

LIMONITTER

JAVIER HERNANDO DIAZ



Sumario

I. Introducción.....	3
a. Presentación del proyecto.....	3
b. Objetivos del proyecto.....	3
c. Justificación del proyecto.....	4
II. Análisis de requerimientos.....	5
a. Identificación de necesidades y requerimientos.....	5
b. Identificación de público.....	6
c. Estudio de mercado y competencia.....	6
III. Diseño y planificación.....	6
a. Definición de la arquitectura del proyecto.....	6
b. Diseño de la interfaz de usuario.....	6
c. Planificación de las tareas y los recursos necesarios.....	6
IV. Implementación y pruebas.....	6
a. Desarrollo de las funcionalidades del proyecto.....	6
b. Pruebas unitarias y de integración.....	6
c. Corrección de errores y optimización del rendimiento.....	6
V. Documentación.....	6
a. Documentación técnica.....	6
b. Documentación de usuario.....	6
c. Manual de instalación y configuración.....	6
VI. Mantenimiento y evolución.....	6
a. Plan de mantenimiento y soporte.....	6
b. Identificación de posibles mejoras y evolución del proyecto.....	6
c. Actualizaciones y mejoras futuras.....	6
VII. Conclusiones.....	7
a. Evaluación del proyecto.....	7
b. Cumplimiento de objetivos y requisitos.....	7
c. Lecciones aprendidas y recomendaciones para futuros proyectos.....	7
VIII. Bibliografía y referencias.....	7
a. Fuentes utilizadas en el proyecto.....	7
b. Referencias y enlaces de interés.....	7

I. Introducción

a. Presentación del proyecto

¡Bienvenidos a Limonitter, la red social de debates para todos los usuarios! Permítanos presentarles nuestra plataforma, diseñada para brindar una experiencia social en línea segura, confiable y con total libertad de expresión.

En un mundo cada vez más conectado, las redes sociales se han convertido en una herramienta fundamental para la expresión y conexión de personas de todo el mundo. Sin embargo, también hemos sido testigos de restricciones y censura en algunas plataformas, limitando la libertad de expresión de los usuarios. Es por eso que hemos creado Limonitter, una red social diseñada para fomentar y proteger la libertad de expresión en línea.

Limonitter se basa en los principios de privacidad y libertad de expresión, brindando a nuestros usuarios un espacio seguro y sin censura para compartir sus pensamientos, ideas y opiniones. Nuestra plataforma promueve la diversidad de voces y opiniones, alentando el intercambio de ideas y el debate saludable.

b. Objetivos del proyecto

Los objetivos del proyecto Limonitter son:

1. **Promover la libertad de expresión:** Permitir a los usuarios expresar libremente sus pensamientos, ideas y opiniones sin temor a la censura o represalias. Fomentar un ambiente inclusivo que valore la diversidad de perspectivas.
2. **Facilitar el debate abierto y constructivo:** Proporcionar una plataforma donde los usuarios puedan participar en debates respetuosos y enriquecedores sobre una amplia variedad de temas. Fomentar la discusión saludable y el intercambio de ideas entre los miembros de la comunidad.
3. **Garantizar la privacidad anónima:** Salvaguardar la identidad y la información personal de los usuarios, permitiendo que se expresen sin temor a ser identificados o perseguidos. Ofrecer opciones de privacidad robustas y respetar la confidencialidad de los datos de los usuarios.
4. **Fomentar el respeto y la legalidad:** Establecer políticas claras para promover un entorno en el que se respeten los derechos y las creencias de los demás. Prohibir contenido ilegal, ofensivo o difamatorio, y tomar medidas para garantizar el cumplimiento de las normas y regulaciones aplicables.
5. **Mejorar constantemente la seguridad y la confiabilidad:** Implementar medidas de seguridad sólidas para proteger la plataforma y los datos de los usuarios contra amenazas ciberneticas. Mantener un sistema de moderación eficiente para detectar y abordar rápidamente cualquier actividad inapropiada o abusiva.
6. **Fomentar la participación activa de la comunidad:** Incentivar la participación de los usuarios en la mejora continua de la plataforma, recopilando comentarios y sugerencias para realizar actualizaciones y mejoras que se alineen con las necesidades de la comunidad.

c. Justificación del proyecto

La justificación de la creación del proyecto Limonitter se basa en la necesidad de brindar una plataforma de redes sociales que promueva la libertad de expresión y aborde las deficiencias presentes en otras plataformas existentes.

En muchas redes sociales actuales, la libertad de expresión se ve limitada debido a políticas restrictivas y al riesgo de censura. Esto dificulta que los usuarios compartan sus opiniones y perspectivas de manera abierta y sin miedo a represalias. Limonitter surge como una alternativa que busca empoderar a los usuarios y permitirles expresarse libremente, sin limitaciones injustificadas.

Además, en muchas plataformas sociales, el control y la moderación inadecuada de las malas conductas y comportamientos inapropiados por parte de los usuarios ha generado un entorno tóxico y perjudicial. Esto ha llevado a situaciones de acoso, abuso y difamación que afectan negativamente la experiencia de los usuarios. Limonitter se compromete a implementar medidas de moderación efectivas para garantizar un entorno seguro y respetuoso, donde los usuarios puedan participar en debates constructivos sin temor a enfrentar situaciones de hostilidad o agresión.

Al abordar estas problemáticas, Limonitter busca ofrecer una plataforma en la que los usuarios se sientan libres de expresar sus ideas y opiniones, debatir de manera constructiva y respetar la diversidad de perspectivas. Además, se busca garantizar la privacidad de los usuarios, ofreciendo opciones de anonimato que les permitan compartir sus pensamientos sin preocupaciones de ser identificados o perseguidos.

Resumiendo un poco, la justificación de la creación de Limonitter radica en la necesidad de una red social que priorice la libertad de expresión, controle de manera efectiva las malas conductas de los usuarios y promueva un entorno seguro y respetuoso. Mediante la implementación de políticas y funcionalidades adecuadas, Limonitter busca marcar la diferencia y brindar una experiencia en línea más satisfactoria para sus usuarios.



