al mismo tiempo también contendría un pequeño foro donde los usuarios tendrán la oportunidad de intercambiar impresiones y discutir, desde el respeto, sobre la temática de los videojuegos con post sobre las novedades y actualidad del mundo del videojuego.

Después de investigar y comparar entre los distintos frameworks, de los que había oído hablar (Laravel, Symfony, CodeIgniter, Slim, etc.) he decidido que para llevarla a cabo utilizaré el framework de PHP *CodeIgniter*.

Requisitos funcionales:

-Autenticación

-Asignación de funcionalidades

- Envío y recepción de archivos
- Operaciones CRUD (mensajes del foro e información del usuario)
- Importación de datos desde archivos externos (JSON)
- Interconexión entre componentes – servicios
- Abstracción de sistemas de bases de datos relacionales

Requisitos no funcionales:

- Esta página web se encontrará disponible para acceso desde internet
- Dispondrá de diseño responsive para todo tipo de dispositivos.
- Tendrá una base de datos propia para el registro y autenticación de los usuarios.
- Habrá una parte de administrador que se encargará de moderar en el foro.
- Será un sitio interoperable, seguro, accesible y usable, ante todo.

DOMINIO DEL PROBLEMA

Se trata de una aplicación web en formato de blog con post o fichas que incluyen información sobre videojuegos de diversa temática.

El funcionamiento general del blog será de la siguiente forma:

- Un usuario accede a la página inicial del blog y puede, sin estar registrado, consultar información sobre estos videojuegos seleccionando o filtrando los posts existentes. Estos posts se generan a partir de la información extraída en formato JSON de una wiki de videojuegos (reseñada en la webgrafía del final).
- Una vez dentro de cada post existe la opción de realizar comentarios, en la parte inferior, o empezar un tema de discusión creando un post, a parte, sobre ello. Estos posts creados por usuarios, registrados previamente, se pueden visualizar en una pestaña anexa a la de los posts que son meramente informativos.
- Los usuarios registrados pueden iniciar sesión desde la página de login y ver sus datos personales en la de perfil. Pueden modificarlos, así como añadir posts a favoritos para poder hacerles un mejor seguimiento si así lo desean.
- El administrador puede iniciar sesión con unas credenciales especiales que le dan la opción extra de acceder al panel de control desde donde puede realizar diversas funciones como son: dar de

alta o de baja usuarios, moderar posts y sus comentarios, bloquear usuarios, o poner el blog en modo mantenimiento si se están desarrollando labores de actualización de algún tipo.

HIPÓTESIS DE TRABAJO

Como explicaba en la introducción, para construir este blog utilizaré el framework de PHP llamado CodeIgniter. En el Anexo I, detallo como realice la instalación y configuración (muy básica) del mismo para empezar a trabajar en pocos minutos. De la misma forma también detallo como he llevado a cabo la configuración del virtual host para, aunque sea en modo local, entrar a <http://argaming.com>.

Tras decidir utilizar CodeIgniter para mi proyecto final, por supuesto entra en juego la formación. Gracias al consejo de mi tutor del proyecto tuve acceso, en Openwebinars, a un curso perfecto para mí, a modo de introducción básica, pero explicando con suficiente detalle los conceptos que abarca este framework. El profesor de este curso puso como ejemplo el diseño de un pequeño blog que fui realizando al mismo tiempo para enterarme bien de cómo funcionaba. Tras 7–8 h estaba listo para empezar mi andadura con CodeIgniter. Tengo que añadir, no obstante, que el curso no duraba este tiempo, sin embargo, tuve que dedicárselo porque tuve algunos problemas al intentar trabajar con la versión 4 de CodeIgniter; finalmente he utilizado la versión 3, puesto que es la que se utilizaba en este curso y con la que sé manejar bien.

Viendo la creciente relevancia que estaba tomando Bootstrap en lo que al diseño web se refiere, a pesar de que durante el curso no lo vimos cómo tal, decidí utilizarlo para llevar a cabo todo el diseño de mi página web. Este framework, de CSS combinado con etiquetas de HTML5 y su propio JavaScript, me va a facilitar mucho la labor y a agilizar el diseño con una vistosidad difícilmente alcanzable si crease yo todos los estilos personalizados.

Tras elegir CodeIgniter para construir el sitio y Bootstrap para darle forma me faltaría el motor de base de datos. Para ello he utilizado MariaDB/MySQL a través de Xampp que al mismo tiempo me proporciona el Apache, todo en uno. En el Anexo II lo explico detalladamente.

Una vez se tiene Xampp instalado, se puede acceder a la base de datos mediante el navegador con <http://localhost/phpmyadmin>. Esta aplicación web permite controlar y administrar fácilmente todas las bases de datos que tengamos.

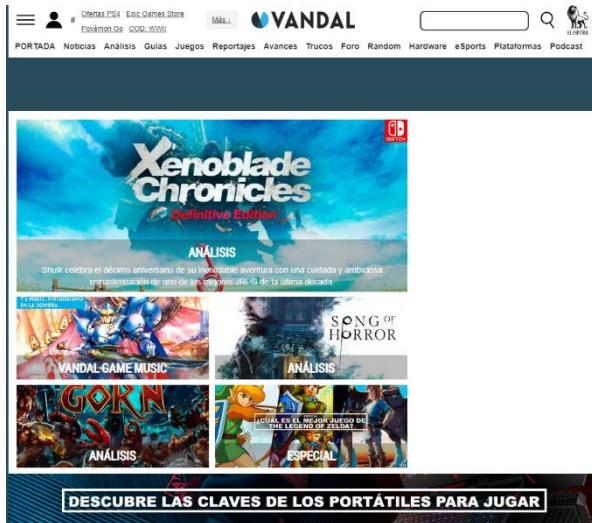
Antes de ponerme como “loco” a programar, me he anotado los pasos que debo seguir:

1. Análisis de los objetivos y cómo los voy a cumplir.
2. Diagramas de componentes (con casos de uso) para entender cómo funcionará la aplicación.

3. Diseño del diagrama entidad-relación y seguidamente inicio del desarrollo de la aplicación en lo que a código se refiere.
4. Pruebas y corrección de posibles fallos durante la construcción de la misma.

Analizando un poco la “competencia” o sitios que ofrecen servicios similares a los que yo quiero ofrecer, he encontrado los siguientes:

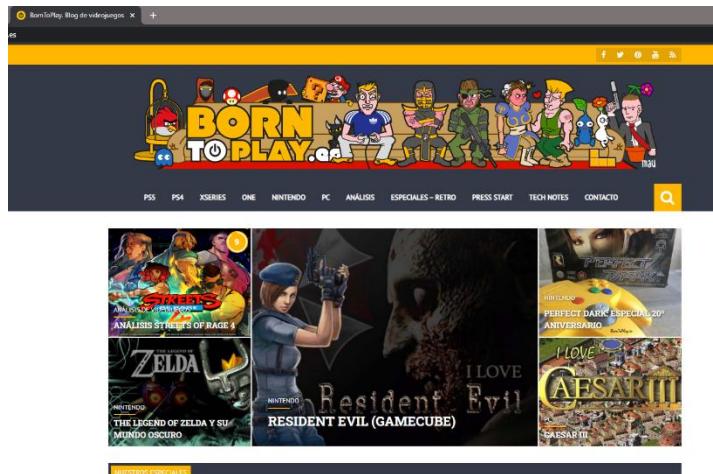
Vandal



Es el máximo exponente que he encontrado en lo que a información sobre videojuegos (en español) se puede pedir, y tiene una web muy completa; quizás demasiado. Es sin duda el mejor ejemplo de que la cantidad no da la calidad porque tiene tantos recursos, opciones, tipos de consolas e imágenes por todas partes que llega a saturar al visitante. Por lo demás, está genial y la tomaré en cuenta para realizar mis entradas del blog.

Borntoplay

Se trata de una página de la misma categoría, pero sin tanto contenido. Está “medio traducida” y me ha gustado su disposición de las entradas en la parte central, bajo la cabecera. Mi idea era algo así también.



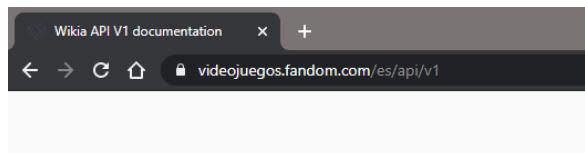
The screenshot shows the homepage of Ultimategame. At the top, there's a search bar with the placeholder "Busca un juego". Below it is a navigation menu with links to "Juegos", "Próximos lanzamientos", "PC", "PS4", "PS3", "Xbox One", "Xbox 360", "Wii U", "3DS", "PS Vita", "Análisis", "Noticias", and "Ofertas en Amazon". A large banner for "Ring Fit Adventure" is prominently displayed, featuring a character from the game. To the right of the banner, there's a sidebar with a news article titled "¿Merece la pena? 5 vs 5, Razones a favor y en contra de Ring Fit Adventure" and another titled "Opinión - Paytowin no es solo pagar por cosas dentro del juego de Mundo del Videojuego". Below the banner, there's a small note about the game being suitable for exercise during the COVID-19 pandemic. At the bottom, there are social sharing buttons for Facebook, Twitter, Google+, and Pinterest, along with a link to "Noticias, wikis, artículos, videos e imágenes de juegos recientes".

Ultimategame

Es una página bastante interesante, del mismo tipo que las dos anteriores, pero además que abarca la venta de contenido digital. Es decir, se pueden comprar licencias de videojuegos en ella. Ofrece la información y el propio videojuego, por lo que está muy completa.

Wiki Juegos – Videojuegos.fandom

The screenshot shows the homepage of Wiki Juegos, a Fandom community. The header features the Fandom logo and navigation links for "JUEGOS", "PELÍCULAS", "TV", and "WIKIS". A large, colorful collage of game screenshots serves as the background. The main content area has a blue header with the text "WikiJuegos" and "Bienvenido a WikiJuegos". Below this, a message says "Hoy es martes, 26 de mayo de 2020 - Actualmente, WikiJuegos tiene 6.813 artículos sobre videojuegos." and links to "Normas de WikiJuegos", "Políticas de WikiJuegos", "Preguntas más frecuentes", "Página principal de ayuda", "Administradores", "Artículos más recientes", and "Fuentes de noticias". A section titled "Artículos destacados" features a thumbnail for "Super Mario Maker 2: ¡El mundo de Mario sin límites!" showing Mario and Luigi working on a level. Other smaller thumbnails for various games like "Call of Duty: Warzone" and "Grand Theft Auto V" are also visible.



Wikia API V1 documentation

/Activity

Get information about the latest user activity on the current wiki

/Articles

Get simplified article contents

/Mercury

Get wiki data, including key values, navigation data, and more

/Navigation

Get wiki navigation links (the main menu of given wiki)

/RelatedPages

Get pages related to a given article ID

/Search

Get results for combined (wiki and cross-wiki) search

/SearchSuggestions

Find suggested phrases for chosen query

/User

Get details about selected users

Es el último, y sin lugar a dudas la web más importante durante el desarrollo de mi proyecto. Es de aquí de donde obtengo el JSON con los datos de los videojuegos y, por tanto, entorno a lo que gira todo mi proyecto, básicamente.

Se realiza a través del apartado API de la misma web y tiene los apartados que se pueden observar en la captura de la izquierda.

He hecho uso, principalmente, del /Search para obtener estos datos.

Posee una base de datos muy completa sobre videojuegos de todas las generaciones, desde los primeros allá por el 2000 hasta los más recientes y para consolas de última generación.

MATERIALES

Aunque he explicado en el anterior apartado los recursos de software que he utilizado para mi proyecto ahora los detallaré algo más, incluyendo las versiones, y en qué consisten, por ejemplo.

El software exclusivo del proyecto:

- **CodeIgniter (3.1.11):** es el programa, o framework, usado para desarrollar mi proyecto
- **Bootstrap (4.4.1):** es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.
- **Jquery (2.1.3 - 3.4.1):** biblioteca multiplataforma de JavaScript que permite interactuar de diversas maneras con los elementos HTML.
- **Font Awesome (4.3.0 - 5.13.0):** librería de fuentes e iconos. Creado para integrarse con Bootstrap.

Software complementario, que me ha servido de ayuda para realizarlo:

- **Visual Studio Code (1.45.1):** es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. De licencia libre.
- **Firefox Browser Developer (77.0b9 - 64-bit):** Firefox Developer Edition está diseñado por Mozilla.

- **Paquete Office (excel, word, powerpoint, etc.):** he usado varios de los programas Office para realizar mi proyecto, puesto que la memoria la he escrito en Word, el historial bibliográfico he ido recopilándolo en un Excel y la presentación final será hecha con PowerPoint.
- **ClipX (1.0.3.7):** es un pequeño administrador del historial del portapapeles para Windows. El cuál se ha convertido en un compañero imprescindible, por la ventaja que ofrece a la hora de crear, mover y editar código, y por lo fluido que funciona.

El proyecto se ha desarrollado íntegramente en mi portátil, un HP 15 Notebook PC. Con un procesador i5-4210 y 8 GB de RAM. Con 500 GB de almacenamiento HDD. Sistema operativo: Windows 10 Pro (64 bits).

Por supuesto, durante el desarrollo del proyecto me he valido de dispositivos de almacenamiento externos (disco duro externo y USB) donde he ido realizando copias de seguridad incrementales y periódicas.

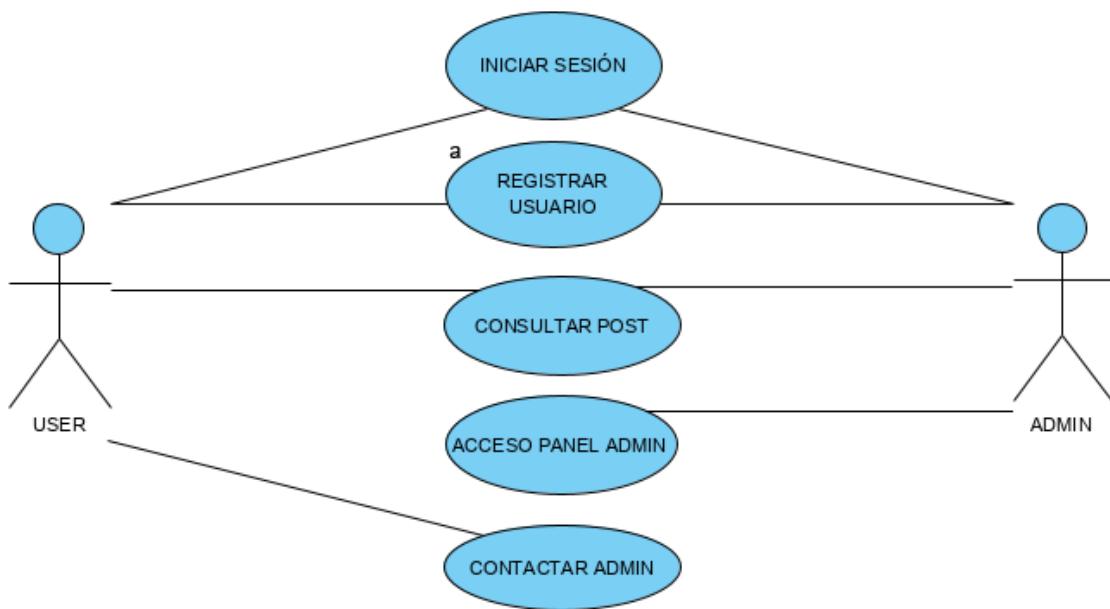
DIAGRAMA DE COMPONENTES

CASOS DE USO

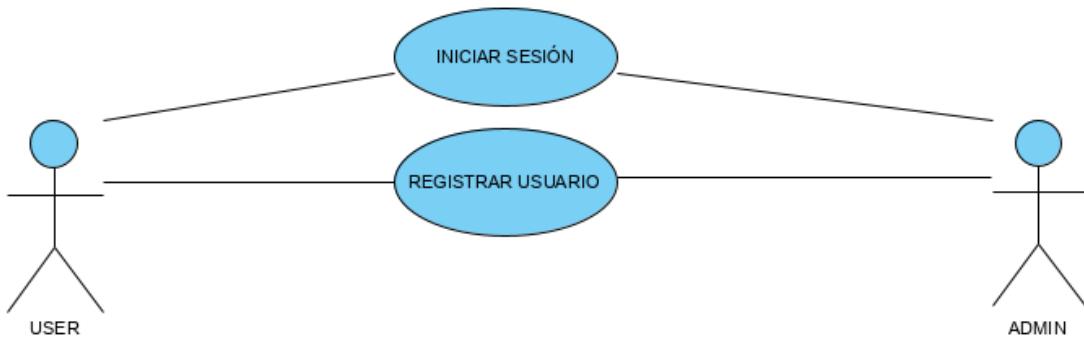
Actores presentes: *usuario no logueado*, *usuario* (registrado y logueado) o *administrador*

Casos de uso que se dan durante la utilización del blog:

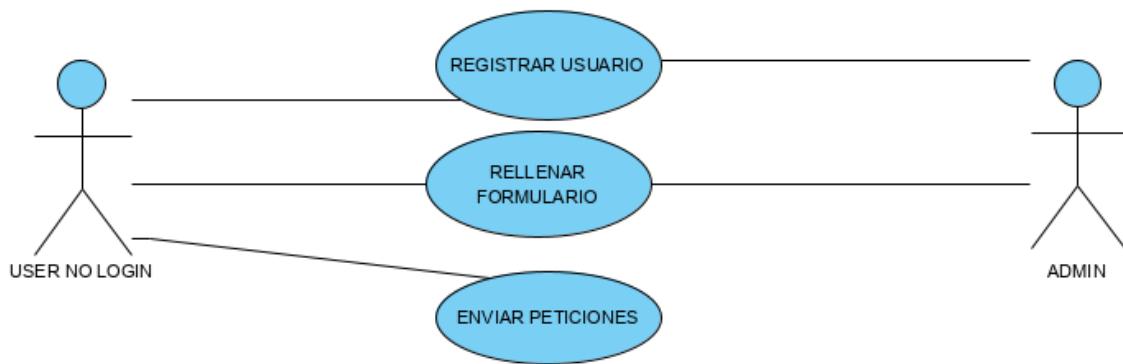
-Acceso a la página de inicio:



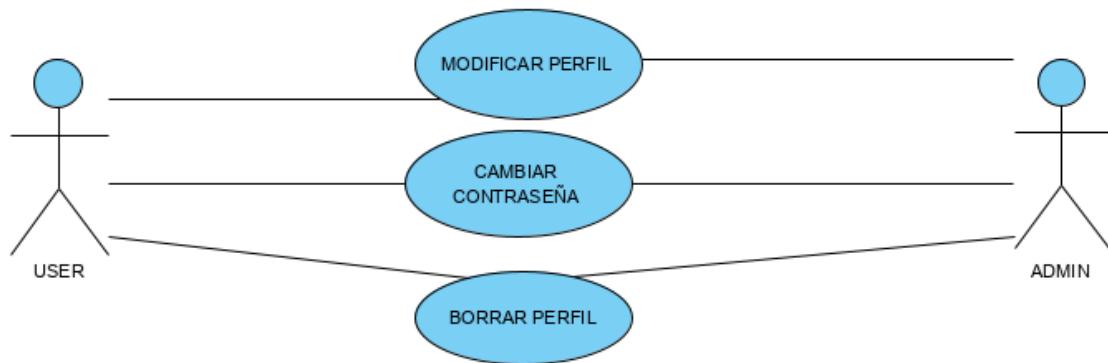
-Login o inicio de sesión:



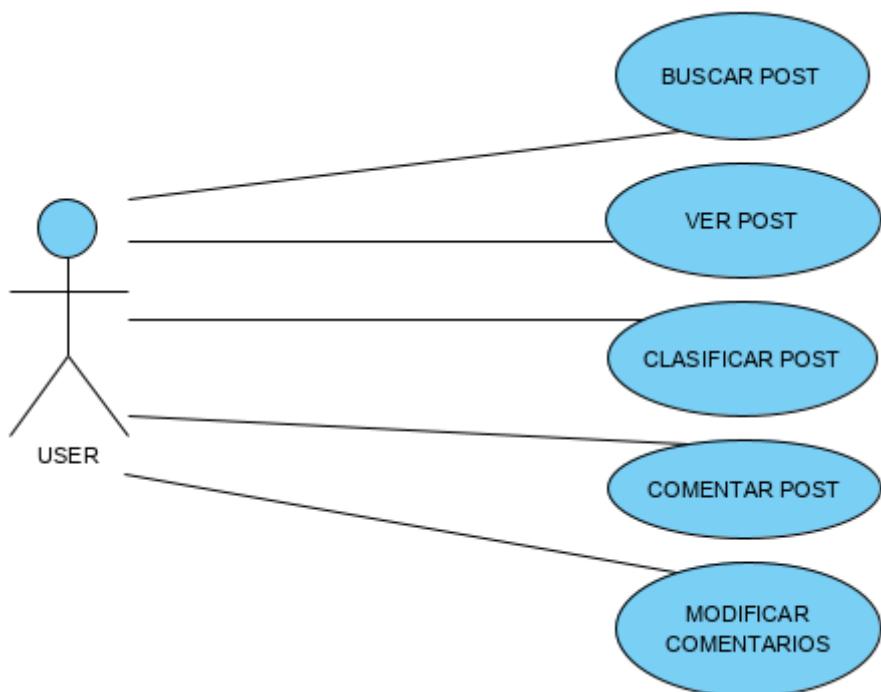
-Registro (para usuarios no registrados):



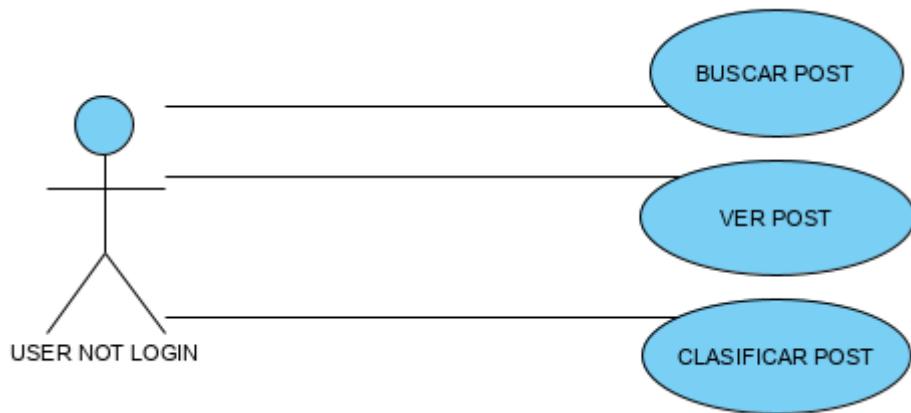
-Página de perfil de cada usuario (una vez esté registrado y logueado):



-Página de posts (información de los videojuegos):



-Para esta página de posts, si el usuario no estuviese logueado solo tendría las siguientes opciones:



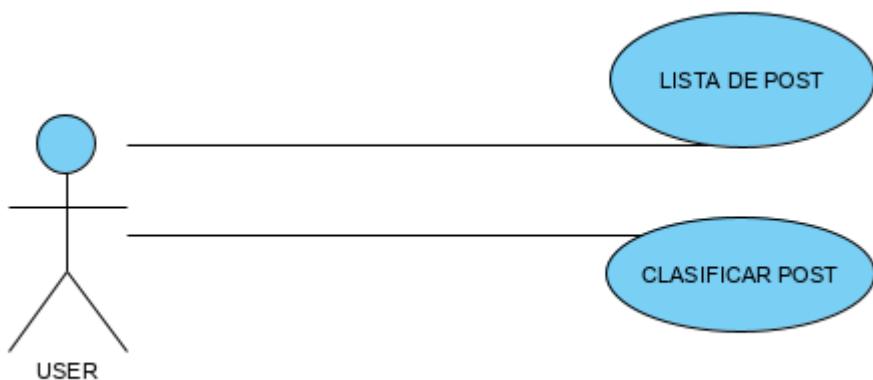
-Página de posts creados por los usuarios:



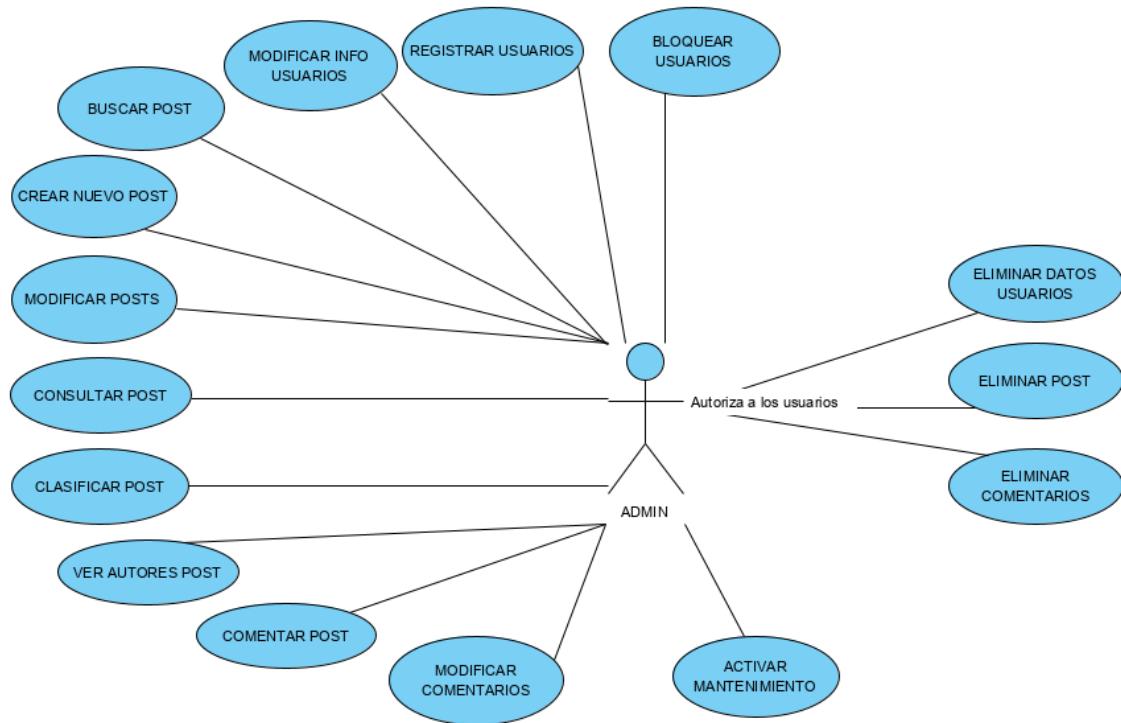
-La página de posts creados por usuarios para un usuario que no haya iniciado sesión:



-Página o listado de post/s creados por un mismo usuario/autor:



-Panel de administración, en el que el usuario con derechos de administración tendría acceso a todas las opciones configurables en el blog:



El usuario administrador solo eliminará datos de usuarios, posts creados por usuarios o comentarios (en cualquier post) si así se lo solicita un usuario cualquiera y él mismo lo ve conveniente.

