



Desarrollo de Aplicaciones Web

Curso 2022-2024

FitSynchronizer

Óscar Fernández-Chinchilla López

IES Juan Bosco

Desarrollo de Aplicaciones Web

Alcázar de San Juan (Ciudad Real)

Índice

Introducción	3
Presentación del proyecto	3
Objetivos del proyecto	3
Justificación del proyecto	3
Análisis de requerimientos	4
Identificación de necesidades y requerimientos	4
Identificación de público	4
Estudio de mercado y competencia	5
Diseño de la interfaz de usuario	5
Planificación de las tareas y los recursos necesarios	8
Implementación de pruebas	8
Desarrollo de las funcionalidades del proyecto	8
Pruebas unitarias y de integración	9
Corrección de errores y optimización del rendimiento	9
Documentación	9
Documentación técnica:	9
Documentación de usuario:	22
Manual de instalación y configuración	31
Mantenimiento y evolución	31
Plan de mantenimiento y soporte	31
Identificación de posibles mejoras y evolución del proyecto	32
Actualizaciones y futuras mejoras	32
Conclusiones	33
Evaluación del proyecto	33
Cumplimiento de objetivos y requisitos	33
Lecciones aprendidas y recomendaciones para futuros proyectos	34
Bibliografía y referencias	34
Fuentes utilizadas en el proyecto	34

Introducción

Presentación del proyecto

El proyecto consiste en la creación y desarrollo de una aplicación web la cual tenga la siguiente funcionalidad:

- a) Un gestor rutinario para dos tipos de usuarios. Los usuarios normales, los cuales podrán crear entrenamientos y dietas dentro de la web y los usuarios profesionales, los cuales serán profesionales del sector del ejercicio o de la salud que podrán crear un perfil público para darse a conocer y proporcionar su información de contacto.
- b) Cabe destacar que se permite asignar un día de la semana a cada entrenamiento creado, es decir, puede haber 1 o más entrenamientos por cada día de la semana, atribuyendo total libertad al usuario.

Objetivos del proyecto

- a) Permitir a los usuarios normales de la página crear entrenamientos, dietas y así poder gestionarse de una manera rápida y sencilla de forma online, así como poder conocer a diversos profesionales del sector del ejercicio y la salud.
- b) Proporcionar la capacidad de crearse perfiles públicos a los usuarios profesionales para promocionar su trabajo y posibilitar la obtención de trabajo, así como poder crear rutinas y dietas para el uso personal (al igual que los demás usuarios normales).

Justificación del proyecto

Desde mi experiencia personal, al buscar crear entrenamientos personalizados y obtener diversas referencias de profesionales para realizar ejercicio físico ya sea bien en un gimnasio o al aire libre, me he encontrado con que muchas plataformas ofrecen rutinas y horarios predefinidos, sin la posibilidad de adaptarlos a necesidades personales específicas.

Esta experiencia me ha llevado a plantear desarrollar una aplicación que permita gestionar entrenamientos y dietas de forma propia, rápida y sencilla, así como acceder a la experiencia de profesionales del sector en un único lugar y de manera intuitiva.

Análisis de requerimientos

Identificación de necesidades y requerimientos

Durante el análisis realizado para identificar las necesidades y requerimientos fundamentales del proyecto, se contempló como fundamental la realización de todas las funcionalidades para el lado público como la creación de rutinas de entrenamiento, dietas saludables por parte del usuario y la posibilidad de ver los perfiles de los usuarios profesionales.

Aun así, también se contemplaron requerimientos básicos relacionados con la seguridad, rendimiento, usabilidad y accesibilidad de la aplicación.

- a) **Registro e inicio de sesión de cualquier tipo de cliente para poder acceder a la aplicación.**
- b) **Autenticación tanto en el registro como al iniciar sesión**
- c) **Posibilidad de crear, editar y modificar entrenamientos y dietas personales por parte de usuarios normales o profesionales.**
- d) **Posibilidad de crear perfiles públicos y modificarlos o eliminarlos por parte de los usuarios profesionales**
- e) **Posibilitar la visualización de cualquier perfil público profesional a ambos tipos de usuarios**
- f) **Utilización de una base de datos para guardar la información de las entidades o poder acceder sus datos.**
- g) **Implementación de medidas de seguridad para evitar accesos no autorizados, proteger los datos personales de los usuarios y la obtención maliciosa de cualquier tipo de información.**

Identificación de público

FitSincronizer está dirigido a varios tipos de clientes del sector del ejercicio y la salud.

Desde usuarios normales los cuales serán:

- Personas interesadas en mejorar su salud y estado físico.
- Personas que intentan organizar sus rutinas de ejercicios y dietas de manera efectiva y rápida.
- Aquellos que quieran seguir un plan de entrenamiento y alimentación personalizados.
- Usuarios que prefieran gestionar sus rutinas y dietas en línea.

A usuarios profesionales como:

- Profesionales del sector del ejercicio y la salud
- Profesionales que quieran promocionarse
- Profesionales que busquen una plataforma donde mostrar su experiencia y sus conocimientos

Estudio de mercado y competencia

Tras realizar un considerable estudio del mercado y la competencia existente, se han logrado observar oportunidades en el mercado que solo FitSincronizer puede cumplir y permitir a su vez diferenciarse de la competencia.

Este estudio posiciona a FitSincronizer como una aplicación web la cual puede servir de gestor y administrador fitness para una gran cantidad de usuarios, así como una potencial futura red social para clientes del sector, dado que aunque de momento solo se vea como una plataforma donde además de la gestión y administración de rutinas y dietas, los profesionales pueden promocionarse a sí mismos y a su trabajo, permitiéndoles obtener posibilidades de empleo, en un futuro podría pasar a ser una red social

a) Entorno servidor

PHPMYADMIN y SQL

Para la parte del servidor web, he decidido utilizar XAMPP, ya que incluye a Apache como servidor web local debido a su simplicidad para el manejo de la BDD, la cual se gestionará a través del sistema de gestión de base de datos de MySQL utilizado en phpMyAdmin. Este sistema permitirá crear, modificar, eliminar y gestionar toda la información relevante de la aplicación.

b) Entorno de desarrollo

Visual Studio Code (IDE)

Debido a la simplicidad de este entorno de desarrollo, he decidido utilizarlo ya que permite incluir varios lenguajes de programación sin problemas y también instalar plugins o extensiones de desarrollo muy útiles tanto para el BackEnd como para el FrontEnd.

Diseño de la interfaz de usuario

A continuación, se mostrará el diseño de la interfaz de usuario de FitSincronizer, la cual ha sido diseñada para garantizar una sencillez, comodidad e interactividad al usuario, con el fin de otorgarle al usuario una experiencia visual excepcional.

Se mostrarán unos diseños comunes y generales que prevalecen en toda la aplicación de igual manera, respetando una estructura individual y única en cualquier parte de la aplicación.

Los siguientes diseños son diseños simples para entender rápidamente y de manera sencilla la estructura visual de la interfaz de usuario

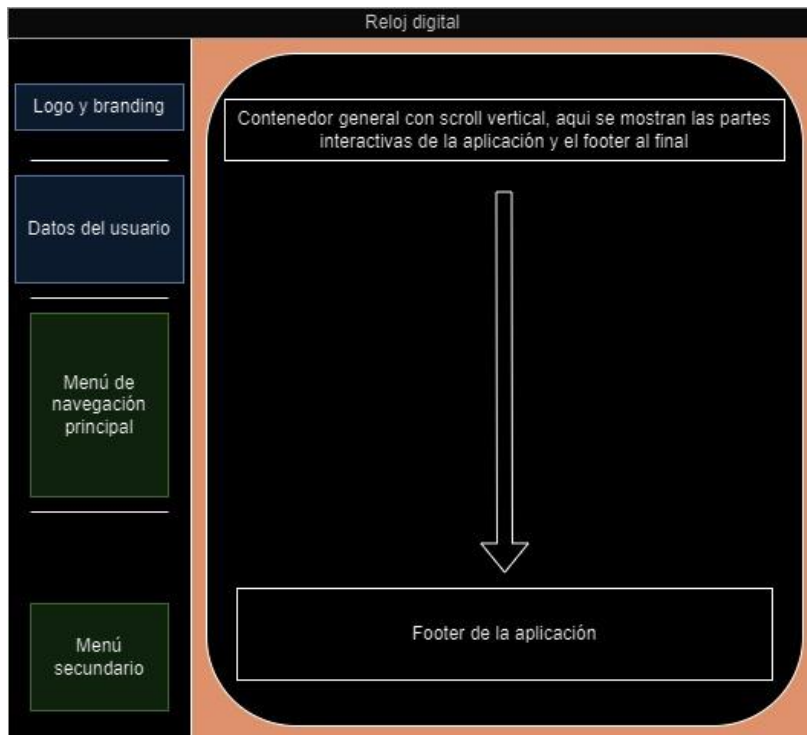
Diseño de la página principal de inicio de sesión



Diseño de la página principal de registro



Diseño de la página una vez se ha iniciado sesión (este diseño se aplica a toda la web, la parte interactiva como crear entrenamientos o dietas vendrá en el contenedor de negro)



Diseño de la página responsive (diseño general). El menú superior contiene los iconos básicos



Planificación de las tareas y los recursos necesarios

Dado que el desarrollo de la aplicación puede llegar a ser bastante compleja, se han establecido diversos tiempos/etapas para llevar una buena organización del proyecto e ir desarrollando cada tarea en su tiempo definido:

- Diseño de una plantilla grafica de la estructura de la web junto con los requisitos y funcionalidades a lograr (1 semana {15/04/2024 – 21/04/2024})
- Diseño y creación de la BDD y configuración del entorno servidor y el entorno de desarrollo del proyecto (3 días {22/04/2024 – 24/04/2024})
- Desarrollar BackEnd y FrontEnd del proyecto (4 semanas {25/04/2024 – 25/05/2024}). Se puede expandir hasta el 1 de junio en el caso de tener dificultades.
- Realización de pruebas de seguridad, calidad y usabilidad tanto de la web como para los usuarios (1 semana {25/05/2024 – 01/06/2024}).
- Contemplar mejoras o fallos y subir la web a un hosting (1 semana {01/06/2024 – 08/06/2024})
- Evaluar la experiencia de los usuarios (tiempo restante hasta la entrega)

Estas etapas no son fijas, ya que puede haber inconvenientes.

