

# PROYECTO FIN DE CICLO DESARROLLO DE APLICACIONES WEB

## TIENDA ONLINE



**Por Alejandro Márquez Aragonés**



## **ÍNDICE**

---

### **1. RESUMEN**

### **2. INTRODUCCIÓN**

### **3. OBJETIVOS Y CARACTERÍSTICAS DEL PROYECTO**

### **4. FINALIDAD**

### **5. MEDIOS Y MATERIALES USADOS**

### **6. PLANIFICACIÓN DEL PROYECTO**

#### **6.1. BOCETO**

#### **6.2. ESPACIO DE TRABAJO (WORKFLOW)**

##### **6.2.1. ESTRUCTURA DE CARPETAS Y ARCHIVOS**

##### **6.2.2. GUÍA DE ESTILOS**

###### **6.2.2.1. COLORES**

###### **6.2.2.2. FUENTES**

##### **6.2.3. FUNCIONAMIENTO NIVEL USUARIO**

##### **6.2.4. FUNCIONAMIENTO NIVEL INTERNO**

### **7. FASE DE PRUEBAS**

#### **7.1. CASOS DE USO**

### **8. CONCLUSIONES Y TRABAJOS FUTUROS O MEJORAS**

#### **8.1. MEJORAS**

#### **8.2. CONCLUSIONES**

### **9. REFERENCIAS BIBLIOGRÁFICAS**



## 1. RESUMEN

Mediante este proyecto tenía la intención de abarcar la creación desde cero de una página web y tienda online. Desde su proceso de login, registro de usuario hasta el carrito de la compra de los productos. Por lo tanto, han sido precisos conocimientos de diversas tecnologías que detallo a continuación:

- **PHP** como lenguaje de scripts y de comunicación con servidor.
- **MYSQL** para extraer los datos de los productos y de los usuarios y así hacer uso de ellos según la necesidad.
- **AJAX/Javascript** para la interacción en la web en el lado cliente.
- **HTML** como lenguaje de marcado de etiquetas y composición de la estructura de las páginas web que verá el usuario.
- **CSS**, hojas de estilos de vital importancia para una navegación agradable a la vista y muy visual.

## 2. INTRODUCCIÓN

La presente memoria pretende describir detalladamente el proceso de creación de dicho proyecto mencionado, dícese pues de una página web con interacción de usuario y venta online.

Se explicarán por lo tanto todas las tareas que se han llevado a cabo desde la idea inicial hasta las pruebas y tests que se han realizado en cada sección de la página para poder comprobar su buen funcionamiento.

También habrá un capítulo especial para futuras mejoras o ampliaciones que se le puede hacer al proyecto con el fin de mejorar su usabilidad y rendimiento.

La tienda web, de nombre **E-FUTURE**, está dirigida principalmente a la venta productos de formación o cursos online. Con el auge de la formación de cursos digitales en los últimos años me he fijado en la base del proyecto en webs tan conocidas como [www.udemy.com](https://www.udemy.com) o [www.domestika.com](https://www.domestika.com), entre otras muy importantes.

Como estandarte del proyecto se ha intentado desarrollarlo de manera que el uso de la aplicación web sea intuitivo, ligero y nada engorroso para el usuario, aunque detrás haya un importante trabajo de programación (a pesar de las mejoras y/o ampliaciones que por tiempo no han podido ser realizadas).

### 3. OBJETIVOS Y CARACTERÍSTICAS DEL PROYECTO

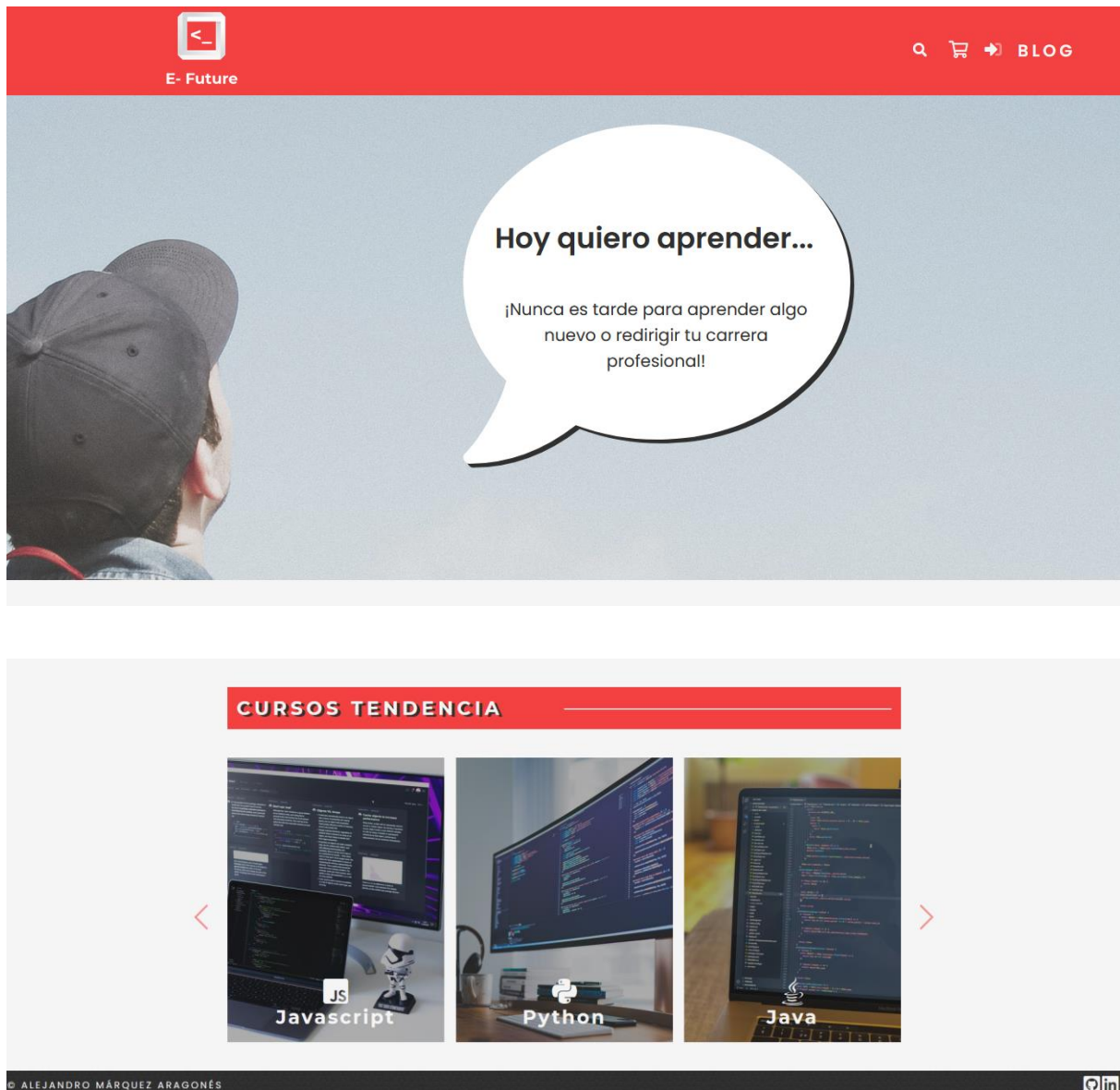


Imagen 1. Captura de pantalla proyecto principal

Sobre estas palabras adjunto imágenes del proyecto definitivo para facilitar la comprensión del objetivo que se pretende alcanzar con el desarrollo.

Como se puede ver fácilmente he creado una plataforma de venta de cursos digitales para poder disfrutar en una futura ampliación en la misma plataforma. Las características más importantes las numero a continuación con exhaustivo detalle:

- En el frontend se ha hecho un uso profesional de los lenguajes de maquetación y estilos html/css. La librería Bootstrap está incluida en los archivos locales del proyecto, pero finalmente y por facilitar la comprensión y futuros bugs, la he ignorado y por lo tanto no la he usado. El código está creado 100% desde cero excepto por partes de consultas en Internet que he adaptado a mis necesidades hasta un nivel de calidad alto. He intentado que predomine la calidad de las animaciones y la claridad visual para no saturar al usuario y que sepa claramente donde buscar cada item en la web.
- Javascript es uno de los corazones del proyecto web. Su uso de AJAX mediante el framework JQuery es crucial para la carga correcta de los componentes que se van a visualizar.
- PHP es de donde van a ir cogidos de la mano los datos que se manejarán entre el servidor y el cliente. La conexión con la base de datos creada en lenguaje MYSQL, otra parte esencial de la web se hace mediante librerías y scripts, los cuales se explicarán en su determinado momento.

Aunando esas tecnologías podemos resumir brevemente qué objetivo se alcanza:

1. Usuario navega por la web.
2. Decide registrarse, si no lo ha hecho con anterioridad.
3. Se realiza el login.
4. Tiene la opción de modificar datos del perfil que se ha creado.