II. Análisis de requerimientos

a. Identificación de necesidades y requerimientos

1. **Requerimientos funcionales:** Como funciones principales a tener en cuenta en Limonitter nos encontramos la capacidad de albergar miles de usuarios en nuestra web, para la mayor interacción posible entre la comunidad, el usuario no solo puede ver todas las publicaciones existentes, si no también las más destacadas, a las personas que sigue, sus propias publicaciones junto a su perfil, todos los perfiles públicos de los usuarios que no sigue o sigue, ademas de poder comentar en ellos , y crear nuevas publicaciones y incluso borrarlas si lo desea, ademas de modificar su propio usuario, por ejemplo la contraseña actual.
2. **Requerimientos no funcionales:** Debemos tener en cuenta la gran potencia de uso en una posible masa de usuarios activos, ya que incluso en ese caso, podemos estar tranquilos por que tenemos una seguridad en la que el usuario no podrá interactuar con el entorno si no es por medio de una cuenta de usuario, lo que llamara la atención de los curioso y aquellos que quieran saber más de las publicaciones que pueden ver sin loguearse, algo importante también es su seguridad y utilización de sesiones y cookies para ayudar al usuario.
3. **Necesidades de privacidad y anonimato:** Hay que tener en cuenta que todas las publicaciones y comentarios son totalmente anónimas, ya que nuestro perfil solo tendrá como información el nombre de usuario, que podemos elegir con total libertad, nada de uso de información confidencial como numero telefónicos o direcciones junto a nombres o apellidos reales.
4. **Requisitos de moderación y control:** Aunque velamos por una total libertad de expresión ante debates o publicaciones con comentarios, hay que tener en cuenta que no permitiremos abuso de leyes y derechos de ningún tipo, y sera castigado con la penalización total de la cuenta e incluso de futuras cuentas que puedas crear por medio de IP por ejemplo.
5. **Interfaz de usuario intuitiva:** En Limonitter destaca la importancia de una interfaz de usuario fácil de usar y atractiva, que permita a los usuarios navegar por el sitio, publicar contenido, interactuar con otros usuarios y acceder a las funciones de forma intuitiva.
6. **Requisitos de escalabilidad:** Limonitter esta hecho de tal forma que un usuario sin cuenta pueda ver las publicaciones de los usuarios más recientes pero no poder interactuar o ver más haya de la publicación sin loguearnos, lo que hará picar la curiosidad los usuarios y se terminaran haciendo una cuenta, por la facilidad que hay en registrarse sabemos que conseguiremos un gran numero de usuarios en poco tiempo y escalará al alza.
7. **Integraciones y compatibilidad:** Limonitter es compatible con todos los navegadores actuales más actualizados, como puede ser Chrome, Firefox, Edge, Duck, Opera...etc. Solo en algunas versiones antiguas de Internet Explorer podría causar problemas de estilos.
8. **Experiencia del usuario:** En la experiencia de usuario nos encontramos respuestas rápidas visuales sin necesidad de recargar la pagina con AJAX, la libertad de crear o eliminar publicaciones y fáciles búsquedas por links de apartados importantes como las Tendencias.

b. Identificación de público

1. **Audiencia objetivo amplia:** Al no tener restricciones de edad, género o nacionalidad, la audiencia objetivo de Limonitter se vuelve más inclusiva y diversa. Esto implica que cualquier persona, sin importar su edad, género o ubicación geográfica, puede unirse y participar en la red social.
2. **Enfoque en la mentalidad y actitud:** Dado que no se consideran criterios demográficos específicos, la identificación de público se basa más en la mentalidad y actitud de los usuarios potenciales. En lugar de enfocarse en la edad o el género, Limonitter busca atraer a personas con una mentalidad abierta, que valoren la libertad de expresión y estén interesadas en debatir y discutir diversos temas.
3. **Intereses y pasiones compartidas:** La identificación de público en Limonitter se basa en los intereses y pasiones compartidas de los usuarios. La red social busca atraer a personas que estén interesadas en discutir una amplia gama de temas, desde política y actualidad hasta entretenimiento y estilo de vida. El objetivo es conectar a personas con intereses comunes y fomentar debates enriquecedores.
4. **Comunidad inclusiva y global:** Al no tener restricciones geográficas, Limonitter busca crear una comunidad inclusiva y global donde personas de diferentes países y culturas puedan conectarse y participar. La identificación de público se centra en atraer a personas que estén abiertas a la diversidad cultural y que estén dispuestas a compartir y aprender de diferentes perspectivas.
5. **Flexibilidad y adaptabilidad:** Dado que no se establecen restricciones, Limonitter debe ser flexible y adaptable para satisfacer las necesidades y preferencias de una audiencia diversa. La identificación de público implica comprender las distintas formas en que los usuarios pueden interactuar con la plataforma y ajustarla en consecuencia para ofrecer una experiencia personalizada



c. Estudio de mercado y competencia

El mercado de las redes sociales ha experimentado un crecimiento significativo en los últimos años, con una demanda creciente de plataformas que ofrezcan mayor privacidad y libertad de expresión. Limonitter busca aprovechar estas tendencias al dirigirse a un público diverso que valora la libertad de expresión y busca una experiencia de red social más privada.

En términos de competencia, Twitter es una red social establecida y popular que ofrece características similares a Limonitter, como la capacidad de compartir contenido en tiempo real y de forma concisa. Sin embargo, Limonitter se diferencia al enfocarse en garantizar una mayor privacidad y anonimato para sus usuarios, lo que aborda una necesidad no satisfecha en el mercado. Esto brinda a los usuarios la oportunidad de debatir y discutir diversos temas sin temor a represalias o violaciones de privacidad.

Limonitter se posiciona como una alternativa a Twitter al ofrecer a los usuarios una mayor libertad de expresión, privacidad y respeto en sus interacciones en línea. Esto le permite diferenciarse en un mercado competitivo y captar la atención de un público diverso que busca una red social más segura y enfocada en la libre expresión.

III. Diseño y planificación

a. Definición de la arquitectura del proyecto

En el proyecto Limonitter se utilizan HTML, CSS y JavaScript para crear la interfaz de usuario y la interactividad en el lado del cliente. PHP se utiliza en el servidor para procesar la lógica del negocio y comunicarse con la base de datos MySQL. AJAX se utiliza para realizar actualizaciones asíncronas de la página sin recargarla por completo. XAMPP con Apache y phpMyAdmin se utilizan como entorno de desarrollo local para ejecutar el servidor web y administrar la base de datos MySQL. Además todo el proyecto sigue la estructura MVC.

HTML (HyperText Markup Language) es el lenguaje de marcado estándar utilizado para crear la estructura y el contenido de las páginas web. Permite definir la estructura de un documento mediante elementos y etiquetas, como encabezados, párrafos, enlaces, imágenes, formularios, entre otros.

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para controlar la presentación y el diseño de las páginas web. Permite definir estilos, como colores, fuentes, márgenes, tamaños, efectos visuales, etc., y aplicarlos a los elementos HTML para lograr una apariencia visual coherente y atractiva.