WIREFRAMES

-Inicio:

The wireframe shows the layout of the ARGaming website's homepage. At the top, there is a header section with a logo placeholder labeled "LOGOTIPO", a "LOGIN" button with a user icon, and a "REGISTRO" button with a pen icon. Below the header is a navigation bar with links for "Inicio", "Videojuegos", "Foro", and "Contacto". The main content area features a teal-colored banner with arrows pointing left and right, and three dots in the center, followed by the heading "NOVEDADES". Below this, there are four placeholder boxes, each containing a square icon with an 'X' and some placeholder text: "Lorem ipsum dolor Nunc maximus, nulla ut lorem felis nec erat...". Underneath this section is a title "POST DE DISCUSIÓN DESTACADOS" above a table. The table has columns for "Activos", "Título", "Descripción", and "Última actualización". It lists five posts, all of which are checked (indicated by a checked checkbox icon). The posts are: Post 1 (25/04/20), Post 2 (03/05/20), Post 3 (28/04/20), Post 4 (23/04/20), and Post 5 (24/04/20). At the bottom of the page is a footer section with a "LOGO CUADRADO" placeholder, social media icons for Facebook, Instagram, YouTube, and Twitter, and a copyright notice: "Creador | Políticas privacidad | Políticas de cookies" and "Copyright © ARGaming 2020. Todos los derechos reservados".

Activos	Título	Descripción	Última actualización
<input checked="" type="checkbox"/>	Post 1	Lorem ipsum lorem	25/04/20
<input checked="" type="checkbox"/>	Post 2	Lorem ipsum lorem	03/05/20
<input checked="" type="checkbox"/>	Post 3	Lorem ipsum lorem	28/04/20
<input checked="" type="checkbox"/>	Post 4	Lorem ipsum lorem	23/04/20
<input checked="" type="checkbox"/>	Post 5	Lorem ipsum lorem	24/04/20

-Página de listado de videojuegos:

The wireframe illustrates a website layout for a video game listing platform. At the top, there is a header section with a logo (a triangle labeled 'LOGOTIPO'), a login/register area (with icons for user and checkmark), and a navigation bar with links for 'Inicio', 'Videojuegos' (highlighted in green), 'Foro', and 'Contacto'. The main content area is titled 'LISTADO DE VIDEOJUEGOS' and contains a table with 10 rows of game data. Below the table is a navigation bar with page numbers («, 1, 2, 3, 4, 5, »). The footer features a square logo (labeled 'LOGO CUADRADO'), social media links for Facebook, YouTube, Instagram, and Twitter, and a copyright notice: 'Creador | Políticas privacidad | Políticas de cookies' and 'Copyright © ARGaming 2020. Todos los derechos reservados'.

Activos	Título	Descripción	Última actualización	Etiquetas
<input checked="" type="checkbox"/>	Juego 1	Lorem ipsum lorem	25/04/20	Acción y fantasía
<input checked="" type="checkbox"/>	Juego 2	Lorem ipsum lorem	03/05/20	Futurista y distópico
<input checked="" type="checkbox"/>	Juego 3	Lorem ipsum lorem	28/04/20	Estrategia
<input checked="" type="checkbox"/>	Juego 4	Lorem ipsum lorem	23/04/20	mmorpg
<input checked="" type="checkbox"/>	Juego 5	Lorem ipsum lorem	24/04/20	simulación y ciencia ficción
<input checked="" type="checkbox"/>	Juego 6	Lorem ipsum lorem	25/04/20	arcade
<input checked="" type="checkbox"/>	Juego 7	Lorem ipsum lorem	03/05/20	deportivo
<input checked="" type="checkbox"/>	Juego 8	Lorem ipsum lorem	28/04/20	juegos de mesa y familiar
<input checked="" type="checkbox"/>	Juego 9	Lorem ipsum lorem	23/04/20	arcade
<input checked="" type="checkbox"/>	Juego X	Lorem ipsum lorem	24/04/20	acción y estrategia

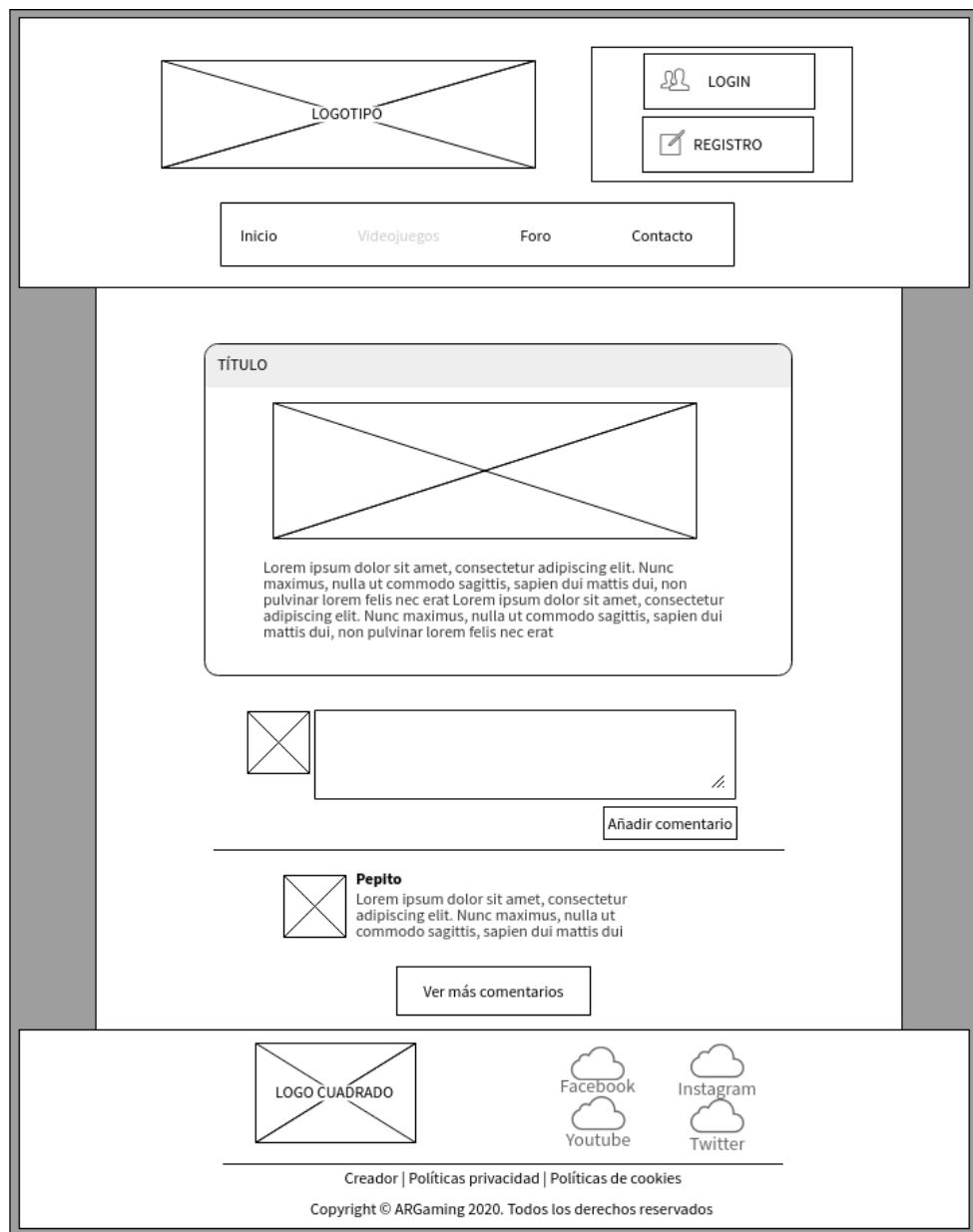
« 1 2 3 4 5 »

LOGO CUADRADO

Facebook
Youtube
Instagram
Twitter

Creador | Políticas privacidad | Políticas de cookies
Copyright © ARGaming 2020. Todos los derechos reservados

-Página de post/videojuego (formato similar, válido para ambos casos):



-Formulario de registro (vista pública/admin):

The image shows a placeholder for a registration form, represented by a large blue rectangular area containing a smaller white rectangular form. The white form is titled "Nuevo usuario" and contains fields for "Nombre de usuario", "Nombre", "Apellidos", "Email", "Contraseña", and "Confirmar contraseña". It also includes a checked checkbox for "Políticas de privacidad y RGPD" and a "Completar registro" button.

Nuevo usuario

Nombre de usuario

Nombre

Apellidos

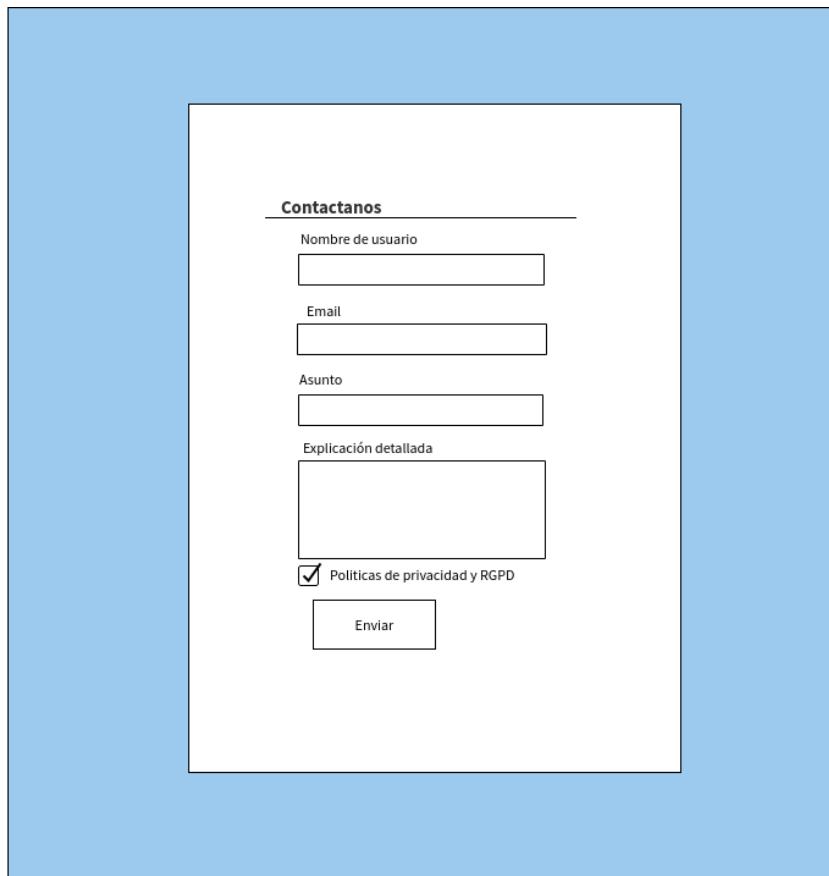
Email

Contraseña

Confirmar contraseña

Políticas de privacidad y RGPD

-Formulario de contacto:



A large blue rectangular placeholder box occupies the left side of the slide, intended to hold a diagram of a contact form.

Contactanos

Nombre de usuario

Email

Asunto

Explicación detallada

Políticas de privacidad y RGPD

Enviar

-Login:



A large blue rectangular placeholder box occupies the left side of the slide, intended to hold a diagram of a login form.

Inicio de sesión

Email

Contraseña

Iniciar sesión

-Panel de control de administrador:

ID	Nombre	Apellidos
1	Antonio	Rodríguez
2	Pepito	Pepe
3	María	Martinez

-Novedades:

NOTICIAS Y ENLACES RELACIONADOS

Noticias y enlaces relacionados
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc maximus, nulla ut commodo sagittis, sapien dui mattis dui, non...

LEER MÁS

LEER MÁS

CREADOR | POLÍTICAS DE PRIVACIDAD | POLÍTICAS DE COOKIES

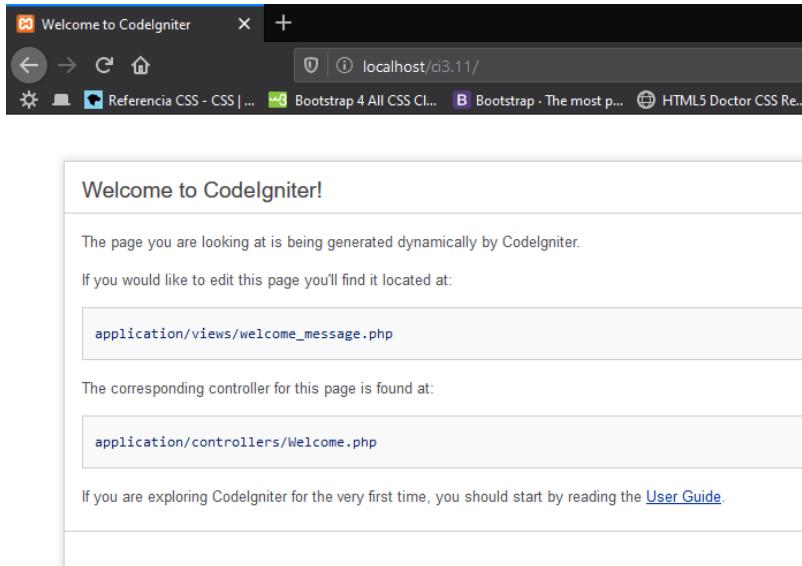
Copyright © ARGaming 2020. Todos los derechos reservados

CODEIGNITER

Una vez están analizados, estudiados y diseñados los diferentes diagramas y casos de uso, paso a crear la aplicación en sí. Para ello, como explico en el Anexo I (sobre el entorno de trabajo) instalo y configuro lo poco que hace falta para empezar a trabajar con CodeIgniter.

Para poder ver la aplicación desde el navegador, primero habría que arrancar el servidor Apache de Xampp, y entonces sí podremos ver lo que muestra nada más empezar.

Se trata de una página de inicio, a modo de bienvenida:



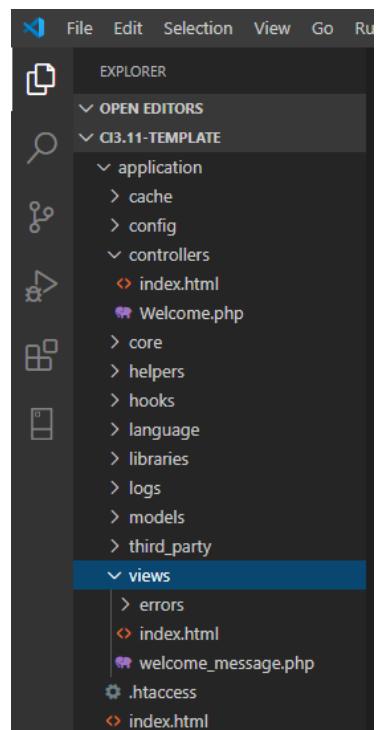
Esta misma página nos da las primeras indicaciones de cómo funciona CodeIgniter y es que explica desde dónde viene todo lo que muestra.

Como sabemos ya, este framework funciona mediante el MVC o Modelo Vista-Controlador, y ese sistema consiste en que un controlador es el encargado de mediar entre datos y vistas, que es lo que finalmente ve el usuario.

Un controlador en CodeIgniter es un archivo que contiene el código de una clase, de programación orientada a objetos, que colocamos en un directorio específico del esquema de carpetas de nuestro sitio. Tiene un nombre cualquiera, que se asociará con una URL de nuestra aplicación web.

El controlador en este caso es [Welcome.php](#) y se sitúa en la carpeta controllers. Desde ahí llama a la vista [welcome_message.php](#) o mensaje de bienvenida, cargado desde la carpeta de views o vistas, que es lo que estamos viendo en el navegador.

A la derecha se puede observar, en el editor Visual Studio Code, toda la carpeta de application. Arriba están los “controllers” y más abajo las “views”.



En welcome.php (el controlador inicial) encontraremos esta función:

```
public function index()
{
    $this->load->view('welcome_message');
```

Es la función index y carga la vista de welcome_message (se refiere a welcome_message.php, es lo mismo).

Y, ¿cómo llega hasta el controlador? Con la ruta, que está en el archivo del directorio /config, en routes.php:

```
$route['default_controller'] = 'Welcome/index';
$route['404_override'] = '';
$route['translate_uri_dashes'] = FALSE;
```

En este archivo se van a ir recogiendo todas las rutas del sitio web.

El ‘default_controller’ indica la ruta por defecto que se cargará, y ahí encontramos asignada la cadena con una estructura de ‘Controlador/función’ que se correspondería con ‘Controlador -> Welcome/ Función -> index’. Y como en esa función hemos visto que carga el mensaje, estaría el camino completo ya explicado.

Cuando la carga de una vista implica recoger datos entran en juego los ‘Model’ de CodeIgniter y lo explicaré más adelante con un ejemplo.

Este sistema de rutas y vistas funciona bien, pero para ser más prácticos y eficaces se puede crear una clase ‘layouts’ que implementa las vistas cargándolas en layouts diseñados por partes: el layout con los archivos de bootstrap, css y js; el archivo de la vista con el contenido principal, y para casi todas las páginas una cabecera y footer que se repetiría.

La clase layouts:

```
class Layouts
{
    //Instance de Codeigniter
    private $CI = null;

    function __construct()
    {
        $this->CI =& get_instance();
    }

    public function view( $data = array())
    {
        if( !empty($data))
        {
            $view_content = $this->CI->load->view($data['vista'], $data['params'], TRUE);
            //A la vista se le pasa (en el directorio view/layouts) el contenido y el título
            $this->CI->load->view('layouts/' . $data['layout'], array(
                'content_for_layout' => $view_content,
                'title_for_layout' => $data['titulo']
            ));
        }
    }
}
```

```
<!-- Page Title -->
<title>
    <?php
        |   echo $title_for_layout;
    ?>
</title>

<!-- Font Awesome -->
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.13.0/css/all.css">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.13.0/css/v4-shims.css">

<!-- Bootstrap Core CSS -->
<link href="/bootstrap/css/bootstrap.min.css" rel="stylesheet">

<!-- Custom CSS -->
<link href="/css/estilos.css" rel="stylesheet">

<!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
<!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
<!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
    <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
<![endif]-->

<head>

<dy>

<!-- Navigation -->
<?php require_once dirname( dirname( dirname(__FILE__))) . "/views/includes/header.php"; ?>

<!-- Page Content -->
<?php
    |   echo $content_for_layout;
?>

<!-- Footer -->
<?php require_once dirname( dirname( dirname(__FILE__))) . "/views/includes/footer.php"; ?>

<!-- jQuery Version 3.4.1 -->
<script src="/bootstrap/js/jquery-3.4.1.min.js"></script>

<!-- Popper JS 1.16.0 -->
<script src="/bootstrap/js/popper.min.js" defer></script>
```

Para que esta clase tenga sentido, es necesario crear en /views/layouts/ los diseños que queramos utilizar. En mi caso he creado varios layouts para las funciones que abarca mi web (versión home, administrador, login, registro o con sesión iniciada, si ya se ha logueado previamente).

Estos layout lo que hacen es gestionar un sistema de carga de diferentes opciones de cabecera o pie de página dentro de una misma estructura. Es decir, dentro de un ly, el de home, por ejemplo, encontraríamos una página de html normal y corriente, si no fuese porque le falta el contenido y en su lugar tiene código php que principalmente indica que se cargan otros ficheros php dentro de éste.

Los apartados que componen todas las vistas son:

La cabecera HTML (head), es la parte superior, que incluye a su vez

- Título de la página.
- Enlaces a los estilos de Font Awesome, Bootstrap CSS y mis estilos personalizados.
- Un apartado comentado que podría ser útil si se va a abrir el proyecto con internet explorer.

El cuerpo HTML (body), que completa el resto del contenido con:

- La parte de navegación, menús y demás, generada a través del ‘header.php’ en ‘/views/include/’.
- El contenido, en sí, de la página que corresponda en cada caso (ya que, consiste en una variable a la que se va asignando desde el controlador un array con los datos).
- La parte inferior, o pie de página, que se genera a partir del ‘footer.php’ en ‘/views/include/’.

Y, por último, dos archivos de javascript encargados de algunas funcionalidades, principalmente relacionadas con la utilización de Bootstrap y sus componentes (menus, formularios, slider, etc.).

Sin embargo, esta estructura no sirve sin un controlador que permita interactuar, mandar y recibir información... Uno de los primeros controladores que hay que crear es el de inicio, index, o como prefiramos llamarle. Consiste en el controlador que llama el framework a través de las rutas por defecto al escribir la “dirección raíz” del sitio. Me explico, cuando escribamos en la barra de direcciones localhost/nombre_proyecto/ cargará este controlador (que podemos cambiar del que trae por defecto y dejar otro).

En mi caso, al trabajar con el virtual host de Apache llamo a la dirección argaming.com y responde la página de inicio gracias a este sistema de rutas.

```
$route['/'] = "UserController/index";  
$route['default_controller'] = 'UserController/index';
```

No importa si se utiliza una fórmula u otra, es indiferente porque si CodeIgniter lee default_controller le da preferencia sobre la barra a secas (/). Al poner nuestra dirección estaríamos seleccionando el UserController, como he llamado al controlador que se utiliza para las funciones desempeñadas por cualquier usuario, esté logueado o no.

En esta clase, UserController, encontramos varias cosas interesantes para empezar.

Extiende de una clase CI_Controller, que necesitan todos los controladores y viene por defecto con CodeIgniter. Dentro de la nueva clase existe un constructor que es llamado a la hora de cargar modelos (para las bases de datos) y a continuación, empiezan las funciones. La primera, es la denominada index() y a partir de dos array llama la vista y le pasa los datos, respectivamente. El primer array almacena los datos, que se cargan como ‘params’ o parámetros. El segundo contiene estas variables que se pasan a la función view de la clase layout que detallaba anteriormente. La vista a la que llama es ‘index.php’ en la carpeta web, dentro de las vistas, utilizando el layout ‘ly_home.php’. Al final aparece el título que se le asigna a la página, Inicio.

```
<?php
defined('BASEPATH') or exit('No direct script access allowed');
class UserController extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
        $this->load->model('FrontEndModel', 'FrontEndModel');
        $this->load->model('BackEndModel', 'BackEndModel');
    }

    public function index()
    {
        $datos = array();

        $vista = array(
            'vista' => 'web/index.php',
            'params' => $datos,
            'layout' => 'ly_home.php',
            'titulo' => 'Inicio'
        );
        $this->layouts->view($vista);
    }
}
```

```
class FrontEndModel extends CI_Model
{
    # Variable donde se guarda la conexión a la base de datos
    private $db = null;

    function __construct()
    {
        parent::__construct();

        # Cargamos la conexión a la base de datos
        $this->db = $this->load->database('default', true);
    }

    # Ejecuta consultas y devuelte los resultados en un array
    public function ExecuteArrayResults( $sql )
    {
        $query = $this->db->query( $sql );
        $rows = $query->result_array();
        $query->free_result();

        return ( $rows );
    }

    public function ExecuteResultsParamsArray( $sql, $params )
    {

        $query = $this->db->query( $sql, $params );
        $rows['data'] = $query->result_array();
        $query->free_result();

        return ( $rows );
    }
}
```

Esa sería una muestra del controlador al principio, pero ahora es algo más extenso, debido a la necesidad de interpretar varios factores al mismo tiempo. Cuando me refiero a los modelos y al trabajo con base de datos, en CodeIgniter funciona de la siguiente manera (imagen de la izquierda).