5. Ver los cursos disponibles en la web.
6. Añadir al carrito los cursos y/o borrar del carrito todos o los que no se desee en concreto.

Como nota se ha de añadir que el proceso de compra no se ha completado puesto que no se ha dispuesto la creación de una pasarela de pago de prueba. La información será dada cuando se explique esa parte del proceso.

#### **4. FINALIDAD**

La creación de aplicaciones webs es como todo en programación, práctica y mucho de ensayo y error. Con este proyecto no se consigue alcanzar la mejor app web del mercado, pero sí conseguir dominar de la mayor y mejor manera posible todas las tecnologías de las que se han hecho uso, al menos al nivel de adaptación que necesita la aplicación.

Una vez alcanzada esa finalidad principal, no se debe olvidar la funcionalidad que se desea tener en el proyecto. He intentado crear una web activa, dinámica y con una interacción buscada del usuario para que no sólo se dedique a leer el contenido que haya sino que con el ratón vaya paseando por las secciones o iconos y vea que se esconde en cada rincón.

## 5. MEDIOS MATERIALES USADOS

La documentación previa y el estudio de lo que se quiere conseguir a la hora de abarcar un proyecto es crucial para poder conseguir el éxito y chocarse lo menos posibles con grandes errores o parones en el desarrollo. Hoy en día con la documentación física, pero sobre todo online, el desarrollo web se ha visto potenciado un 200%. Sitios web tan importantes como [Stackoverflow](#) o [W3Schools](#) son una guía básica para cualquier tipo de desarrollo actual. A continuación, detallo mis principales materiales utilizados para el proyecto, tanto software como documentación:

### - SOFTWARE:

- **Visual Studio Code** como IDE principal además de extensiones complementarias para facilitar el espacio de trabajo o workflow. Entre ellas las más importantes son “Live Server” para ver los cambios que realice en la web en tiempo real (ya sea en cliente o en servidor) y “Live Sass Compiler” que compilará a CSS los archivos SASS de estilos los cuales he utilizado para la parte visual de la web.
- **SASS**, consiste en un compilador de hojas de estilo que facilita el desarrollo de la maquetación gracias a los anidamientos, importaciones de archivos, variables e incluso estructuras de control. Gracias a ello la manipulación más personalizada de CSS es mucho más rápida, versátil y completa.
- **XAMPP** para crear el entorno servidor, con sus complementos para Apache/PHP y Mysql/PHPMyadmin. El primero para poder generar los scripts de comunicación con el servidor y ver los cambios en pantalla y el segundo para poder tener una base de datos en la que apoyar PHP y realizar todas las consultas necesarias.

- **Notepad++** como aplicación complementaria para realizar, editar o extraer notas necesarias, o copias y pegadas de código en caso de necesitar unas líneas extra por seguridad.
  - **GIT**, software de control de versiones básico para poder subir el código al repositorio alojado en [Github](https://github.com/reddevman/ProyectoFinal) (<https://github.com/reddevman/ProyectoFinal>).  
Con ello se consigue una mayor seguridad a la hora de realizar cambios y deshacer /controlar errores.
- **HARDWARE:**
- Portátil de uso personal con sistema operativo Windows 10.
- **DOCUMENTACIÓN:**

La detallo en el apartado correspondiente al final de esta memoria.

## 6. PLANIFICIACIÓN DEL PROYECTO

### 6.1 BOCETO

La planificación se ha realizado según distintos pasos. Primeramente, pinté en papel un esquema básico y a partir de ahí concretar la maquetación principal e intentar abarcar con el tiempo disponible la mayor parte posible.