JavaScript es un lenguaje de programación que se utiliza principalmente en el desarrollo web para crear interactividad en las páginas. Permite agregar comportamientos dinámicos, como validación de formularios, animaciones, cambios en tiempo real, interacciones con el usuario y comunicación con el servidor.

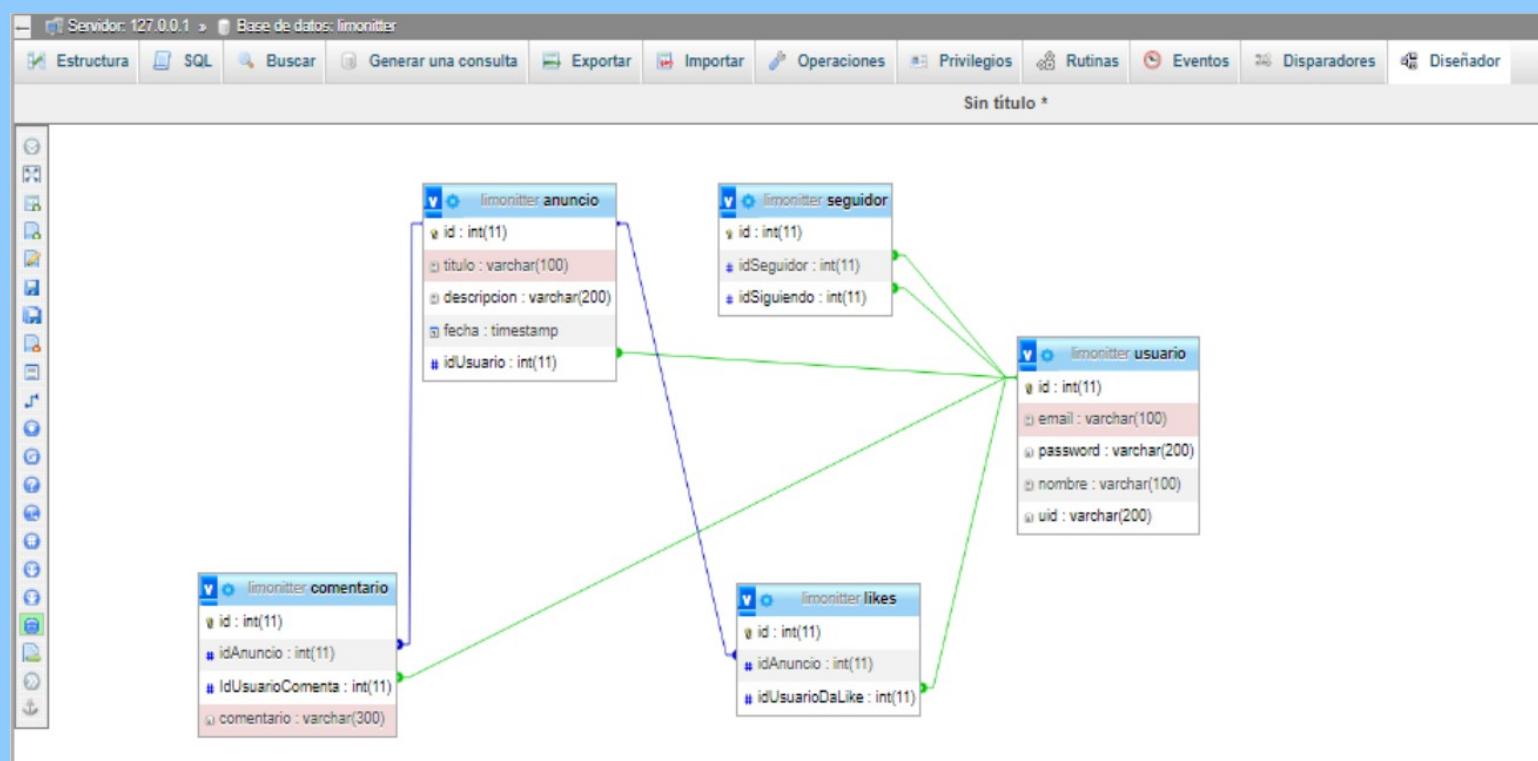
PHP (Hypertext Preprocessor) es un lenguaje de programación del lado del servidor ampliamente utilizado en el desarrollo web. Permite generar contenido dinámico, interactuar con bases de datos, procesar formularios, manejar sesiones de usuario, realizar cálculos, entre otras tareas del lado del servidor.

AJAX (Asynchronous JavaScript and XML) es una técnica de desarrollo web que utiliza JavaScript para enviar y recibir datos en segundo plano sin necesidad de recargar toda la página. Permite realizar solicitudes asíncronas al servidor y actualizar partes específicas de la página, lo que mejora la experiencia del usuario al proporcionar respuestas rápidas y fluidas.

XAMPP es un paquete de software gratuito que proporciona un entorno de servidor web local para el desarrollo y la prueba de aplicaciones web. Incluye Apache como servidor web, MySQL como sistema de gestión de bases de datos y PHP como lenguaje de programación del lado del servidor. Proporciona una configuración predefinida y fácil de usar para crear un entorno de desarrollo web en local.

phpMyAdmin es una herramienta de administración web basada en PHP que permite gestionar bases de datos **MySQL** de forma gráfica. Proporciona una interfaz intuitiva para crear, modificar y eliminar bases de datos, tablas, registros, ejecutar consultas SQL y realizar tareas de mantenimiento en la base de datos.

A continuación te muestro la estructura de Base de Datos que utiliza Limonitter:



Como podemos observar tenemos 5 tablas, anuncio son las publicaciones de los usuarios, los comentarios son aquellos creados por los usuarios y que se contienen en las publicaciones de los mismos, los likes son los me gusta de cada publicación o anuncio por parte de los usuarios, y los seguidores contiene una doble **FK o foreign key** para saber quien es el que sigue al usuario indicado, todas las tablas contienen una **PK o primary key**.

b. Diseño de la interfaz de usuario

El diseño de la interfaz de usuario es algo básico pero agradable y divertido, los colores más notables son los morados, rosas, azules, amarillos, blancos y negros, de fondo principal tenemos una imagen GIF en movimiento tipo pixel game y todas las vistas contiene un div apropiado con un fondo gradiante, se utilizan algunas animaciones en los títulos, la barra de navegación nav da un tono no totalmente oscuro pero llevadero para aquellos que no les guste los colores claros y tengan más amor al modo oscuro. Tenemos un logo como icono al lado del nombre Limonitter que a la vez es un link directo a inicio donde encontraras los Limonitters, he utilizado distintos tipos de botones, ventanas modales y favicons para las publicaciones, comentarios, likes...etc.