Lo primero es tener creado el modelo, que va en la carpeta models. Una vez creado, funciona como los layout, se crea su clase que se extiende de CI_Model, la clase por defecto para modelos, y dentro éste contiene una llamada a la base de datos, y consultas, tanto predefinidas con órdenes básicas de CodeIgniter, como personalizadas.

Con las llamadas “básicas” me refiero a las que se utilizan para realizar cualquier consulta (execute -> query) crear nuevos registros (insert), actualizar los registros presentes en la base de datos (update) o borrarlos (delete).

Sin embargo, yo dentro de los modelos he querido diferenciar, de ahí que los llame frontend y backend.

Dado que el primero lo uso tanto en el [UserController](#), para todas las acciones que puedan necesitar los usuarios estándar como en [LoginController](#), que es para ciertas acciones relacionadas con el inicio de sesión y los datos de perfil del usuario estándar.

```
# Ejecuta querys sin devolver resultados deletes, inserts o updates
public function Execute( $sql )
{
    $this->db->query( $sql );
}

public function insert( $tabla, $datos )
{
    $this->db->insert( $tabla, $datos );
}

public function update( $tabla, $datos, $WHERE )
{
    $this->db->update( $tabla, $datos, $WHERE );
}

public function delete( $tabla, $WHERE )
{
    $this->db->delete( $tabla, $WHERE );
}
```

Por otra parte, están las funciones personalizadas a las necesidades que me han ido surgiendo. De igual forma se dividen entre usuario estándar en el [FrontEndModel](#) y otra, las acciones reservadas para el usuario Administrador del blog, se llevan a cabo con el modelo que he llamado [BackEndModel](#) y aunque se producen repeticiones me parecía lo más lógico para evitar un caos de funciones llamando a la base de datos desde todas partes.

Para que funcione la base de datos, puntualizar que he tenido que irme al fichero de configuración [database.php](#) y añadirla de la siguiente forma:

```
$db['default'] = array(
    # Suponiendo que nuestra base de datos se llama base_de_datos
    # y estamos conectando a localhost
    'dsn'      => 'mysql:host=localhost;dbname=argaming_db',
    'hostname'  => 'localhost',
    'username'  => 'admin', #El nombre de usuario
    'password'  => '1234', #La contraseña del usuario de arriba
    'database'  => '',
    'dbdriver'  => 'pdo', #Justo aquí utilizamos PDO
    'dbprefix'  => '',
    'pconnect'  => FALSE,
    'db_debug'  => (ENVIRONMENT !== 'development'),
    'cache_on'   => FALSE,
    'cachedir'  => '',
    'char_set'   => 'utf8',
    'dbcollat'   => 'utf8_general_ci',
    'swap_pre'   => '',
    'encrypt'   => FALSE,
    'compress'  => FALSE,
    'stricton'   => FALSE,
    'failover'   => array(),
    'save_queries' => TRUE
);
```

CodeIgniter permite elegir entre PDO y Mysqli pero, por supuesto, prefería PDO después de estar todo el curso trabajando con él.

Mi base de datos, que funciona con MySQL, como se puede observar se llama ‘argaming_db’.

He trabajado con ella desde la línea de comandos de Xampp y sobre todo desde PHPMyAdmin, dónde ha quedado tal que así:

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
categorias	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	0 B
comentarios	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	6	InnoDB	utf8mb4_general_ci	48.0 KB	-
pertenecen	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
post	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	5	InnoDB	utf8mb4_general_ci	48.0 KB	-
usuarios	Examinar, Estructura, Buscar, Insertar, Vaciar, Eliminar	9	InnoDB	utf8mb4_general_ci	32.0 KB	-
5 tablas	Número de filas	20	InnoDB	utf8mb4_general_ci	160.0 KB	0 B

Contiene las tablas de usuarios, post, categorías, comentarios y pertenecen (categorías y pertenecen son tablas simbólicas, puesto que no he llevado a cabo ningún uso con ellas).

Más adelante, en el siguiente punto, detallaré mejor el tema de bases de datos.

Volviendo al punto del que hablaba antes, cuando se trabaja con modelos lo siguiente sería llamar a ‘cargar’ la base de datos con ‘load’ en la función construct que se añade en cada controlador, en la parte superior, que vaya a necesitar acceder a los datos. Después, ya solo hace falta llamar a la función que necesitemos de este modelo, desde las funciones del controlador.

En la de `index()`, por ejemplo, se recurre al método `Lista('tabla', 'campo clasificador')`, el cuál devuelve una lista de los registros contenidos en la tabla que le indiquemos y ordenando de forma descendente a partir del campo con el que se quiera clasificar.

```
/*Lista la tabla que se le solicite y clasifica la información a partir del parámetro indicado*/
public function Lista($tabla,$clasif)
{
    $sql = "select * from ".$tabla." order by ".$clasif." desc";
    return ( $this->ExecuteArrayResults( $sql ) );
}
```

También decir que la devolución de los datos se hace mediante un método propio de CodeIgniter, que procesa los datos en formato Array y los devuelve como si se hubiera pasado por un bucle con un `for()` o un `foreach()`.

```
# Ejecuta consultas y devuelte los resultados en un array
public function ExecuteArrayResults( $sql )
{
    $query = $this->db->query( $sql );
    $rows = $query->result_array();
    $query->free_result();

    return ( $rows );
}
```

```

public function index()
{
    $posts = $this->FrontEndModel->Lista('post','visitas');
    //Se almacenan los datos en el array para pasarselo a la vista que corresponda
    $datos = array(
        'posts' => $posts,
    );
    //debug($datos);
    /*Tras obtener los datos que se van a mostrar,
    se comprueba si hay una sesión abierta por parte del usuario*/
    $verif = comprobar_login();
    if (!empty($verif)) {
        $datos['rol'] = $verif['rol'];
        $vista = array(
            'vista' => 'web/index.php',
            'params' => $datos,
            'layout' => 'ly_session.php',
            'titulo' => 'Inicio'
        );
        $this->layouts->view($vista);
    } else {
        $vista = array(
            'vista' => 'web/index.php',
            'params' => $datos,
            'layout' => 'ly_home.php',
            'titulo' => 'Inicio'
        );
        $this->layouts->view($vista);
    }
}

```

Tras conseguir estos datos del modelo, los almacena en el array con la referencia que corresponda y que luego será con las que se le llame en la vista.

Pero antes de mostrar la vista, además, se comprueba si el usuario hubiera iniciado sesión para decidir si mostrarle una u otra cabecera, ya que, ésta es la diferencia principal del ly_session.php o el ly_home.php. Al verificar si existe la sesión también almacena una variable en el array de datos, y dentro del propio layout será necesaria para darle a entender si es el usuario administrador o simplemente uno estándar y conceder así la opción de acceder al panel de control. Esta decisión se realiza con un **operador ternario** tal que así:

```

<body>

    <!-- Navigation -->
    <?php
        $rol == 'administrador'
        ? require_once dirname( dirname( dirname( __FILE__ ) ) ) . "/views/includes/header_session_admin.php"
        : require_once dirname( dirname( dirname( __FILE__ ) ) ) . "/views/includes/header_session_user.php";
    ?>

```

Este operador es similar a un if, ya que consiste en realizar una comparación con una variable y seguidamente añadir una interrogación (?) para lo que queremos que realice si la comparación es positiva o dos puntos (:) para que haga otra cosa si la comparación es negativa.

De esta manera se completa el proceso de mostrar la página de inicio, desde que se ejecuta la ruta raíz del sitio web, en la url, hasta que el usuario ve la información por pantalla.

BASES DE DATOS

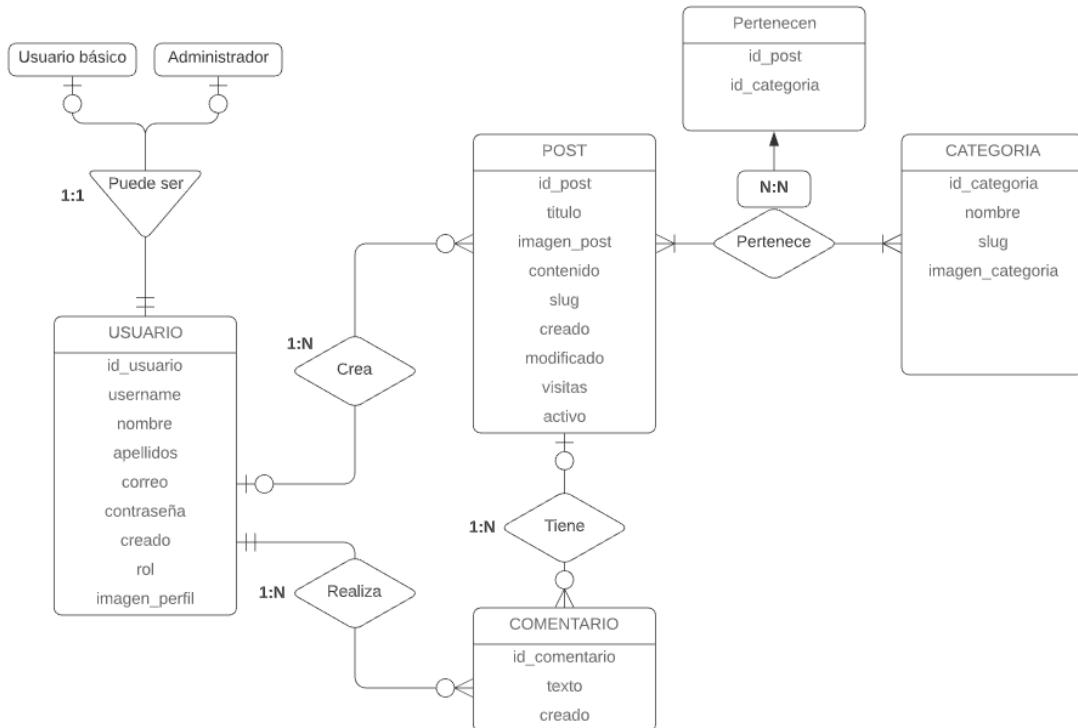
Mi proyecto en lo que se refiere a base de datos se reduce a utilizar MySQL para las consultas y la administración, y PHPMyAdmin, desde localhost, para visualizar en ciertos momentos las tablas o registros, o probar las consultas antes de pasárlas al modelo en CodeIgniter.

Para algunas consultas relacionadas con búsquedas he implementado unas cuantas expresiones regulares muy útiles y potentes que devuelven los resultados con exactitud milimétrica.

Lo primero, tras diagramas de casos de uso, narrativa, diagrama de secuencia, etc. Es realizar el diagrama entidad relación para ver qué tablas necesito y desarrollar las relaciones pertinentes entre ellas.

- **DIAGRAMA ENTIDAD-RELACIÓN**

Ha sido mi primer paso para proceder a crear la base de datos de mi aplicación. Las entidades son: el usuario, los posts, sus categorías y los comentarios que se pueden realizar en los posts.



Al combinar estas entidades, originales, aparecerán los Autores y “Pertenecen” que aunán o integran los usuarios que han creado al menos 1 post y las categorías que aparecen en los posts, respectivamente.

Una vez tengo el diagrama, me he ido al gestor de bases de datos, que en este caso es PHPMyAdmin, a través de Xampp, y entrando por localhost.

Desde ahí he creado para empezar la base de datos con la siguiente instrucción:

```
CREATE DATABASE IF NOT EXISTS `ARGAMING_DB` CHARACTER SET utf8mb4;
```

(La codificación utf8mb4 es muy utilizada en la actualidad y permite ‘ñ’ y tildes)

A continuación he ido creando cada tabla con sus respectivos campos, y claves primarias y foráneas. Todas las tablas, detalladas, quedarían así:

Usuarios

- Id_usuario: tipo entero, es la clave primaria de la tabla usuarios, no puede ser nulo y tiene como máximo 11 dígitos de longitud.
- Username: tipo varchar, es el nombre (o alias) de cada usuario, no permite espacios y no puede ser nulo. Es un campo imprescindible.
- Nombre: tipo varchar, es el nombre propio del usuario, sí permite espacios, pero tampoco puede ser nulo.
- Apellidos: tipo varchar, son los apellidos del usuario, no es obligatorio.
- Email: tipo varchar, campo del correo del usuario, campo imprescindible también.
- Password: tipo varchar, la contraseña para acceder, no puede estar vacío y está codificado en md5.
- Modificado: tipo timestamp, se trata de la fecha de creación del usuario, que irá variando si posteriormente se actualiza la información del mismo. Al crear un usuario toma la fecha y hora de forma automática con: *current_timestamp()*.
- Estado: es entero, por defecto es activo, con un 1 y si no, será un 0. Prácticamente un campo boolean. Está pensado para poder desactivar usuarios a modo de “banearlos”.
- Rol: campo entero, muy importante, por defecto es 0 (usuario estándar) y si se cambia desde la administración a un 1 ese usuario tendrá permisos de administrador.
- Imagen_perfil: campo varchar, muy opcional, para añadir el nombre de la imagen que aparece con cada usuario.

Post

- Id_post: tipo entero, es la clave primaria de la tabla de post, no puede ser nulo y tiene como máximo 11 dígitos de longitud.
- Id_usuario: tipo entero, es la clave foránea de la tabla usuarios, no puede ser nulo y tiene como máximo 11 dígitos de longitud. Indicará el usuario creador de cada post.
- Título: tipo varchar, es el texto que proporciona la idea de que va el post. Tampoco puede ser nulo.
- Imagen_perfil: campo varchar, un tanto opcional, para añadir el nombre de la imagen que aparece con cada post.
- Contenido: campo varchar, muy extenso (lo he limitado a 5000 caracteres), puesto que alberga la parte principal del post como es la información del mismo.
- Slug o enlace: es una especie de meta descripción que se utiliza a modo de referencia del post para los enlaces. No puede tener espacios, solo guiones entre las palabras y tampoco caracteres especiales. No puede ser nulo.
- Creado: fecha de creación del post, es tipo timestamp, no puede ser nulo y se registra por defecto con la fecha en la que crea.
- Modificado: última fecha en la que se ha modificado el post, es tipo timestamp, no puede ser nulo y se actualiza por defecto con la fecha en la que crea. Para ello se incluye al final esta expresión:

```
 `modificado` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
```

- Visitas: tipo entero, puede ser nulo y coge por defecto el valor 1, simbólicamente. Es la cifra que va recogiendo las veces que se muestra o abre un post.
- Estado: como pasaba con el usuario, es un entero (por defecto 1) que indica la disponibilidad del post y que en un caso concreto podría pasarse a 0 y evitar que apareciese listado.

Comentarios

- Id_comentario: es de tipo entero, es la clave primaria de la tabla de comentarios, no puede ser nulo y tiene como máximo 11 dígitos de longitud.
- Id_post: tipo entero, es la clave foránea de la tabla de post, no puede ser nulo y tiene como máximo 11 dígitos de longitud.
- Id_usuario: tipo entero, es la clave foránea de la tabla usuarios, no puede ser nulo y tiene como máximo 11 dígitos de longitud.
- Texto: tipo varchar, no puede ser nulo y con un máximo de caracteres en 500.
- Creado: fecha de creación del post, es tipo timestamp, no puede ser nulo y se registra por defecto con la fecha en la que crea.

Categorías

- Id_categoria: tipo entero, es la clave primaria de la tabla de categorías, no puede ser nulo y tiene como máximo 11 dígitos de longitud.
- Nombre: tipo varchar, se refiere al nombre de la categoría y tampoco puede ser nulo.
- Descripción: tipo varchar no nulo que incluye una breve descripción de la categoría.
- Slug o enlace: es una especie de meta descripción que se utiliza a modo de referencia del post para los enlaces. No puede tener espacios, solo guiones entre las palabras y tampoco caracteres especiales. No puede ser nulo.
- Imagen_categoria: tipo varchar, es opcional y consiste en el nombre de la imagen que se le quiera asignar a la categoría.

Pertenecen

- Id_post: tipo entero, es la clave foránea de la tabla de post, no puede ser nulo y tiene como máximo 11 dígitos de longitud.
- Id_categoria: tipo entero, es la clave foránea de la tabla de categorías, no puede ser nulo y tiene como máximo 11 dígitos de longitud.

Algunas de los campos establecidos, además de valores por defecto, aumentan de forma automática, es lo que se denomina auto-incremento:

```
-- AUTO_INCREMENT de la tabla `categorias`  
--  
ALTER TABLE `categorias`  
    MODIFY `id_categoria` int(11) NOT NULL AUTO_INCREMENT;  
  
--  
-- AUTO_INCREMENT de la tabla `comentarios`  
--  
ALTER TABLE `comentarios`  
    MODIFY `id_comentario` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;  
  
--  
-- AUTO_INCREMENT de la tabla `post`  
--  
ALTER TABLE `post`  
    MODIFY `id_post` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=12;
```

Se encuentran en todas las tablas, puesto que los id de cada tabla (post, usuarios, categorías...) funcionan de esta manera, al añadir un registro nuevo aumentan el id respecto del último registro de la tabla.

Las claves primarias en otras tablas relacionadas, pasan a ser foráneas, y las he añadido con esta orden:

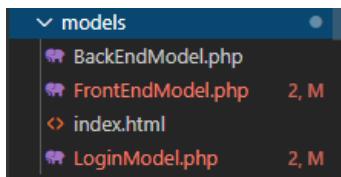
```
ALTER TABLE `post`  
    ADD CONSTRAINT `post_ibfk_1` FOREIGN KEY (`id_usuario`) REFERENCES `usuarios` (`id_usuario`);  
COMMIT;
```

DESARROLLO DEL PROYECTO

En el apartado de CodeIgniter anterior, he explicado mediante un ejemplo como procesa la información esta framework para que llegue a visualizarla el usuario desde que accede a la página inicial. Pero ese solo es un breve proceso de los muchos que se llevan a cabo, ahora procederé a detallar el resto y a explicar más en profundidad, principalmente, los controladores, modelos y vistas que he creado, y por qué.

Como lo denominamos MVC (Modelo Vista-Controlador), seguiré ese orden y empezaré mostrando los Modelos que he diseñado y por qué funciones se componen, algunas las enseñé y nombré antes pero ahora las repasaré más en detalle.

MODELOS/MODELS



Como dije antes, el modelo está compuesto por todo lo referente a la conexión con base de datos y a gestión de la información, en ambos sentidos, tanto de la base de datos a los usuarios, como al revés.

Mi idea ha sido siempre separar la parte del FrontEnd del BackEnd y es lo que he procurado al crear estos modelos y diferenciar sus funciones.

Hay elementos comunes para todos los modelos, desde el constructor que conecta con base de datos a través del archivo database.php que nombré y enseñé antes, hasta las funciones `ExecuteArrayResults()` y `ExecuteResultsParamsArray()` que se encargan de devolver un array con los resultados (su diferencia es a la hora de tratar los parámetros pasados, simplemente). Y, por descontado, una función `Execute()` para ejecutar cualquier consulta, sin más.

```
# Ejecuta consultas y devuelte los resultados en un array
public function ExecuteArrayResults( $sql)
{
    $query = $this->db->query( $sql);
    $rows = $query->result_array();
    $query->free_result();

    return ( $rows);
}

# Ejecuta consultas, con los valores pasados por separado,
# y devuelte los resultados en un array
public function ExecuteResultsParamsArray( $sql, $params)
{
    $query = $this->db->query( $sql, $params);
    $rows[ 'data' ] = $query->result_array();
    $query->free_result();

    return ( $rows);
}

public function Execute( $sql)
{
    $this->db->query( $sql);
}
```

Empezaré por el [LoginModel](#), es el modelo más simple, solo se utiliza para verificar si el usuario está registrado y si en ese caso, las credenciales son correctas.

```
    # Método para validar el email y contraseña que nos han pasado desde el formulario
    public function comprobar_usuario( $datos )
    {
        $sql = "SELECT * FROM usuarios WHERE username = ? AND password = ? ";
        $params = array( $datos['username'],$datos['password'] );

        return ( $this->ExecuteResultsParamsArray( $sql, $params ) );
    }
```

Otro sería el [BackEndModel](#), que sí contiene bastantes más funciones a parte de las típicas y las tres por defecto que acabo de comentar.

```
public function ListarPost( $post_id )
{
    $sql = "SELECT * FROM post WHERE id_post = ?";
    $params = array( $post_id );
    return ( $this->ExecuteResultsParamsArray( $sql, $params ) );
}

public function Listado_post_y_usuarios()
{
    $sql = "SELECT post.id_post,post.id_usuario,post.titulo,post.imagen_post,post.contenido,post.slug,post.creado,post.modificado,post.visitas,post.estado,
    usuarios.username
    FROM post,usuarios
    WHERE post.id_usuario = usuarios.id_usuario";

    return ( $this->ExecuteArrayResults( $sql ) );
}

public function Filtrar_post($filtro,$orden = null)
{
    $sql = "SELECT post.id_post,post.id_usuario,post.titulo,post.imagen_post,post.contenido,post.slug,post.creado,post.modificado,post.visitas,post.estado,
    usuarios.username
    FROM post,usuarios
    WHERE post.id_usuario = usuarios.id_usuario
    ORDER BY $filtro $orden ";
    return ( $this->ExecuteArrayResults( $sql ) );
}

# Método para mostrar todos los comentarios que existen en un post, buscando la coincidencia con usuarios
public function Listado_comentarios_post_y_usuarios($filtro = null, $orden = null)
{
    $sql = "SELECT comentarios.id_comentario,comentarios.id_post,comentarios.id_usuario,comentarios.texto,comentarios.creado,
    usuarios.id_usuario,usuarios.username,
    post.id_post,post.titulo,post.slug
    FROM comentarios,usuarios,post
    WHERE comentarios.id_usuario = usuarios.id_usuario and comentarios.id_post = post.id_post
    ORDER BY $filtro $orden ";

    return ( $this->ExecuteArrayResults( $sql ) );
}

public function ListarUsuario($usuario_id)
{
    $sql = "SELECT * FROM usuarios WHERE id_usuario = ?";
    $params = array( $usuario_id );
    return ( $this->ExecuteResultsParamsArray( $sql, $params ) );
}
```

Funciones del BackEndModel:

- ListarPost(\$post_id): devuelve el registro de post con el id que se le haya pasado.
- Listado_post_y_usuarios(): devuelve todos los registros relacionados con post y los datos del usuario que lo ha creado.
- Filtrar_post(\$filtro, \$orden = null): se encarga de hacer exactamente lo mismo que la anterior pero dándole un orden concreto en función del campo que se pida filtrar mediante ‘order by’. Ésta se aplica en las tablas de listados para ordendar de forma ascendente o descendente.



```
//Función que permite buscar coincidencias en la tabla de usuarios mediante las expresiones regulares
public function busqueda_usuarios($campo)
{
    $sql = "SELECT * FROM usuarios
    WHERE username REGEXP '^.*' . $campo . '.*$' or
    nombre REGEXP '^.*' . $campo . '.*$' or
    apellidos REGEXP '^.*' . $campo . '.*$' or
    email REGEXP '^.*' . $campo . '.*$'";
    /*$sql = "SELECT * FROM usuarios WHERE nombre ='" . $campo . "' or username ='" . $campo . "'";*/
    return ($this->ExecuteArrayResults($sql));
}

//Función que permite buscar coincidencias en la tabla de post (vinculada a la de usuarios) mediante las expresiones regulares
public function busqueda_post($campo)
{
    $sql = "SELECT post.id_post,post.id_usuario,post.titulo,post.imagen_post,
    post.contenido,post.slug,post.creado,post.modificado,post.visitas,post.estado,
    usuarios.username,usuarios.id_usuario
    FROM post,usuarios
    WHERE post.id_usuario = usuarios.id_usuario and
    (usuarios.username REGEXP '^.*' . $campo . '.*$' or
    post.titulo REGEXP '^.*' . $campo . '.*$' or
    post.contenido REGEXP '^.*' . $campo . '.*$' or
    post.slug REGEXP '^.*' . $campo . '.*$' or
    post.visitas REGEXP '^.*' . $campo . '.*$')");
    return ($this->ExecuteArrayResults($sql));
}

//Función que permite buscar coincidencias en la tabla de comentarios (vinculada a la de usuarios y post) mediante las expresiones regulares
public function busqueda_comentario($campo)
{
    $sql = "SELECT comentarios.id_comentario,comentarios.id_post,comentarios.id_usuario,comentarios.texto,comentarios.creado,
    usuarios.id_usuario,usuarios.username,
    post.id_post,post.titulo,post.slug
    FROM comentarios,usuarios,post
    WHERE comentarios.id_usuario = usuarios.id_usuario and comentarios.id_post = post.id_post and
    (comentarios.texto REGEXP '^.*' . $campo . '.*$' or
    post.titulo REGEXP '^.*' . $campo . '.*$' or
    post.slug REGEXP '^.*' . $campo . '.*$' or
    usuarios.username REGEXP '^.*' . $campo . '.*$')");
    /*$sql = "SELECT * FROM usuarios WHERE nombre ='" . $campo . "' or username ='" . $campo . "'";*/
    return ($this->ExecuteArrayResults($sql));
}
```

```
/* Paginación de usuarios */

function pagination($tabla,$pag_size,$offset)
{
    $this->db->select();
    $this->db->from($tabla);
    $this->db->limit($pag_size,$offset);
    $query = $this->db->get();
    $rows = $query->result_array();
    return $rows;
}

function count($tabla)
{
    $number = $this->db->query("SELECT count(*) as number FROM $tabla")->row()->number;
    return intval($number); //devuelve en un valor entero el número de filas de la tabla
}
```

- Listado_comentarios_post_y_usuarios(\$filtro = null, \$orden = null): cumple con la misma labor que la anterior pero aplicadado a la tabla de comentarios.
- ListarUsuario(\$usuario_id): obtiene el registro del usuario que se indique a partir de su id. El valor se introduce en una consulta mediante sentencia preparada al estilo de CodeIgniter.
- Pagination(\$tabla,\$pag_size,\$offset): permite paginar a partir de los registros de la tabla que le indiquemos. He realizado una paginación de prueba con el listado de usuarios.