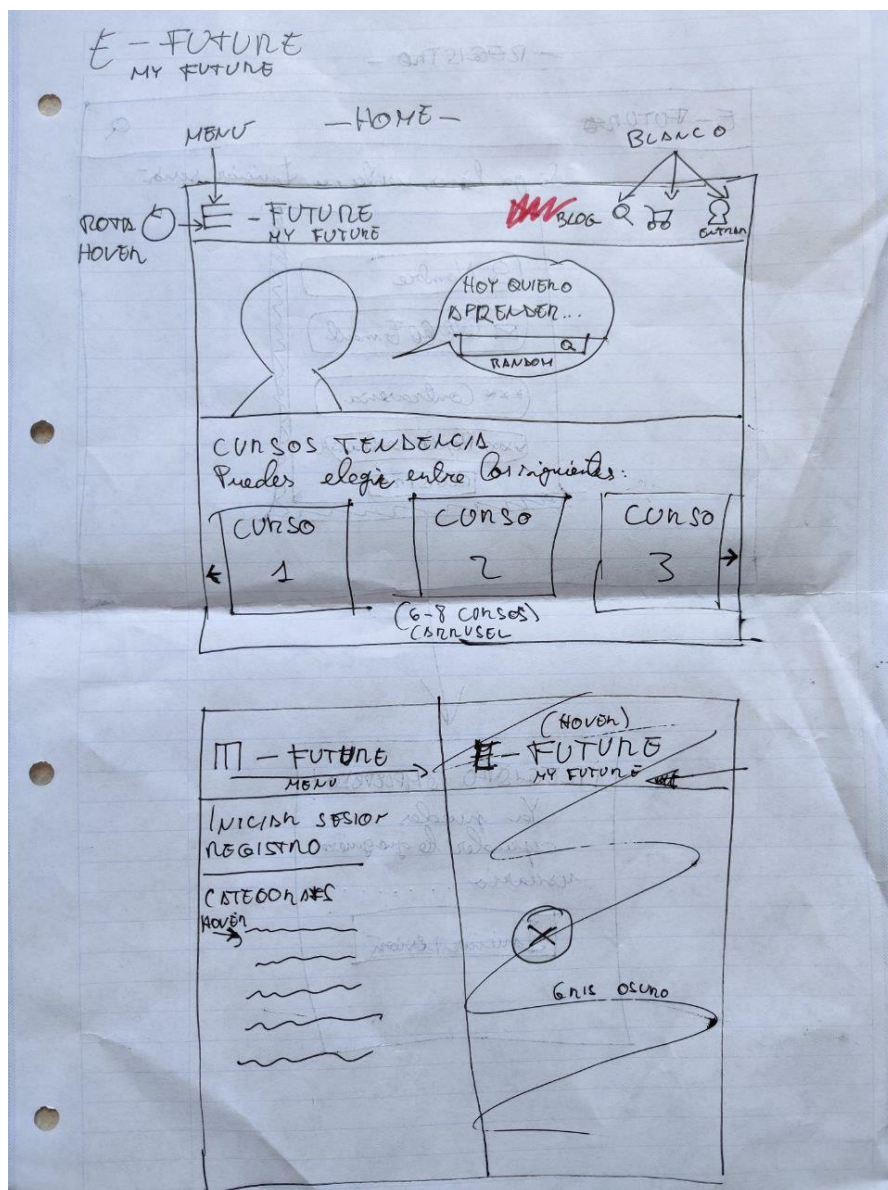


Imagen 2. Boceto 1/2

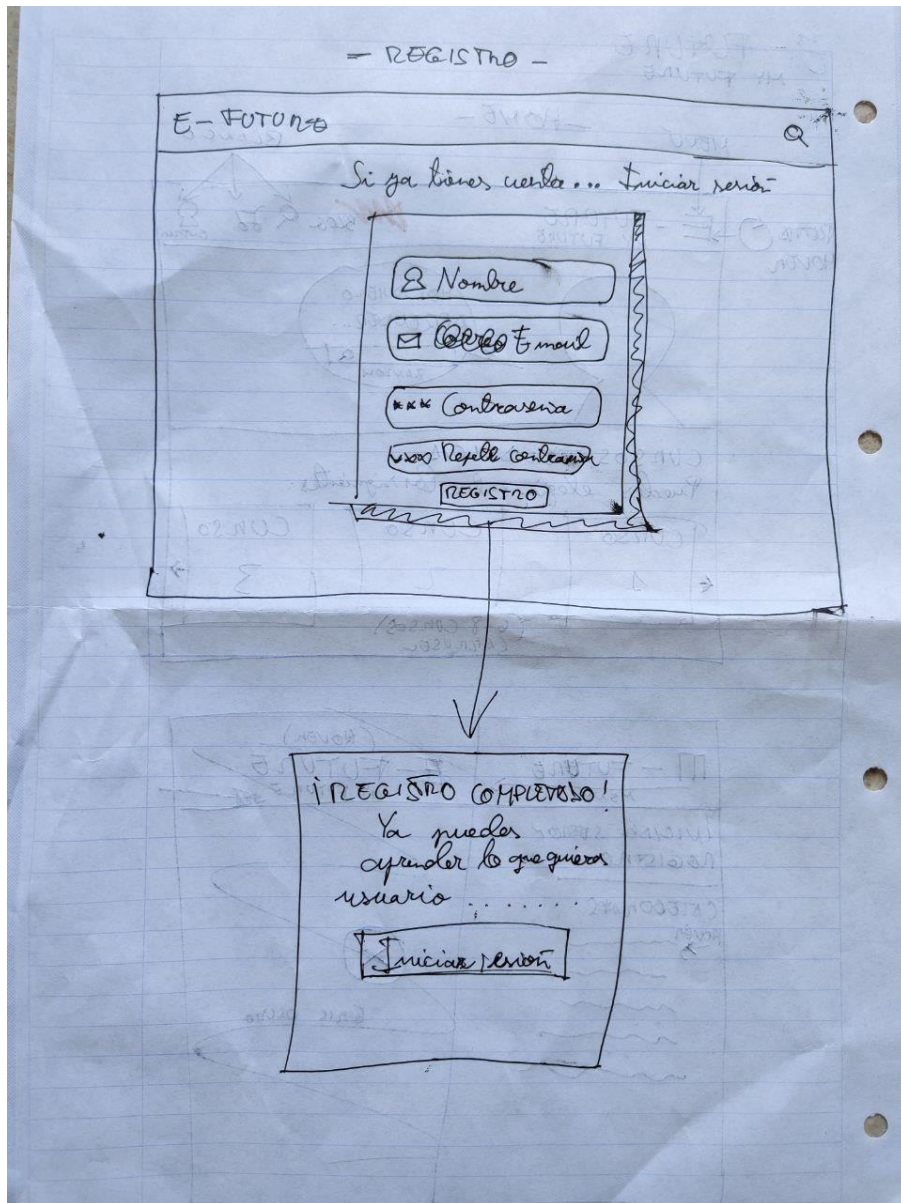


Imagen 3. Boceto 2/2

En las fotografías adjuntas se ve claramente el esqueleto principal de la aplicación cuyo home y login intenté reflejar lo más exacto posible para que con posterioridad fuera más fácil maquetar el resto de páginas adyacentes.

Cuando se realiza un boceto se intenta ajustar a las marcas de tiempo establecidas (fecha de entrega del proyecto 21 de mayo de 2021) pero hay ocasiones que por factores externos o por complicaciones en el desarrollo a veces se deben dejar cosas atrás para posibles mejoras y aplicaciones las cuales irán detalladas en su apartado correspondiente de esta memoria.

## 6.2 ESPACIO DE TRABAJO (WORKFLOW)

Abordar un proyecto, por pequeño que sea, siempre requerirá de una estructura y conexiones entre archivos que tengan una gran usabilidad, reutilización y que en un futuro sean fáciles de comprender en caso de que un programador ajeno (o uno mismo) retome el proyecto dentro de un tiempo indeterminado. Es por ello que he tomado como base la estructura de árbol que en muchos desarrollos se suelen utilizar, pero adaptada a mi proyecto.

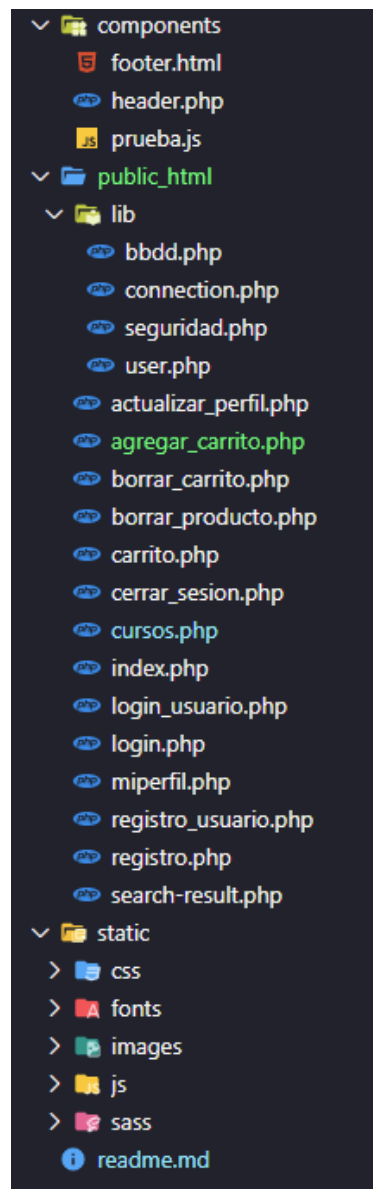


Imagen 4. Árbol de estructura principal

### 6.2.1 ESTRUCTURA DE CARPETAS Y ARCHIVOS

- **/components:** esta carpeta se crea para recoger todos los componentes que se verán repetidos en las distintas páginas de la aplicación. En el caso de este proyecto son en un principio **header** y **footer**.
  - **Header:** va a ser la cabecera que compartirán todas las páginas. Contendremos el logo, buscador, carrito de la compra, login, perfil de usuario, salir de la sesión, y un link al blog.



Imagen 5. Header

El logo fue generado mediante una aplicación web que genera logos preestablecidos (<https://hatchful.shopify.com/es/>) cuyo uso es muy práctico para este tipo de proyectos personales y que no van a salir a un ámbito de producción real ya que normalmente el logo ha de ser diseñado por uno mismo. Bien es cierto que el logo base ha sido editado para poder adecuarse a los requisitos del boceto.



Imagen 6. Logo



Para el resto de los iconos se hizo uso de la librería Font Awesome Pro (<https://fontawesome.com>) ya que proporciona gran cantidad de iconos de calidad y muy personalizables en CSS.

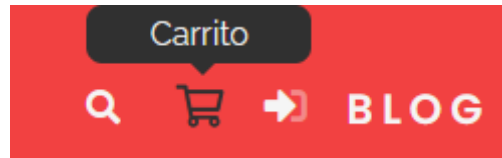


Imagen 7. Iconos header

Como se puede observar todos los iconos poseen un efecto *hover* que al situar el ratón encima cambia de color (los colores y sus variables las especificaré en la parte de código que pertenece al archivo correspondiente). A ello se le suma el *tooltip* o información sobre la herramienta que indicará al usuario qué significa dicho icono en caso de que no entienda su significado, ello ayudará también en cierta medida a ampliar la accesibilidad de público de la página.

```
<div class="header-full-wrapper">
  <div class="header-content container12">
    <!-- HEADER LOGO -->
    <div class="logo-wrapper">
      <a href="index.php">
        
```

Javascript se usa en este caso para darle acción al botón de buscar y que cuando hagamos click encima se abra el cuadro de búsqueda y podamos hacer una consulta entre los cursos disponibles y que cuando pulsemos en buscar en una de las sugerencias que nos aparezcan se rediriga a la web que queramos (en este caso la de descripción de los cursos `cursos.php`). La función que se encarga de ello es la llamada “*autocomplete*” y a continuación muestro un trozo de esta.

```
function autocomplete(inp, arr) {  
  var currentFocus;  
  inp.addEventListener('input', function(e) {  
    var a,  
        b,  
        i,  
        val = this.value;  
    closeAllLists();  
    if (!val) {  
      return false;  
    }  
  })
```

- **Footer:** El footer no tiene más que una posición fijada a la parte inferior de la página y links a mis perfiles de Github y LinkedIn.
- **/public\_html:** contendrá todos los archivos principales de la web. Desde el index.php hasta los archivos intermediarios que se usan para el registro o el carrito. Dentro de dicha carpeta hay otra llamada **/lib** en la que se hayan los archivos controladores que se encargan de la seguridad de la sesión(seguridad.php), la conexión a la base de datos (connection.php), la creación del objeto usuario(user.php) y las funciones de consultas sql(bbdd.php). Dicha distribución está basada en el Modelo Vista Controlador (MVC) y en la programación orientada objetos con el fin de conseguir mayor encapsulación además de las múltiples ventajas que dan los objetos.
- **/static:** Tiene gran parte del núcleo de la aplicación, los archivos estáticos que no serán cacheables. La carpeta **/sass** contiene los archivos .scss que se compilarán para luego ser transformados a hojas de estilos css que serán almacenadas en la carpeta **/css** que luego podrá interpretar fácilmente cualquier navegador compatible. De hecho, se han establecido configuraciones predeterminadas para que haya prefijos que aseguren dicha compatibilidad a versiones antiguas y/o modernas de navegadores web. El directorio **/js** contiene las librerías Javascript utilizadas que han sido Owl Carousel, JQuery, Font Awesome y el script principal nombrado mainScript.js. Y ahora he de detenerme para explicar una parte **crucial** de la distribución y configuración que se ha mantenido para armar la aplicación.

Vamos a tener 2 archivos que prácticamente son los que consiguen que todo el estilo y componentes de la página se armen y hagan un todo.

- **mainStyles.scss**, su función es realizar la importación de todas las hojas estilos conjuntas que usaremos para no repetir el código constantemente. Es decir, se harán import de los estilos que tengan las secciones que

maquetamos además de las variables de sass que se usan para las fuentes tipográficas y los colores. Además, aplicamos un reset universal para todas las páginas que hagan uso de mainStyles y unos estilos generales con la clase **container12** (mencionada más arriba) y el tamaño y color de fondo que queramos que tenga el body.