Arriba a la derecha encontraremos un despliegue en el que encontraremos la posibilidad de Iniciar Sesión si no lo hemos echo ya, ademas de unos links apropiados que te dirigen a Inicio y Sobre Mi, si estas logueado o te has registrado en esta misma columna también aparecerán Apartados, que contienen el Para Ti, Mis Limonitters, Tendencias y Descubrir Personas.

Si lo deseamos tenemos un botón decorado con gradiant para Cerrar la Sesión del usuario activo.

Se han utilizado fuentes directas de CSS y GoogleFont. La pagina también esta preparada para ser Responsive y adaptarse a diferentes tamaños tanto para Monitor como un Smartphone.

Todo lo mencionado se ha organizado con la información para facilitar la navegación y la comprensión de la interfaz y el usuario este lo más cómodo posible.

c. Planificación de las tareas y los recursos necesarios

Definición de objetivos:

- Crear una plataforma de redes sociales temática que fomente la interacción y participación activa de los usuarios, garantizando su privacidad y anonimato.
- Proporcionar un espacio en línea seguro donde los usuarios puedan expresarse libremente sin temor a represalias, promoviendo la libertad de expresión de manera responsable y ética.
- Establecer políticas y mecanismos para garantizar que el contenido compartido en Limonitter cumpla con las leyes y reglamentos vigentes, promoviendo un entorno en línea saludable y seguro.

Identificación de funcionalidades:

1. Registro y autenticación segura:

- Permitir a los usuarios crear una cuenta en Limonitter de forma anónima, utilizando alias en lugar de nombres reales. 1 Hora.
- Implementar un proceso de autenticación robusto para proteger la identidad de los usuarios y evitar cuentas falsas. 2 Horas.

2. Configuración de privacidad y seguridad:

- Ofrecer opciones avanzadas de configuración de privacidad para que los usuarios controlen quién puede ver y acceder a su contenido. 2 Horas.
- Establecer filtros y mecanismos de bloqueo para prevenir interacciones no deseadas o abusivas. 2 Horas.

3. Publicación de comentarios :

- Permitir a los usuarios publicar comentarios de forma anónima, sin revelar su identidad real. 1 Hora.
- Implementar un límite de caracteres para las publicaciones y proporcionar opciones de formato básico. 1 Hora.

4. Interacción y seguimiento:

- Habilitar la opción de seguir a otros usuarios y recibir notificaciones sobre sus publicaciones. 2 Horas.
- Proporcionar funciones de interacción como "me gusta", comentarios y compartición de mensajes. 2 Horas.

5. Reporte y moderación de contenido:

- Establecer un sistema de reporte para que los usuarios puedan denunciar contenido inapropiado o abusivo. En proceso a futuro.
- Implementar un equipo de moderación que revise y tome acciones sobre los informes recibidos. En proceso a futuro.

Desglose de tareas:

1. Diseño y desarrollo de la interfaz de usuario:

- Crear una interfaz intuitiva y atractiva que refleje la identidad de Limonitter y facilite la navegación de los usuarios. 2 Horas.
- Diseñar la estructura de las páginas, incluyendo perfiles de usuarios, línea de tiempo, opciones de configuración, etc. 2 Horas.

2. Implementación del sistema de registro y autenticación:

- Desarrollar un proceso de registro seguro que valide la información del usuario sin revelar su identidad real. 2 Horas.
- Implementar un sistema de autenticación robusto utilizando tecnologías criptográficas y protocolos de seguridad. 3 Horas.

3. Desarrollo de funcionalidades de publicación y seguimiento:

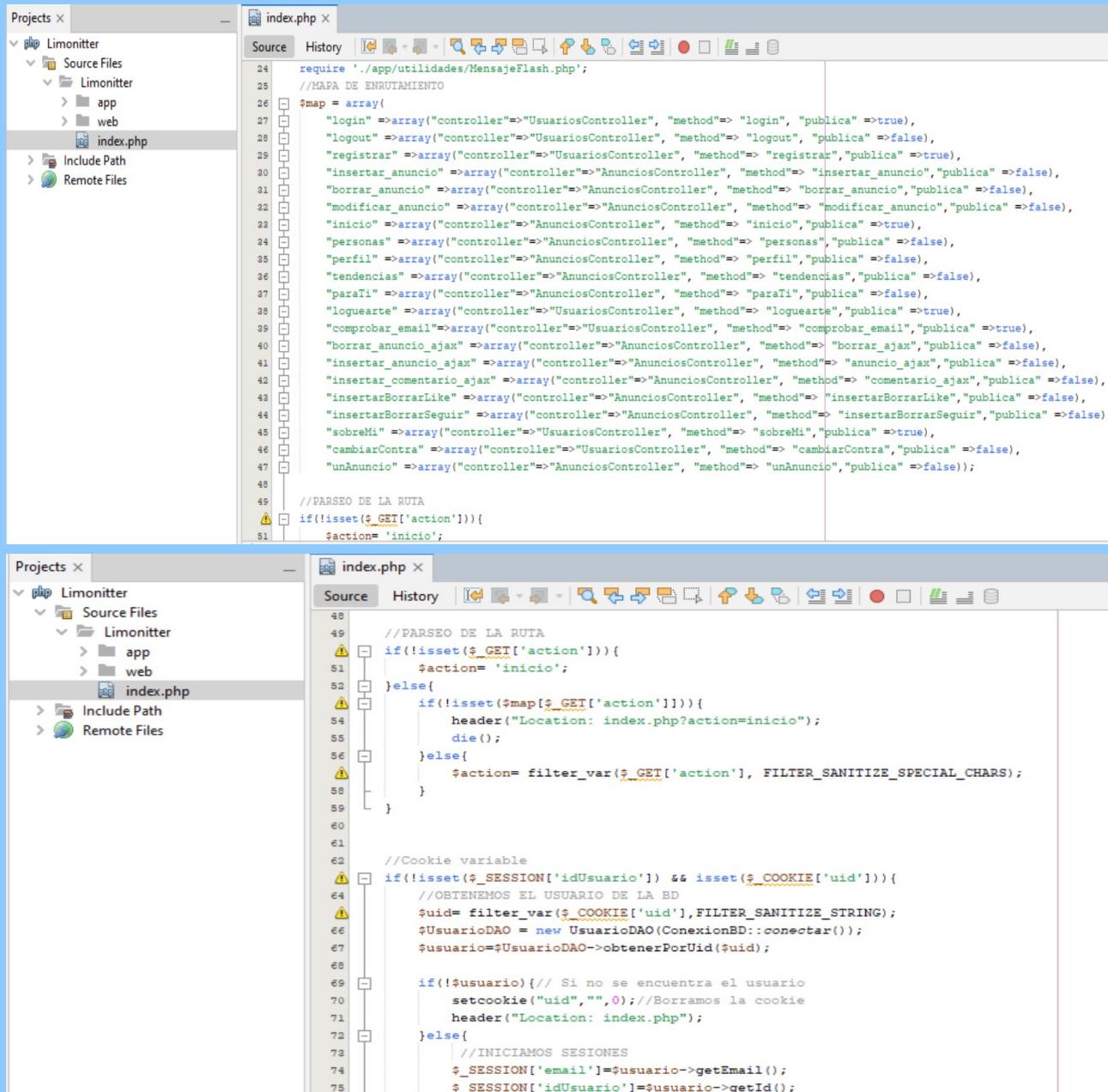
- Permitir a los usuarios publicar mensajes de forma anónima y establecer un límite de caracteres. 2 Horas.
- Desarrollar funciones de seguimiento de usuarios y notificaciones de actividades relevantes. 2 Horas.