Después de estas funciones, aparecen 3 similares relacionadas con búsquedas a partir de expresiones regulares. Son:

- Busqueda_usuarios(\$campo): realiza una consulta que compara varios campos de la tabla de usuarios, con el valor que se le pase, mediante expresiones regulares.
- Busqueda_post(\$campo): igual pero con la tabla de post.
- Busqueda_comentarios(\$campo): de la misma manera, pero con los comentarios.

Funciones del FrontEndModel:

```

/*Lista la tabla que se le solicite y clasifica la información a partir del parámetro indicado*/
public function Lista($tabla,$clasif)
{
    $sql = "select * from ".$tabla." order by ".$clasif." desc";
    return ( $this->ExecuteArrayResults( $sql ) );
}

/*Devuelve el registro que coincide, de la tabla, y con el valor indicados*/
public function Buscar($tabla,$campo_clave,$cadena)
{
    $sql = "select * from ".$tabla." where ".$campo_clave." = '".$cadena."'";
    return ( $this->ExecuteArrayResults( $sql ) );
}

Buscar coincidencia de dos campos, el primero que no existe
en ningún otro registro y el segundo que sí coincide para que se cumpla*/
public function Buscar_campo_existente($tabla,$campo1,$valor1,$campo2,$valor2)
{
    $sql = "select * from ".$tabla." where ".$campo1." != '".$valor1."' and ".$campo2." = '".$valor2."'";
    return ( $this->ExecuteArrayResults( $sql ) );
}

# Método para mostrar un post cuando se seleccione, a través de su id
public function Listar_post($post_id)
{
    $sql = "Select * From post Where id_post = ?";
    $params = array($post_id);
    return ( $this->ExecuteResultsParamsArray( $sql, $params ) );
}

# Método para mostrar un comentario cuando se seleccione, a través de su id
public function Listar_comentario($comentario_id)
{
    $sql = "Select * From comentarios Where id_comentario = ?";
    $params = array($comentario_id);
    return ( $this->ExecuteResultsParamsArray( $sql, $params ) );
}

# Método para mostrar todos los comentarios que existen en un post, según su id
public function Listar_comentarios($post_id)
{
    $sql = "Select comentarios.id_comentario,comentarios.id_post,comentarios.id_usuario,
    comentarios.texto,comentarios.creado,usuarios.username,usuarios.email,usuarios.imagen_perfil
    From comentarios,usuarios
    WHERE comentarios.id_usuario = usuarios.id_usuario and id_post = ?";
    $params = array($post_id);
    return ( $this->ExecuteResultsParamsArray( $sql, $params ) );
}

/* Actualizar número de visitas */
public function actualizar_visitas_post( $id_post )
{
    $this->db->set('visitas', 'visitas+1', FALSE);
    $this->db->WHERE('id_post', $id_post);
    $this->db->update('post');
}

# Método para listar el número de post creados por cada usuario
public function Autores_post()
{
    $sql = "SELECT usuarios.username,usuarios.id_usuario,
    count(post.id_post) as numero_post
    FROM usuarios,post
    WHERE post.id_usuario = usuarios.id_usuario
    GROUP BY usuarios.username ";
    return ( $this->ExecuteArrayResults($sql));
}

# Método para listar los datos de los post creados por un usuario concreto
public function Datos_autor_post($id)
{
    $sql = "SELECT post.id_post,post.id_usuario,post.titulo,post.imagen_post,
    post.contenido,post.slug,post.creado,post.modificado,post.visitas,post.estado,
    usuarios.username,usuarios.id_usuario
    FROM post,usuarios
    WHERE post.id_usuario = usuarios.id_usuario AND usuarios.id_usuario = $id";
    return ( $this->ExecuteArrayResults($sql));
}

```

- Actualizar_visitas_post(\$id_post): cada vez que se entra en uno de los posts creados por usuarios, esta función actualiza el campo de visitas del post y le suma 1.
- Lista(\$tabla, \$clasif): obtiene los datos de la tabla que se le indique, clasificados a partir del campo que se le pase.
- Buscar_campo_existente(\$tabla, \$campo1, \$valor1, \$campo2, \$valor2): es una función algo más enreversada, puesto que la creé para resolver la eventualidad de que al editar la información del usuario no se pudiese cambiar el correo o nombre de usuario por uno que ya estuviese registrado. Por lo tanto, lo que hace es comparar, en la tabla que se le pida, un primer campo que no debe encontrarse con ese valor indicado y un segundo que sí debe encontrarse para devolver algo. Si coinciden ambas es porque ya existe el usuario o correo que intentamos registrar.
- Listar_post(\$post_id): obtiene el registro de un post que se le indique mediante el id del mismo.
- Listar_comentario(\$comentario_id): obtiene el registro del comentario según el id de comentario que se le indique.
- Listar_comentarios(\$post_id): saca todos los comentarios del post que se le diga mediante el id de este post.
- Autores_post() y Datos_autor_post(): las he añadido al final y han servido para proporcionar información extra sobre los usuarios que han publicado posts, ya que con esos datos he listado en una página los post de cada usuario (en su perfil) y desde otra se puede ver cada usuario cuantos posts ha creado.

CONTROLADORES/CONTROLLERS

Después de los modelos toca ver lo que hay entre medias, el engranaje que mueve este framework, como son los controladores. Se crean como una nueva clase que se extiende de la clase ya existente ‘CI_Controller’.

He creado unos cuantos, para diferenciar también las distintas funciones y usos que se dan en el blog. Empezaré por el que ya nombré y expliqué antes, encargado de las funciones que afectan a todos los usuarios por igual, el UserController.

Funciones del UserController:

```
class UserController extends CI_Controller
{
    function __construct()
    { ...
    }
    public function index()
    { ...
    }
    public function cambiar_contraseña_sin_acceso()
    { ...
    }
    public function datos_usuario()
    { ...
    }
    public function error_404()
    { ...
    }
    public function juegos()
    { ...
    }
    public function post()
    { ...
    }
    public function ver_post()
    { ...
    }
    public function autores_post()
    { ...
    }
    public function aviso_legal()
    { ...
    }
    public function política_cookies()
    { ...
    }
    public function política_privacidad()
    { ...
    }
    public function creador_blog()
    { ...
    }
    public function agregar_comentarios($datos_nuevos)
    { ...
    }
```

```
# RUTAS PARA TODOS LOS USUARIOS
$route['/'] = "UserController/index";//de todas formas, coge la ruta por defecto
$route['lista-juegos'] = "UserController/juegos";
$route['lista-post'] = "UserController/post";
$route['post/(:num)'] = "UserController/ver_post";
$route['post/(:num)/agregar-comentario'] = 'UserController/ver_post';
$route['nuevo-post'] = 'FormController/nuevo_post';
$route['responder-comentario/(:num)'] = 'FormController/responder_comentario';
$route['contacto'] = "FormController/contactar";
$route['contacto/enviar-email/(:any)'] = "EmailController/enviar/$1";
$route['creador-blog'] = "UserController/creador_blog";
$route['aviso-legal'] = "UserController/aviso_legal";
$route['política-cookies'] = "UserController/política_cookies";
$route['política-privacidad'] = "UserController/política_privacidad";
$route['cambiar-password'] = 'UserController/cambiar_contraseña_sin_acceso';
```

- [Index\(\)](#): como expliqué en el ejemplo del apartado de CodeIgniter, se encarga de lo relativo a visualizar la página de inicio al cargar la ruta por defecto o raíz para acceder al blog argaming.com

- [Cambiar contraseña sin acceso\(\)](#): como su propio nombre indica es la utilidad encargada de enviar un correo a un usuario que solicite cambiar su contraseña porque no la recuerda o no puede acceder. El mayor logro de esta función es buscar el correo solicitado y si no está en la tabla de usuarios, mostrar un error y derivar al usuario al registro, puesto que sino está ni registrado difícilmente puede haber olvidado la contraseña...

- [Error 404\(\)](#): administra lo que va a aparecer en la “típica” página de error 404, es decir, en la página que se muestra cuando no se encuentra la ruta solicitada por la razón que sea.
- [Juegos\(\)](#): función importante, vital se diría, ya que es la que obtiene los datos en formato JSON de la página videojuegos.fandom.com para mostrarlos en formato de lista. Responde a la ruta de /lista-juegos. Muestra los juegos según lo que le solicite mediante la URL a través de las etiquetas que se pueden seleccionar en cabecera.
- [Post\(\)](#): esta función devuelve el listado de post, desde la base de datos, y no solamente entra en juego la tabla de post, sino que como explicaba los coge de una consulta donde se combinan post y usuarios para poder visualizar así también que usuario ha creado cada post.
- [Ver_post\(\)](#): se trata de la función más extensa de este controlador, puesto que cumple con varias condiciones para mostrar un post en concreto: si recibe el id de un post que existe, si el usuario tiene la sesión iniciada para comentar en el post, errores derivados de la

validación de los comentarios tras introducir mal los datos, inserción del comentario en este post una vez tenga éxito, etc.

- Datos_usuario(): muestra la información del usuario seleccionado a partir del id y debajo, si los hay, también lista los post creados por éste.
- Autores_post(): lista los autores de post. Busca usuarios que hayan creado post, los lista y muestra cuantos post han hecho. Al seleccionar un usuario lleva a la función datos_usuario().
- Aviso_legal()
- Política_cookies()
- Política_privacidad()
- Creador_blog(): muestra la información sobre mí, como el creador del blog.

Funciones del AdminController:

```
# ADMIN
$route['admin'] = 'AdminController/index';
$route['admin/inicio'] = 'AdminController/index';
$route['admin/perfil-admin'] = 'AdminController/perfil_admin';
$route['admin/panel-control'] = 'AdminController/panel_control';

# Admin - juegos
$route['admin/panel-control/juegos'] = 'AdminController/listado_juegos';

# Admin - usuarios
$route['admin/panel-control/usuarios'] = 'AdminController/listado_usuarios';
$route['admin/panel-control/usuarios/ordenar-username/desc'] = 'AdminController/listado_usuarios';
$route['admin/panel-control/usuarios/ordenar-username/asc'] = 'AdminController/listado_usuarios';
$route['admin/panel-control/usuarios/ordenar-nombre/desc'] = 'AdminController/listado_usuarios';
$route['admin/panel-control/usuarios/ordenar-nombre/asc'] = 'AdminController/listado_usuarios';
$route['admin/panel-control/usuarios/ordenar-modificado/desc'] = 'AdminController/listado_usuarios';
$route['admin/panel-control/usuarios/ordenar-modificado/asc'] = 'AdminController/listado_usuarios';
$route['admin/buscar-usuario'] = 'AdminController/listado_usuarios';
$route['admin/nuevo-usuario'] = 'FormController/registro_admin';
$route['admin/registrar-usuario'] = 'AdminController/registrar_nuevo_usuario';
$route['admin/editar-usuario/(:num)'] = 'FormController/editar_admin';
$route['actualizar-usuario'] = 'AdminController/actualizar_usuario';
$route['admin/eliminar-usuario/(:num)'] = 'AdminController/eliminar_usuario';

# Admin - post
$route['admin/panel-control/post'] = 'AdminController/listado_post';
$route['admin/panel-control/post/ordenar-creado/desc'] = 'AdminController/listado_post';
$route['admin/panel-control/post/ordenar-creado/asc'] = 'AdminController/listado_post';
$route['admin/panel-control/post/ordenar-modificado/desc'] = 'AdminController/listado_post';
$route['admin/panel-control/post/ordenar-modificado/asc'] = 'AdminController/listado_post';
$route['admin/panel-control/post/ordenar-visitas/desc'] = 'AdminController/listado_post';
$route['admin/panel-control/post/ordenar-visitas/asc'] = 'AdminController/listado_post';
$route['admin/buscar-post'] = 'AdminController/listado_post';
$route['admin/nuevo-post'] = "FormController/nuevo_post_admin";
$route['admin/crear-post'] = 'AdminController/registrar_nuevo_post';
$route['admin/editar-post/(:num)'] = 'FormController/editar_post_admin';
$route['actualizar-post'] = 'AdminController/actualizar_post';
$route['admin/eliminar-post/(:num)'] = 'AdminController/eliminar_post';

# Admin - comentarios
$route['admin/panel-control/comentarios'] = 'AdminController/listado_comentarios';
$route['admin/panel-control/comentarios/ordenar-creado/asc'] = 'AdminController/listado_comentarios';
$route['admin/panel-control/comentarios/ordenar-creado/desc'] = 'AdminController/listado_comentarios';
$route['admin/buscar-comentario'] = 'AdminController/listado_comentarios';
$route['admin/nuevo-comentario'] = "FormController/nuevo_comentario_admin";
$route['admin/registrar-comentario'] = 'AdminController/registrar_comentario';
$route['admin/responder-comentario/(:num)'] = 'FormController/responder_comentario_admin';
$route['actualizar-comentario'] = 'AdminController/actualizar_comentario';
$route['admin/eliminar-comentario/(:num)'] = 'AdminController/eliminar_comentario';
```

- Inicio(): se encarga de mostrar la página principal para el administrador. Previamente inicia su sesión y después se le redirige a aquí. La página solo varía en su cabecera, ya que la normal muestra perfil y cerrar sesión, y si es administrador además le da la posibilidad de acceder al panel de control.

- Panel_control(): si elige la opción de panel de control, llega a esta función que llama a la vista pertinente y muestra esta página de administración.
- Perfil_admin(): igual que el perfil de usuario normal (está en el LoginController), pero en este caso para el usuario con rol de administrador.
- Listado_juegos(): no es que sea similar a la de ‘juegos()’ que nombraba antes en UserController, es que es lo mismo exactamente. Muestra los juegos según lo que le solicite mediante la URL a través de las etiquetas que se pueden seleccionar en cabecera.
- Listado_usuarios(): genera una tabla con toda la información de los usuarios registrados en el blog.
- Eliminar_usuario(): lleva a cabo la eliminación del usuario que se le pida mediante el id. Al hacerse por ruta, en la url, primero se comprueba que el usuario que lo solicita está registrado y es administrador, de lo contrario no sería posible. Esta función se ayuda de un pequeño código de JavaScript para llegar hasta aquí con el id de usuario.
- Listado_post(): genera una tabla con toda la información sobre los post del blog.
- Eliminar_post(): lleva a cabo la eliminación del post que se le pida mediante el id. Por lo demás funciona de la misma forma que la de eliminar usuarios.
- Listado_comentarios(): genera una tabla con toda la información sobre los comentarios en los post del blog.
- Eliminar_comentario(): lleva a cabo la eliminación del comentario que se le pida mediante el id. Por lo demás funciona de la misma forma que la de eliminar usuarios y post.

Funciones de LoginController:

- Login(): esta función

```
class LoginController extends CI_Controller
{
    function __construct()
    {
        ...
    }
    public function login()
    {
        ...
    }
    public function login2()
    {
        ...
    }
    public function inicio_logueado()
    {
        ...
    }
    public function perfil_usuario()
    {
        ...
    }
    public function actualizar_usuario()
    {
        ...
    }
    public function cerrar_sesion()
    {
        ...
    }
}
```

```
# LOGIN
$route['login'] = 'LoginController/login';
$route['login2'] = 'LoginController/login2';
$route['login/error'] = 'LoginController/login';
$route['login/no-registrado'] = 'LoginController/login';
$route['registro'] = "FormController/registro";
$route['inicio'] = 'LoginController/inicio_logueado';

# USUARIO
$route['perfil-usuario'] = 'LoginController/perfil_usuario';
$route['usuario/editar-perfil/(:num)'] = 'FormController/editar_perfil';
$route['usuario/cambiar-password/(:num)'] = 'FormController/cambiar_contraseña';
$route['usuario/cerrar-sesion'] = 'LoginController/cerrar_sesion';
```

permite visualizar la página de inicio de sesión, mediante el ly_login.php, diseñado en especial para esta parte. Si detecta errores mediante la url, los muestra.

- Login2(): es a donde se redirige después de escribir el usuario y contraseña en el inicio de sesión, comprueba estos datos y si hay cualquier incoherencia devuelve una ruta a modo de error. En caso contrario, se inicia sesión almacenando los datos básicos del usuario. Además, si todo es correcto redirige a la página de inicio pero mostrando la sesión iniciada.

- Inicio_logueado(): comprueba si el usuario ya estaba logueado y le manda a la página de inicio en todo caso.
- Perfil_usuario(): muestra los datos de sesión del usuario en el típico formato de perfil.
- Actualizar_usuario(): función que creé con la idea de redirigir en momento puntuales, los datos para acabar de actualizar usuarios o post; sin embargo, no he llegado a ponerla en uso finalmente.
- Cerrar_sesión(): como indica, acaba con la sesión del usuario, borra todos los datos y devuelve a la página de inicio (con la cabecera original, de acceso / registro).

Funciones de EmailController:

```

class EmailController extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
    }

    public function enviar()
    {
        /* Configuración para mailtrap.io */
        $config = Array(
            'protocol' => 'smtp',
            'smtp_host' => 'smtp.mailtrap.io',
            'smtp_port' => 2525,
            'smtp_user' => '46fff8744270f6b',
            'smtp_pass' => 'b39367f2273361',
            'charset' => 'utf-8',
            'crlf' => "\r\n",
            'newline' => "\r\n"
        );

        $this->email->initialize($config);

        $datos_email = array();

        $url = urldecode($this->uri->segment(3)); //se extrae la parte de la url con los datos

        $array_url = explode('&', $url); //se separan los datos a partir del separador insertado previamente y se dejan en forma de array

        debug($array_url);

        $long_array = count($array_url); //se guarda la longitud del array generado a partir de la url

        //si esta longitud vale 5 es porque contiene el teléfono y sino no lo tiene
        $long_array == 5 ? $campos_email = ['username', 'email', 'phone', 'asunto', 'mensaje'] : $campos_email = ['username', 'email', 'asunto', 'mensaje'];

        //Se asignan los diferentes segmentos de URL a cada valor del array, el primero, el número 3 será el nombre de usuario y le corresponde a 'username', por ejemplo.
        for($i = 0; $i < $long_array; $i++) {
            if($campos_email[$i] != 'email'){
                strpos($array_url[$i], '-') !== null ? ($cad1 = str_replace('-', ' ', $array_url[$i])) : ($cad1 = $array_url[$i]);
                strpos($array_url[$i], 'br') !== null ? ($cad2 = str_replace('br', PHP_EOL, $cad1)) : ($cad2 = $cad1);
                $datos_email [$campos_email[$i]] = $cad2;
            } else {//si se trata del email, se salta el filtro, porque no puede contener saltos de línea y sí guiones que no habría que sustituirlos por nada..
                $datos_email [$campos_email[$i]] = $array_url[$i];
            }
        }
    }

    isset($datos_email['phone']) ? $datos_email['mensaje'] = $datos_email['mensaje'].PHP_EOL.PHP_EOL.'Teléfono de contacto: '.$datos_email['phone'] : '';
    debug($datos_email);

    //Se estructuran las partes de la clase email
    $this->email->from($datos_email['email'], $datos_email['username']);
    $this->email->to('info@argaming.com');
    $this->email->cc('antonir96@gmail.com');
    $this->email->bcc('them@their-example.com');

    $this->email->subject($datos_email['asunto']);
    $this->email->message($datos_email['mensaje']);
    //Y se ejecuta, es decir se manda, con una respuesta positiva si se envía con éxito, o negativa si da error al enviarlo
    if($this->email->send()){
        $mensaje = '<p class="alert alert-success alert-dismissible">
                    <button type="button" class="close" data-dismiss="alert">&times;</button>El correo ha sido enviado con éxito</p>';
    } else $mensaje = '<p class="alert alert-danger alert-dismissible">
                    <button type="button" class="close" data-dismiss="alert">&times;</button>El correo no ha podido ser enviado</p>';

    //Se carga la página de contacto, con un "mínimo" de opciones, se pasa el mensaje de que el correo ha sido enviado o no y después se refresca para devolver a la ventana original
    $this->load->helper(array('form', 'url'));

    $this->load->library('form_validation');

    $datos = array(
        'mensaje' => $mensaje
    );

    $vista = array(
        'vista' => 'web/contacto.php',
        'params' => $datos,
        'layout' => 'ly_contacto.php',
        'titulo' => 'Contactar',
    );

    $this->layouts->view($vista);
    header("refresh:10;url=/contacto");
}

```

Este controlador solo tiene dos funciones, `enviar()` para enviar un correo con la información introducida en el formulario de contacto y otra para enviar correos automáticos a modo de solicitud para borrar el perfil de usuario, `solicitar_borrado()`. Se basa en la librería de CodeIgniter `email`. Carga la configuración del servidor que se le asigne al principio, en mi caso lo he hecho con el servidor para testeo de mailtrap.io (recomendado por el tutor) y ha funcionado a las mil maravillas. Con Gmail lo intenté y no hubo tanto éxito.

El formato se lo he dado desde la función `enviar()` y el orden de todos los argumentos, que tras muchos intentos, errores y problemas varios conseguí que apareciese como debía.

Aquí se puede observar el mensaje, enviado desde contacto, en mi buzón de entrada del mailtrap.io:

Prueba

From: Prueba2 <pruebas@gmail.com>
To: <info@argaming.com>
Message-Id: 5ee4038af3d53@gmail.com
Date: Sat, 13 Jun 2020 00:36:58 +0200
X-Mailer: CodeIgniter
Cc: antonirg96@gmail.com
Reply-To: pruebas@gmail.com
Return-Path: pruebas@gmail.com
Bcc: **⚠ Upgrade your plan to get access to this feature**

[Hide Info](#)

HTML HTML Source Text Raw Analysis SMTP info

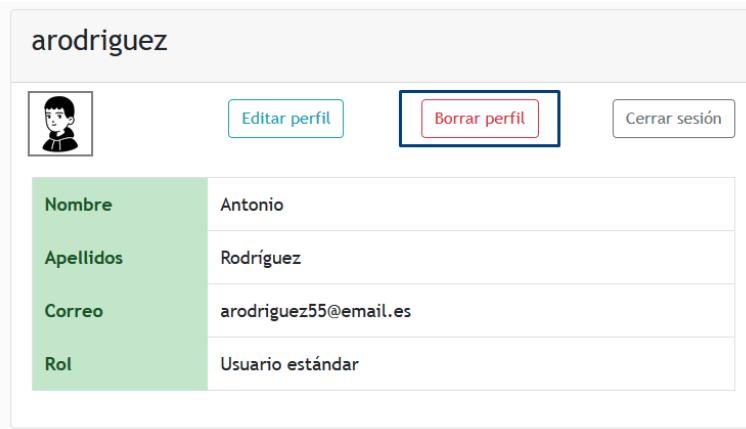
Hola, esto es un mensaje de prueba para mostrarlo en forma de debug en el proyecto,

Saludos!

Teléfono de contacto: 612345678

La segunda función, al igual que la primera, necesita la configuración smtp de mailtrap.io para cargar la librería correctamente.

Esta función se utiliza para solicitar que se borre la cuenta de un usuario desde su perfil, a través de una enlace que manda el id (después de asegurarse de esta decisión mediante un pop-up con javascript).