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

// basic imports
@import './fonts';
@import './colors';

// sections
@import './header';
@import './footer';
@import './index';
@import './login';
@import './loginUsuario';
@import './courses';
@import './carrito';

@viewport{
  zoom: 1.0;
  width: extend-to-zoom;
}

.container12 {
  width: 1140px;
  margin: 0 auto;
}

html,
body {
  height: 100%;
  min-height: 100vh;
  background-color: $blanco;
}
```

- **mainScript.js**, nos va a permitir mediante la función de JQuery `$.load()` cargar footer y header en todas las páginas que contengan un link a dicho script. Además, he incorporado una pequeña configuración de Ajax para no tener problemas de caché a la hora de cargar la web.

```
$.ajaxSetup({
    cache: false,
});

// Carga del componente html y comprobación en caso de error
$('#header_container').load(
    '/ProyectoFinal/components/header.php',
    function (response, status, xhr) {
        if (status == 'error') {
            var msg = 'Sorry but there was an error: ';
            $('#error').html(msg + xhr.status + ' ' + xhr.statusText);
        }
    }
);
```

- **/images** lógicamente contendrá todas las imágenes almacenadas en local y algunas a las que la base de datos se referirá a la hora de armar la sección de los cursos.
- **/fonts** la cree en caso de que necesitara almacenar localmente las fuentes que he utilizado en el proyecto, pero gracias al servicio de Google Fonts con un simple sass donde he almacenado las url a dichas fuentes y sus tamaños, no he necesitado dicha carpeta pero es posible que como medida de seguridad en una futura mejora las guarde localmente.

```
// TEXT0 1
@import url('https://fonts.googleapis.com/css2?family=Lato:ital,wght@0,100;0,300;0,400;0,700;0,900;1,100;1,300;1,400;1,700;1,900&display=swap');
$font1: 'Lato', sans-serif;
```

## 6.2.2. GUÍA DE ESTILOS

No es el propósito de esta memoria ser una guía de estilos de la web, pero sí quisiera mostrar lo más importante de los estilos utilizados como base que son tanto los colores como las tipografías utilizadas. Como nota adyacente quiero indicar que estos valores se han tomado como prioridad pero que es posible que durante el desarrollo haya habido alguna modificación como por ejemplo una fuente de títulos haya sido utilizada para un botón, etc.

### 6.2.2.1 COLORES



Imagen 9. Colores