Implementación de pruebas

Desarrollo de las funcionalidades del proyecto

El desarrollo de las funcionalidades principales del proyecto se ha ido realizando en orden para ir cumpliendo todas las características necesarias para seguir con las funcionalidades siguientes con el mejor tiempo posible.

Las funcionalidades se han ido creando y desarrollando en el siguiente orden:

- Creación de usuarios e inicio de sesión
- Creación de entrenamientos y dietas
- Modificación y eliminación de entrenamientos y dietas
- Creación, modificación y eliminación de perfiles públicos sobre Profesionales.
- Listado de todas las funcionalidades anteriores para la aplicación

Pruebas unitarias y de integración

Conforme se han ido desarrollando las funcionalidades del proyecto, se han ido realizando a su vez distintas pruebas y diferentes correcciones de código para asegurar la usabilidad y el correcto funcionamiento de la aplicación

Primero se han realizado pruebas unitarias para comprobar el funcionamiento de componentes individuales de la aplicación para posteriormente realizar pruebas de integración en la misma.

Corrección de errores y optimización del rendimiento

La corrección de errores se ha ido desarrollando mediante la creación y propio desarrollo de la aplicación en sí. Se han intentado corregir los errores principales que se iban obteniendo para permitir el correcto funcionamiento y posteriormente se han ido optimizando cada funcionalidad individualmente para evitar obtener más errores y pulir los que ya se obtenían.

Documentación

Documentación técnica:

1. Estructura de archivos:
 1. **Index.php**: Archivo principal para almacenar el mapa de enrutamiento y manejar las rutas y solicitudes de la aplicación.
 2. **app/**: Directorio que contiene toda la parte lógica y el BackEnd de la aplicación.
 - **config/**:
 - config.php: archivo donde se guardan los datos para la conexión a la bdd
 - **controladores/**: Carpeta que almacena todos los controladores.
 - ControladorDietas: Controlador con funciones para la gestión de dietas.
 - ControladorEntrenamientos: Controlador con funciones para la gestión de entrenamientos.
 - ControladorLikes: Controlador con funciones para la gestión de likes.
 - ControladorPerfiles: Controlador con funciones para la gestión de perfiles.
 - ControladorUsuarios: Controlador con funciones para la gestión de usuarios.

- **modelos/**: Carpeta que contiene las clases necesarias para los modelos de la bdd.
 - ConexionBD.php: clase para conectar con la base de datos
 - Dieta.php
 - Dietas.php
 - Entrenamiento.php
 - EntrenamientosDAO.php
 - Like.php
 - LikesDAO.php
 - PerfilesProfesionalesDAO.php
 - PerfilProfesional.php
 - Sesion.php: clase para las funcionalidades de la sesión
 - Usuario.php
 - UsuariosDAO.php
- **utils/**: Carpeta con un archivo para diferentes funcionalidades de la aplicación.
 - funciones.php
- **vistas/**: carpeta donde se almacenan todas las vistas necesarias para la aplicación.

Aquí se almacenan todas las vistas necesarias y utilizadas por los controladores de la aplicación.

Existen más de 20 vistas.
- 3. **web/**: carpeta donde se almacenan todos los archivos públicos de la aplicación.
 - **archivosJS/**: carpeta donde se almacenan todos los archivos js para utilizar en la aplicación.
 - **css/**: carpeta donde se almacenan todos los archivos css para las vistas.
 - **fotosPerfiles/**: carpeta donde se almacenan todas las fotos de los perfiles públicos de los profesionales.

Documentación de cada archivo con funcionalidad en la aplicación:

Explicación del archivo Index.php:

En este archivo se realizan todos los requerimientos necesarios de la aplicación, instanciación de variables de sesión.

Además, se realiza un mapa de enrutamiento con todas las rutas de la aplicación, indicando cuales son públicas o privadas para proteger la integridad y privacidad de la aplicación y sus usuarios. Cada ruta contiene el nombre de la acción que va a realizar, el controlador que va a utilizar, el método correspondiente del controlador y su privacidad.

Si no se especifica ninguna acción por la URL, por defecto utilizará la acción de inicio por lo que controlamos el acceso a la aplicación mediante variables de sesión y estableciendo cookies.

```
index.php > ...
1  <?php
2  // Requiere los archivos necesarios para los modelos, controladores, configuración y utilidades.
3  require_once "app/modelos/Usuario.php";
4  require_once "app/modelos/UsuariosDAO.php";
5  require_once "app/modelos/Entrenamiento.php";
6  require_once "app/modelos/EntrenamientosDAO.php";
7  require_once "app/modelos/Dieta.php";
8  require_once "app/modelos/DietasDAO.php";
9  require_once "app/modelos/PerfilProfesional.php";
10 require_once "app/modelos/PerfilesProfesionalesDAO.php";
11 require_once "app/config/config.php";
12 require_once "app/modelos/ConexionBD.php";
13 require_once "app/controladores/ControladorUsuarios.php";
14 require_once "app/controladores/ControladorPerfiles.php";
15 require_once "app/controladores/ControladorEntrenamientos.php";
16 require_once "app/controladores/ControladorDietas.php";
17 require_once "app/modelos/Sesion.php";
18 require_once "app/utiles/funciones.php";
19
20 // Inicia la sesión PHP
21 session_start();
```

```
// Mapa de rutas que define las acciones permitidas y sus respectivos controladores, métodos y si requieren sesión iniciada
$mapa = array(
    'inicio' => array(
        'controlador' => 'ControladorUsuarios',
        'metodo' => 'inicio',
        'privada' => false
    ),
    'registrar' => array(
        'controlador' => 'ControladorUsuarios',
        'metodo' => 'registrar',
        'privada' => false
    ),
    'login' => array(
        'controlador' => 'ControladorUsuarios',
        'metodo' => 'login',
        'privada' => false
    ),
    'vistaNormal' => array(
        'controlador' => 'ControladorUsuarios',
        'metodo' => 'vistaNormal',
        'privada' => true
    ),
    'vistaProfesional' => array(
        'controlador' => 'ControladorUsuarios',
        'metodo' => 'vistaProfesional',
        'privada' => true
    ),
);
```

```
// Parseo de la ruta y selección de acción
if (isset($_GET['accion'])) {
    // Comprueba si se ha pasado una acción concreta, sino usa la acción por defecto 'inicio'
    if (isset($mapa[$_GET['accion']])) {
        // Comprueba si la acción existe en el mapa, sino muestra error 404
        $accion = $_GET['accion'];
    } else {
        // La acción no existe
        header('Status: 404 Not found');
        echo 'Página no encontrada';
        die();
    }
} else {
    // Acción por defecto
    $accion = 'inicio';
}

// Verificación de la cookie y inicio de sesión automático
if (!Sesion::existeSesion() && isset($_COOKIE['usuario'])) {
    // Conectamos con la BD
    $conexionDB = new ConexionBD(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConnexion();

    // Nos conectamos para obtener el id
    $usuariosDAO = new UsuariosDAO($conn);
    if ($usuario = $usuariosDAO->getByNombreUsuario($_COOKIE['usuario'])) {
        // Inicia sesión automáticamente
        Sesion::iniciarSesion($usuario);
        // Redirigir a la página principal después de iniciar sesión automáticamente
        header('location: index.php?accion=vistaNormal');
        die();
    }
}

// Si la acción es privada, comprueba que el usuario ha iniciado sesión, sino redirige a index
if (!Sesion::existeSesion() && $mapa[$accion]['privada']) {
    header('location: index.php');
    guardarMensaje("Debes iniciar sesión para acceder a $accion");
    die();
}
}
```

Archivo de configuración para la conexión con la BDD:

Este archivo contiene los datos de las variables necesarias para la conexión con la BDD. Sin este archivo, la aplicación no funcionaría ya que no se podría establecer conexión con la BDD

```
app > config > config.php > ...
1  <?php
2
3  define('MYSQL_USER', 'root');
4  define('MYSQL_PASS', '');
5  define('MYSQL_DB', 'fitsincronizer');
6  define('MYSQL_HOST', 'localhost');
7
```

Clase de Sesión:

Esta clase se utiliza para poder guardar los usuarios en una variable de sesión mediante serialización y poder utilizar sus datos posteriormente, así como hacer verificaciones de seguridad, iniciar sesión, cerrar sesión, etc.

```
app > modelos > Sesion.php > ...
1  <?php
2  class Sesion
3  {
4      static public function getUsuario(): Usuario|false
5      {
6          if (isset($_SESSION['usuario'])) {
7              return unserialize($_SESSION['usuario']);
8          } else {
9              return false;
10         }
11     }
12
13     static public function iniciarSesion($usuario)
14     {
15         $_SESSION['usuario'] = serialize($usuario);
16     }
17
18     static public function cerrarSesion()
19     {
20         unset($_SESSION['usuario']);
21     }
22
23     static public function existeSesion()
24     {
25         if (isset($_SESSION['usuario'])) {
26             return true;
27         } else {
28             return false;
29         }
30     }
31 }
32
```