IV. Implementación y pruebas

a. Desarrollo de las funcionalidades del proyecto

Comenzaremos dividiendo y viendo como se compone las carpetas del proyecto y el MVC:

El proyecto se compone principalmente de una carpeta llamada Limonitter, en ella se encuentran 2 carpetas llamadas app y web y fuera de ellos el archivo index.php, primero veremos el contenido de el archivo index.php:



```
index.php (Top Version)
24 require './app/utilidades/MensajeFlash.php';
25 //MAPA DE ENRUTAMIENTO
26 $map = array(
27     "login" =>array("controller"=>"UsuariosController", "method"=> "login", "publica" =>true),
28     "logout" =>array("controller"=>"UsuariosController", "method"=> "logout", "publica" =>false),
29     "registrar" =>array("controller"=>"UsuariosController", "method"=> "registrar", "publica" =>true),
30     "insertar_anuncio" =>array("controller"=>"AnunciosController", "method"=> "insertar_anuncio", "publica" =>false),
31     "borrar_anuncio" =>array("controller"=>"AnunciosController", "method"=> "borrar_anuncio", "publica" =>false),
32     "modificar_anuncio" =>array("controller"=>"AnunciosController", "method"=> "modificar_anuncio", "publica" =>false),
33     "inicio" =>array("controller"=>"AnunciosController", "method"=> "inicio", "publica" =>true),
34     "personas" =>array("controller"=>"AnunciosController", "method"=> "personas", "publica" =>false),
35     "perfil" =>array("controller"=>"AnunciosController", "method"=> "perfil", "publica" =>false),
36     "tendencias" =>array("controller"=>"AnunciosController", "method"=> "tendencias", "publica" =>false),
37     "paraTi" =>array("controller"=>"AnunciosController", "method"=> "paraTi", "publica" =>false),
38     "loguearte" =>array("controller"=>"UsuariosController", "method"=> "loguearte", "publica" =>true),
39     "comprobar_email"=>array("controller"=>"UsuariosController", "method"=> "comprobar_email", "publica" =>true),
40     "borrar_anuncio_ajax" =>array("controller"=>"AnunciosController", "method"=> "borrar_ajax", "publica" =>false),
41     "insertar_anuncio_ajax" =>array("controller"=>"AnunciosController", "method"=> "anuncio_ajax", "publica" =>false),
42     "insertar_comentario_ajax" =>array("controller"=>"AnunciosController", "method"=> "comentario_ajax", "publica" =>false),
43     "insertarBorrarLike" =>array("controller"=>"AnunciosController", "method"=> "insertarBorrarLike", "publica" =>false),
44     "insertarBorrarSeguir" =>array("controller"=>"AnunciosController", "method"=> "insertarBorrarSeguir", "publica" =>false),
45     "sobreMi" =>array("controller"=>"UsuariosController", "method"=> "sobreMi", "publica" =>true),
46     "cambiarContra" =>array("controller"=>"UsuariosController", "method"=> "cambiarContra", "publica" =>false),
47     "unAnuncio" =>array("controller"=>"AnunciosController", "method"=> "unAnuncio", "publica" =>false);
48
49 //PARSEO DE LA RUTA
50 if(!isset($_GET['action'])){
51     $action= 'inicio';
52 }
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
```

```
index.php (Bottom Version)
48 //PARSEO DE LA RUTA
49 if(!isset($_GET['action'])){
50     $action= 'inicio';
51 }else{
52     if(!isset($map[$_GET['action']])){
53         header ("Location: index.php?action=inicio");
54         die();
55     }else{
56         $action= filter_var($_GET['action'], FILTER_SANITIZE_SPECIAL_CHARS);
57     }
58 }
59
60
61
62 //Cookie variable
63 if(!isset($_SESSION['idUsuario']) && isset($_COOKIE['uid'])){
64     //OBTEMOS EL USUARIO DE LA BD
65     $uid= filter_var($_COOKIE['uid'],FILTER_SANITIZE_STRING);
66     $UsuarioDAO = new UsuarioDAO(ConexionBD::conectar());
67     $usuario=$UsuarioDAO->obtenerPorUid($uid);
68
69     if(!$usuario){ // Si no se encuentra el usuario
70         setcookie("uid","",0); //Borramos la cookie
71         header("Location: index.php");
72     }else{
73         //INICIAMOS SESIONES
74         $_SESSION['email']=$usuario->getEmail();
75         $_SESSION['idUsuario']=$usuario->getId();
```

Este archivo es muy importante ya que contiene el Controlador Frontal, Variables de sesión, Requires de los Modelos, Mapa de Enrutamiento otros archivos y funciones requeridas para el correcto funcionamiento de la pagina web. Su funcionamiento es sencillo, llegada de acciones y métodos junto a creación de sesiones y cookies para la seguridad de la pagina.