Después de inicializar la configuración, se recoge el valor recibido por la url y se utiliza, a modo de id, para buscar al usuario correspondiente en la tabla de usuarios. Una vez devuelve el registro mandamos el correo con la estructura preformateada al administrador. En esa estructura van incluidos el id del usuario, el nombre, los apellidos y su correo para que el administrador pueda buscarlo y finalmente eliminar sus datos de acuerdo con la decisión del usuario

```

public function solicitar_borrado()
{
    /* Configuración para mailtrap.io */
    $config = array(
        'protocol' => 'smtp',
        'smtp_host' => 'smtp.mailtrap.io',
        'smtp_port' => 2525,
        'smtp_user' => '46ff8744270f6b',
        'smtp_pass' => 'b39367f2273361',
        'charset' => 'utf-8',
        'crlf' => "\r\n",
        'newline' => "\r\n"
    );

    $this->email->initialize($config);

    $datos_email = array();

    $id = $this->uri->segment(3); //se extrae la parte de la url con los datos

    $datos_usuario = $this->FrontEndModel->Buscar('usuarios', 'id_usuario', $id);

    //debug($datos_usuario);

    $mensaje = 'Aviso automático. Solicito la eliminación de mi cuenta de usuario.' . PHP_EOL . 'Datos de la cuenta:' . PHP_EOL .
        'ID: ' . $datos_usuario[0]['id_usuario'] . PHP_EOL . 'Nombre: ' . $datos_usuario[0]['nombre'] . PHP_EOL . 'Correo: ' . $datos_usuario[0]['email'];

    $datos_email = array(
        'email' => $datos_usuario[0]['email'],
        'username' => $datos_usuario[0]['username'],
        'asunto' => 'Eliminación de perfil',
        'mensaje' => $mensaje
    );

    //debug($datos_email);
    //Se estructuran las partes de la clase email
    $this->email->from($datos_email['email'], $datos_email['username']);
    $this->email->to('info@argaming.com');
    $this->email->cc('antonirg96@gmail.com');
    $this->email->bcc('them@their-example.com');

    $this->email->subject($datos_email['asunto']);
    $this->email->message($datos_email['mensaje']);

    //Y se ejecuta, es decir se manda, con una respuesta positiva si se envía con éxito, o negativa si da error al enviarlo
    if ($this->email->send()) {
        $mensaje = '<p class="alert alert-success alert-dismissible">
            <button type="button" class="close" data-dismiss="alert">&times;</button>La solicitud ha sido enviada con éxito</p>';
    }
}

```

En la bandeja de entrada se recibe algo similar a esto:

Eliminación de perfil	
To: <info@argaming.com>	a day ago
Prueba	
To: <info@argaming.com>	4 days ago
Hola	
To: <info@argaming.com>	9 days ago
Hola	
To: <info@argaming.com>	9 days ago
Buenos días	
To: <info@argaming.com>	9 days ago
ok	
To: <nicolauacamacho@hotmail.fr>	9 days ago
Mail sent by smtp.mailtrap.io: successful	
To: <nicolauacamacho@hotmail.fr>	9 days ago
Ultima prueba con mensaje	
To: <info@argaming.com>	10 days ago

Eliminación de perfil

From: arodriguez <arodriguez55@email.es>
 To: <info@argaming.com>
 Message-ID: See7ba5cb1d8@email.es
 Date: Mon, 15 Jun 2020 20:13:48 +0200
 X-Mailer: Codeigniter
 Cc: antonirg96@gmail.com
 Reply-To: arodriguez55@email.es
 Return-Path: arodriguez55@email.es
 Bcc: **⚠ Upgrade your plan** to get access to this feature

[Hide Info](#)

[HTML](#) [HTML Source](#) [Text](#) [Raw](#) [Analysis](#) [SMTP Info](#)

Aviso automático. Solicito la eliminación de mi cuenta de usuario.
 Datos de la cuenta:
 ID: 2
 Nombre: Antonio
 Correo: arodriguez55@email.es

Funciones del FormController:

Todas las funciones tienen en común que trabajan mediante la librería de validación de CodeIgniter, llamada `form_validation`.

```
$this->load->helper(array('form', 'url'));

$this->load->library('form_validation'); //llamamos a las reglas de validación
```

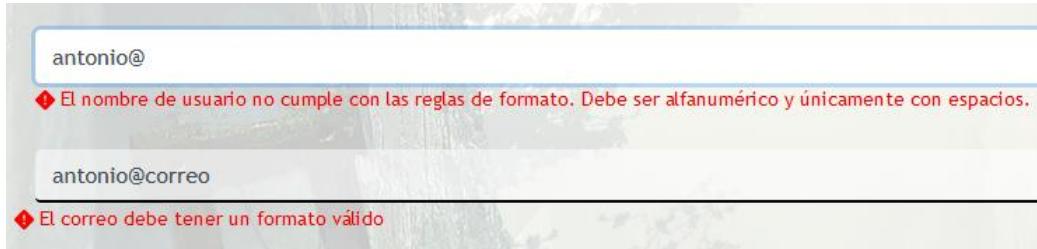
El proceso es el mismo en todas las funciones que utilizan la librería de validación. Tras cargar la librería, es necesario aplicarle las reglas de configuración con ‘set_rules’, éstas se le pasan mediante arrays de arrays y se corresponden con cada campo del formulario que vayamos a procesar (nombre de usuario, correo, mensaje, etc.). La validación del formulario de contacto:

```
$config = array(
    array(
        'field' => 'username',
        'label' => 'nombre de usuario',
        'rules' => 'trim|required|min_length[2]|max_length[30]|regex_match/[^\0-9A-ZáéíóúñÁÉÍÓÚñáéíóúñ\s]+$/',
        'errors' => array(
            'required' => 'El %s no debe contener espacios, solo puede llevar \'_\' .',
            'min_length' => 'El %s debe tener al menos 2 caracteres de longitud',
            'max_length' => 'El %s debe tener, como mucho, 30 caracteres de longitud',
            'regex_match' => 'El %s no cumple con las reglas de formato. Debe ser alfanumérico y únicamente con espacios.'
        ),
    ),
    array(
        'field' => 'email',
        'label' => 'correo',
        'rules' => 'trim|required|valid_email',
        'errors' => array(
            'required' => 'El %s es obligatorio',
            'valid_email' => 'El %s debe tener un formato válido'
        ),
    ),
    array( ... ),
    array( ... ),
    array( ... ),
    array( ... )
);

$this->form_validation->set_rules($config);

if ($this->form_validation->run() == TRUE) {
```

Para mostrar los errores en español yo mismo fui traduciendo los originales. Según las reglas aplicadas en cada caso, muestra cada error de forma personalizada. Por ejemplo, si no introducimos un formato de nombre del usuario válido nos dice esto:



He escrito el nombre y el correo mal para que se puedan observar los errores. Si se dejan vacíos, al ser campos ‘required’ nos dice que son obligatorios. Y también tienen que tener entre una longitud mínima y una máxima.

Cuando todos los campos obligatorios sean correctos se procede a realizar la acción estimada para una validación positiva. En este caso, sería enviar el correo con los datos introducidos, como ya se vió en el ejemplo anterior del email.

Y así, básicamente, funciona con todos los formularios. El resto de las funciones (y las rutas de acceso) de este controlador serían las siguientes:

```
class FormController extends CI_Controller
{
    function __construct()
    {
        ...
    }

    public function registro()
    {
        ...
    }

    public function editar_perfil()
    {
        ...
    }

    public function nuevo_post()
    {
        ...
    }

    public function contactar()
    {
        ...
    }

    public function cambiar_contraseña()
    {
        ...
    }

    public function mensajes_vista($tipo, $texto, $ruta)
    {
    }

    /*-----Apartado para las funciones que abarca el administrador
    public function registro_admin()
    {
        ...
    }

    public function editar_admin()
    {
        ...
    }

    public function nuevo_post_admin()
    {
        ...
    }

    public function editar_post_admin()
    {
        ...
    }
}

$route['nuevo-post'] = 'FormController/nuevo_post';
$route['responder-comentario/(:num)'] = 'FormController/responder_comentario';
$route['contacto'] = "FormController/contactar";
$route['registro'] = "FormController/registro";

$route['usuario/editar-perfil/(:num)'] = 'FormController/editar_perfil';
$route['usuario/cambiar-password/(:num)'] = 'FormController/cambiar_contraseña';

$route['admin/nuevo-usuario'] = "FormController/registro_admin";
$route['admin/editar-usuario/(:num)'] = 'FormController/editar_admin';
$route['admin/nuevo-post'] = "FormController/nuevo_post_admin";
$route['admin/editar-post/(:num)'] = 'FormController/editar_post_admin';
```

- [Registro\(\)](#): carga el formulario de registro para los nuevos usuarios desde la parte pública. Pide nombre de usuario, nombre y apellidos, correo del usuario, contraseña con confirmación y aceptar las políticas de privacidad y cookies. La imagen de usuario es opcional y se puede añadir más tarde.
- [Editar_perfil\(\)](#): permite, una vez se registra y loguea un usuario, ver sus datos del perfil y modificarlos. Al igual que el registro, no permite “tocarlo” todo, solo lo que está al alcance de los usuarios estándar, el resto de opciones le pertenecen al administrador.
- [Nuevo_post\(\)](#): muestra el formulario para crear un nuevo post en el blog. Contiene los campos de título del post, meta descripción, creador (selecciona automáticamente al usuario que tiene la sesión iniciada para crearlo) y contenido del post. Además, se puede elegir una imagen para ponerla en la cabecera del post.
- [Contacto\(\)](#): (explicada en el ejemplo).
- [Cambiar_contraseña\(\)](#): a la hora de editar el perfil del usuario también debe existir la posibilidad de cambiar la contraseña, por ello lo separé y en esta función es a donde se manda para cambiarla. Es un formulario que valida dos campos, nueva contraseña y confirmación de contraseña. Por seguridad, para evitar posibles hackeos, se restringe de tal forma que cada usuario solo puede cambiar su propia contraseña (tienen que coincidir el id de ese usuario y un hash de la contraseña, que se guardan al iniciar sesión a modo de token de seguridad), y el usuario administrador es el único que puede cambiar cualquier una de las contraseñas de usuarios registrados.

La idea, además, es que también se pueda acceder al pedir cambiar la contraseña si la hemos olvidado, con un enlace que reciba en el correo el usuario. Esta funcionalidad no está implementada, pero sin duda es viable.

- Mensajes vista(): se trata de una función que tenía en mente para mostrar mensajes positivos o negativos según las diferentes acciones que se pueden realizar en el blog. Finalmente, al no ser importante, lo he dejado sin implementar también.

Y ahora empezamos con las funciones que abarcan al usuario administrador exclusivamente. Lo principal es comprobar antes de nada que el usuario está logueado y con rol ‘1’ o de ‘administrador’. Una vez es así, permite acceder a diversas opciones de las tablas de usuarios, post y comentarios como son crear nuevos registros, modificar los existentes o borrarlos.

- Registro admin(): igual que el registro del nuevo usuario descrito antes, pero esta vez con todos los campos que tienen los usuarios en base de datos al alcance del administrador. Es decir, al crear el usuario podría elegir que tuviese el rol de administrador o que apareciese como inactivo/bloqueado.
- Editar admin(): y de la misma manera se pueden editar todos los usuarios con esta función que solo restringe la edición de un campo del formulario y es la fecha de modificación del usuario, ya que es de actualización automática; como comenté en el apartado que describía los campos de la tabla usuarios.
- Nuevo post admin(): formulario de creación de nuevos post en la página, no tiene nada de especial, excepto el hecho de que permite seleccionar cualquier usuario registrado para asignarle la autoría o creación del post.
- Editar post admin(): igual que la edición de usuarios pero con post. Se comprueba la existencia previa del mismo título o meta descripción para evitar posibles conflictos.

Me gustaría añadir, como detalle que no nombré antes en el controlador de administración, y es que en las tablas o listados de usuarios, post y comentarios que posee la interfaz del administrador, he implementado un sistema de búsqueda mediante expresiones regulares (que sí apareció en mis capturas sobre bases de datos y los modelos que había desarrollado).

Estas búsquedas consisten en un campo, un input “normal”, que manda la cadena que queremos buscar al AdminController y desde ahí, según la función que sea, se encarga de pasarlala por base de datos y devuelve las posibles coincidencias. Muestro un ejemplo con capturas:

Listado de usuarios										
ID	Foto	Username	Email	Nombre	Apellidos	Modificado	Admin	Activo	Editar	Eliminar
1		anrogo	anrogo@email.es	Antonio	Rodríguez González	2020-06-04 23:44:38	Sí			
2		arodriguez	arodriguez55@email.es	Antonio	Rodríguez	2020-06-03 18:24:06	No			
4		jorge_1	jorge-1@gmail.com	Jorge	García Torres	2020-06-02 19:59:02	No			
6		Federico_Ibañez	federico_ibaez@gmail.com	Federico	Ibañez	2020-05-31 11:36:33	No			
10		Destroyer	alfre_destro@gmail.com	Alfredo	Martinez De los Cobos	2020-06-07 13:44:55	No			
11		anrogo2	anrogo2@email.es	Antonio	Rodríguez González	2020-06-09 09:30:34	Sí			
12		Naaraa	arodriguez55@gmail.es	Arancha	Rodríguez González	2020-06-04 23:42:34	Sí			

Si escribo, por ejemplo “anto” refiriéndome a Antonio, al tener varios los lista rápidamente:

Listado de usuarios										
ID	Foto	Username 🕒🕒	Email	Nombre 🕒🕒	Apellidos	Modificado 🕒🕒	Admin	Activo	Editar	Eliminar
1		anrogo	anrogo@email.es	Antonio	Rodríguez González	2020-06-04 23:44:38	Sí			
2		arodriguez	arodriguez55@email.es	Antonio	Rodríguez	2020-06-03 18:24:06	No			
11		anrogo2	anrogo2@email.es	Antonio	Rodríguez González	2020-06-09 09:30:34	Sí			
14		antonio	antonio@email.es	antonio	alferez	2020-06-09 21:11:28	No			

Si me equivoco, y escribo un carácter especial que no coincide con nada (y no es @ ni -) no encontrará nada y se lo comunica al administrador:

Listado de usuarios										
ID	Foto	Username	Email	Nombre	Apellidos	Modificado	Admin	Activo	Editar	Eliminar
No se han encontrado resultados										

Y no solo busca del principio de la cadena en adelante, es decir, al poner “anto” que es el comienzo de “Antonio” resulta sencillo, pero es que, si se pusiera el dominio de los correos, ya sea email o gmail lo busca sin problemas gracias a la expresión regular:

Listado de usuarios										
ID	Foto	Username 🕒🕒	Email	Nombre 🕒🕒	Apellidos	Modificado 🕒🕒	Admin	Activo	Editar	Eliminar
4		jorge_1	jorge-1@gmail.com	Jorge	García Torres	2020-06-02 19:59:02	No			
6		Federico_Ibañez	federico_ibaez@gmail.com	Federico	Ibañez	2020-05-31 11:36:33	No			
10		Destroyer	alfre_destro@gmail.com	Alfredo	Martinez De los Cobos	2020-06-07 13:44:55	No			
12		Naaraa	arodriguez55@gmail.es	Arancha	Rodríguez González	2020-06-04 23:42:34	Sí			
13		Luisa	luisa_ventu@gmail.com	Luisa	Ventura	2020-06-09 08:49:18	No			

Expresión regular que utilizo: `usuarios.username REGEXP '^.*' . $campo . ".*?$/'`

Ese \$campo es la cadena que le paso por parámetro y al igual que en el username busca en todos los demás campos de la tabla de usuarios, o de post o la que sea.

Otras funciones a destacar:

Hay una función, que a nivel general ha sido vital. Realmente me ha salvado la vida en este proyecto, y es **debug(\$array)**. Una función que se carga desde los helpers de CodeIgniter (en concreto `utilles_helpers.php`) y está basada en el debugger, es decir, devuelve en forma de ‘`<pre>`’ combinado con ‘`print_r`’ el array o variable que le indiquemos. La he utilizado infinidad de veces para analizar post, datos recogidos de la base de datos o simplemente ver como funcionaban las uris a la hora de seleccionar fragmentos de lo que se recibía en ciertas funciones de los controladores.

```
debug($datos_email);
```

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

function debug( $var)
{
    $debug = debug_backtrace();
    echo "<pre>";
    echo $debug[0]['file']." ".$debug[0]['line']."<br><br>";
    print_r($var);
    echo "</pre>";
    echo "<br>";
}
```

En el caso de utilizarla para mostrar los datos recogidos en el formulario, tras verificarlos y almacenarlos debidamente, y enviados mediante email, aparece en el navegador así:

C:\xampp\htdocs\argaming\application\controllers>EmailController.php 33

```
Array
(
    [0] => Prueba2
    [1] => pruebas@gmail.com
    [2] => 612345678
    [3] => Prueba
    [4] => Hola,-esto-es-un-mensaje-de-prueba-para-mostrarlo-en-forma-de-debug-en-el-proyecto,-brbrSaludos!
)
```

El correo ha sido enviado con éxito

Contáctenos

Su nombre de usuario, o el nombre que desee, si no está registrado

Dirección de correo electrónico para poder responderle

Su número de teléfono (opcional)

Asunto del mensaje

Introduzca aquí el mensaje que quiera transmitirnos. Tendrá nuestra respuesta lo antes posible! Gracias 😊

Enviar

De paso muestro lo que aparece tras enviar un mensaje, desde el contacto, satisfactoriamente.

Como cualquiera, esté registrado o no y haya iniciado sesión o no, podrá enviar correos no se comprueba si el usuario se encuentra en la base de datos registrado o si ha iniciado sesión antes.

Pero hay otros casos en los que sí será necesario realizar esta comprobación y ahí es donde entra la utilidad práctica de mi función **comprobar_login()** que se encuentra en el mismo archivo de ‘helpers’ que la de debug:

```
function comprobar_login(){

    $datos = array();
    if (isset($_SESSION['logueado']) && $_SESSION['logueado'] === TRUE) {
        $datos['nombre'] = $_SESSION['nombre'];
        $datos['rol'] = $_SESSION['rol'] == 1 ? 'administrador' : 'usuario normal';
        $datos['id'] = $_SESSION['id'];
        $datos['password_hash'] = $_SESSION['password_hash'];
        $datos['imagen_perfil'] = $_SESSION['imagen_perfil'];
    }
    return $datos;
}
```

```
$verif = comprobar_login();

if (!empty($verif) && $verif['rol'] == 'administrador') {
    $datos['rol'] = $verif['rol'];
    $vista = array(
        'vista' => 'admin/listado-videojuegos.php',
        'params' => $datos,
        'layout' => 'ly_admin.php',
        'titulo' => 'Videojuegos - Admin'
    );

    $this->layouts->view($vista);
} else {

    $vista = array(
        'vista' => 'web/listado-videojuegos.php',
        'params' => $datos,
        'layout' => 'ly_home.php',
        'titulo' => 'Videojuegos'
    );

    $this->layouts->view($vista);
}
```

De esta forma se busca la variable de sesión ‘logueado’ que fue creada al iniciar sesión con un usuario y contraseña válidos. Además, se almacenaron los datos básicos del usuario que ahora se recuperan y devuelven a través de la variable ‘datos’ para que desde donde llamemos a dicha función tengamos acceso a estos datos para mostrar el nombre del usuario, tener su id a mano, etc.

En el caso que presento en la captura, se verifica que está logueado y si es administrador, para mostrar la interfaz propia de él, sino se mostrará la normal para un usuario estándar logueado.

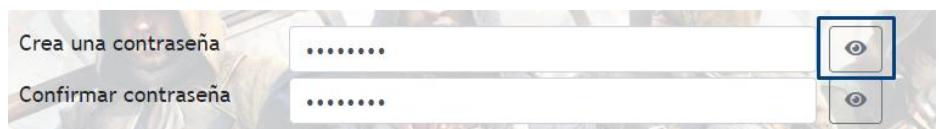
También se puede observar que almaceno el valor de ‘rol’ y se lo paso al propio layout, ya que es en éste donde se encuentra el operador ternario que elige entre una cabecera u otra y necesita saber si es un usuario administrador o no.

Esta pequeña, aunque extremadamente útil, función no solo se usa aquí y en todos los demás controladores explicados, sino que además es imprescindible para el último controlador que falta por detallar, el FormController, encargado de todas las partes con validaciones de formularios que contiene la web (excepto el login, que lo hago desde LoginController directamente).

Cambiando de funcionalidad, estoy satisfecho con como han quedado mis campos de contraseña en los formularios puesto que he añadido la posibilidad de **revelar la contraseña u ocultarla**.

Para ello, me he valido de varias líneas de jquery desde la misma página que buscan el id de 2 botones con iconos interactivos:

Por defecto la contraseña está oculta siempre.



Al pulsar en el ícono del ojo, se muestra cambiando el tipo de input de ‘password’ a ‘text’.





Si se pulsa en ocultar vuelve a cambiar el tipo y revierte los cambios, quedando como al principio.



El código JavaScript, mediante jquery, sería el siguiente para todos los campos de contraseña de mis formularios:

```
<script src="/bootstrap/js/jquery-3.4.1.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    $('#ver-pass').on('click', function() {
        $('#password').attr('type', 'text');
        if ($('#ocultar-pass').length == 0) {
            $('#botones-pass').append($('').attr("class", "btn btn-outline-secondary").attr("type", "button").text("Ocultar contraseña"));
        }
        $('#ocultar-pass').on('click', function() {
            $('#password').attr('type', 'password');
            $('#ocultar-pass').remove();
        });
    });
    $('#ver-pass-conf').on('click', function() {
        $('#password_confirm').attr('type', 'text');
        if ($('#ocultar-pass-conf').length == 0) {
            $('#botones-pass-conf').append($('').attr("class", "btn btn-outline-secondary").attr("type", "button").text("Ocultar contraseña"));
        }
        $('#ocultar-pass-conf').on('click', function() {
            $('#password_confirm').attr('type', 'password');
            $('#ocultar-pass-conf').remove();
        });
    });
});
</script>
```

También me gustaría añadir, que he podido finalmente realizar la **paginación** mediante la librería que CodeIgniter trae para ello, no sin muchas pruebas e intentos infructuosos.

La paginación la he realizado, desde el apartado de administración, sobre los usuarios registrados (que ya van por 16) y era interesante ver cómo queda y no solamente ver una lista interminablemente larga haciendo scroll hacia abajo cada vez que los listaba...

Se compone de varias partes, primero de todo la función desde el modelo tiene que devolver los datos de forma “limitada”, es decir, aprovechando la orden de MySQL limit para mandar la tabla de usuarios por partes al controlador. La he denominado pagination() y se compone por 3 parámetros, el de la tabla, el número de páginas total en el que se va a distribuir nuestro listado y el offset, que responde al índice de comienzo del listado (es lo que va variando en cada página).

```
/* Paginación de usuarios */

function pagination($tabla,$pag_size,$offset)
{
    $this->db->select();
    $this->db->from($tabla);
    $this->db->limit($pag_size,$offset);
    $query = $this->db->get();
    $rows = $query->result_array();
    return $rows;
}

function count($tabla)
{
    $number = $this->db->query("SELECT count(*) as number FROM $tabla")->row()->number;
    return intval($number); //devuelve en un valor entero el número de filas de la tabla
}
```

Otra función del modelo que he usado es `count($tabla)` para contar el número de registros que recibiremos de la tabla.

```
function count($tabla)
{
    $number = $this->db->query("SELECT count(*) as number FROM $tabla")->row()->number;
    return intval($number); //devuelve en un valor entero el número de filas de la tabla que le pidamos
}
```

Una vez tenemos el modelo, en el controlador habría que añadir algo así:

```
$numero_filas = $this->BackEndModel->count('usuarios'); //obtenemos el número de filas de la tabla usuarios

$this->load->library('pagination');

//Configuración básica de la paginación
$config['base_url'] = '/admin/listado';
$config['total_rows'] = $numero_filas;
$config['per_page'] = 5;
$config['uri_segment'] = 3;
$config['num_links'] = 3;

//Otros parámetros de páginación, con bootstrap:
$config['full_tag_open'] = '<ul class="pagination">';
$config['full_tag_close'] = '</ul>';
$config['first_link'] = false;
$config['last_link'] = false;
$config['first_tag_open'] = '<li class="page-item">';
$config['first_tag_close'] = '</li>';
$config['prev_link'] = '&laquo;';
$config['prev_tag_open'] = '<li class="prev page-item">';
$config['prev_tag_close'] = '</li>';
$config['next_link'] = '&raquo;';
$config['next_tag_open'] = '<li class="page-item">';
$config['next_tag_close'] = '</li>';
$config['last_tag_open'] = '<li class="page-item">';
$config['last_tag_close'] = '</li>';
$config['cur_tag_open'] = '<li class="active page-item"><a href="#" class="page-link">';
$config['cur_tag_close'] = '</a></li>';
$config['num_tag_open'] = '<li class="page-item">';
$config['num_tag_close'] = '</li>';

$this->pagination->initialize($config);
$result = $this->BackEndModel->pagination('usuarios', $config['per_page'], $this->uri->segment(3));

$data['usuarios'] = $result;
$data['pagination'] = $this->pagination->create_links();
```

Se carga la librería ‘pagination’, se configura la ruta desde la que vamos a acceder, el número total de registros (a través de la función `count($tabla)`) y 3 valores más: los registros que se van a mostrar por página, de dónde se obtiene el número de secuencia (`uri->segment()`) y el número de links o páginas que queremos mostrar, es decir, los numeritos que aparecen debajo a modo de paginación.

El resto de parámetros los he dejado prácticamente como venían por defecto. Por cierto, puntualizar que reseñaré más adelante, en la webgrafía, de dónde he sacado éste y otros recursos a lo largo de mi proyecto.

Ya solo necesita inicializar la configuración, cargar la información mediante la función del modelo y los resultados devueltos pasarlos a la vista. También tuve que configurar las rutas previamente para indicarles que pasasen el número de página por la url y que apuntasen a la misma función para cargar la página inicial y las demás.