Archivo de funciones:

Este archivo contiene diversas funciones para ser utilizadas en diversas partes del BackEnd de la aplicación:

```
app > utils > funciones.php > ...
1  <?php
2
3  /**
4   * Genera un hash aleatorio para un nombre de archivo manteniendo la extensión original
5   */
6  function generarNombreArchivo(string $nombreOriginal): string
7  {
8      $nuevoNombre = md5(time() + rand());
9      $partes = explode('.', $nombreOriginal);
10     $extension = $partes[count($partes) - 1];
11     return $nuevoNombre . '.' . $extension;
12 }
13
14 function guardarMensaje($mensaje)
15 {
16     $_SESSION['error'] = $mensaje;
17 }
18
19 function imprimirMensaje()
20 {
21     if (isset($_SESSION['error'])) {
22         echo '<div class="error" id="mensajeError">' . $_SESSION['error'] . '</div>';
23         unset($_SESSION['error']);
24     }
25 }
26
```

Estructura de las clases de la carpeta modelos/:

A continuación, se muestra una única clase de la aplicación para ahorrar extensión en el documento. Esta clase servirá para ver la estructuración de las clases y sus DAOS correspondientes:

Clase de Perfil Profesional:

Cada clase representa un campo/tabla de la BDD, por lo que contienen sus correspondientes variables y métodos Getter y Setter.

```
app > modelos > PerfilProfesional.php > PerfilProfesional > getNombreUsuarioProfesional
1  <?php
2  class PerfilProfesional {
3      private $id;
4      private $id_usuario;
5      private $imagen;
6      private $nombre_personal;
7      private $nombre_usuario_profesional;
8      private $descripcion_personal;
9      private $edad;
10     private $anos_experiencia;
11     private $datos_contacto;
12     private $trabajos_anteriores;
13
14     // Métodos getter
15     public function getId() {
16         return $this->id;
17     }
18
19     public function getIdUsuario() {
20         return $this->id_usuario;
21     }
22
23     public function getImagen() {
24         return $this->imagen;
25     }
26
27     public function getNombrePersonal() {
28         return $this->nombre_personal;
29     }
30
31     public function getNombreUsuarioProfesional() {
32         return $this->nombre_usuario_profesional;
33     }
34
35     public function getDescripcionPersonal() {
36         return $this->descripcion_personal;
37     }
38
39     public function getEdad() {
40         return $this->edad;
41     }
42 }
```

Clase de Perfiles Profesionales DAO:

Los DAO son los intermediarios entre una aplicación y una base de datos. Ofrecen los métodos necesarios para usar en la aplicación e interactuar con la BDD al mismo tiempo.

Se deben crear tantos DAO por objeto/clase que tengamos creados ya que los necesitaremos para poder obtener, modificar, eliminar o crear los datos de la base de datos.

Los DAOS tendrán siempre métodos para la obtención de una o más entidades (Ejemplo: para obtener un perfil individualmente o varios en un array), métodos para crear entidades en la BDD, borrarlas o modificarlas.

```
app > modelos > PerfilesProfesionalesDAO.php > PerfilesProfesionalesDAO > update
1  <?php
2  require_once "PerfilProfesional.php";
3  class PerfilesProfesionalesDAO
4  {
5      private mysqli $conn;
6
7      public function __construct($conn)
8      {
9          $this->conn = $conn;
10     }
11     //Función para obtener un perfil por su id
12     public function getById($id): PerfilProfesional|null
13     {
14         if (!$stmt = $this->conn->prepare("SELECT * FROM perfil_profesional WHERE id = ?")) {
15             echo "Error en la SQL: " . $this->conn->error;
16             return null;
17         }
18
19         $stmt->bind_param('i', $id);
20         $stmt->execute();
21         $result = $stmt->get_result();
22
23         if ($result->num_rows == 1) {
24             $perfil = $result->fetch_object(PerfilProfesional::class);
25             return $perfil;
26         } else {
27             return null;
28         }
29     }
}
```

```

30 //Función para obtener un perfil por su id de usuario
31 public function getIdUsuario($id_usuario): PerfilProfesional|null
32 {
33     if (!$stmt = $this->conn->prepare("SELECT * FROM perfil_profesional WHERE id_usuario = ?")) {
34         echo "Error en la SQL: " . $this->conn->error;
35         return null;
36     }
37
38     $stmt->bind_param('i', $id_usuario);
39     $stmt->execute();
40     $result = $stmt->get_result();
41
42     if ($result->num_rows == 1) {
43         $perfil = $result->fetch_object(PerfilProfesional::class);
44         return $perfil;
45     } else {
46         return null;
47     }
48 }
49 //Función para obtener todos los perfiles
50 public function getAll(): array
51 {
52     if (!$stmt = $this->conn->prepare("SELECT * FROM perfil_profesional")) {
53         echo "Error en la SQL: " . $this->conn->error;
54     }
55     $stmt->execute();
56     $result = $stmt->get_result();
57
58     $array_perfiles = array();
59
60     while ($perfil = $result->fetch_object(PerfilProfesional::class)) {
61         $array_perfiles[] = $perfil;
62     }
63     return $array_perfiles;
64 }

```

```

65 //Función para insertar un perfil
66 public function insert(PerfilProfesional $perfil): int|bool
67 {
68     $sql = "INSERT INTO perfil_profesional (id_usuario, imagen, nombre_personal, nombre_usuario_profesional, descripcion_personal, edad, anos_experiencia, datos_contacto, trabajosanteriores) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
69
70     if (!$stmt = $this->conn->prepare($sql)) {
71         echo "Error al preparar la consulta insert: " . $this->conn->error;
72         return false;
73     }
74
75     $id_usuario = $perfil->getIdUsuario();
76     $imagen = $perfil->getImagen();
77     $nombre_personal = $perfil->getNombrePersonal();
78     $nombre_usuario_profesional = $perfil->getNombreUsuarioProfesional();
79     $descripcion_personal = $perfil->getDescripcionPersonal();
80     $edad = $perfil->getEdad();
81     $anos_experiencia = $perfil->getAnosExperiencia();
82     $datos_contacto = $perfil->getDatosContacto();
83     $trabajosanteriores = $perfil->getTrabajosAnteriores();
84
85     $stmt->bind_param('issssiis', $id_usuario, $imagen, $nombre_personal, $nombre_usuario_profesional, $descripcion_personal, $edad, $anos_experiencia, $datos_contacto, $trabajosanteriores);
86
87     if ($stmt->execute()) {
88         return $stmt->insert_id;
89     } else {
90         echo "Error al ejecutar la consulta: " . $stmt->error;
91         return false;
92     }
93 }

```



```

94 //Función para borrar un perfil
95 public function delete($id): bool
96 {
97     if (!$stmt = $this->conn->prepare("DELETE FROM perfil_profesional WHERE id = ?")) {
98         echo "Error en la SQL: " . $this->conn->error;
99         return false;
100     }
101
102     $stmt->bind_param('i', $id);
103     $stmt->execute();
104
105     if ($stmt->affected_rows == 1) {
106         return true;
107     } else {
108         return false;
109     }
110 }

```

```

111 //Función para modificar un perfil
112 public function update(PerfilProfesional $perfil): bool
113 {
114     if (!$stmt = $this->conn->prepare(
115         "UPDATE perfil_profesional SET id_usuario=?, imagen=?, nombre_personal=?, nombre_usuar
116     )) {
117         echo "Error al preparar la consulta update: " . $this->conn->error;
118         return false;
119     }
120
121     $id = $perfil->getId();
122     $id_usuario = $perfil->getIdUsuario();
123     $imagen = $perfil->getImagen();
124     $nombre_personal = $perfil->getNombrePersonal();
125     $nombre_usuario_profesional = $perfil->getNombreUsuarioProfesional();
126     $descripcion_personal = $perfil->getDescripcionPersonal();
127     $edad = $perfil->getEdad();
128     $anos_experiencia = $perfil->getAnosExperiencia();
129     $datos_contacto = $perfil->getDatosContacto();
130     var_dump("DAO");
131     var_dump($datos_contacto);
132     $trabajos_anteriores = $perfil->getTrabajosAnteriores();
133
134     $stmt->bind_param(
135         'isssisssi',
136         $id_usuario,
137         $imagen,
138         $nombre_personal,
139         $nombre_usuario_profesional,
140         $descripcion_personal,
141         $edad,
142         $anos_experiencia,
143         $datos_contacto,
144         $trabajos_anteriores,
145         $id
146     );
147
148     return $stmt->execute();
149 }

```

Controladores de la aplicación:

De la misma manera, para ahorrar extensión se mostrará la estructura y diversas funcionalidades del Controlador de perfiles como ejemplo de los demás.

Cada controlador se hará responsable de manejar las acciones relacionadas con sus entidades, en este caso el Controlador de Perfiles se hará cargo de crear, eliminar, editar y mostrar los perfiles junto con la utilización de las clases explicadas anteriormente.

Cada método de los controladores valida que la entrada de datos sea válida y segura antes de introducirlos a la BDD.