```
$linksColor: #166b8c;  
$linksColor2: #26a699;  
$accentColor: #f2b035;  
$mainColor: #f24141;  
$secondaryColor: #f29191;  
$negro: #303030;  
$blanco: #f5f5f5;
```

Imagen 10. Variables de color

Como se puede observar arriba detallo los colores hexadecimales y las variables con un nombre identificativo a utilizar en la aplicación web. Como extra es importante destacar que los negros y blancos utilizados no son los puros #000000 ni #ffffff. He elegido otras tonalidades más agradables para la vista y la lectura que son las que se pueden observar en las variables \$negro y \$blanco. Esas las he utilizado por ejemplos en efectos *hover*.

### 6.2.2.2 FUENTES

Google Fonts (<https://fonts.google.com/>) proporciona un gran servidor de fuentes de uso gratuito que actualmente son tendencia en el desarrollo web. Las elegidas son las que he visto oportunas por su gran legibilidad y atractivo visual, además de ser tendencia en el desarrollo web actual.

#### Texto 1

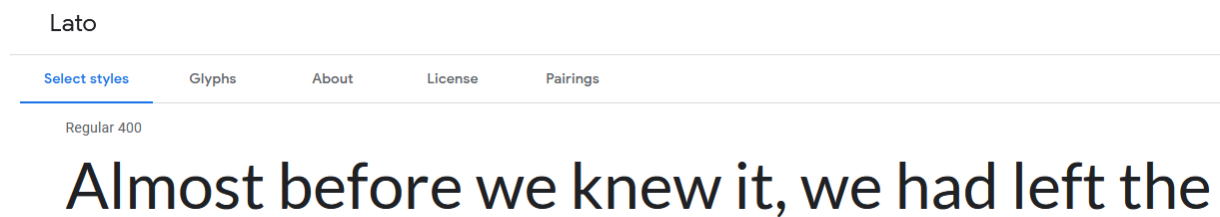


Imagen 11. Fuentes

#### Títulos 1

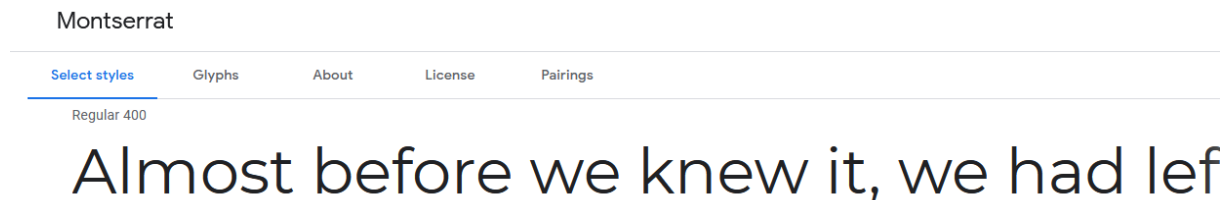


Imagen 12. Fuentes

#### Links

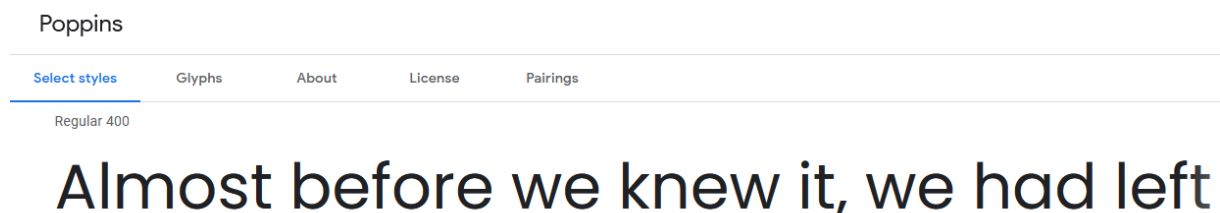


Imagen 13. Fuentes

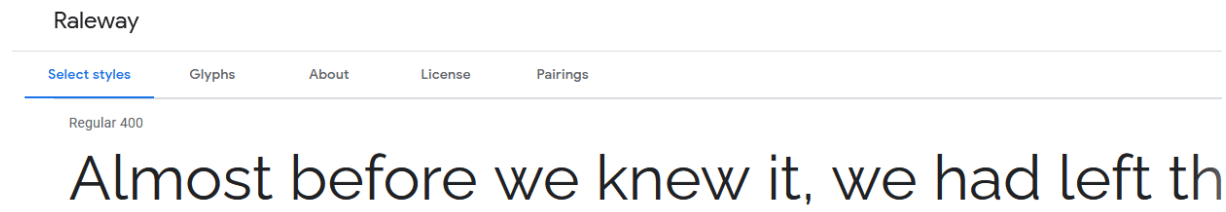


Imagen 14. Fuentes

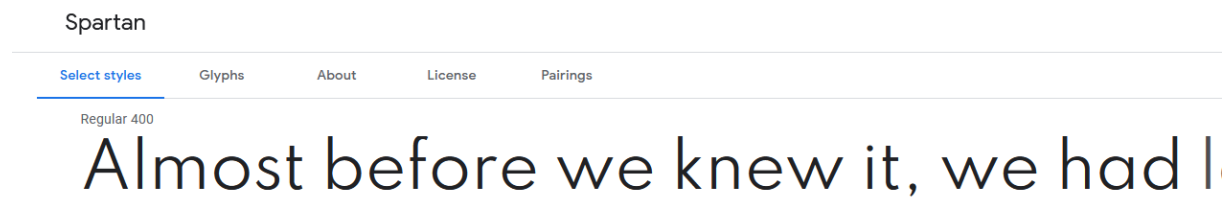
**Texto 2**

Imagen 15. Fuentes



### 6.2.3 FUNCIONAMIENTO NIVEL USUARIO

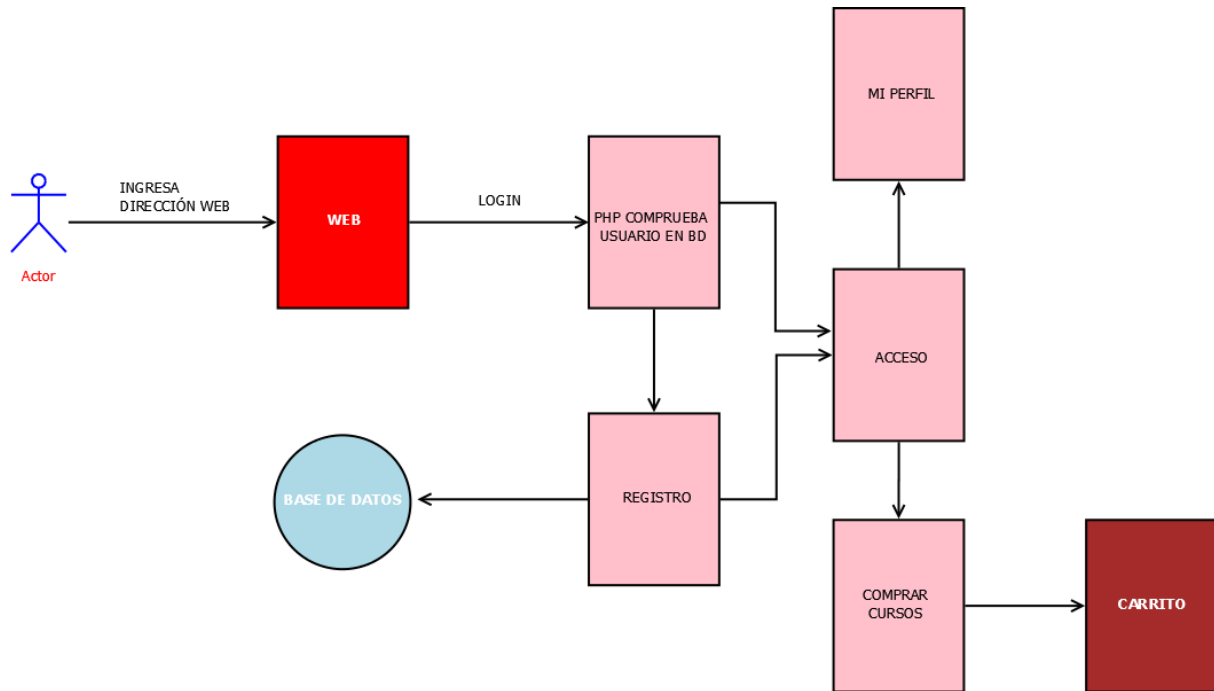


Imagen 16. Diagrama

El funcionamiento de una aplicación web es básicamente los pasos que queremos que siga el usuario cuando accede a la misma. Arriba detallo un sencillo diagrama con pasos básicos que contiene mi web, pero a continuación explico en mayor cantidad de palabras y exhaustivamente qué es lo que hace cada archivo en su proceso, aunque algo he llegado a mencionar con anterioridad es en este apartado donde se debe destacar de la manera más amena posible para quién lea esta memoria.

- Usuario accede a la dirección web implementada en el servidor. Se resuelve la ip y por lo tanto le aparece en pantalla en su navegador. Mientras navega de fondo Ajax va cargando en cada página el mismo footer y header. La única consulta que realizamos de momento a la base de datos es la que se hace cuando en la sección de los cursos aparecen

los mismos con su precio y es entonces cuando PHP se comunica con la base de datos para mostrar dicha información almacenada.

- Usuario decide hacer login, el header sólo muestra el icono de login ya que lógicamente la persona no está logueada. Suponiendo que no se haya registrado con anterioridad y no posea datos de acceso, decide darle al botón de registrar. Aparece pues una pantalla muy parecida a la del login pero con unos campos extras. Dicho formulario realiza una consulta de inserción en la base de datos y por lo tanto indica que se ha realizado el registro correctamente si no ha habido ningún error y da la opción de hacer login, una vez se realiza el login se da la bienvenida.

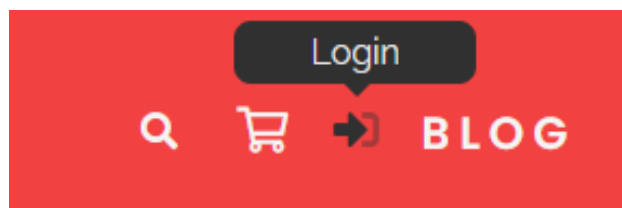