```
$route['admin/listado'] = 'AdminController/listado_paginado';
$route['admin/listado/(:num)'] = 'AdminController/listado_paginado/$1';
```

Quedaría tal que así:

The screenshot shows a web application interface for managing users. At the top, there's a navigation bar with links for 'Inicio', 'Panel de control', and 'Mi perfil'. Below this is a secondary navigation bar with tabs for 'Videojuegos', 'Usuarios', 'Usuarios con paginación', 'Post', and 'Comentarios'. The main content area is titled 'Listado de usuarios' and contains a table of user data. The table has columns for ID, Foto, Username, Email, Nombre, Apellidos, Modificado, Admin, Activo, Editar, and Eliminar. There are 5 rows of data, each with a small profile picture and some placeholder names. At the bottom of the table is a set of pagination links: '<', '1', '2', '3', and '>'. The number '2' is highlighted with a red box.

ID	Foto	Username	Email	Nombre	Apellidos	Modificado	Admin	Activo	Editar	Eliminar
11		anrogo2	anrogo2@email.es	Antonio	Rodríguez González	2020-06-09 09:30:34	Sí			
12		Naaraa	a Rodriguez55@gmail.es	Arancha	Rodríguez González	2020-06-04 23:42:34	Sí			
13		Luisa	luisa_ventu@gmail.com	Luisa	Ventura	2020-06-09 08:49:18	No			
14		antonio	antonio@email.es	antonio	alferez	2020-06-09 21:11:28	No			
15		Maria2	maria18@email.es	Maria José	Lorite Casas	2020-06-15 09:19:08	No			

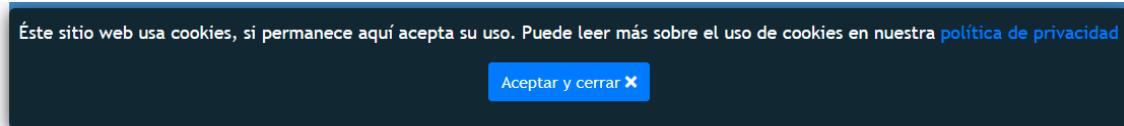
Se muestra cómo al pasar a la página 2, carga desde el punto 5 del listado, es decir, la 2^a parte. Esto se debe a que al tener 15 usuarios en ese momento y dividir entre 3 sale a 5 por página.

Por si no queda suficientemente claro, tengo los siguientes Ids de usuarios en relación a páginas:

- 1,2,4,6,10 -> página 1 (carga con /admin/listado)
- 11,12,13,14,15 -> página 2 (/admin/listado/5)
- 16,17,18,19,20 -> página 3 (/admin/listado/10)

Para la última funcionalidad interesante he reservado la de las cookies. Y es que CodeIgniter, como no podía ser de otra manera, tiene una particular forma de gestionar los cookies y genera la suya propia en forma de “Cookie de Sesión CI”, pero yo he creado la mía propia incluyendo un mensaje en el inicio que advierte de la presencia de cookies y que si es aceptado genera una de inmediato de forma local en el equipo desde el que se accede.

En la página de inicio se ve el cartel así:



Y si se acepta, desde las herramientas de administración en el apartado de almacenamiento/Cookies aparecerá nuestra cookie con el nombre ‘cookies_aceptadas’.



Para crear la cookie he utilizado la función setcookie(), que recibe el nombre que queremos que tenga la cookie, el valor de la misma, el tiempo de expiración y si es segura

```
class CookieController extends CI_Controller {

    function __construct()
    {
        parent::__construct();

        // $this->load->helper('cookie');
        $this->load->helper(array('cookie', 'url'));
    }

    function set()
    {
        // Valores que se le asignan a la cookie
        $nombre = 'cookies_aceptadas';
        $valor = '1';
        $expira = time() + 604800;           // ese tiempo es una semana
        $seguro = TRUE;

        setcookie($nombre,$valor,$expira,$seguro);

        //echo "Congratulation Cookie Set";
        header('Location: /');
    }

    function get()
    {
        // muestra el contenido de la cookie
        //echo $this->input->cookie('cookies_aceptadas',true);
        debug($_COOKIE);
    }