Método para crear:

```
class ControladorPerfiles
{
    // Método para crear un nuevo perfil profesional
    public function crear()
    {
        if ($_SERVER['REQUEST_METHOD'] == 'POST') {
            // Recogida de datos del formulario
            $id_usuario = Sesion::getUsuario()->getId();
            $foto = $_FILES['foto']['name'];
            $nombre_personal = htmlentities($_POST['nombre_personal']);
            $nombre_usuario_profesional = htmlentities($_POST['nombre_usuario_profesional']);
            $descripcion_personal = htmlentities($_POST['descripcion_personal']);
            $edad = htmlentities($_POST['edad']);
            $anos_experiencia = htmlentities($_POST['anos_experiencia']);
            $datos_contacto = htmlentities($_POST['datos_contacto']);
            $trabajos_anteriores = htmlentities($_POST['trabajos_anteriores']);
            $extension = pathinfo($foto, PATHINFO_EXTENSION);

            // Conexión a la BD
            $conexionDB = new ConexionBD(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
            $conn = $conexionDB->getConnexion();

            // Instancia del DAO de perfiles
            $perfilesDAO = new PerfilesProfesionalesDAO($conn);
            $perfil = new PerfilProfesional();

            // Validar y Copiar la foto al disco
            if ($_FILES['foto']['type'] != 'image/jpeg' && $_FILES['foto']['type'] != 'image/webp' && $_FILES['foto']['type'] != 'image/png') {
                $error = 'La foto no tiene el formato adecuado';
            } else {
                // Generar nombre único para la foto
                $foto = generarNombreArchivo($foto);

                // Asegurar que el nombre del archivo sea único
                while (file_exists("web/fotosPerfiles/$foto")) {
                    $foto = generarNombreArchivo($foto);
                }

                if (!move_uploaded_file($_FILES['foto']['tmp_name'], "web/fotosPerfiles/$foto")) {
                    die("Error al subir el archivo");
                }

                // Asignar los datos al perfil
                $perfil->setIdUsuario($id_usuario);
                $perfil->setImagen($foto);
                $perfil->setNombrePersonal($nombre_personal);
                $perfil->setNombreUsuarioProfesional($nombre_usuario_profesional);
                $perfil->setDescripcionPersonal($descripcion_personal);
                $perfil->setEdad($edad);
                $perfil->setAnosExperiencia($anos_experiencia);
                $perfil->setDatosContacto($datos_contacto);
                $perfil->setTrabajosAnteriores($trabajos_anteriores);

                // Insertar perfil en la BD
                if ($perfilesDAO->insert($perfil)) {
                    header("location: index.php?accion=vistaNormal");
                    die();
                } else {
                    header("location: index.php?accion=crearPerfil");
                }
            }
        }
    }
}
```

Método para editar

```
// Método para editar un perfil profesional existente
public function editarPerfil()
{
    // Verificar que se recibe un ID
    if (!isset($_GET['id'])) {
        echo "Error: No se ha proporcionado un ID";
        return;
    }

    $id = $_GET['id'];

    // Conexión a la BD
    $conexionDB = new ConexionBD(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConexion();

    // Instancia del DAO de perfiles
    $perfilesDAO = new PerfilesProfesionalesDAO($conn);

    // Obtener el perfil actual
    $perfil = $perfilesDAO->getById($id);

    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        // Recoger datos del formulario
        $nombrePersonal = htmlspecialchars($_POST['nombre_personal']);
        $nombreUsuarioProfesional = htmlspecialchars($_POST['nombre_usuario_profesional']);
        $descripcionPersonal = htmlspecialchars($_POST['descripcion_personal']);
        $edad = htmlspecialchars($_POST['edad']);
        $anosExperiencia = htmlspecialchars($_POST['anos_experiencia']);
        $datosContacto = htmlspecialchars($_POST['datos_contacto']);
        $trabajosAnteriores = htmlspecialchars($_POST['trabajos_anteriores']);

        // Manejar subida de imagen
        $imagen = $perfil->getImagen();
        if (isset($_FILES['imagen']) && $_FILES['imagen']['error'] == 0) {
            $imagenPath = 'web/fotosPerfiles/' . basename($_FILES['imagen']['name']);
            if (move_uploaded_file($_FILES['imagen']['tmp_name'], $imagenPath)) {
                // Eliminar la imagen antigua
                if (file_exists('web/fotosPerfiles/' . $perfil->getImagen())) {
                    unlink('web/fotosPerfiles/' . $perfil->getImagen());
                }
                $imagen = basename($_FILES['imagen']['name']);
            } else {
                echo "Error al subir la imagen";
                return;
            }
        }

        // Actualizar el perfil
        $perfil->setNombrePersonal($nombrePersonal);
        $perfil->setNombreUsuarioProfesional($nombreUsuarioProfesional);
        $perfil->setDescripcionPersonal($descripcionPersonal);
        $perfil->setEdad($edad);
        $perfil->setAnosExperiencia($anosExperiencia);
        $perfil->setDatosContacto($datosContacto);
        $perfil->setTrabajosAnteriores($trabajosAnteriores);
    }
}
```

Método para borrar

```
// Método para borrar un perfil profesional existente
public function borrar()
{
    $mensaje=null;
    // Verificar que se recibe un ID
    if (!isset($_GET['id'])) {
        echo "Error: No se ha proporcionado un ID";
        return;
    }

    $id = $_GET['id'];

    // Conexión a la BD
    $conexionDB = new ConexionBD(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConnexion();

    // Instancia del DAO de perfiles
    $perfilesDAO = new PerfilesProfesionalesDAO($conn);

    // Obtener el perfil para obtener el nombre de la imagen
    $perfil = $perfilesDAO->getById($id);
    if ($perfil) {
        // Obtener la ruta de la imagen
        $imagenPath = 'web/fotosPerfiles/' . $perfil->getImagen();

        // Condición para borrar el perfil y la imagen
        if ($perfilesDAO->delete($id)) {
            // Eliminar la imagen del sistema de archivos
            if (file_exists($imagenPath)) {
                unlink($imagenPath);
            }
            if (Sesion::getUsuario()->getRol() == "Administrador") {
                $mensaje="Se ha borrado el perfil";
                header("location: index.php?accion=vistaAdmin");
                die();
            } else {
                header("location: index.php?accion=miPerfil");
                die();
            }
        } else {
            echo "No se ha podido borrar el perfil";
        }
    } else {
        echo "Perfil no encontrado";
    }
}
```

Método para listar todos los perfiles y mostrar el del usuario propio

```
// Método para listar todos los perfiles profesionales
public function listarPerfiles()
{
    $id_usuario = Sesion::getUsuario()->getId();

    // Conexión a la BD
    $conexionDB = new ConexionBD(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConnexion();

    // Instancia del DAO de perfiles
    $perfilesDAO = new PerfilesProfesionalesDAO($conn);

    // Obtener todos los perfiles
    $perfiles = $perfilesDAO->getAll();

    // Creas la instancia de LikesDAO
    $likeDAO = new LikesDAO($conn);
    // Obtén la cantidad de perfiles en el array
    $cantidadPerfiles = count($perfiles);

    require 'app/vistas/listarPerfiles.php';
}

// Método para mostrar el perfil del usuario actual
public function miPerfil()
{
    // Conexión a la BD
    $conexionDB = new ConexionBD(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConnexion();

    // Instancia del DAO de perfiles
    $perfilesDAO = new PerfilesProfesionalesDAO($conn);

    $id_usuario = Sesion::getUsuario()->getId();

    // Obtener el perfil del usuario actual
    $perfil = $perfilesDAO->getByIdUsuario($id_usuario);

    require 'app/vistas/miPerfil.php';
}
```

Método para mostrar un perfil de un usuario ajeno a nosotros:

```
// Método para mostrar un perfil profesional específico
public function mostrarPerfil()
{
    // Verificar que se recibe un ID
    if (!isset($_GET['id'])) {
        echo "Error: No se ha proporcionado un ID";
        return;
    }

    $id = $_GET['id'];

    // Conexión a la BD
    $conexionDB = new ConexionBD(MYSQL_USER, MYSQL_PASS, MYSQL_HOST, MYSQL_DB);
    $conn = $conexionDB->getConexion();

    // Instancia del DAO de perfiles
    $perfilesDAO = new PerfilesProfesionalesDAO($conn);

    // Obtener el perfil específico
    $perfil = $perfilesDAO->getById($id);

    require 'app/vistas/mostrarPerfil.php';
}
}
```

Documentación de usuario:

En esta sección se explicará que pasos hay que seguir para poder utilizar correctamente la aplicación por parte del usuario.

Desde como registrarse en la aplicación e iniciar sesión, crear entrenamientos y dietas personalizados o mostrar todos los perfiles públicos e ir viendo sus detalles.

Observación: en todo momento se valida que los datos introducidos por la parte de usuario/cliente sean válidos. Si no lo son, aparecerá un aviso con el error correspondiente cometido y volverá a la página en la que estábamos.

Inicio de la aplicación

En la pantalla principal nos saldrá una vista para iniciar sesión si ya tenemos un usuario creado. Si tenemos una sesión ya creada con su cookie correspondiente, prescindirá de esta vista e iniciará sesión automáticamente hasta que se cierre o pase 1 semana.

FitSynchronizer

Iniciar Sesión | Registrarse

¿Que es FitSynchronizer?

FitSynchronizer es una aplicación web destinada a aquellos usuarios del mundo fitness y la salud, los cuales buscan un lugar de administración y gestión de sus propias rutinas y dietas de forma online, rápida y sencilla.

En FS podrás gestionar todos tus entrenamientos y comidas con la interfaz de la aplicación en cero coma.

A su vez, también podrás encontrar perfiles públicos profesionales destinados a darse a conocer con el objetivo de proporcionar información y ayuda a aquellos que busquen algún entrenador o algún dietista.