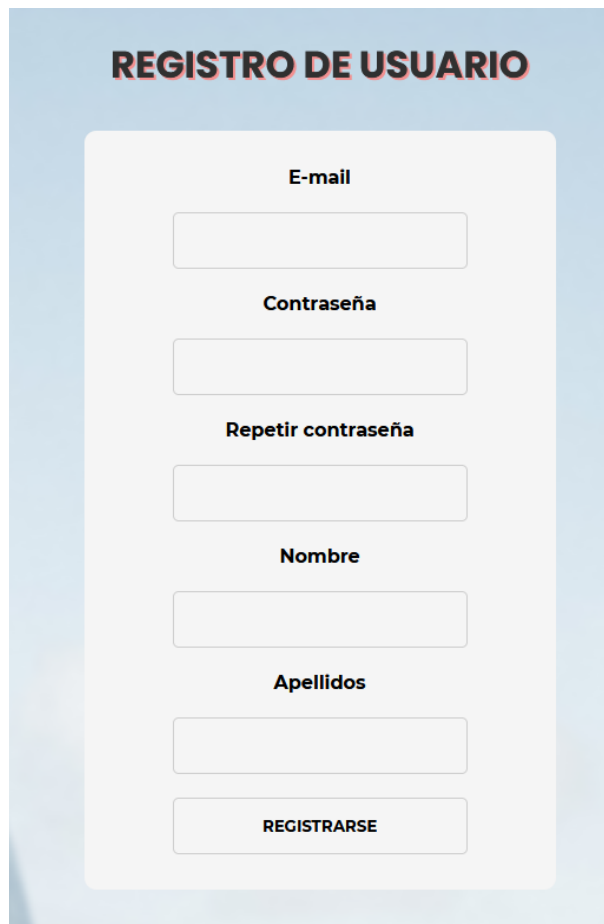


Imagen 17. Login



Imagen 18. Login



**REGISTRO DE USUARIO**

**E-mail**

**Contraseña**

**Repetir contraseña**

**Nombre**

**Apellidos**

**REGISTRARSE**

Imagen 19. Registro

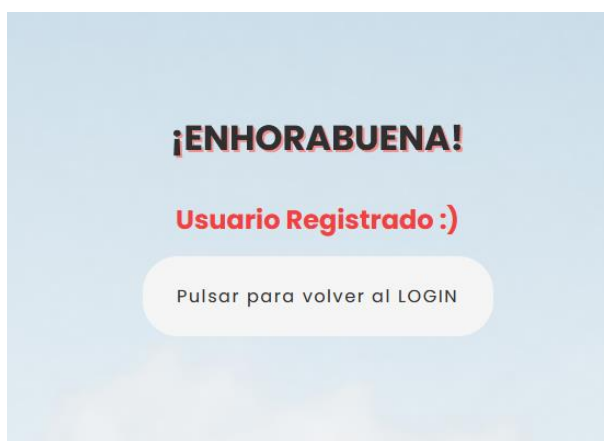


Imagen 20. Registro

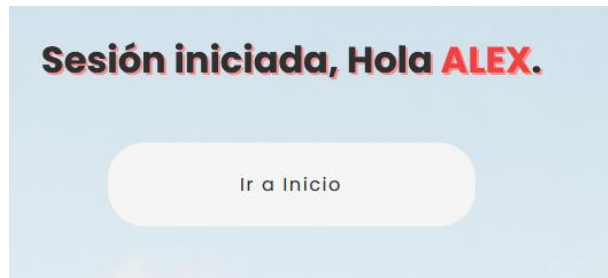


Imagen 21. Registro

- Es en este momento en el que el header cambia, no muestra ya el icono de login pero sí el de Mi perfil y el de Cerrar Sesión. Mi Perfil da la opción de editar datos del usuario y el segundo cierra la sesión para desloguear.

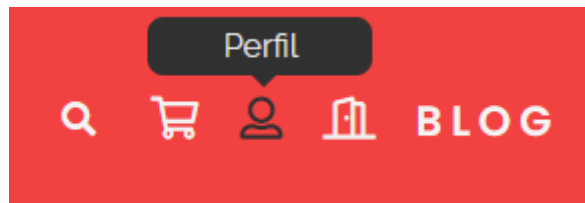
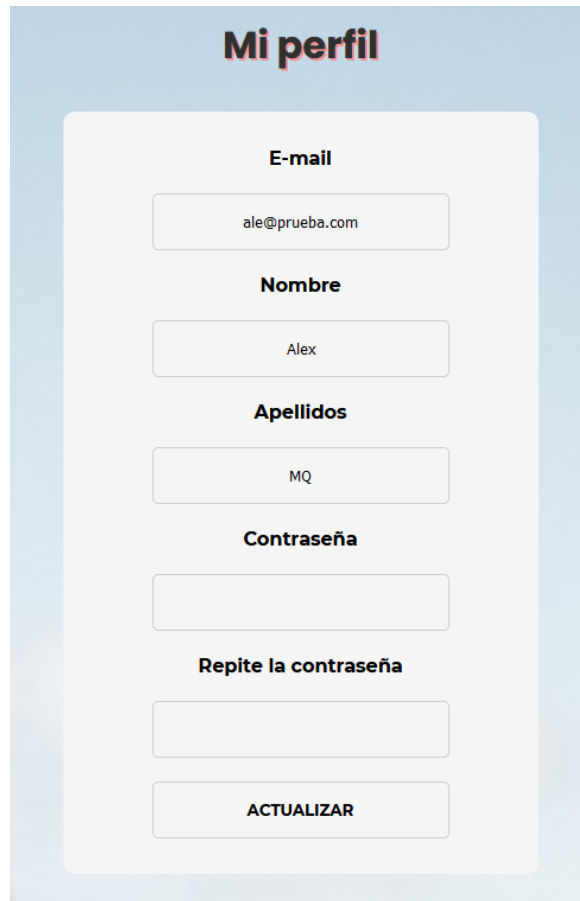


Imagen 22. Perfil



**Mi perfil**

**E-mail**

ale@prueba.com

**Nombre**

Alex

**Apellidos**

MQ

**Contraseña**

**Repite la contraseña**

**ACTUALIZAR**

Imagen 23. Perfil

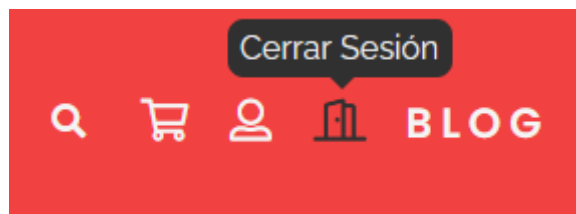


Imagen 24. Cerrar sesión

- Si se sitúa en la página de index (click en el logo principal del header) observamos un carrusel de cursos cuyo VER MÁS le lleva a la página de los cursos, allí al darle a “añadir al carrito” realiza dicha acción mediante el id de los cursos y los va mostrando en el carrito. Además, se da la opción de borrar un producto o todos del carrito.

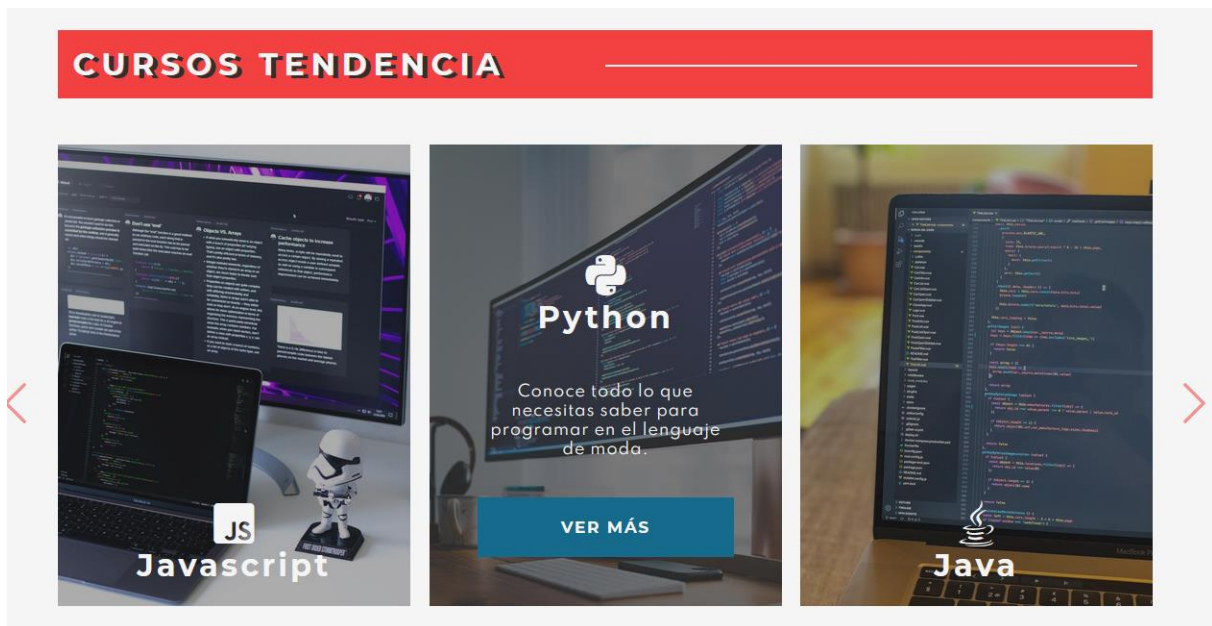


Imagen 25. Carrusel

---

## MÁSTER CERTIFICADO CSS



body {  
font: x-small  
background: #  
color: black;  
margin: 0;  
padding: 0;



**Oferta SÓLO hoy por 10.99€**

[AÑADIR AL CARRITO](#)

Desde animaciones nivel experto, compiladores tipo **SASS**, frameworks como **Bootstrap** y muchos más conocimientos son los que adquirirás en este curso.

---

## MÁSTER CERTIFICADO JAVA



Java



**Oferta SÓLO hoy por 10.99€**

[AÑADIR AL CARRITO](#)

Imagen 26. Cursos

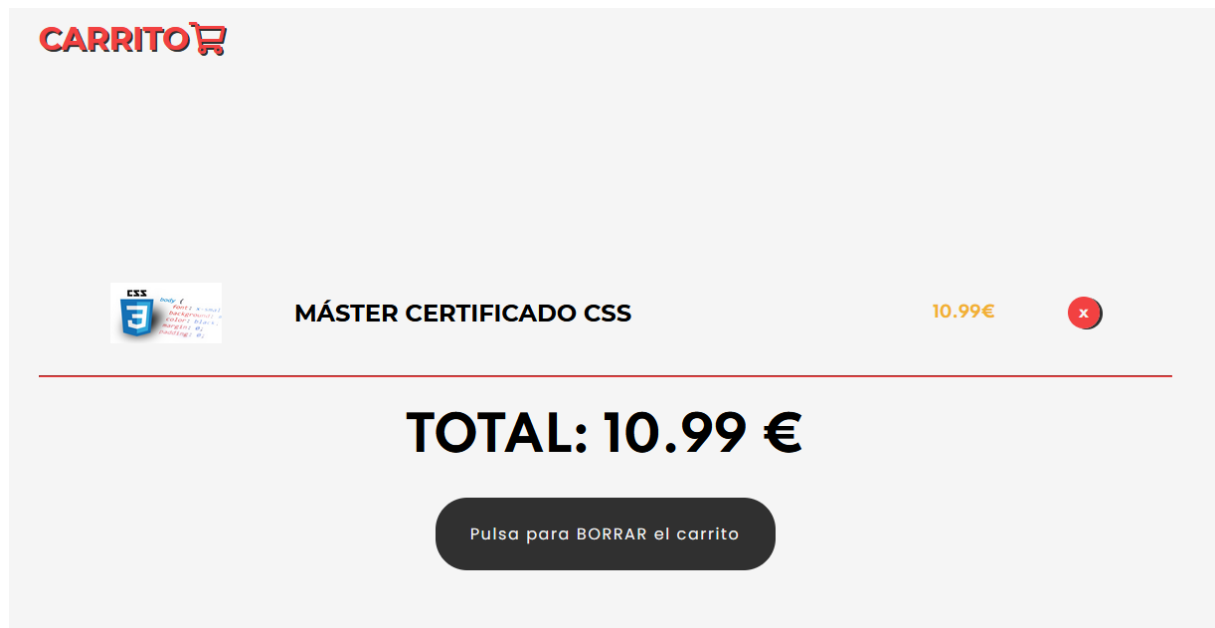


Imagen 27. Carrito

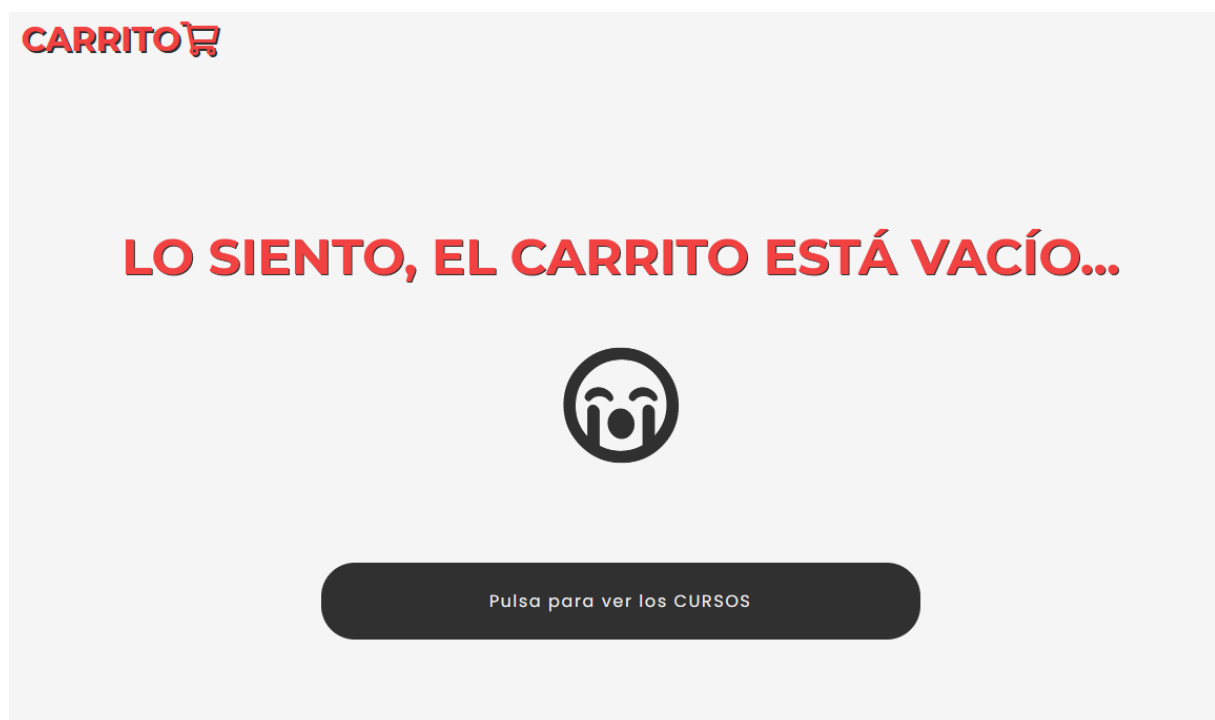


Imagen 28. Carrito

- Por último, hay que añadir que existe una función muy interesante a través de la cual el usuario puede buscar cursos y al hacerlo le lleva a la página del curso o cursos que haya elegido. Dicho buscador incluye sugerencias para que sea más intuitiva la elección.

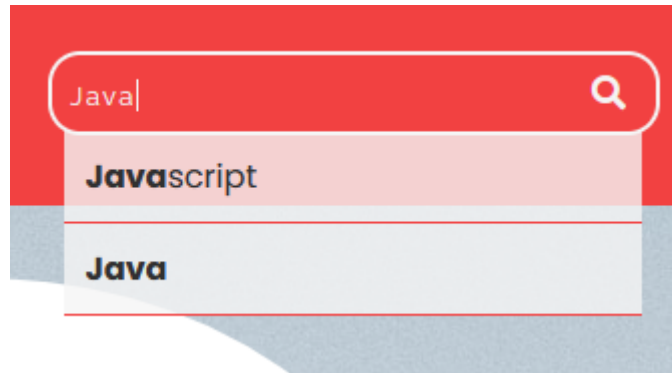


Imagen 29. Buscador



### 6.2.4 FUNCIONAMIENTO NIVEL PROGRAMADOR

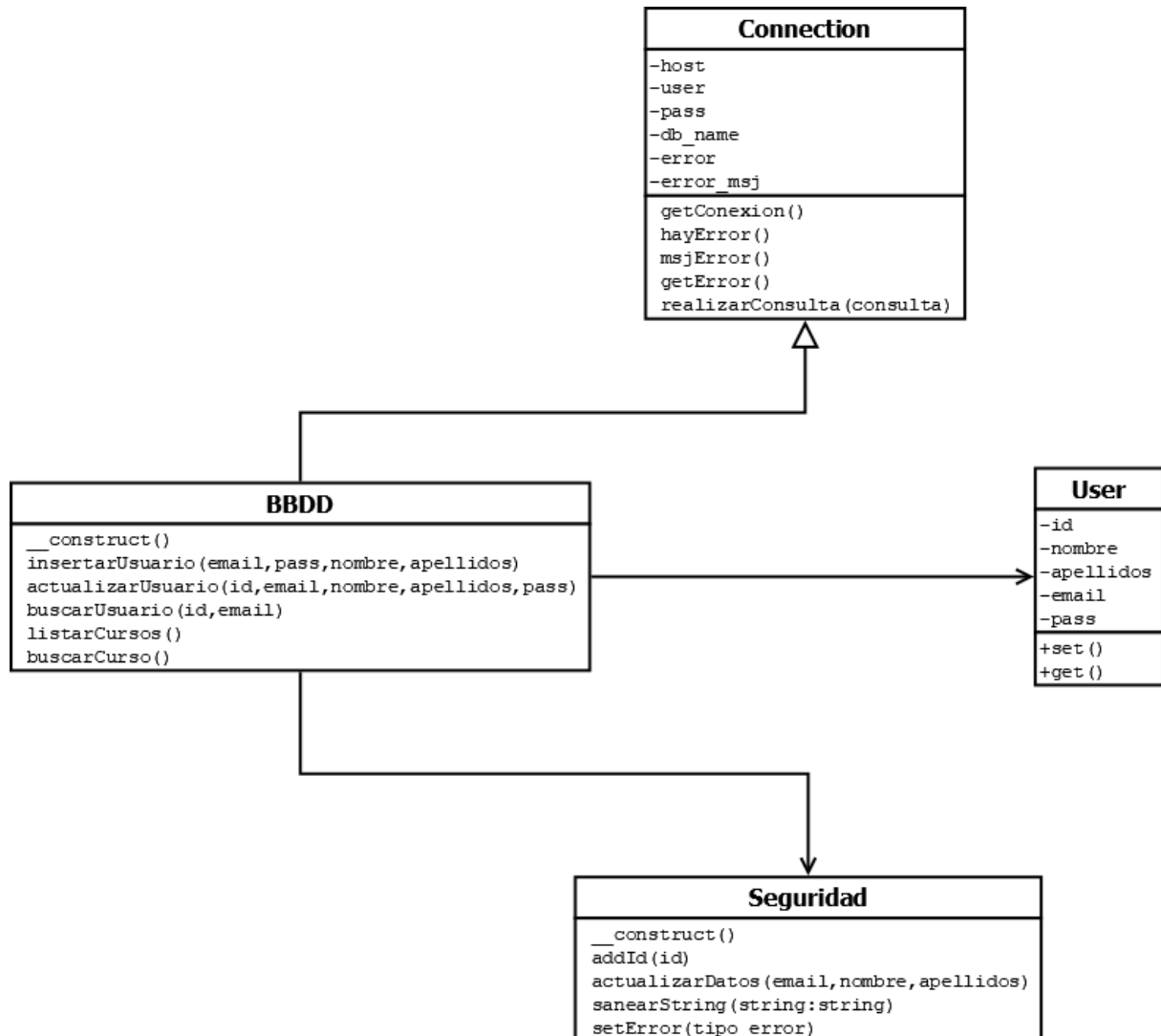


Imagen 30. Diagrama de clases

Guiándonos por el Modelo Vista Controlador para tener una aplicación eficiente y orientada a objetos se debe seguir una guía para la asociación de las clases. Arriba detallo dicho diagrama mediante la imagen adjunta.

A partir de la conexión a la base de datos es cuando se van construyendo todas las clases, pero la única que hereda es BBDD. Seguridad la usaremos para iniciar la sesión y almacenar las variables en el id de sesión. Luego User es la que hace funcionar la orientación a objetos ya que las funciones de BBDD contienen creación de objetos. Posteriormente los datos de cada

objeto User creado es lo que se usará para almacenar la id en la sesión, nombre para formularios, etc.