Ahora pasamos a las carpetas web, en ella se encuentran archivos sencillos, como el CSS o las imágenes utilizadas para el estilo de la página web, en ella también podría contenerse el código Javascript utilizado, pero por motivos de funcionamiento individual este se contiene en las vistas individuales del proyecto, en un futuro podría ser por separada perfectamente.

```

4  /*
5  * @author Javier Hernando Diaz
6  * @date 06/06/23
7  * @version 2.0
8  */
9  @import url('https://fonts.googleapis.com/css2?family=Pacifico&display=swap');
10 *{margin: 0; box-sizing: border-box;}
11 body{
12     background: black;
13     background-image: url("../web/img/body.gif");
14     background-position: center;
15     background-repeat: no-repeat;
16     background-size: cover;
17 }
18 .logo-image {
19     width: 40px;
20     height: 40px;
21     object-fit: contain;
22     border-radius: 10px;
23 }
24 .flash-message {
25     display: inline-block;
26     padding: 10px 20px;
27     background: linear-gradient(135deg, #1976d2, #9c27b0);
28     color: #ffffff;
}

```

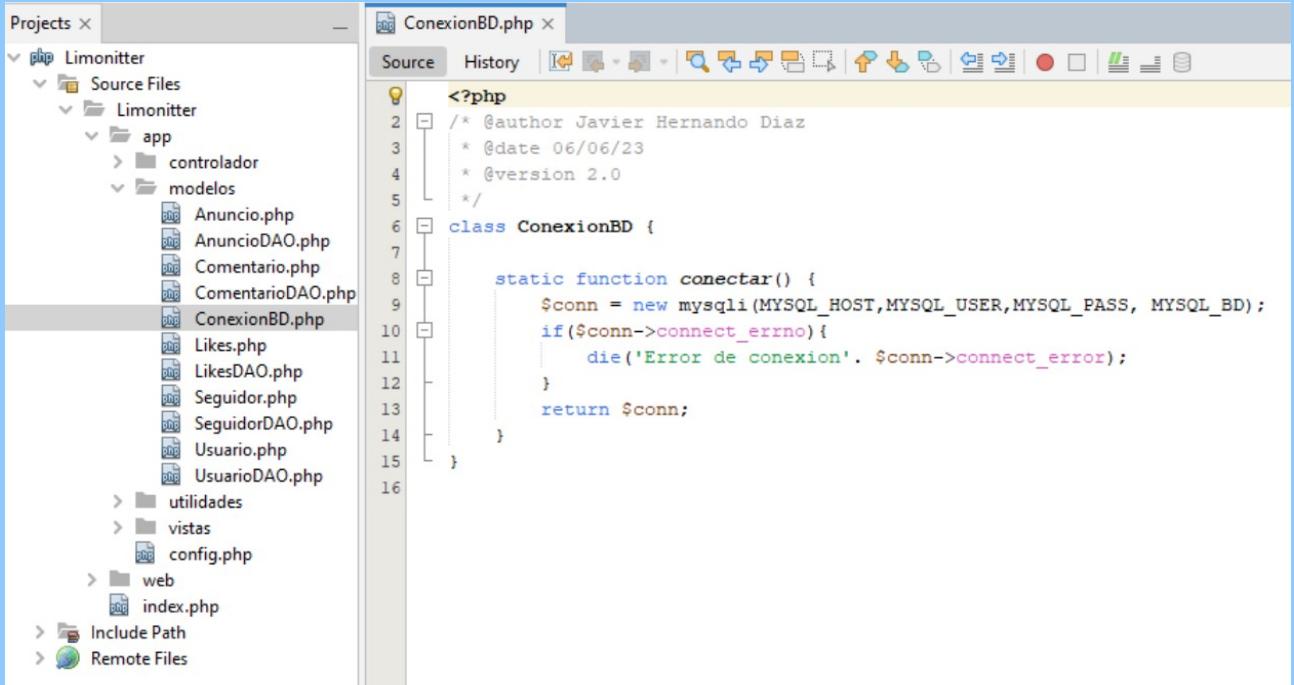
Ahora llegamos al apartado más importante del proyecto, la carpeta app contiene todo el MVC y otras utilidades del proyecto web, se dividen en carpetas que son, controlador, modelos, vistas, utilidades y fuera de ellas un archivo llamado config.php que es el que contiene la configuración a la conexión de la base de datos .

```

1 ?php
2 define("MYSQL_USER", "root");
3 define("MYSQL_PASS", "");
4 define("MYSQL_HOST", "localhost");
5 define("MYSQL_DB", "limonitter");
6 //Constantes de conexión MySQL
7 /* @author Javier Hernando Diaz
8 * @date 06/06/23
9 * @version 2.0
10 */

```

Para empezar vamos a desglosar la carpeta Modelos, en ella se encuentran todas las clases y clases DAO necesarias para la creación de objetos de tipo partiendo de la BBDD, en ella están Anuncio (son las Publicaciones),Comentario, Likes, Seguidor y Usuario. Cada uno con sus respectivos DAO que contienen las Querys y Funciones a BBDD para el completo CRUD, también se puede observar un archivo extra llamado ConexionBD.php que contiene una function static que es la hace la conexión.