Iniciar sesión

Nombre de usuario

Correo electrónico

Contraseña

Acceder

¿Todavía no tienes cuenta?
 Pincha [aquí](#) para ir a la pagina de registro

Registro:

Para crear una cuenta en la aplicación deberemos pinchar en registrarse, de modo que llegaremos a la siguiente vista. Se dan dos posibilidades, crear un usuario normal o profesional. El profesional podrá hacerse un perfil público para promocionarse en la aplicación.

FitSynchronizer

Iniciar Sesión | Registrarse

Plan para usuarios básicos

Nombre personal

Nombre de usuario

Correo electrónico

Contraseña

Confirmar contraseña

Acceder

Plan para usuarios profesionales

Nombre personal

Nombre de usuario

Correo electrónico

Correo electrónico

Teléfono

Nº Teléfono

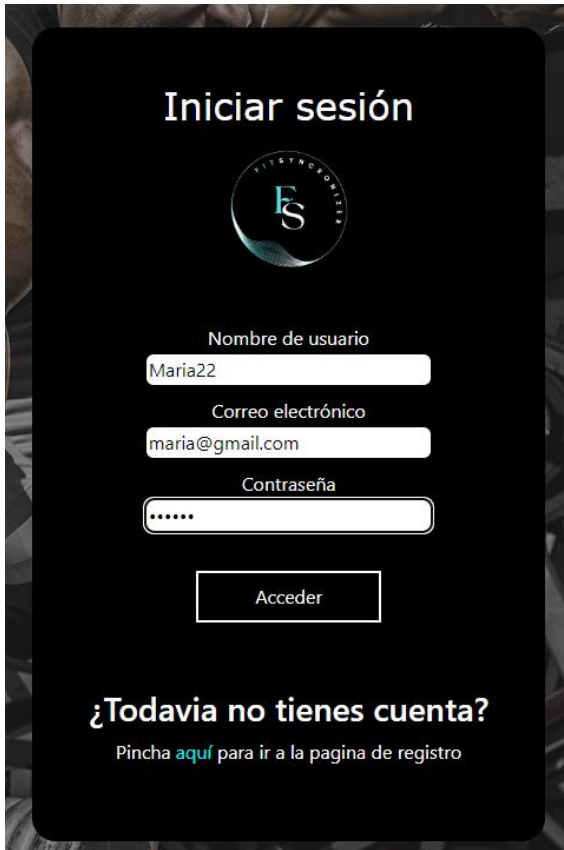
Contraseña

Ingrese su contraseña


Confirmar contraseña

Acceder

Tras crear el usuario, nos volverá a la vista anterior y podremos iniciar sesión.



Iniciar sesión



Nombre de usuario

Correo electrónico

Contraseña

¿Todavía no tienes cuenta?
 Pincha [aquí](#) para ir a la pagina de registro

Vista de Usuario:

Al iniciar sesión, nos redirigirá a la siguiente vista para los usuarios. Se proporciona una barra lateral a modo de navegación con todas las opciones posibles. Si te has registrado como usuario normal, no saldrá la opción de “Mi perfil”.

En la vista te darán la bienvenida y podrás empezar a crear tus entrenamientos y tus dietas. Además, en la barra lateral te saldrán tus datos y las opciones adicionales para cerrar sesión, ver el “sobre mí” del desarrollador y los ajustes para modificar los datos del usuario.



Vista de entrenamientos:

Al principio no tendremos entrenamientos, por lo que se te indicará que debes crear alguno. Pinchando en el botón saldrá un formulario para crear nuestro entrenamiento y al crearlo nos redirigirá a la vista anterior para ver nuestros entrenamientos, con las posibilidades de editarlo, borrarlos o mostrarlo individualmente.

The image displays two screenshots of the FitSynchronizer web application interface.

Top Screenshot: Entrenamientos creados: 0

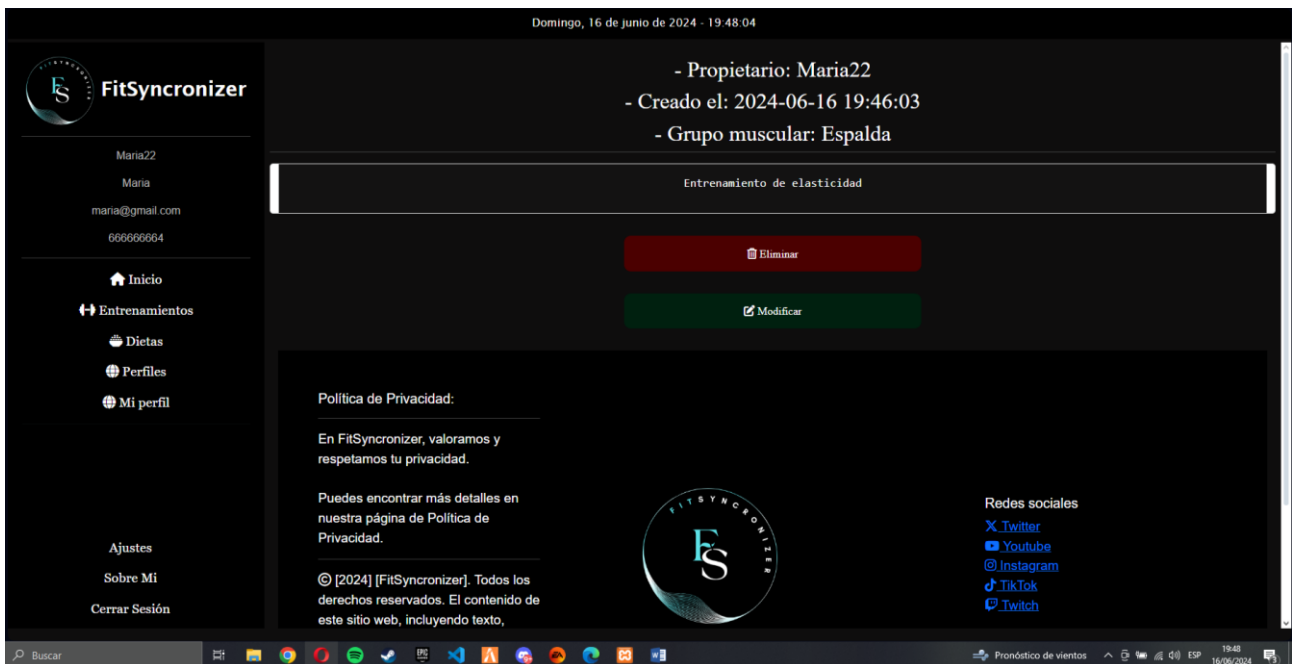
- Header:** FitSynchronizer logo and user profile (Maria22, Maria, maria@gmail.com, 666666664).
- Navigation:** Inicio, Entrenamientos, Dietas, Perfiles, Mi perfil, Ajustes, Sobre Mi, Cerrar Sesión.
- Main Content:**
 - Entrenamientos creados: 0
 - Añadir Entrenamiento button
 - No tienes ningun entrenamiento
 - Política de Privacidad: En FitSynchronizer, valoramos y respetamos tu privacidad. Puedes encontrar más detalles en nuestra página de Política de Privacidad.
 - © [2024] [FitSynchronizer]. Todos los derechos reservados. El contenido de este sitio web, incluyendo texto, imágenes, gráficos y otros materiales, está protegido por las leyes de derechos de autor y otras leyes de propiedad intelectual.
 - Redes sociales: Twitter, Youtube, Instagram, TikTok, Twitch.

Bottom Screenshot: Formulario de Entrenamiento

- Header:** FitSynchronizer logo and user profile (Maria22, Maria, maria@gmail.com, 666666664).
- Navigation:** Inicio, Entrenamientos, Dietas, Perfiles, Mi perfil, Ajustes, Sobre Mi, Cerrar Sesión.
- Main Content:**
 - Formulario de Entrenamiento
 - Día: Jueves
 - Rutina: Entrenamiento de elasticidad
 - Grupo Muscular: Espalda
 - Guardar button
 - Política de Privacidad: En FitSynchronizer, valoramos y respetamos tu privacidad. Puedes encontrar más detalles en nuestra página de Política de Privacidad.
 - Redes sociales: Twitter, Youtube, Instagram.