    function delete()
    {
        delete_cookie('cookies_aceptadas');
    }
}
```

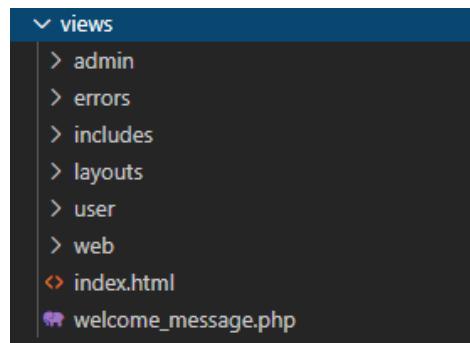
He creado un controlador para las Cookies, y dentro 3 funciones: para añadir cookies, obtenerlas y borrarlas. La idea era crear una cookie para el login, que permitiese recordar el usuario y contraseña de los usuarios; sin embargo, no pude desarrollarla.



VISTAS

Son la última parte del camino, antes de llegar al usuario. Se encargan de mostrar los datos recogidos por el modelo y enviados desde el controlador. Están construidas principalmente con bootstrap, se insertan los datos mediante variables de php, y bucles, en el caso de los arrays.

Una gran ventaja de trabajar de esta forma con las vistas es que se pueden reutilizar y no hacen falta una infinidad de ficheros para construir el sitio web. La estructura principal se mantiene mediante los layouts y el resto, el contenido, viene desde estas vistas. Hay diferentes clasificaciones dentro de mi carpeta de vistas o ‘views’, que contiene:



- Admin (todo lo que maneja el administrador)
 - Editar_post.php
 - Editar_usuario.php
 - Index.php
 - Lista-comentarios.php
 - Lista-post.php
 - Listado_paginacion.php
 - Listado_usuarios.php
 - Listado-videojuegos.php
 - Listados.php
 - Nuevo_post.php
 - Nuevo_usuario.php
 - Perfil_admin.php
- Includes (partes llamadas desde los layouts)
 - Footer.php
 - Header.php
 - Header_panel_control.php
 - Header_session_admin.php
 - Header_session_user.php
- Layouts (diseños de la estructura de las diferentes páginas del blog)
 - Ly_admin.php
 - Ly_admin_basico.php
 - Ly_contacto.php
 - Ly_home.php
 - Ly_login.php
 - Ly_registro.php
 - Ly_session.php
- User (una vez el usuario ha iniciado sesión)
 - Cambiar_contraseña.php
 - Editar-perfil.php
 - Login.php
 - Perfil-usuario.php
- Web (lo que puede ver cualquier usuario)
 - Autores.php
 - Aviso-legal.php
 - Cambiar-contraseña-sin-acceso.php
 - Contacto.php
 - Error-404.php
 - Index.php
 - Lista-post.php
 - Listado-videojuegos.php
 - Novedades-1.php
 - Novedades-2.php
 - Novedades-3.php
 - Nuevo_post.php
 - Nuevo_usuario.php
 - Politica-cookies.php
 - Politica-privacidad.php
 - Post.php
 - Post-novedades.php
 - Usuario.php

RESULTADOS

Una vez están explicadas todas las partes que han intervenido en la creación de mi proyecto, queda mostrar los resultados cuando visualizamos el blog desde cualquier navegador. Enseñaré capturas de la mayoría de las páginas junto a la referencia de éstas, para que se entienda:

Página de inicio

(cabecera)



(sección de novedades con 3 noticias recientes sobre videojuegos y consolas)

Un analista predice que PS5 venderá casi el doble que Xbox Series X para 2024

La firma Ampere estima que PlayStation 5 venderá 66 millones de unidades para ese año...

[Ver más »](#)

Silent Hill regresa por la puerta pequeña de la mano de una colaboración con Behaviour Interactive

Aunque no es lo que muchos esperan, los creadores de Dead By Daylight nos traen...

[Ver más »](#)

La temporada 4 de 'Call of Duty: Modern Warfare' y 'Warzone' ya está aquí: estas son sus novedades

La temporada 4 de 'Call of Duty: Modern Warfare' y 'Warzone' ya se encuentra...

[Ver más »](#)

Listado de posts							
Imagen	Título	Creador	Descripción	Meta descripción	Fecha creación	Última modificación	Visitas
	Prueba, creado mi primer post desde formulario	anrogo2	"Sed ut perspiciatis unde omnis iste natus error sit volupta..."	post-de-prueba	2020-06-04 23:07:28	2020-06-16 14:09:11	11
	Mi post con foto	jorge_1	"Sed ut perspiciatis unde omnis iste natus error sit volupta..."	post-con-foto	2020-06-04 23:29:03	2020-06-16 17:59:10	11
	Cómo unirse a nuestra nueva comunidad	anrogo	¿Quieres unirte a nuestra comunidad? Pues es sencillo solo ...	nuevos-usuarios-argaming	2020-06-04 23:16:51	2020-06-16 17:30:34	19
	Bienvenidos al blog ARGaming	anrogo	Hola a todos! Con este post arrancamos el nuevo blog especia...	primer-post	2020-06-04 21:41:05	2020-06-16 18:00:38	23
	Final Fantasy VII: Remake	aRodriguez	Nueva adaptación de la obra maestra del rol japonés. El re...	final-fantasy-remake-7	2020-06-08 08:31:25	2020-06-11 23:20:37	28

(footer)



Página de novedades

Home / Novedades / 2020 / Call of Duty: Modern Warfare y Warzone

La temporada 4 de 'Call of Duty: Modern Warfare' y 'Warzone' ya está aquí: estas son sus novedades

La temporada 4 de 'Call of Duty: Modern Warfare' y 'Warzone' ya se encuentra entre nosotros tras el retraso. Repasamos sus novedades.

By Anrogo



Post de usuarios

¿Cómo unirse?	Bienvenidos al Final Fantasy VII: blog!
Remake	Mi post con foto

GTA V, no hay que decir más



Otros enlaces que te pueden interesar

[GTA V fue el videojuego más](#)

Post de novedades (al acceder a uno)

Novedades

La actualidad sobre el mundo de los videojuegos



[La temporada 4 de 'Call of Duty: Modern Warfare' y 'Warzone' ya está aquí: estas son](#)

Post de usuarios

[¿Cómo unirse?](#) [Final Fantasy VII: Remake](#) [Bienvenidos al blog!](#) [Mi post con foto](#)

GTA V, no hay que decir más



Otros enlaces que te pueden interesar

[GTA V fue el videojuego más vendido durante el mes de mayo](#)

Lista de videojuegos

Listado de videojuegos

Resultados encontrados: 126

ID	Título	Descripción	URL	Etiqueta
36345	Mario Bros.	Mario Bros. (マリオブラザーズ; Mario Burazāzu; lit. Hermanos Mario) es un videojuego de arcade desarrollado por Nintendo en el año 1983. Fue creado por Shigeru Miyamoto. Ha sido presentado	Pulsa para ver más información.	mariobros
2597	Super Mario Bros.	Super Mario Bros. (スーパーマリオブラザーズ; Sūpā Mario Burazāzu; lit. Súper Hermanos Mario) es un videojuego de plataformas, diseñado por Shigeru Miyamoto, lanzado el 13 de	Pulsa para ver más información.	mariobros
2605	Super Mario Bros. 3	Super Mario Bros. 3 (también conocido como Mario 3 o Super Mario 3). Se publicó el 23 de octubre de 1988 en Japón y el 12 de febrero	Pulsa para ver más información.	mariobros

argaming.com/lista-juegos/buscar

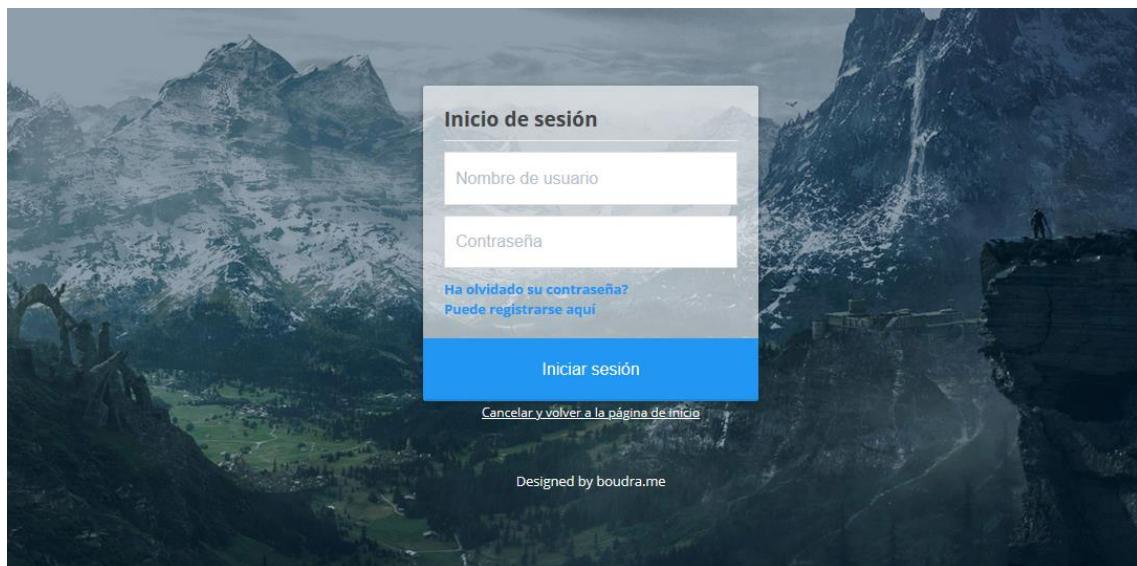
Resultados encontrados: 40

Listado de videojuegos

Resultados encontrados: 40

ID	Título	Descripción	URL	Etiqueta
3496	Age of Mythology (juego)	Age of Mythology es un juego de estrategia en tiempo real (RTS) desarrollado por Ensemble Studios y publicado por Microsoft Games en 2002. Aunque similar a Age of Empires, Age of Mythology cuenta con gráficos	Pulsa para ver más información.	ageof
6594	Age of Empires	Age of Empires (en español "Era de los Imperios" o "Edad de Imperios"), es una serie de juegos de estrategia en tiempo real de gran popularidad inspirados en diversas épocas y civilizaciones de la historia	Pulsa para ver más información.	ageof
7359	Age of Empires II: The Conquerors	publicado por Microsoft Games para las plataformas Microsoft Windows y Apple Macintosh. El título expande al videojuego Age of Empires II: The Age of Kings y fue lanzado a mediados del año 2000. AoC inicia	Pulsa para ver más información.	ageof
23930	Age of Empires III: The WarChiefs	Age of Empires III The WarChiefs trae 3 nuevas civilizaciones americanas: La Confederación Iroquesa, Nación Sioux y Imperio	Pulsa para ver más información.	ageof

Login



Registro

A registration form titled "Nuevo usuario". It includes fields for "Nombre de usuario", "Nombre", "Apellidos", and "Correo". Below these are fields for creating a password ("Crea una contraseña") and confirming it ("Confirmar contraseña"), each with an "eye" icon for password visibility. There is also a checkbox for accepting terms and conditions ("Acepto las políticas de [privacidad](#) y [cookies](#) de Argaming"). A blue button labeled "Completar registro" is at the bottom left. A link to "Cancelar y volver a la página de inicio" is at the bottom right.

Post (ordenados por visitas)

Listado de posts					Crear nuevo post	<input type="text" value="Buscar..."/>	Búsqueda
Imagen	Título	Creador	Descripción	Meta descripción	Fecha creación <small>↓ ↑</small>	Última modificación <small>↓ ↑</small>	Visitas <small>↓ ↑</small>
	Final Fantasy VII: Remake	arodriguez	Nueva adaptación de la obra maestra del rol japonés. El re...	final-fantasy-remake-7	2020-06-08 08:31:25	2020-06-11 23:20:37	28
	Bienvenidos al blog ARGaming	anrogo	Hola a todos! Con este post arrancamos el nuevo blog especia...	primer-post	2020-06-04 21:41:05	2020-06-16 18:00:38	23
	Cómo unirse a nuestra nueva comunidad	anrogo	¿Quieres unirte a nuestra comunidad? Pues es sencillo solo ...	nuevos-usuarios-argaming	2020-06-04 23:16:51	2020-06-16 17:30:34	19
	Prueba, creado mi primer post desde formulario	anrogo2	"Sed ut persipciatis unde omnis iste natus error sit volupta...	post-de-prueba	2020-06-04 23:07:28	2020-06-16 14:09:11	11
	Mi post con foto	jorge_1	"Sed ut persipciatis unde omnis iste natus error sit volupta...	post-con-foto	2020-06-04 23:29:03	2020-06-16 17:59:10	11

Contacto

Contáctenos

Introduzca aquí el mensaje que quiera transmitirnos. Tendrá nuestra respuesta lo antes posible! Gracias 😊

Recuperación de contraseña

No recuerdo mi contraseña, ¿alguna solución?

Puedes decirnos tu correo para que te mandemos una nueva contraseña

Perfil de usuario

arodriguez


[Editar perfil](#)
[Borrar perfil](#)
[Cerrar sesión](#)

Nombre	Antonio
Apellidos	Rodríguez
Correo	arodriguez55@email.es
Rol	Usuario estándar

Autores

Listado de autores

Usuario	Post creados
anrogo	2
anrogo2	1
arodriguez	1
jorge_1	1

Perfil de autor

arodriguez



Nombre	Antonio
Apellidos	Rodríguez
Correo	arodriguez55@email.es
Rol	usuario estándar

Posts creados por arodriguez

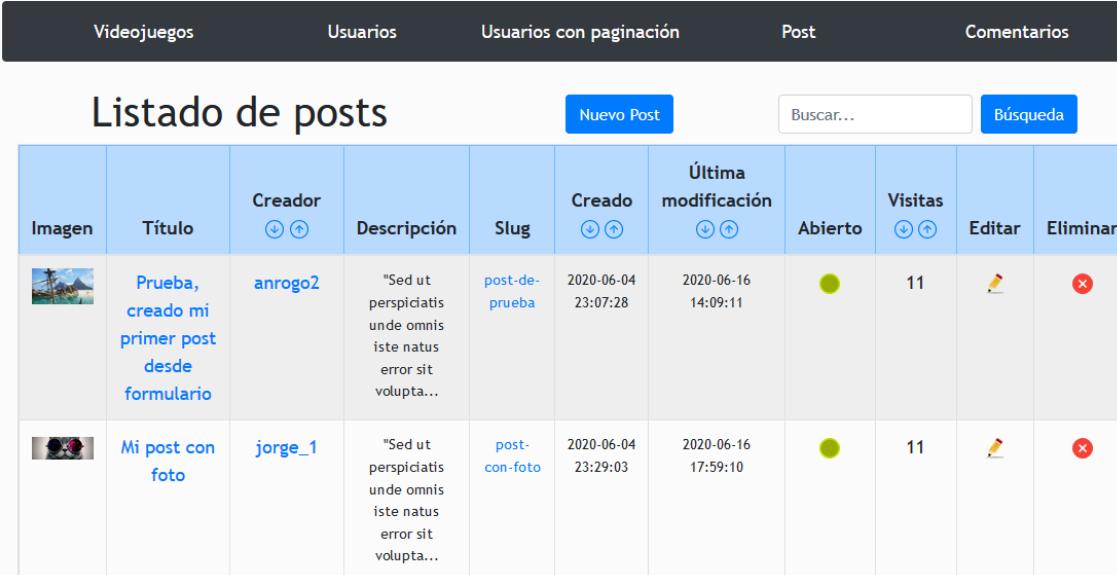
Imagen	Título	Creador	Descripción	Meta descripción	Última modificación	Visitas
	Final Fantasy VII: Remake	arodriguez	Nueva adaptación de la obra maestra del rol japonés. El re...	final-fantasy-remake-7	2020-06-16 23:23:26	32

La parte de admin:
(cabecera y menú en el panel de control)



The screenshot shows the top navigation bar with four items: Inicio, Panel de control (selected), Mi perfil, and Cerrar sesión. Below this is a secondary navigation bar with five items: Videojuegos, Usuarios, Usuarios con paginación, Post (selected), and Comentarios. The main content area is titled "Listados disponibles:" and contains a bulleted list of options: - Usuarios registrados en la plataforma, - Post disponibles, - Comentarios de los usuarios, and - Lista de videojuegos. A note at the bottom says "Seleccione la opción en el menú superior para ver los detalles."

CRUD de admin para los post:



The screenshot shows the "Post" section of the admin panel. At the top, there is a header with tabs: Videojuegos, Usuarios, Usuarios con paginación, Post (selected), and Comentarios. Below the header is a search bar with "Nuevo Post" and "Buscar..." fields, and a "Búsqueda" button. The main area is titled "Listado de posts" and displays a table with two rows of data. The columns are: Imagen, Título, Creador, Descripción, Slug, Creado, Última modificación, Abierto, Visitas, Editar, and Eliminar. The first post has a title "Prueba, creado mi primer post desde formulario" and a description starting with "Sed ut perspiciatis unde omnis iste natus error sit volupta...". The second post has a title "Mi post con foto" and a similar description. Both posts have a green status circle in the "Abierto" column, 11 visits, and edit/delete icons.

REFERENCIAS - Webgrafía

Voy a enumerar las páginas webs y recursos que he utilizado durante el desarrollo del proyecto y la formación en CodeIgniter, antes y durante.

Como anécdota, diré que cuando me empecé a informar más en profundidad sobre la temática “gaming” me surgieron conceptos con los que ni yo estoy familiarizado, aunque me gusten los juegos de PC, y tuve que consultar en diccionarios online:

Por ejemplo, con el término **MMORPG**:

--Para aquellos que no conozcáis el término, un MMORPG (Massively Multiplayer Online Role-Playing Game) son juegos de rol online que permiten a un número masivo de jugadores introducirse en un mundo virtual de manera simultánea e interactuar entre ellos.

<https://www.hobbyconsolas.com/reportajes/cual-mejor-mmorpg-actualidad-404373>

Cabe destacar, que hasta el 2019 el mejor juego MMORPG que existe es el World of Warcraft

También decir que hubo un momento en el que dudé sobre cómo extraer la información de los videojuegos y surgieron otras posibilidades, como por ejemplo el **scrapping**. Y qué es? Pues consiste en técnicas que permiten obtener información de otras webs e introducirla en la tuya. Me he apoyado en varias fuentes para investigar sobre el tema:

-Las técnicas de screen scrapping y web scrapping son poderosas aliadas de las tiendas de eCommerce, pues proporcionan una forma rápida de recopilar y comparar precios o contenidos de otros sitios de Internet, ya sean proveedores o competidores. Esto es más importante cuantos más productos haya en una tienda y cuanto más completa se quiera publicar la información. O analizar a la competencia.

<https://blog.seur.com/web-scrapping-commerce/>

Mucho antes, además, estuve buscando APIs que proporcionasen información de videojuegos y di con alguna cosa bastante interesante, como este repositorio de Github con datos de varias APIs públicas.

<https://github.com/public-apis/public-apis>

Las referencias más importantes de mi proyecto:

- API con los datos de videojuegos que proceso para listarlos (02/04/20)
<https://videojuegos.fandom.com/es/api/v1>
- Recursos consultados sobre CodeIgniter (10/04/2020)
<https://code.tutsplus.com/es/tutorials/how-to-work-with-session-data-in-codeigniter--cms-28658>
<https://stackoverflow.com/questions/5409485/codeigniter-post-variables>
<https://fernando-gaitan.com.ar/codeigniter-parte-9-libreria-session/>
<https://stringinarray.wordpress.com/2014/02/07/introduccion-al-patron-mvc-con-codeigniter/>
- Por supuesto, el curso de CodeIgniter en OpenWebinars (7h en total) – Acabado 11/04/20:
<https://openwebinars.net/academia/aprende/codeigniter/>

Las demás referencias son más simples, consultas puntuales, normales y corrientes.

- Guía sobre git y sus diferentes funcionalidades disponibles para hacer control de versiones (14/04/2020)
<https://guides.github.com/introduction/git-handbook/>

- Información adicional sobre videojuegos (25/04/2020)
<https://www.borntoplay.es/>
<https://www.ultragame.es/juegos/dead-by-daylight-nightmare-edition/noticias#36290>
- Página desde la que he diseñado mis wireframes (25/04/2020)
<https://mockflow.com>
- Web con numerosos recursos para consultar tanto de bootstrap como de javascript (30/04/2020)
https://www.w3schools.com/bootstrap4/bootstrap_ref_all_classes.asp
- Sitio oficial de bootstrap, con toda la información y los repositorios (30/04/2020)
<https://getbootstrap.com/>
- Vídeo - curso completo sobre Bootstrap 4 (30/04/2020)
<https://www.youtube.com/watch?v=-83eiJ9EaD4&t=3872s>
- Plantillas para Bootstrap (02/05/2020)
<https://bootswatch.com/>
- Iconos de todo tipo, gratuitos (02/05/2020)
<https://www.flaticon.es/icono-gratis/>
- Detallada guía sobre cómo configurar inicios de sesión en CodeIgniter (desde el login hasta la sesión con cookies). Fue muy muy útil, la verdad. (03/05/2020)
<https://fernando-gaitan.com.ar/codeigniter-parte-9-libreria-session/>
- Consultas varias sobre manejar los datos de sesión en CodeIgniter (04/05/2020)
<https://es.stackoverflow.com/questions/170840/como-usar-datos-de-un-session-codeigniter>
- Curso básico de Git (05/05/2020)
<https://www.youtube.com/watch?v=HiXLkL42tMU&t=587s>
- Sitio online para construir todo tipo de diagramas y modelos de datos (07/05/2020)
<https://www.lucidchart.com>
- Ampliando conocimientos de Git (09/05/2020)
<https://diego.com.es/ramas-y-uniones-en-git>
<https://www.it-swarm.dev/es/git/como-salir-de-git-log-o-git-diff/942153601/>
<https://git-scm.com/book/es/v2/Fundamentos-de-Git-Deshacer-Cosas>
- Validación de formularios en HTML5 (11/05/2020)
<https://lenguajehtml.com/p/html/formularios/validaciones-html5>
- Ejemplo de login realizado con JQUERY, PHP Y AJAX (13/05/2020)
<https://www.jose-aguilar.com/blog/jquery-ajax-php-form/>
- Cómo utilizar los iconos de Font Awesome en su versión 4 (14/05/2020)
<https://fontawesome.com/how-to-use/on-the-web/setup/upgrading-from-version-4>

- Validaciones de formularios con CodeIgniter (15/05/2020)
<http://rickharrison.github.io/validate.js/> (no me convenció)
https://codeigniter.com/userguide3/libraries/form_validation.html (es la que he terminado usando)
- Pros y contras de CodeIgniter. Ventajas de usar Laravel (19/05/2020)
<https://styde.net/porque-elegir-laravel-en-vez-de-codeigniter/>
- Fondos HD (de todas las temáticas y con posibilidad de redimensionar) gratis. Es, sin duda, la mejor web de fondos que he visitado nunca (21/05/2020)
<https://wall.alphacoders.com>
- Dudas sobre cómo cambiar dinámicamente el tipo de input en el formulario (21/05/2020)
<https://es.stackoverflow.com/questions/189665/c%C3%B3mo-puedo-cambiar-el-valor-del-type-en-un-input-din%C3%A1mico-con-jquery>
- Dudas relacionadas con eventos de formulario en jquery (21/05/2020)
<https://informaticapc.com/curso-de-jquery/eventos-formulario.php>
- Ejemplo de formulario, con validación por js, que envía email al correo (22/05/2020)
<https://www.raulprietofernandez.net/blog/webs/como-crear-un-formulario-de-contacto-con-bootstrap-4-y-php>
- Pruebas de email, para conectar con el servidor smtp (26/05/2020)
<https://mailtrap.io/>
- Ejemplos panel de control admin (26/05/2020)
<https://medium.com/hackernoon/10-fascinating-php-admin-templates-4acfb113db7>
- Diseño del login gracias a “boudra.me”. Su repositorio de github: (Mediados de mayo)
<https://gist.github.com/boudra/16f32dfedd19c6f2d8b8>
- Funciones útiles para trabajar con cadenas en MySQL (29/05/2020)
<https://www.tutorialesprogramacionya.com/mysqlia/temarios/descripcion.php?inicio=21&cod=75&punto=25>
- Expresiones regulares en MySQL (29/05/2020)
<https://blog.openalfa.com/como-usar-expresiones-regulares-en-mysql>
- Crear un buscador con codeigniter y MySQL (08/06/2020)
<https://www.youtube.com/watch?v=ob0xVe6RWI&vl=es-419>
- Opciones para crear una presentación con javascript (09/06/2020)
<https://www.dariobf.com/6-plugins-javascript-increibles-para-crear-presentaciones/>
- Posible mejora del panel de administración (13/06/2020)
<https://colorlib.com/wp/bootstrap-sidebar/>
- Configurar paginación en CodeIgniter (16/06/2020)
<https://pastebin.com/nQBbUJ0x>

CONCLUSIONES

Este proyecto de blog/foro, que más bien a terminado siendo un blog con la posibilidad de realizar comentarios y responderlos, ha alcanzado la mayor parte de objetivos, aunque evidentemente no todos. Siempre existe la posibilidad de mejorar y mi proyecto no está exento de fallos y de posibles mejoras. Para empezar, quería introducir una especie de sistema de “post favoritos” que se almacenesen y el usuario los tuviera más a mano, digamos. Esta al igual que otras funcionalidades (mejores categorías y mejores enlaces, alguna animación que tenía en mente, etc.) se han quedado sobre el papel, aunque sin duda podrían añadirse para darle más valor al propio blog.

Creo sinceramente que he llevado a cabo un gran trabajo, durante el curso de DAW haciéndolo lo mejor posible; y más tarde, tras terminar y empezar el período de FCT, instruyéndome previamente en el curso de CodeIgniter, y desarrollando después el proyecto con este framework de PHP.

Y, aunque no me ha costado mucho esfuerzo dedicarle todas las horas que han sido precisas, puesto que me ha gustado trabajar con CodeIgniter y estaba motivado con el tema del proyecto, hay que admitir que le he dedicado mucho más tiempo del que preveía en un principio. Aún así, le dedicaría un extra para sacar las funcionalidades extra y que quedase prácticamente perfecto.

Por ejemplo, la interfaz de la administración habría estado bien meterle una plantilla atractiva y diferenciarla del resto del blog. Y también me quedo con la gana de experimentar más con la parte de los correos y automatizarlo mejor, para poder recuperar la contraseña en caso de olvido sin ir más lejos.

A pesar de este apunte, puedo decir que estoy contento ya que he aprendido muchísimo, he recordado cosas que creía no saber pero que estaban ahí de mi andadura por ASIR; principalmente de todo lo relacionado con bases de datos, servidores FTP y Apache. Y, por tanto, ¡Ha sido un confinamiento muy provechoso!

Estoy seguro de que paso a paso, y nivel a nivel, si hablamos en el más puro contexto “gamer”, llegaré a ser un auténtico Desarrollador Web.

AGRADECIMIENTOS

Tengo que agradecer, en primer lugar, a mi familia con la que he convivido durante la crisis del COVID-19 y han sido un pilar fundamental mientras realizaba el proyecto, después el profesorado de DAW que me ha ayudado a llegar hasta aquí. Y, por supuesto, al tutor del proyecto que ha sido Rafael García Cabrera, gran profesor y persona, y el auténtico artífice de que haya utilizado este framework, puesto que lo hablamos y decidimos que podría irme bien con él, y así ha sido.

Ahora dejo por aquí los anexos con información extra sobre el desarrollo del proyecto:

ANEXOS:

ANEXO I

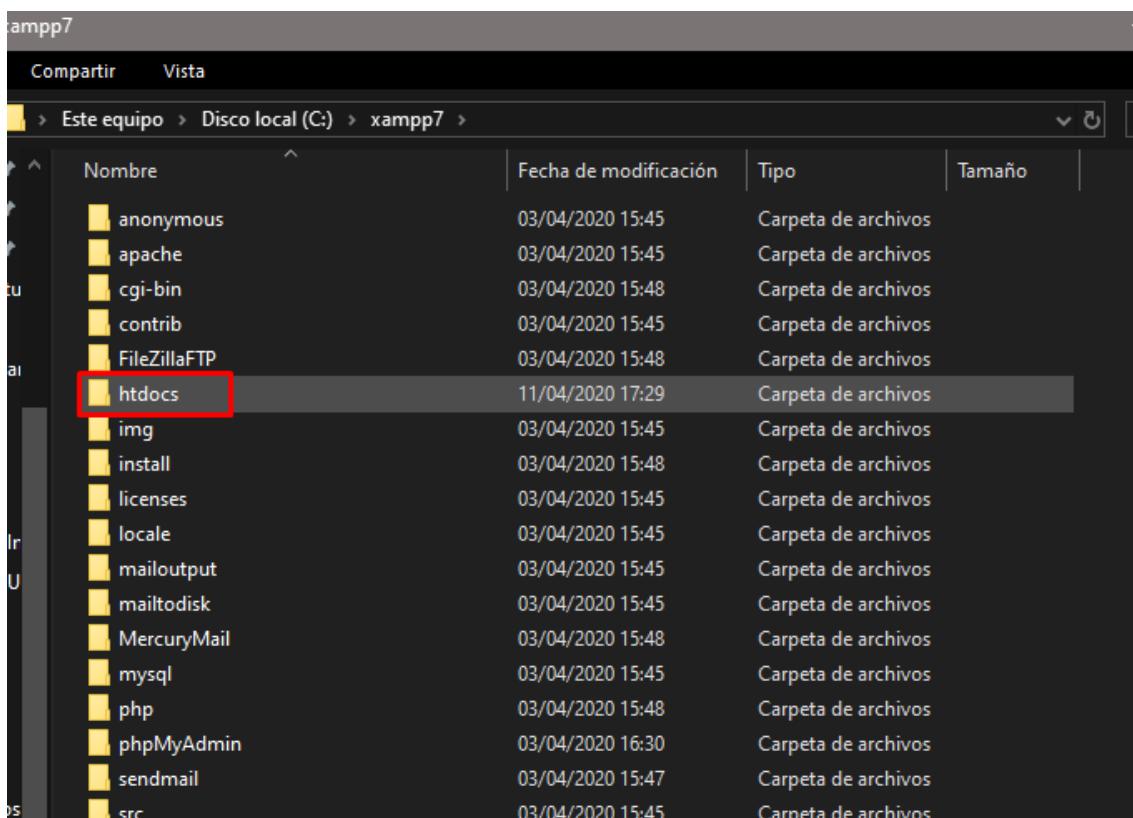
Entorno de trabajo

Lo primero, explicación de cómo he organizado mi entorno de trabajo con CodeIgniter y Visual Studio Code.

CodeIgniter es un potente framework de PHP, muy liviano, construido para desarrolladores que necesitan un kit de herramientas simple y elegante para crear aplicaciones web completas.

Básicamente necesitaremos un servidor web con apache, PHP, MySQL ... Tipo Xampp, por ejemplo. Una vez lo tengamos instalado consiste en descargar la versión de CodeIgniter que sea y descomprimirla en la carpeta htdocs de Xampp:

Ruta: <C:\xampp7\htdocs> (para mí es xampp7, pero depende de cada instalación)



Nombre	Fecha de modificación	Tipo	Tamaño
anonymous	03/04/2020 15:45	Carpeta de archivos	
apache	03/04/2020 15:45	Carpeta de archivos	
cgi-bin	03/04/2020 15:48	Carpeta de archivos	
contrib	03/04/2020 15:45	Carpeta de archivos	
FileZillaFTP	03/04/2020 15:48	Carpeta de archivos	
htdocs	11/04/2020 17:29	Carpeta de archivos	
img	03/04/2020 15:45	Carpeta de archivos	
install	03/04/2020 15:48	Carpeta de archivos	
licenses	03/04/2020 15:45	Carpeta de archivos	
locale	03/04/2020 15:45	Carpeta de archivos	
mailoutput	03/04/2020 15:45	Carpeta de archivos	
mailtodisk	03/04/2020 15:45	Carpeta de archivos	
MercuryMail	03/04/2020 15:48	Carpeta de archivos	
mysql	03/04/2020 15:45	Carpeta de archivos	
php	03/04/2020 15:48	Carpeta de archivos	
phpMyAdmin	03/04/2020 16:30	Carpeta de archivos	
sendmail	03/04/2020 15:47	Carpeta de archivos	
src	03/04/2020 15:45	Carpeta de archivos	

Este equipo > Disco local (C:) > xampp7 > htdocs >			
Nombre	Fecha de modificación	Tipo	Tamaño
blog	07/04/2020 12:56	Carpeta de archivos	
ci3.11	04/04/2020 14:04	Carpeta de archivos	
ci3.11-template	05/04/2020 22:51	Carpeta de archivos	
ci4	03/04/2020 16:32	Carpeta de archivos	
dashboard	03/04/2020 15:45	Carpeta de archivos	
img	03/04/2020 15:45	Carpeta de archivos	
myblog	06/04/2020 14:21	Carpeta de archivos	
webalizer	03/04/2020 15:45	Carpeta de archivos	
xampp	03/04/2020 15:45	Carpeta de archivos	
.htaccess	19/09/2019 5:08	Archivo HTACCESS	1 KB
applications.html	27/08/2019 16:02	Firefox HTML Doc...	4 KB
bcit-ci-CodeIgniter-3.1.11-0-gb73eb19.zip	04/04/2020 13:33	Archivo WinRAR Z...	2.694 KB

En mi caso ese es el archivo que me he bajado de la versión 3.1.11 de CodeIgniter. Una vez lo tengamos, se descomprime pulsando con el botón derecho “Extraer aquí” y encontraremos lo siguiente en su interior:

bcit-ci-CodeIgniter-b73eb19			
Compartir Vista			
Este equipo > Disco local (C:) > xampp7 > htdocs > bcit-ci-CodeIgniter-b73eb19 >			
Nombre	Fecha de modificación	Tipo	Tamaño
application	19/09/2019 5:08	Carpeta de archivos	
system	19/09/2019 5:08	Carpeta de archivos	
user_guide	19/09/2019 5:08	Carpeta de archivos	
.editorconfig	19/09/2019 5:08	Archivo EDITORC...	1 KB
.gitignore	19/09/2019 5:08	Documento de te...	1 KB
composer.json	19/09/2019 5:08	JSON File	1 KB
contributing.md	19/09/2019 5:08	Archivo MD	7 KB
index.php	19/09/2019 5:08	Archivo PHP	11 KB
license.txt	19/09/2019 5:08	Archivo TXT	2 KB
readme.rst	19/09/2019 5:08	Archivo RST	3 KB

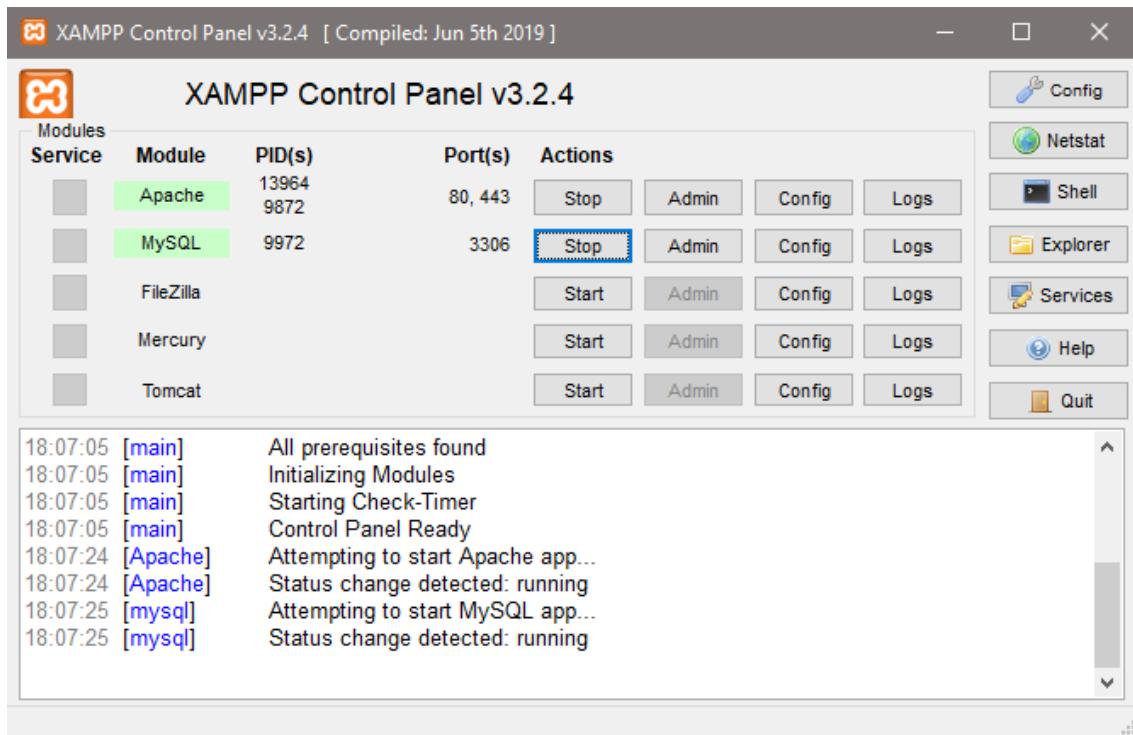
Esa era la carpeta original, ahora muestro como ha quedado la mía del proyecto:

Disco local (C:) > xampp > htdocs > argaming >			
Nombre	Fecha de modificación	Tipo	Tamaño
.git	26/05/2020 23:56	Carpeta de archivos	
application	12/05/2020 16:41	Carpeta de archivos	
pruebas	26/05/2020 16:24	Carpeta de archivos	
system	25/04/2020 18:59	Carpeta de archivos	
tests	25/04/2020 18:59	Carpeta de archivos	
user_guide	25/04/2020 19:00	Carpeta de archivos	
.editorconfig	19/09/2019 5:08	Archivo EDITORC...	1 KB
.gitignore	06/05/2020 17:49	Documento de te...	1 KB
.htaccess	26/05/2020 16:33	Archivo HTACCESS	2 KB
argaming.sql	26/05/2020 16:33	SQL Text File	4 KB
composer.json	19/09/2019 5:08	JSON File	1 KB
contributing.md	19/09/2019 5:08	Archivo MD	7 KB
license.txt	19/09/2019 5:08	Archivo TXT	2 KB
readme.rst	19/09/2019 5:08	Archivo RST	3 KB

El directorio de Argaming contiene las siguientes carpetas:

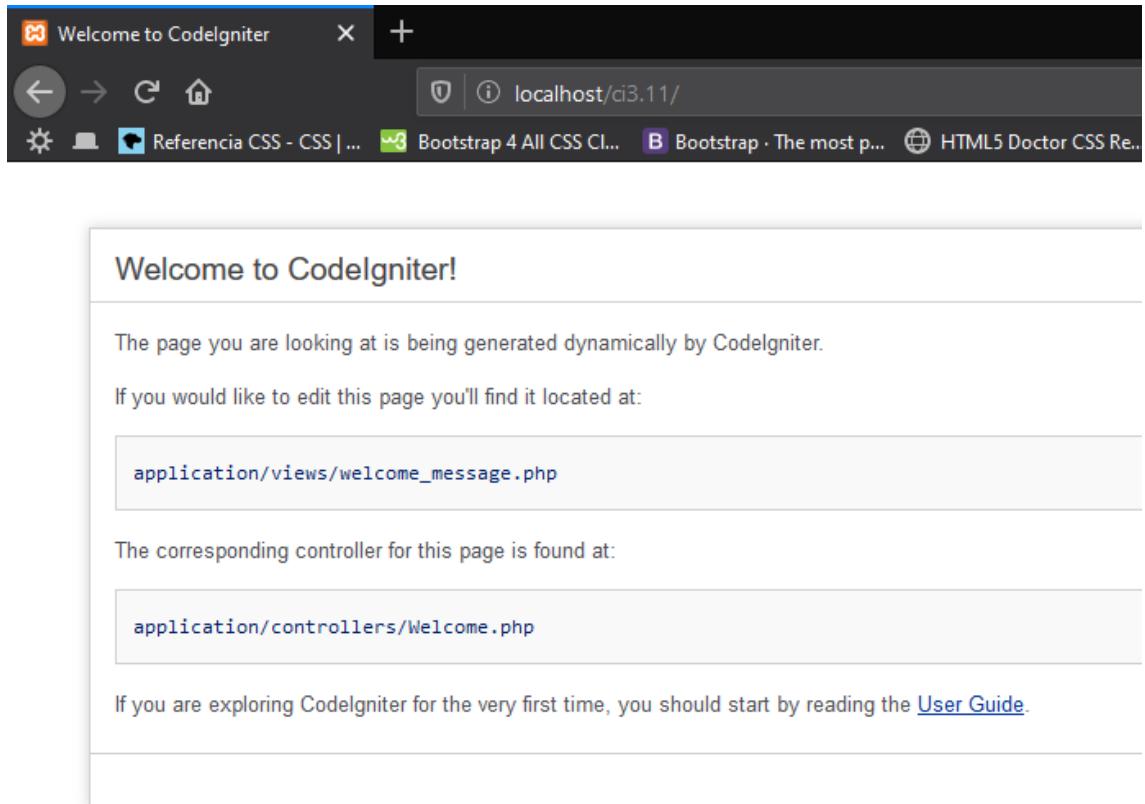
- Application**: todo lo relativo a la aplicación creada
- System**: desde donde se lleva a cabo el funcionamiento del framework
- User_guide**: guías y tutoriales de uso para principiantes.
- Pruebas**: es la carpeta donde he realizado pruebas durante el proyecto.

Como ya tendremos nuestro servidor web solo habrá que encender Apache (y MySQL, aunque ahora mismo no se va a utilizar ninguna base de datos).



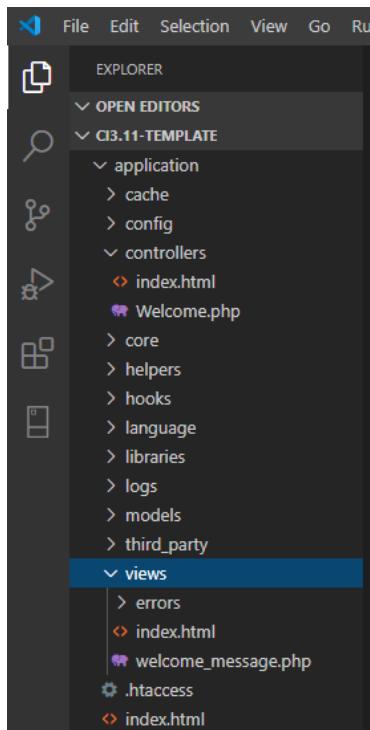
Ya solo faltaría irnos al navegador y escribir en la barra de búsqueda:

localhost/nombre_aplicación/ y el framework con la configuración inicial se encarga de mostrarnos una página de inicio de bienvenida.



The screenshot shows a web browser window with the title "Welcome to CodeIgniter". The address bar indicates the URL is "localhost/ci3.11/". Below the title, the page content reads: "The page you are looking at is being generated dynamically by CodeIgniter. If you would like to edit this page you'll find it located at: application/views/welcome_message.php. The corresponding controller for this page is found at: application/controllers/Welcome.php. If you are exploring CodeIgniter for the very first time, you should start by reading the [User Guide](#)".

Esta misma página nos da las primeras indicaciones de cómo funciona CodeIgniter y es que explica desde donde viene todo lo que muestra.



Como sabremos ya este framework funciona mediante el MVC o Modelo Vista-Controlador, y ese sistema consiste en que un controlador es el encargado de mediar entre datos y vistas, que es lo que finalmente ve el usuario.

El controlador en este caso es [Welcome.php](#) y se sitúa en la carpeta controllers. Desde ahí llama a la vista [welcome_message.php](#) o mensaje de bienvenida, cargado desde la carpeta de views o vistas, que es lo que estamos viendo en el navegador.

En mi caso lo he configurado mediante los virtualhost de Apache para que me responda a mi dominio personal. Por lo tanto, puedo acceder de las dos formas, o bien localhost + “nombre de la carpeta principal” o con el dominio que haya configurado.

Para configurarlo es necesario editar el archivo desde donde se cargan los virtualhost, [httpd-vhost.conf](#), en la siguiente ruta:

C:\xampp7\apache\conf\extra

Nombre	Fecha de modificación	Tipo	Tamaño
httpd-ajp.conf	30/03/2013 13:29	Archivo CONF	1 KB
httpd-autoindex.conf	03/04/2020 15:48	Archivo CONF	3 KB
httpd-dav.conf	03/04/2020 15:48	Archivo CONF	3 KB
httpd-default.conf	03/04/2020 15:48	Archivo CONF	3 KB
httpd-info.conf	03/04/2020 15:48	Archivo CONF	2 KB
httpd-languages.conf	03/04/2020 15:48	Archivo CONF	6 KB
httpd-manual.conf	03/04/2020 15:48	Archivo CONF	2 KB
httpd-mpm.conf	03/04/2020 15:48	Archivo CONF	5 KB
httpd-multilang-errordoc.conf	03/04/2020 15:48	Archivo CONF	3 KB
httpd-proxy.conf	30/03/2013 13:29	Archivo CONF	1 KB
httpd-ssl.conf	03/04/2020 15:48	Archivo CONF	14 KB
httpd-userdir.conf	03/04/2020 15:48	Archivo CONF	1 KB
httpd-vhosts.conf	11/04/2020 17:32	Archivo CONF	3 KB
httpd-xampp.conf	07/04/2020 11:23	Archivo CONF	3 KB
proxy-html.conf	11/08/2019 14:23	Archivo CONF	4 KB

La instrucción es relativamente sencilla y consta de indicar el directorio raíz utilizado, el nombre del servidor o ServerName, que deberá coincidir con el que indiquemos más adelante; y por supuesto, debajo a la misma ruta del directorio se le asignarán los siguientes permisos:

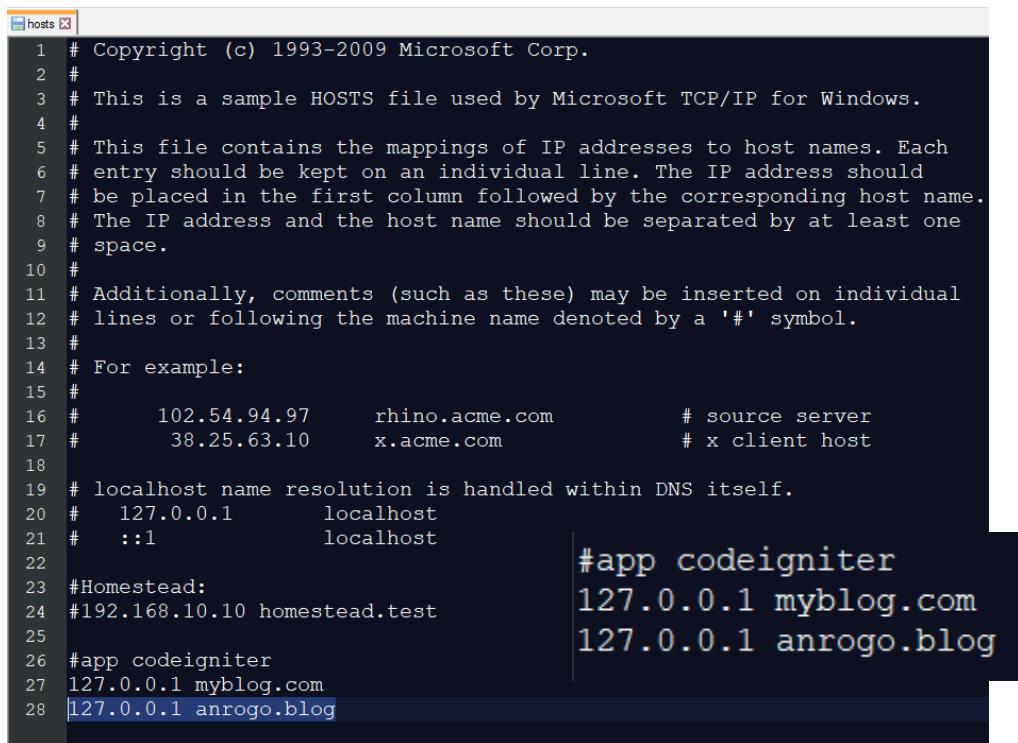
AllowOverride All

Require all granted