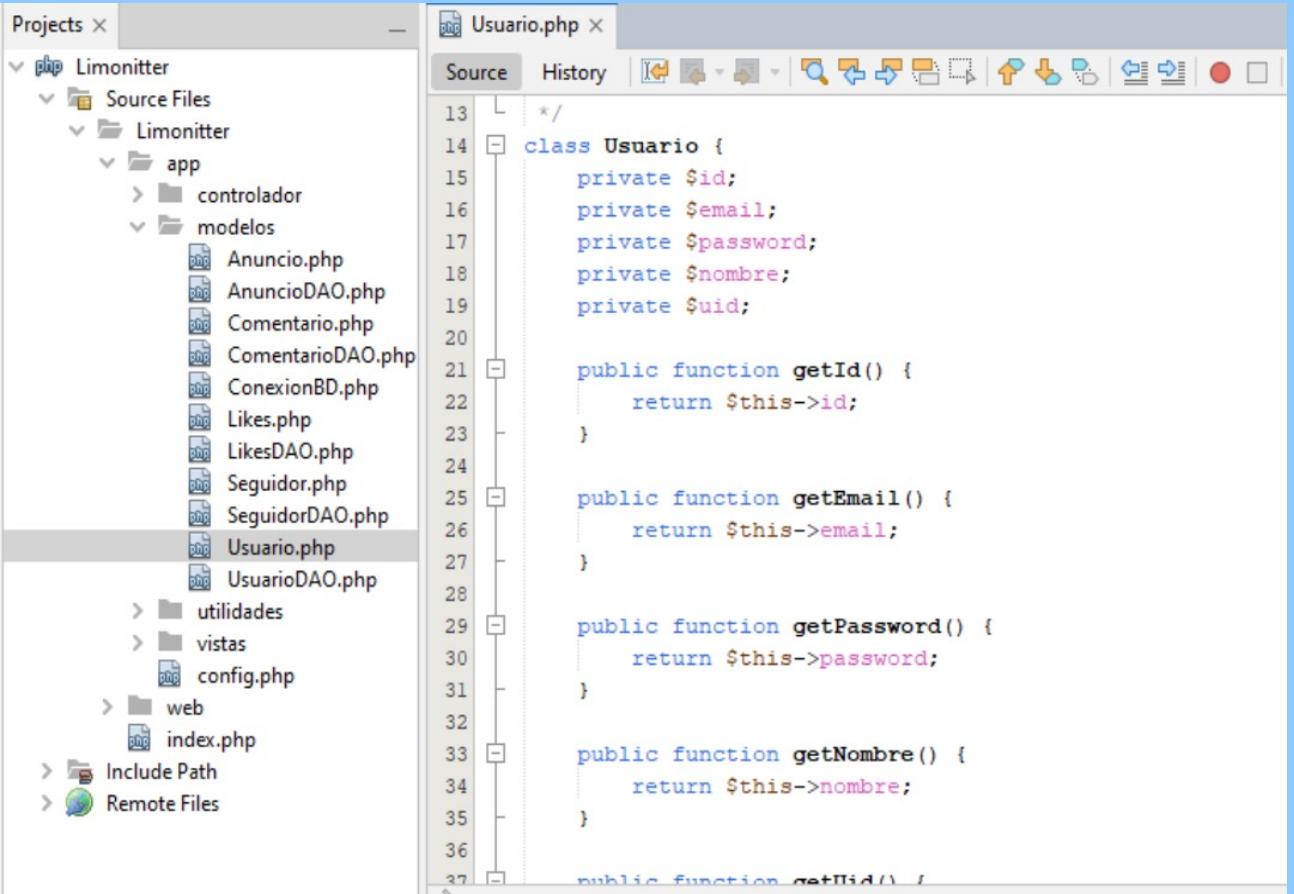


```

<?php
/*
 * @author Javier Hernando Diaz
 * @date 06/06/23
 * @version 2.0
 */
class ConexionBD {
    static function conectar() {
        $conn = new mysqli(MYSQL_HOST,MYSQL_USER,MYSQL_PASS, MYSQL_DB);
        if($conn->connect_errno){
            die('Error de conexion'. $conn->connect_error);
        }
        return $conn;
    }
}

```

Ahora mostrare la **Clase Usuario y UsuarioDAO** como ejemplo para observar su contenido:



```

/*
class Usuario {
    private $id;
    private $email;
    private $password;
    private $nombre;
    private $uid;

    public function getId() {
        return $this->id;
    }

    public function getEmail() {
        return $this->email;
    }

    public function getPassword() {
        return $this->password;
    }

    public function getNombre() {
        return $this->nombre;
    }

    public function getApellido() {
        return $this->apellido;
    }
}

```

```

6  class UsuarioDAO {
7      private $conn;
8
9      public function __construct($conn) {
10         if(!$conn instanceof mysqli){
11             echo "Error: No se pudo conectar a MySQL . Error" . mysqli_connect_errno() . ":" . mysqli_
12             return false;
13         }
14         $this->conn=$conn;
15     }
16
17     public function obtenerUsuarios() {
18         $sql="SELECT * FROM usuario";
19         if(!$result=$this->conn->query($sql)){
20             die("Error al ejecutar la SQL ".$this->conn->error);
21         }
22         $arrayusu= array();
23         while($usuários = $result->fetch_object('Usuario')){
24             $arrayusu[]=$usuários;
25         }
26         return $arrayusu;
27     }
28     public function obtenerUsu(int $id){
29         $sql="SELECT * FROM usuario WHERE id= ?";

```

Aunque parece complicado en realidad las Clases normales solo contienen los get y set correspondientes a las variables establecidas por las tablas de la BBDD, y los DAO comienzan con el constructor de la conexión que es la misma para todos los DAO, y después las Function a Queries que deseamos hacer con CRUD.

Ahora vamos a centrarnos en la Carpeta Controlador ,en ella se existen 2 archivos que dividen las tareas en Anuncios Controller que son todas las funciones necesarias para implementar las utilidades de la web, ya sea crear una Publicación, Dar Like, Comentar una publicación, ver perfiles, Seguir a alguien ...etc.

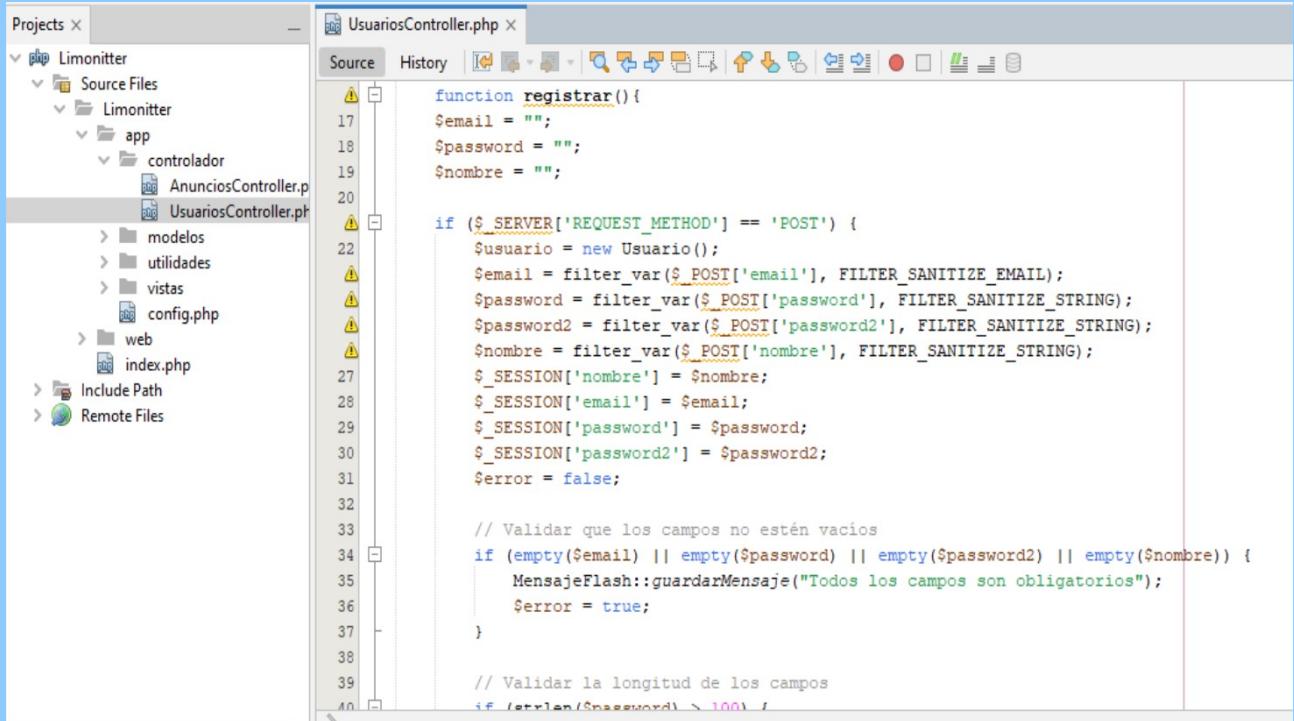
```

7  class AnunciosController {
9      function inicio(){
10         if(isset($_SESSION['idUsuario'])){
11             $idUsuario=$_SESSION['idUsuario'];
12         }else{
13             $idUsuario=0;
14         }
15         $anuncioDAO = new AnuncioDAO(ConexionBD::conectar());
16         $usuarioDAO = new UsuarioDAO(ConexionBD::conectar());
17         $comentarioDAO = new ComentarioDAO(ConexionBD::conectar());
18         $likesDAO = new LikesDAO(ConexionBD::conectar());
19         //Obtengo todos los mensajes de la BD
20         $anuncios = $anuncioDAO->obtenerTodos();
21         $comentarios = $comentarioDAO->obtenerTodos();
22         $usuarios = $usuarioDAO->obtenerUsuarios();
23         $likes = $likesDAO->obtenerTodos();
24
25         //Incluimos la vista
26         require 'app/vistas/inicio.php';
27     }
28     function perfil(){
29         $idPerfil = filter_var($_GET['id'], FILTER_SANITIZE_NUMBER_INT);
30         if(isset($_SESSION['idUsuario'])){
31             $idUsuario=$_SESSION['idUsuario'];