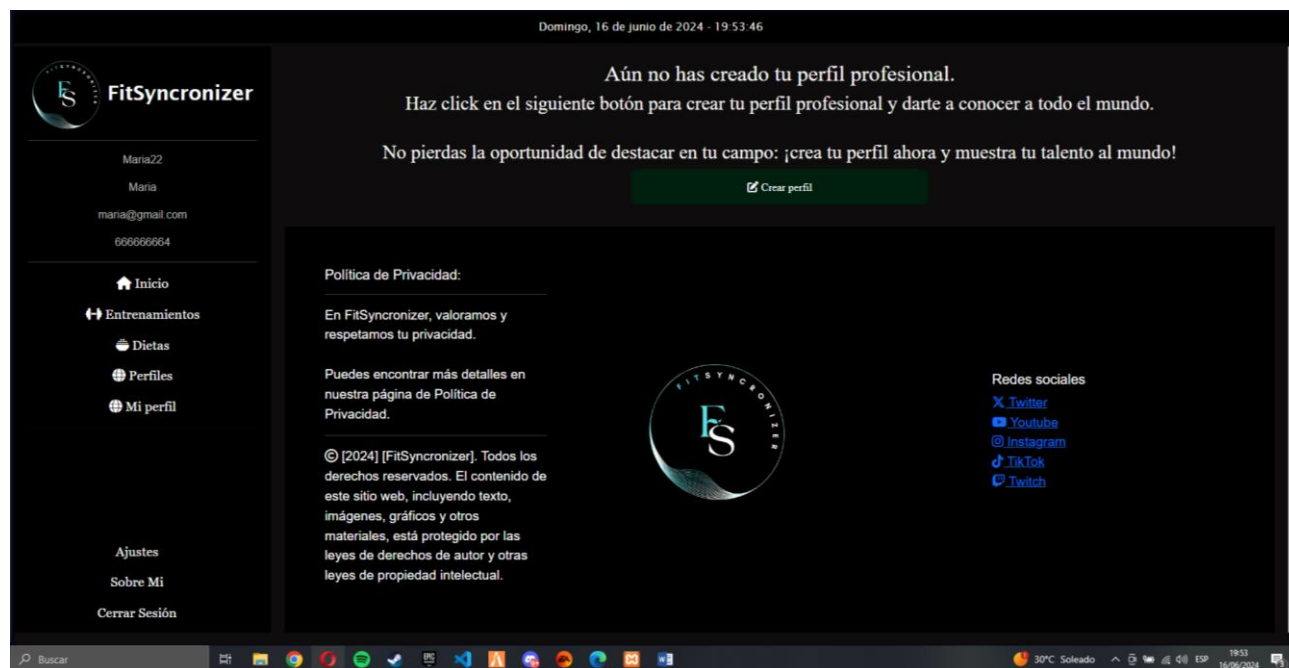
Para la creación de dietas es similar a la de entrenamientos, ya que sigue la misma estructura. En cada entrenamiento o dieta, tendremos varias opciones tanto para eliminarlos de forma asíncrona, editarlos o mostrarlos.



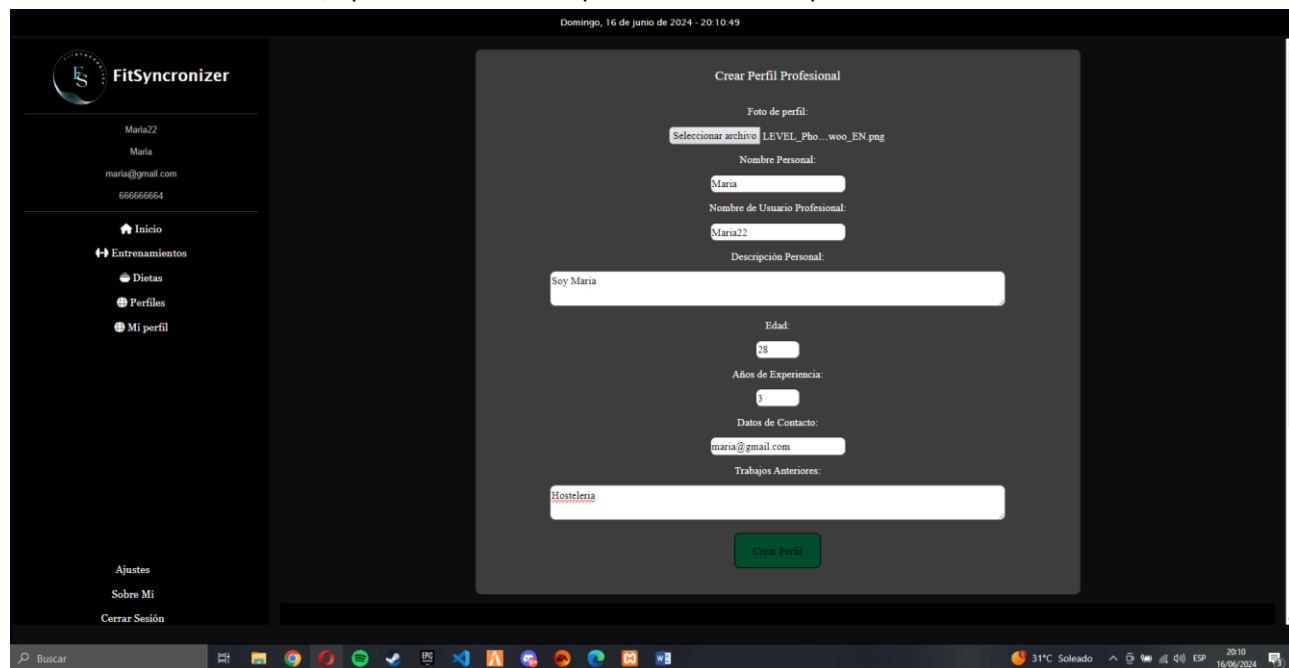
Vista de perfil:

Si somos usuarios profesionales, tendremos la opción de tener un apartado para ver nuestro perfil y modificarlo o eliminarlo. FitSincronizer da la posibilidad de crear un perfil cuando sea, no es obligatorio al principio crearse un perfil público si el usuario no quiere.

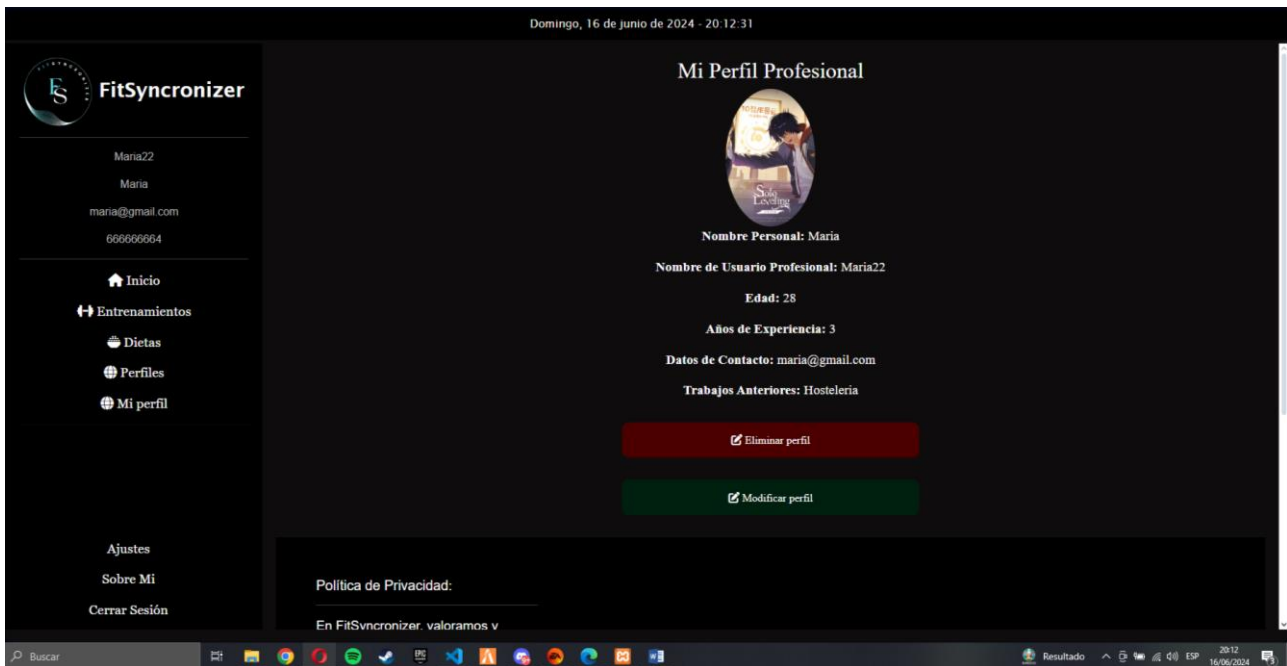
Al principio nos dirá que tendremos que crear uno, por lo que dándole al botón se redirigirá a un formulario para crearlo y posteriormente tras guardar los datos, nos volverá a la vista anterior para ver nuestro perfil y modificarlo o eliminarlo.



Cuando le demos al botón, aparecerá esta vista para crear nuestro perfil.



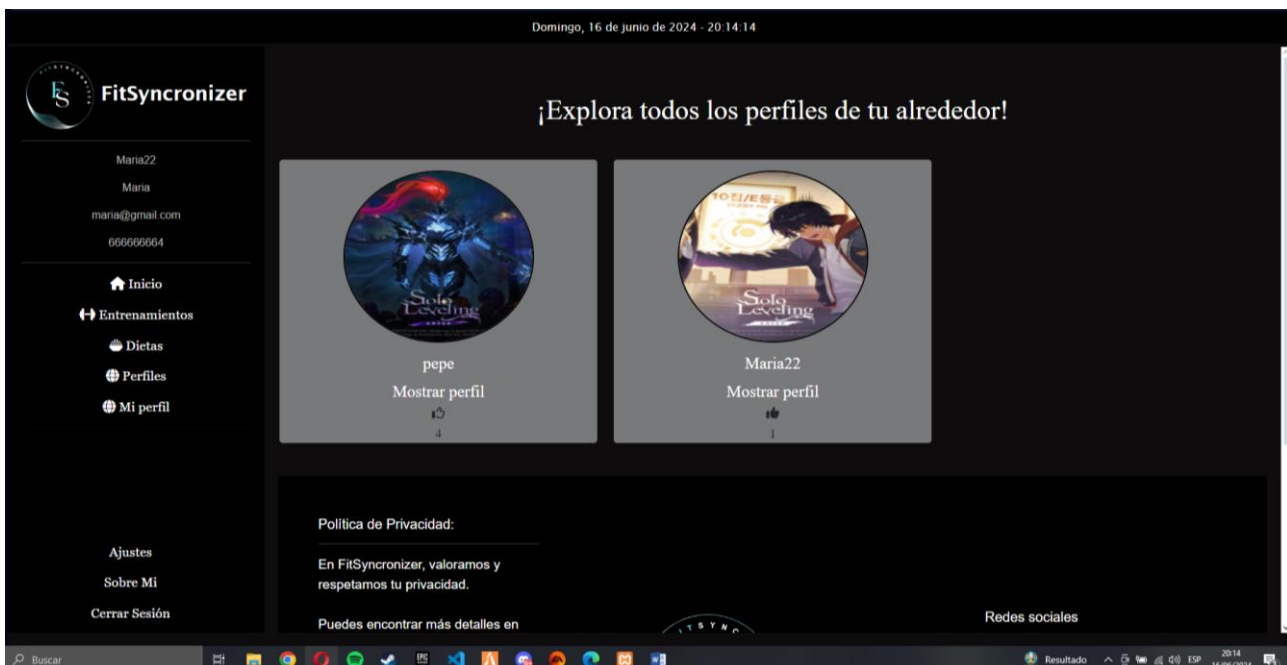
Tras rellenar correctamente todos los campos, nos volverá a la pantalla de inicio y si volvemos a ir a nuestro perfil, lo veremos creado con la posibilidad de modificarlo o verlo.