```
<VirtualHost *>
    DocumentRoot "C:\xampp7\htdocs\myblog\application\webroot"
    ServerName myblog.com
    <Directory "C:\xampp7\htdocs\myblog\application\webroot">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>

<VirtualHost *>
    DocumentRoot "C:\xampp7\htdocs\blog\application\webroot"
    ServerName anrogo.blog
    <Directory "C:\xampp7\htdocs\blog\application\webroot">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Además, de por supuesto añadir la siguiente línea en el fichero de hosts del sistema:



```
hosts
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #      102.54.94.97      rhino.acme.com      # source server
17 #      38.25.63.10      x.acme.com          # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 #      127.0.0.1      localhost
21 #      ::1            localhost
22 #
23 #Homestead:
24 #192.168.10.10 homestead.test
25 #
26 #app codeigniter
27 127.0.0.1 myblog.com
28 127.0.0.1 anrogo.blog
```

The screenshot shows the Windows hosts file being edited. A new section is added at the bottom:

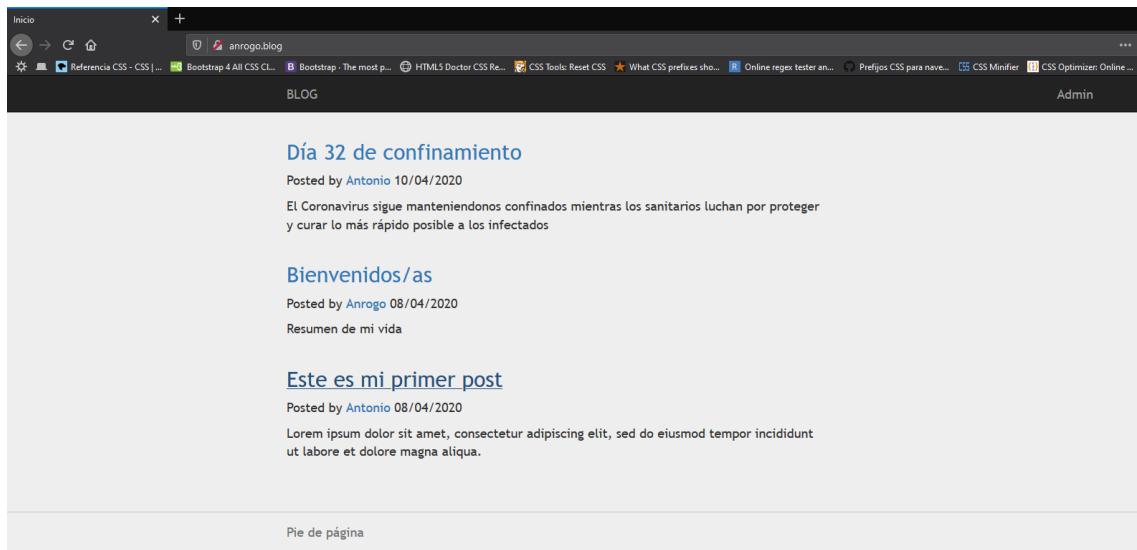
```
#app codeigniter
127.0.0.1 myblog.com
127.0.0.1 anrogo.blog
```

Este

fichero lo encontramos, en Windows, en: <C:\Windows\System32\drivers\etc>

Para esta aplicación he utilizado “anrogo.blog” pero cada uno puede añadir el dominio lo que sea que quiera. Mientras sea el mismo nombre que pusimos en el virtualhost (el ServerName que indicaba antes).

Veríamos lo siguiente:



Hay que tener cuidado porque de lo contrario cualquier mínimo error de configuración puede desembocar en este mensaje al intentar cargar nuestra aplicación:



ANEXO II

Lo segundo, detallar las narrativas de casos de uso que he diseñado para mi blog:

Narrativa de casos de uso

Caso de uso	<i>Acceso a la página inicio</i>
Actores	Usuario (no logueado), administrador
Precondición	Ninguna
Postcondición	Tampoco habría
Breve descripción	El usuario o administrador entra en la página de inicio del blog. Se encuentra la presentación de esta y tiene a su alcance diferentes opciones.
Escenario principal	Puede realizar unas acciones u otras dependiendo de si es un usuario previamente registrado, si es un usuario administrador o si simplemente es un usuario que entra por primera vez. -Un usuario público que no está registrado puede registrarse e iniciar sesión desde la página habilitada para ello. -Un usuario registrado puede iniciar sesión y volver a esta página. -Un usuario administrador, previo inicio de sesión, puede acceder al panel de control desde inicio. Por lo demás cualquiera puede buscar y ver los posts publicados.
Escenario alternativo	No habría

Caso de uso	<i>Inicio de sesión</i>
Actores	Usuario (no logueado), administrador
Precondición	Estar registrado en el blog e irse a la página del login
Postcondición	Se inicia sesión y se devuelve a la página inicial. Esta sesión se mantiene mientras no la cierre el mismo usuario manualmente o no exceda el límite de tiempo.
Breve descripción	El usuario o administrador introducen sus credenciales y acceden a su cuenta del blog. Se encuentran en la página de inicio y tienen a su alcance diferentes opciones.
Escenario principal	Puede realizar unas acciones u otras dependiendo de si es un usuario previamente registrado o si es un usuario administrador. -El usuario ya registrado y con la sesión iniciada puede ver, clasificar y comentar los posts de videojuegos; además de crear posts personalizados desde la página de posts de usuarios. También puede editar la información personal de su perfil de usuario.
Escenario alternativo	Si escribe mal las credenciales, tanto correo como contraseña, recibe por pantalla un error. Si olvida la contraseña puede solicitar un cambio de la misma. A través del correo recibe un mensaje del administrador y desde ahí puede volver a la página y resetearla.

Caso de uso	<i>Registro</i>
Actores	Usuario (no logueado), administrador
Precondición	Rellenar el formulario solicitado
Postcondición	El usuario queda registrado en el blog.
Breve descripción	Se rellena el formulario de registro.
Escenario principal	El usuario o administrador introduce como mínimo los campos requeridos por el formulario de registro, se pulsa enviar y una vez es correcto se le envía al usuario un correo de confirmación, que debe aceptar y se le redirecciona a la página de inicio de sesión.
Escenario alternativo	<p>Si el usuario escribe mal algún campo recibe por pantalla un error que le insta a corregirlo.</p> <p>Si no acepta la casilla de políticas y condiciones del blog no puede registrarse.</p> <p>Hasta que el usuario no confirme desde su correo no queda oficialmente registrado.</p>

Caso de uso	<i>Página de perfil</i>
Actores	Usuario (registrado y logueado)
Precondición	El usuario debe estar registrado y logueado para poder acceder.
Postcondición	Si realiza cambios se actualizarán sus datos.
Breve descripción	El usuario tiene acceso a su información personal.
Escenario principal	<p>El usuario accede a sus datos y si selecciona editar puede cambiarla y posteriormente guardarla y actualizar. De la misma manera que en el registro estos campos serán verificados para que se guarden adecuadamente.</p> <p>Desde aquí el usuario podrá elegir eliminar su cuenta y sus datos. Si lo hace el administrador recibirá el aviso y confirmará la eliminación.</p>
Escenario alternativo	<p>Si el usuario escribe mal algún campo recibe por pantalla un error que le insta a corregirlo. Y si sale de esta página de edición, sin guardar sus cambios, éstos no tendrán efecto.</p>

Caso de uso	<i>Página de posts</i>
Actores	Usuario y administrador
Precondición	Acceder seleccionando la opción en la página de inicio
Breve descripción	Todos los usuarios independientemente de su rol o de si tienen la sesión iniciada pueden ver, clasificar o buscar cualquier post del blog.

Escenario principal	<p>Además de ver, clasificar u ordenar cualquier post se puede realizar unas acciones u otras dependiendo de si es un usuario previamente registrado o si es un usuario administrador.</p> <ul style="list-style-type: none"> -El usuario ya registrado y con la sesión iniciada puede ver, clasificar y comentar los posts de videojuegos; además de crear posts personalizados desde la página de posts de usuarios. También puede editar la información personal de su perfil de usuario. -Si el usuario no está registrado o lo está, pero no inicia sesión antes, únicamente puede buscar y ver posts, ya sea de videojuegos o de los creados por otros usuarios. No puede comentar o responder comentarios hasta que inicie sesión. -El administrador en cambio una vez inicie sesión puede irse a su panel de control/administración seleccionando esa opción.
----------------------------	--

Caso de uso	<i>Lista de autores</i>
Actores	Usuario, administrador
Precondición	Seleccionar un usuario que haya creado como mínimo 1 post
Breve descripción	Listado con los posts de un autor
Escenario principal	Aparece un listado con los post/s del autor previamente seleccionado. El usuario dispone de las mismas opciones que en la página de post: buscar, ordenar, elegir un post para verlo, etc. Si está registrado e inicia sesión antes, también puede comentar en estos posts de lo contrario si elige comentar se le pide iniciar sesión y cuando lo haga volverá a la opción de comentar.

Caso de uso	<i>Panel de administración</i>
Actores	Administrador
Precondición	Haber iniciado sesión, sino no tendrá acceso.
Postcondición	Seguirá activa su sesión de administrador hasta que la cierre o se cumpla un límite de tiempo.
Breve descripción	Menú exclusivo para el usuario administrador del blog.
Escenario principal	<p>El administrador desde aquí puede realizar funciones como:</p> <ul style="list-style-type: none"> -Dar de alta o baja de usuarios. Revisar reportes que puedan hacer. -Clasificar, ordenar, buscar, crear, editar y borrar posts. -Moderar, modificar y borrar comentarios en caso de ser ofensivos. -Ver autores de post. -Activar el modo mantenimiento del blog, restringiendo el acceso a todos los usuarios temporalmente.
Escenario alternativo	<p>El administrador podría olvidar sus credenciales de acceso, así que también puede seleccionar la opción para resetear su contraseña a través del mensaje al correo.</p> <p>En otro caso también podría darse que con el mantenimiento del blog activo, se cerrase la sesión del administrador y no pudiese acceder para restablecerlo. Por ello, la parte de administración tendrá su propio login aunque igualmente podrá acceder desde el inicio de sesión clásico.</p>

ANEXO III

Configuración del repositorio con github

Para empezar, he realizado la instalación de Git bash, un terminal que “emula” los entornos de Unix y Linux con el que poder trabajar en Windows. Me lo he descargado gratuitamente de la web: <https://git-scm.com/downloads> (versión 2.26).

Una vez iniciado, me he situado en mi carpeta del proyecto, que se encuentra en el directorio de C:\xampp\htdocs\argaming. Tras ejecutar git init para indicarle que se va a convertir en un directorio administrado por git, he añadido en el archivo .gitignore los ficheros y directorios que no me interesaba agregar al repositorio, y por último he agregado con “git add .” los demás archivos, imágenes, ... Que si voy a tener en mi repositorio del proyecto.

```
Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git add .
warning: LF will be replaced by CRLF in .editorconfig.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in .htaccess.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in system/.htaccess.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in system/core/Benchmark.php.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in system/core/CodeIgniter.php.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in system/core/Common.php.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in system/core/Config.php.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in system/core/Controller.php.
```

Una vez ha terminado simplemente he realizado mi primer commit para confirmar que quiero que estos archivos pasen de mi “directorio de trabajo” a mi “área de trabajo”, con este comando:

```
git commit -m "origen"
```

```
Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git commit -m "commit origen"
[master (root-commit) ff32848] commit origen
 316 files changed, 81877 insertions(+)
 create mode 100644 .editorconfig
 create mode 100644 .gitignore
 create mode 100644 .htaccess
 create mode 100644 application/cache/index.html
 create mode 100644 application/config/autoload.php
 create mode 100644 application/config/config.php
 create mode 100644 application/config/constants.php
 create mode 100644 application/config/database.php
 create mode 100644 application/config/doctypes.php
 create mode 100644 application/config/foreign_chars.php
 create mode 100644 application/config/hooks.php
 create mode 100644 application/config/index.html
 create mode 100644 application/config/memcached.php
 create mode 100644 application/config/migration.php
 create mode 100644 application/config/mimes.php
 create mode 100644 application/config/profiler.php
 create mode 100644 application/config/routes.php
 create mode 100644 application/config/smileys.php
 create mode 100644 application/config/user_agents.php
 create mode 100644 application/controllers/AdminController.php
 create mode 100644 application/controllers/LoginController.php
 create mode 100644 application/controllers/UserController.php
 create mode 100644 application/controllers/Welcome.php
 create mode 100644 application/controllers/index.html
```

Después he comprobado que no quedaban archivos sin añadir con:

```
git status
```

```
Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

A continuación, he procedido a subirlo a mi repositorio “Proyecto_final_daw”, de Github.

Antes de nada, he tenido que configurar el correo y usuario asociados a mi cuenta.

```
git config --global user.email <correo> y git config --global user.name <nombre>
```

He comprobado por última vez el commit realizado, con `git log` donde se almacena el historial de commit en cada rama o branch, y he ejecutado

```
git remote add origin https://github.com/Anrogo/proyecto_final_daw
```

```
Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git config --global user.email "antoniorogo96@gmail.com"

Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git config --global user.name "Anrogo"

Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git status
On branch master
nothing to commit, working tree clean

Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git log
commit ff3284802b631e4e607df5b933badfa4112384d6 (HEAD -> master)
Author: Proyectos_Antonio <antoniorogo96@gmail.com>
Date:   Wed May 6 17:50:51 2020 +0200

    commit origen

Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git remote add origin https://github.com/Anrogo/proyecto_final_daw

Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ git push -u origin master
Enumerating objects: 323, done.
Counting objects: 100% (323/323), done.
Delta compression using up to 4 threads
Compressing objects: 100% (320/320), done.
Writing objects: 100% (323/323), 1.22 MiB | 246.00 KiB/s, done.
Total 323 (delta 106), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (106/106), done.
To https://github.com/Anrogo/proyecto_final_daw
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

Antonio@ANTONIO-HP MINGW64 /c/xampp/htdocs/argaming (master)
$ |
```

Entrando a https://github.com/Anrogo/proyecto_final_daw se pueden ver mis archivos de la aplicación.

The screenshot shows the GitHub repository page for 'Anrogo / proyecto_final_daw'. At the top, there are links for Pull requests, Issues, Marketplace, and Explore. Below the header, it says 'No description, website, or topics provided.' and 'Manage topics'. It displays 1 commit, 1 branch, 0 packages, 0 releases, and 1 contributor. The commit list shows a single entry from 'Anrogo' with the message 'origen' made 6 minutes ago. At the bottom, there's a note to 'Add a README'.

Tras el último commit, el repositorio ha quedado de esta forma:

The screenshot shows the same GitHub repository page after several commits. A prominent 'Join GitHub today' modal is displayed in the center. Below the modal, it says 'No description, website, or topics provided.' and 'Manage topics'. It shows 37 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. The commit list includes multiple entries from 'Anrogo' with messages like 'más correcciones de última hora', 'commit origen', and 'probando form validation', all made within the last 8 hours.

ANEXO IV

Errores y problemas durante el proyecto

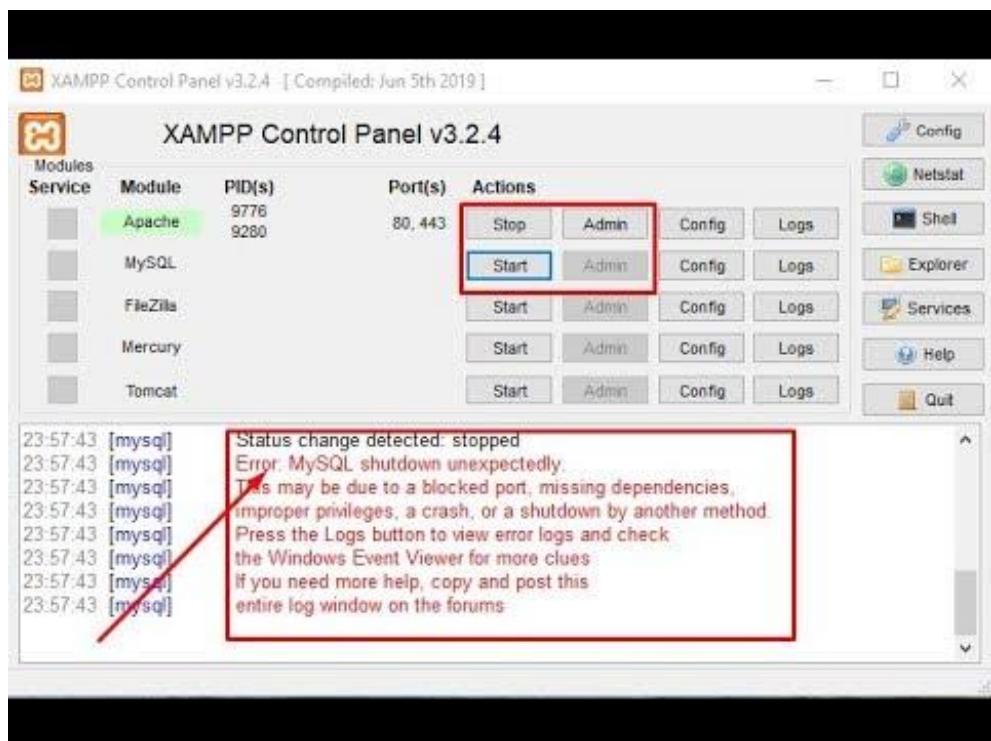
Recopilación de los errores que me he ido encontrando durante el desarrollo del proyecto web.

- Para empezar, no sabía como funcionaba CodeIgniter y lo primero fue realizar un curso básico sobre el mismo. En el cuál fui siguiendo los ejemplos y tratando de imitarlos, pero me topé con un problema, hay muchas versiones de este framework, la más reciente es la 4 y en el curso usaban las 3.11. Así que, pensando que no importaba seguí los pasos que me marcaban con la versión 4 del mismo. Al cabo de 5 vídeos me di cuenta de que no me cuadraban ciertos aspectos de su proyecto en comparación con el mío, puesto que las diferencias no son exageradas, pero si se aprecian claramente, y como no soy un “maestro” de este framework todavía decidí pasarme a la versión 3.11 y realizar mi proyecto a partir de ésta para ahorrarme una dificultad añadida. En esta web se pueden encontrar algunas de las diferencias entre ambas versiones:
<https://dev.to/jonathanlamim/10-differences-between-codeigniter-3-and-4-5526>
- Solo fue el principio, cuando todo marchaba bien me encuentro con la problemática de crear un host virtual en Windows 10 con el Apache de Xampp (solo lo había hecho en Linux con el propio Apache). Tras darme algunos recursos el profesor, el encargado del curso de CodeIgniter, pude continuar y todo marchó bien hasta completar el blog de ejemplo que se desarrollaba en este breve curso.

El host virtual funcionaba, pero más adelante tuve que realizar ciertos cambios de directorios, permisos, el htaccess que él aconsejaba, etc. Y finalmente quedó como he explicado en el anexo anterior de configuración de CodeIgniter y del dominio local mediante host virtual con Apache.

- Tras tres semanas de proyecto, de repente phpMyAdmin un día, sin que yo tocase nada de su configuración o ajustes, dejó de funcionar y empezó a mostrar errores por todas partes. Al no encontrar ninguna solución válida para este problema me vi obligado a reinstalar Xampp desde cero y volver a configurar CodeIgniter y Apache. Además de restaurar la base de datos de mi proyecto por completo. Me costó un día entero, pero pude seguir adelante.
- Y hasta unos días después no tuve más dificultades extra. Si hay que remarcar que de nuevo la base de datos de Xampp me dio problemas, esta vez desde su Panel de Control. Debido a que al intentar arrancar MySQL se detenía seguidamente mostrando un error en la pantalla inferior:

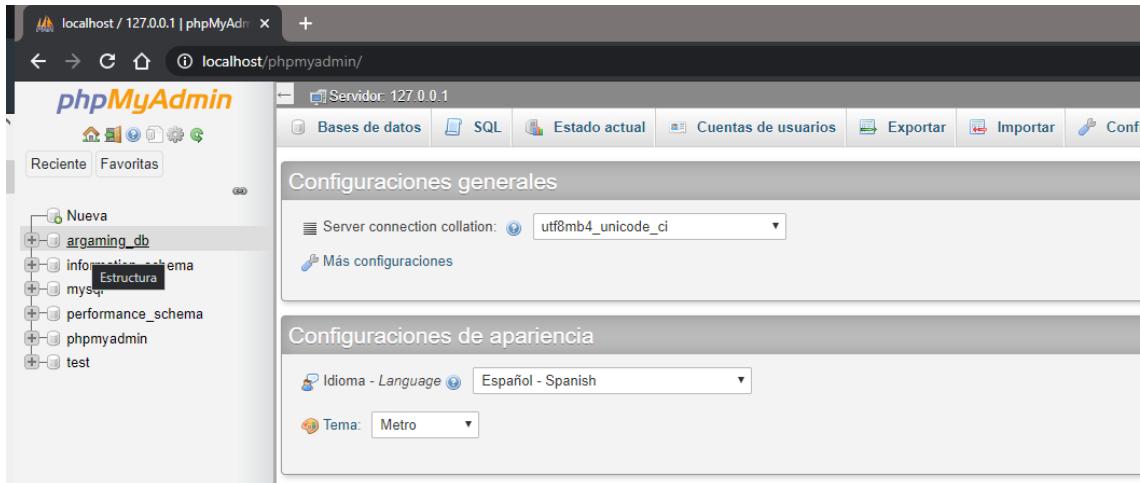
La



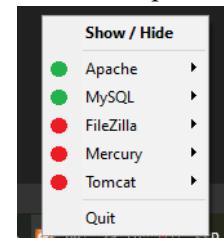
solución fue simple y práctica, recuperar sus archivos originales mediante la carpeta backup, contenida dentro del mismo directorio de C:/xampp/mysql.

Nombre	Fecha de modificación	Tipo	Tamaño
backup	04/05/2020 19:48	Carpeta de archivos	
bin	04/05/2020 19:50	Carpeta de archivos	
data	18/05/2020 22:56	Carpeta de archivos	
scripts	04/05/2020 19:47	Carpeta de archivos	
share	04/05/2020 19:48	Carpeta de archivos	
COPYING	10/12/2019 14:47	Archivo	18 KB
CREDITS	10/12/2019 14:47	Archivo	3 KB
EXCEPTIONS-CLIENT	10/12/2019 14:47	Archivo	9 KB
mysql_installservice.bat	30/03/2013 13:29	Archivo por lotes ...	1 KB
mysql_uninstallservice.bat	30/03/2013 13:29	Archivo por lotes ...	1 KB
README.md	10/12/2019 14:47	Archivo MD	4 KB
resetroot.bat	03/06/2019 13:39	Archivo por lotes ...	2 KB
THIRDPARTY	10/12/2019 14:47	Archivo	85 KB

El contenido de la carpeta backup se pasa a data y se sobreescreiben los archivos. Luego borré la base de datos y la volví a generar, desde mi copia de seguridad argaming_db.sql (que ya tenía lista por si pasaba algo por el estilo).



Tras muchos días sin apenas incidencias, últimamente no me ejecutaba el visor de gráfico de Xampp y he tenido que echarle paciencia hasta que me dejaba activarlo desde sus “opciones reducidas”



También volví a sufrir el fallo de PHPMyAdmin y tuve que recuperar los datos originales de la carpeta de mysql/data para poder seguir trabajando...

Y tras estos diversos “paréntesis”, pude continuar desarrollando mi aplicación.