```
# FUNCIÓN INSERTAR
function insertarUsuario($email, $pass, $nombre, $apellidos)
{

    // Creación del usuario como un nuevo objeto
    $newUser = new User();
    // Hash de contraseña (encriptar)
    $pass_hash = password_hash($pass, PASSWORD_DEFAULT);

    // Las propiedades del objeto toman el valor de los parámetros
    // Serán los valores introducidos por el usuario en el formulario
    $newUser->setEmail($email);
    $newUser->setPass($pass);
    $newUser->setNombre($nombre);
    $newUser->setApellidos($apellidos);

    // Consulta Sql de inserción
    $sql = "INSERT INTO usuarios (email, nombre, apellidos, pass)
        VALUES ('" . $email . "', '" . $nombre . "', '" . $apellidos . "', '" .
$pass_hash . "')";

    // Capturar id de la consulta y añadirlo a la propiedad ID del usuario
    $resultado = $this->realizarConsulta($sql);
    if ($resultado) {
        $newUser->setId($this->getConexion()->insert_id);

        // Se devuelve el objeto
        return $newUser;
    } else {
        return null;
    }
}
```

Clase BBDD funciones haciendo uso de objetos

```

<?php

    if (
        isset($_POST['email']) && !is_null($_POST['email']) &&
        isset($_POST['pass']) && !is_null($_POST['pass'])
    ) {

        $pass = $_POST['pass'];

        $existUser = $bdd->buscarUsuario(NULL, $_POST['email']);

        if ($existUser != null) {

            if (password_verify($pass, $existUser->getPass())) {
                $id = $existUser->getId();
                $seguridad->addId($id);
                $_SESSION['nombre'] = $existUser->getNombre();
                $_SESSION['apellidos'] = $existUser->getApellidos();
                $_SESSION['email'] = $existUser->getEmail();
                $_SESSION['pass'] = $existUser->getPass();
            }
        }
    }