Mostrar todos los perfiles:

Por último, seamos del tipo de usuario que seamos, podremos ver una vista general de todos los perfiles públicos en la aplicación.

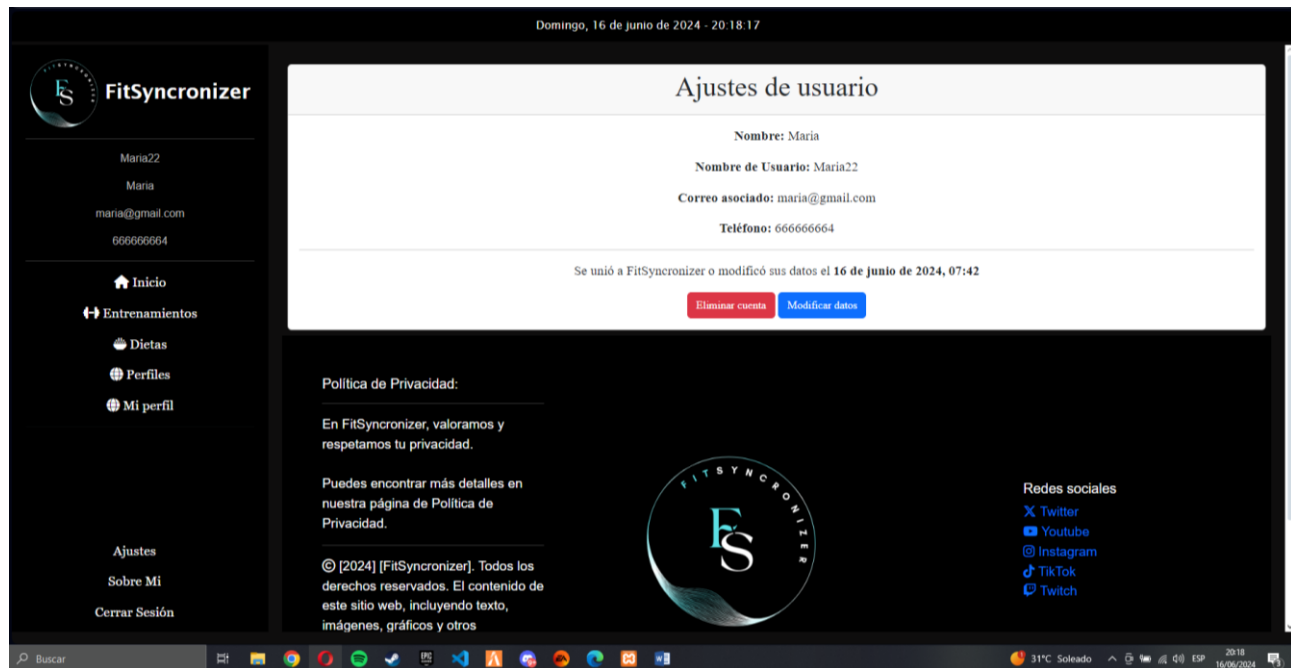
Al darle a “Perfiles” en el menú de navegación lateral, nos aparecerá esta vista donde podremos ver todos los perfiles. Podremos darle like, quitar el like o ver el perfil profesional.



Vista de ajustes de usuario

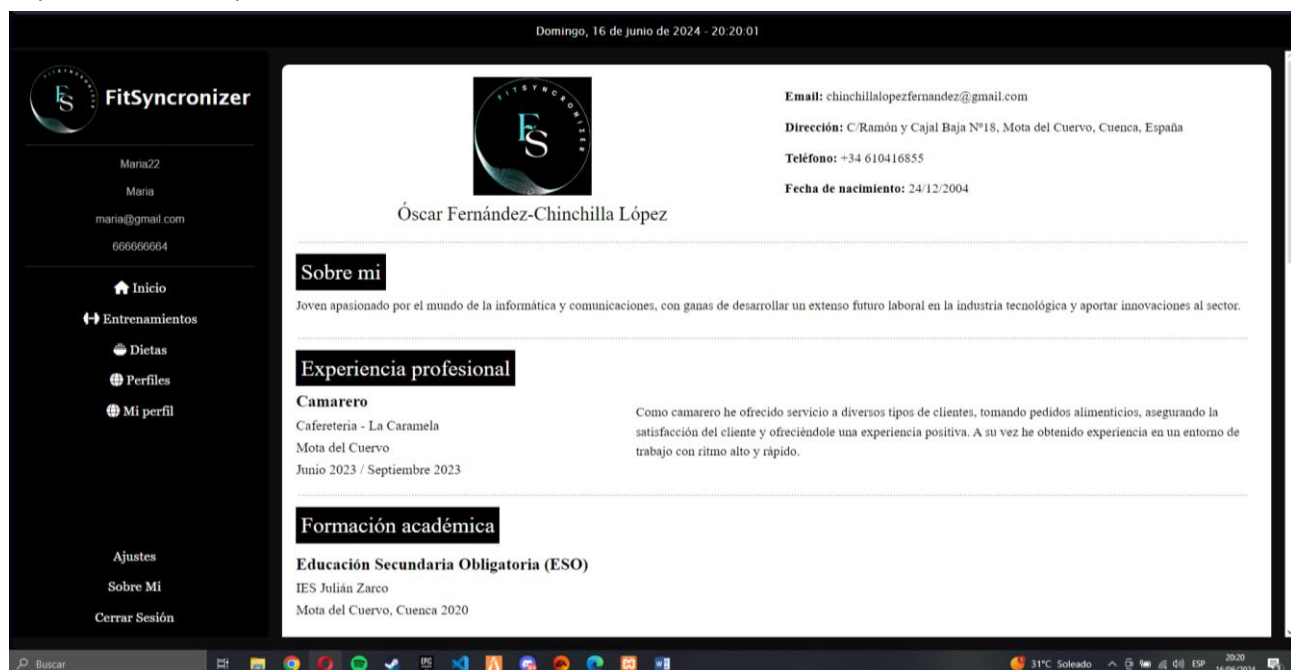
En el menú lateral de navegación tendremos una opción para irnos a la vista de ajustes del usuario, donde podremos ver todos los datos del mismo, borrar la cuenta o modificar sus datos.

Si borramos la cuenta del usuario, en consecuencia, perderemos también el perfil, los entrenamientos y las dietas del mismo.



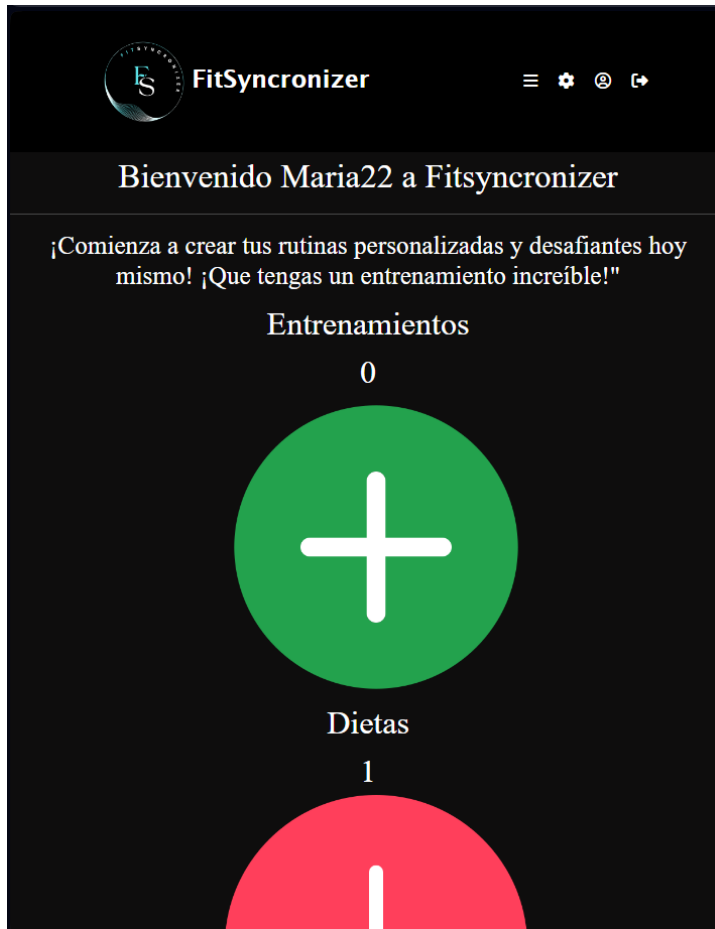
Vista Sobre el Desarrollador

La aplicación contiene un apartado con el CV del desarrollador para poder ver sus años de experiencia, datos personales, etc.

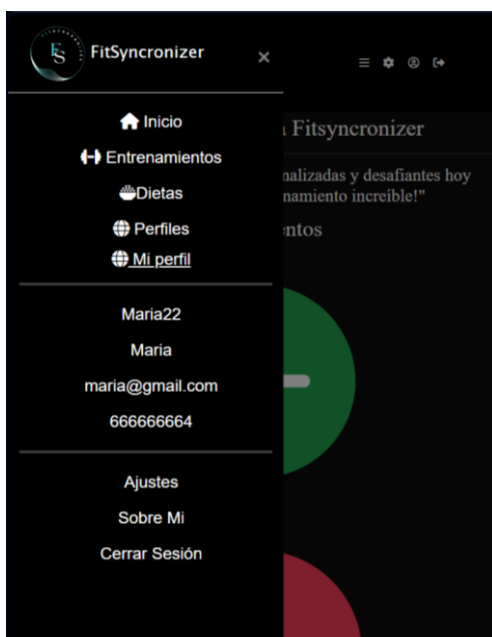


Demostración Responsive de la Web

La aplicación ha sido diseñada completamente responsive, por lo que tengamos el tamaño de pantalla que tengamos, podremos hacer uso de la aplicación, sin dificultades en cualquier parte de la aplicación.



Junto con esto, para navegar por la aplicación se ha implementado en la cabecera superior un desplegable que muestra un modal con el menú de navegación al completo.



Manual de instalación y configuración

Siguiendo este manual, podrás implementar la aplicación web en local en tu dispositivo.

Requisitos:

- Tener un servidor web (en mi caso Apache).
- Crear la BDD de la aplicación.
- PHP (versión 8 recomendada).
- Tener un IDE para utilizar, en mi caso utilizo VSC.

Pasos de instalación:

1. Obtener/Descargar los archivos de la aplicación de repositorio de GitHub.
2. Extraer los archivos o la carpeta en la carpeta raíz del servidor que estes utilizando.
3. Descargar el IDE si no lo tenemos o abrir la carpeta desde el IDE si ya lo tenemos.

Configuración de los archivos:

1. Crearemos/Importaremos en nuestro sistema de gestión de BDD correspondiente la base de datos de FitSynchronizer.
2. Una vez ya podamos acceder desde el servidor en local a la carpeta de la aplicación, deberemos modificar el archivo de configuración de la aplicación para poder conectar la aplicación y la BDD.
3. Modificar las variables del archivo de configuración para poder conectarnos y verificarlo.

Con estos pasos realizados, ya podrás manejar tanto en BackEnd como el FrontEnd desde tu IDE personal y probarlo en local con tu servidor web.

Mantenimiento y evolución

Plan de mantenimiento y soporte

El plan de mantenimiento de la aplicación consistiría en llevar a cabo una revisión de todas las funcionalidades de la aplicación, así como de todos los archivos de la misma. Además, se deberían llevar a cabo unas comprobaciones sobre los usuarios y todos sus datos correspondientes para verificar que no están inactivos, que los perfiles no contienen nada inapropiado dado que no tiene restricción de edad, y verificar que no se está haciendo uso inapropiado de los datos proporcionado en los perfiles.

En cuanto a la lógica de la aplicación, se debería ir actualizando con el paso del tiempo y realizando comprobaciones de que todo funciona correctamente

Identificación de posibles mejoras y evolución del proyecto

La identificación de mejoras para la aplicación podría llevar un tiempo, esto es debido a que cualquier parte de la misma puede ir en constante evolución, así como mejorando y eliminando errores de cualquier tipo.

Algunas de las posibles mejoras que se le podrían realizar a la aplicación serían las siguientes:

- Mejorar el sistema de edición de textos a la hora de crear y modificar ya sean entrenamientos, dietas o perfiles.
- Mejorar el sistema de perfiles públicos, permitiendo que se puedan dar dislikes, escribir o contactar con el usuario del perfil a través de la aplicación, implementar notificaciones de mensajes, etc.
- Optimizar el código lo máximo posible para llevar una buena organización y limpieza de código.
- Añadir o mejorar las validaciones de los campos introducidos por los usuarios.
- Mejorar el sistema de respuestas asíncronas con AJAX y JavaScript nativo.

Entre otras muchas cosas, las enumeradas anteriormente son las que más prioridad llevaría mejorar en la aplicación para mejorar la experiencia del usuario lo máximo posible.

En cuanto a la evolución del proyecto, ha seguido una evolución normal y constante, aunque debido a la falta de tiempo puede que no se hayan logrado cumplir ciertos objetivos o de mejor manera, con tiempo es posible mejorar muchísimo la aplicación para obtener unos resultados aún mejores que los obtenidos

Actualizaciones y futuras mejoras

Las posibles actualizaciones que se podrían añadir a la aplicación consistirían en:

- La creación de un chat para el usuario profesional y el normal
- Subida de publicaciones como dietas o entrenamientos a modo de apoyo para aquellos que vean nuestro perfil
- Mejora de edición de textos para las dietas, entrenamientos o perfiles
- Mejora del panel de administración
- Posibilidad de descargar entrenamientos o dietas en formato de foto jpg.
- Sistema para compartir perfiles

Conclusiones

Evaluación del proyecto

- **Objetivos cumplidos:**
Se han logrado completar los objetivos principales de la aplicación, los cuales son la creación de dietas y entrenamientos a modo de administración y gestión por parte del usuario de manera fácil, rápida e intuitiva.
Además, se ha logrado crear una plataforma publica para dar a conocer diferentes profesionales del sector y promocionar su trabajo a través de la aplicación.
- **Requisitos cumplidos:**
Se ha llevado a cabo la realización de todos los requisitos pedidos en la aplicación, con un previo análisis para lograr obtener un buen resultado final y otorgarle al usuario una aplicación útil, sencilla e intuitiva para su uso personal.
- **Funcionalidad y rendimiento de la aplicación:**
Se han evaluado todas las funcionalidades que permite FitSincronizer y se ha ido optimizado su rendimiento conforme se iba desarrollando la aplicación, de tal manera que se han logrado desarrollar todas las funcionalidades principales y necesarias de la misma.
- **Diseño de la interfaz de usuario:**
Desde el inicio del desarrollo de FitSincronizer, que ha buscado diseñar una interfaz única, bonita y sencilla para los usuarios, logrando obtener todo lo que se buscaba y con un buen resultado para todo tipo de usuarios y pantallas.
Se ha obtenido un diseño agradable que permite al usuario tener una experiencia satisfactoria.

La evaluación de FitSincronizer ha logrado cumplir con las expectativas del proyecto y la idea inicial, cumpliendo con el desarrollo de todo lo previsto, otorgando una buena interfaz de usuario y de la misma manera pudiendo satisfacer las necesidades de utilización del usuario.

Cumplimiento de objetivos y requisitos

- 1- **Base de datos:** Se ha cumplido el requisito mínimo de tener una base de datos relacional con al menos cuatro tablas. FitSincronizer está compuesta de 5 tablas comúnmente relacionadas entre sí.
Además, varias tablas contienen un campo que incluya fechas y todas ellas tienen un identificador único.
- 2- **Desarrollo BackEnd:** La aplicación ha sido desarrollada con PHP nativo, haciendo uso de MVC y optimizando todos los archivos de la mejor manera posible.
Los datos importantes como las contraseñas han sido encriptadas antes de introducirlas en la BDD, mejorando la seguridad y privacidad de los usuarios, así como estableciendo mensajes de error para los casos de que algo no vaya correctamente e informar al usuario.
- 3- **Consultas:** Todas las consultas necesarias que requieran de varias tablas se han realizado apropiadamente, aunque normalmente no son necesarias en FitSincronizer.

- 4- **FrontEnd:** Se ha realizado la aplicación web haciendo uso de HTML, CSS, JavaScript e incluso frameworks como Bootstrap, cumpliendo con el requisito.
- 5- **Diseño Responsive:** toda la aplicación contiene un diseño responsive para permitir a los usuarios utilizarla desde diferentes dispositivos con diferentes tamaños de pantalla, cumpliendo con el requisito.
- 6- **Formularios:** todos los formularios contienen comprobaciones y métodos de seguridad tanto en las vistas como en los controladores para asegurar el ingreso de datos y la seguridad de la aplicación.
- 7- **Conexiones asíncronas:** existen varias funcionalidades que implementan AJAX para mejorar la experiencia del usuario y cumplir con el requisito necesario.

Lecciones aprendidas y recomendaciones para futuros proyectos

Gracias a la realización de este proyecto, he logrado comprender y utilizar mejor todas las tecnologías y lenguajes de programación que he ido viendo a lo largo del curso, desde crear correctamente la BDD y asociarlas con el proyecto hasta el desarrollo BackEnd y FrontEnd de la aplicación. Además, he podido comprender mejor cómo funciona el MVC (Modelo-Vista-Controlador) para estructurar lo mejor posible todos los archivos y mantener una buena organización.

También he podido utilizar herramientas de control de versiones como Git para ir manteniendo la seguridad del código y no perder nada innecesariamente, además de poder mantener diversas copias de seguridad.

Las recomendaciones para futuros proyectos serían mantener una buena estructuración de los archivos y atribuirles buenas metodologías para tener todo bien organizado, así como mejorar con la utilización de Git he ir mejorando, aprendiendo y adaptando a diversos lenguajes de programación.

Bibliografía y referencias

Fuentes utilizadas en el proyecto

- <https://www.php.net/manual/es/intro-what-is.php>
- https://developer.mozilla.org/es/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data