?>

```

Login\_usuario.php , creando la sesión con addId del objeto \$seguridad

```

<?php
class Seguridad
{

    # CONSTRUCTOR BÁSICO QUE ES EL QUE CREARÁ Y MANTENDRÁ LA SESIÓN
    function __construct()
    {
        session_start();
        if (isset($_SESSION['id'])) {
            $this->id = $_SESSION['id'];
        } else {
            $_SESSION['id'] = null;
        }
    }
}

```

Clase Seguridad con su constructor y el inicio de sesión cada vez que se crea un objeto de Seguridad

```
# FUNCIÓN PARA AÑADIR EL USUARIO A LA SESIÓN
public function addId($id)
{
    $_SESSION["id"] = $id;
    $this->id = $id;
}
```

Clase Seguridad añadiendo id a la superglobal de \$\_SESSION

Los pasos detallados para las comunicaciones entre clases son los siguientes:

- login.php → login\_usuario.php (comprueba post, comprueba hash password, añade id a la sesión y busca usuario).
- miperfil.php (con variables de sesión inicializadas) → actualizar\_perfil.php (actualiza datos mediante sentencia sql).
- registro.php → registro\_usuario.php (comprueba si existe usuario e inserta nuevo usuario en la base de datos).
- cursos.php (envía id del curso obtenido de la base de datos) → agregar\_carrito.php (array de sesión y array con los ids de los cursos).
- borrar\_carrito.php (borra carrito entero al vaciar variable global de sesión).
- borrar\_producto.php (borra un producto en concreto por su comprobación de la id que se recibe por \$\_GET).

Por lo tanto, debemos destacar lo importante que es para el desarrollo de este proyecto los siguientes puntos a nivel de PHP:

- Sesiones mediante \$\_SESSION
- Obtención de datos con \$\_POST y \$\_GET.
- Orientación a objetos.
- Objeto mysqli para la conexión a la base de datos.
- Encapsulación.

## 7. FASE DE PRUEBAS

### 7.1 CASOS DE USO (problemas y soluciones)

Para el entorno de pruebas y configuración/puesta en marcha de la aplicación web es necesario un servidor local (XAMPP en Windows, LAMP en Linux) que contenga simplemente servidor MySQL en funcionamiento y PHP para los scripts de servidor.

CASOS DE USO	DESCRIPCIÓN
CU-001	Acceso a páginas exclusivas del usuario.
CU-002	Correcta repetición de contraseña.
CU-003	Introducir e-mail válido.
CU-004	Mostrar perfil y cerrar sesión.

Identificación	
Identificador:	CU-001
Título:	Acceso a páginas exclusivas del usuario ( <b>solucionado</b> ).
Detalle:	Mediante una estructura de control se filtra si el usuario intenta acceder a determinadas páginas como registro_usuario.php o miperfil.php sin haber pasado por registro.php o sin haberse logueado en el caso de la segunda.
Solución:	<pre> &lt;?php if (     isset(\$nombre) &amp;&amp; !is_null(\$nombre) &amp;&amp;     isset(\$apellidos) &amp;&amp; !is_null(\$apellidos) &amp;&amp;     isset(\$email) &amp;&amp; !is_null(\$email) ) {     ?&gt; </pre>

Identificación	
<b>Identificador:</b>	CU-002
<b>Título:</b>	Correcta repetición de contraseña ( <b>por resolver</b> ).
<b>Detalle:</b>	En el registro de usuario la segunda contraseña introducida por el usuario debe ser igual que la primera. Dicha comprobación se debe realizar en el lado cliente de manera liviana en recursos (Javascript).
<b>Solución:</b>	Uso de expresiones regulares regex para la contraseña y función que compruebe si pass1 es igual a pass2.

Identificación	
<b>Identificador:</b>	CU-003
<b>Título:</b>	Introducir e-mail válido ( <b>por resolver</b> ).
<b>Detalle:</b>	En registro de usuario y login evitar errores de e-mail y comprobar su validez.
<b>Solución:</b>	Uso de expresiones regulares regex mediante Javascript.

Identificación	
Identificador:	CU-004
Título:	Mostrar perfil y cerrar sesión ( <b>solucionado</b> ).
Detalle:	En header no mostrar Mi Perfil y Cerrar Sesión si el usuario no está logueado.
Solución:	<p>Estructuras de control en el archivo header.php comprobando que exista la sesión de un usuario logueado y su id.</p> <pre>&lt;?php require_once '../public_html/lib/seguridad.php'; \$seguridad = new Seguridad(); if (\$_SESSION['id'] == null) : ?&gt;</pre>



## 7. CONCLUSIONES Y TRABAJOS FUTUROS O MEJORAS

### 7.1 MEJORAS

Identificación	
<b>Título:</b>	PDO y bind parameter.
<b>Descripción:</b>	Implementación de parámetros bind y PDO para consultas preparadas prepare() en PHP dentro de un workflow en MVC. Dicha mejora proporcionaría mayor seguridad.

Identificación	
<b>Título:</b>	Envío de post con Ajax asincrónico.
<b>Descripción:</b>	Para que el cliente/usuario no notara ningún cambio en la web ni refresco, el envío de los datos GET y POST de los formularios se podría hacer mediante Ajax a las páginas PHP encargadas de dichas funciones o sentencias SQL.

Identificación	
<b>Título:</b>	Limpieza de strings para MYSQL.
<b>Descripción:</b>	Las funciones están para usarlas en el archivo seguridad.php pero no se han utilizado en pos de tener finalizada la mayor cantidad de código útil disponible. La limpieza de los

	strings evitaría la inyección de sentencias sql que pudieran hackear la base de datos.
--	--

Identificación	
<b>Título:</b>	Contenido páginas para cursos.
<b>Descripción:</b>	Proveer a cada curso de una página sólo para ese curso en vez de una página común como está actualmente. A su vez el buscador del header se personalizaría para cada curso y escribiría sugerencias complementarias en caso de no hayarse el curso buscado.

Identificación	
<b>Título:</b>	Asociar productos a usuario.
<b>Descripción:</b>	Para cada usuario logueado que compre cursos, poder asignar los cursos comprados al usuario en concreto que haya pagado.

Identificación	
<b>Título:</b>	Pasarela de pago.
<b>Descripción:</b>	Pasarela de pago PayPal mediante el uso de una cuenta sandbox de pruebas.

Identificación	
<b>Título:</b>	Página visualizadora de cursos.
<b>Descripción:</b>	Crear páginas de acceso al curso con el contenido que se ha comprado y con videotutoriales.

Identificación	
<b>Título:</b>	Responsive Design y accesibilidad.
<b>Descripción:</b>	Adaptar a dispositivos móviles y Tablet la web además de dar opciones de accesibilidad por ejemplo modo oscuro.

## 7.2 CONCLUSIONES

Una vez realizada la puesta en marcha de nuestra aplicación y analizados errores y posibles mejoras es la hora de realizar una conclusión para el trabajo obtenido y realizado.

Mediante el desarrollo de la página web de E-Future se ha conseguido aprender a utilizar en un entorno local las siguientes tecnologías de programación:

- MVC con orientación a objetos en PHP.
- Compilación de hoja de estilos SASS.
- Scripts en lado cliente mediante Javascript.
- Asincronía en la carga de componentes separados de una web con Ajax y JQuery.
- Realización de un login, registro y carrito de la compra.

Con el desarrollo y aprendizaje de dichas acciones el cliente navega en un entorno seguro, amigable y en el que poder centrarse para captar su atención y realizar las compras necesarias para obtener una monetización del sitio web. Bien es cierto que la temática ha estado especialmente dirigida a los cursos online, el modelo de datos es totalmente portable a cualquier tipo de producto que se desee.

Dentro de las valoraciones personales hay que indicar que se ha generado todo el código desde cero tanto hojas de estilos como el lenguaje de servidor y consultas. Es por ello por lo

que se han detectado errores con cierta facilidad, aunque eso no evitara que aún siendo una web sencilla con acciones simples y cotidianas que vemos en multitud de páginas, detrás hay un código elaborado y que ha conllevado investigación y horas de desarrollo. Bien es cierto que al crear una aplicación de este carácter se han alterado plazos a la hora de dar por finalizadas ciertas secciones, como en el entorno laboral hay que cumplir fechas, pero durante la creación hay que dejar cosas atrás y priorizar la parte más importante del trabajo que se desea presentar. Es por ello por lo que indico en la sección anterior las mejoras y ampliaciones que necesitaría si se lanzara a un entorno de producción real y posiblemente con la elaboración de esas mejoras surgieran otras interesantes que añadir al entorno.

## 8. REFERENCIAS BIBLIOGRÁFICAS

- Libro, Aprender PHP,MYSQL y Javascript. Editorial O'Reilly.
- Web: <https://www.w3schools.com/>
- Web: <https://stackoverflow.com/>
- Web: <https://code.tutsplus.com/es/categories/php>
- Web: <https://academy.leewayweb.com/como-armar-un-carrito-de-compras-con-php/>
- Web: Manual Oficial PHP en español <https://www.php.net/manual/es/index.php>
- Web: MDN Javascript <https://developer.mozilla.org/es/docs/Web/JavaScript>