```

Por ultimo tenemos Usuarios Controller que es el que da las ordenes y funciones necesarias para la seguridad, creación y logueo de usuarios de toda la pagina, es el que nos protege de todo mal si alguien quisiera hacer trampas en nuestra web o romper la privacidad y seguridad del mismo.

Algo interesante que podemos comentar es la creación de contraseñas con Hash lo que permite la encriptado de las contraseñas creadas.



The screenshot shows the PHP code for the `UsuariosController.php` file. The code defines a `registrar()` function that handles user registration. It starts by initializing variables `$email`, `$password`, and `$nombre`. It then checks if the request method is POST. If so, it creates a new `Usuario` object and sanitizes the input fields (`$email`, `$password`, `$password2`, `$nombre`) using `filter_var` with appropriate filters. It also sets session variables for the user's information. The code then validates that all required fields are filled and checks if the password length is greater than 100 characters. If any validation fails, it sets an error flag and displays a flash message indicating that all fields are mandatory.

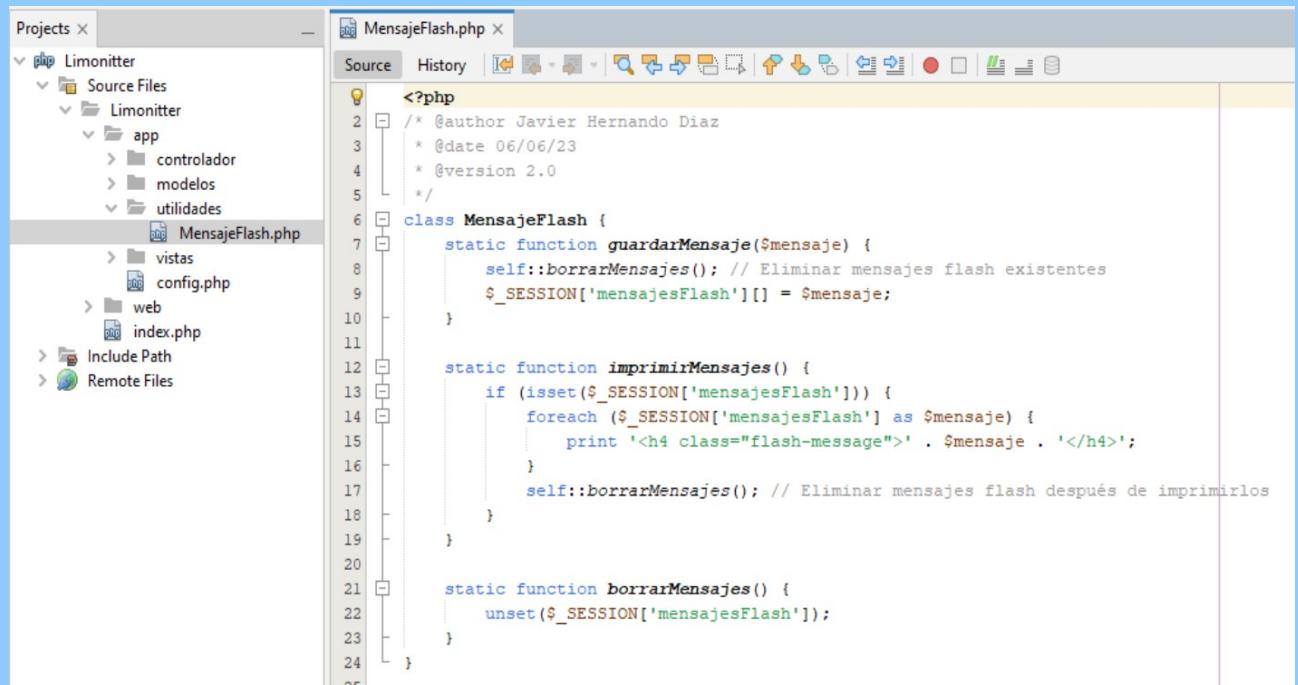
```
function registrar() {
    $email = "";
    $password = "";
    $nombre = "";

    if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $usuario = new Usuario();
        $email = filter_var($_POST['email'], FILTER_SANITIZE_EMAIL);
        $password = filter_var($_POST['password'], FILTER_SANITIZE_STRING);
        $password2 = filter_var($_POST['password2'], FILTER_SANITIZE_STRING);
        $nombre = filter_var($_POST['nombre'], FILTER_SANITIZE_STRING);
        $_SESSION['nombre'] = $nombre;
        $_SESSION['email'] = $email;
        $_SESSION['password'] = $password;
        $_SESSION['password2'] = $password2;
        $error = false;

        // Validar que los campos no estén vacíos
        if (empty($email) || empty($password) || empty($password2) || empty($nombre)) {
            MensajeFlash::guardarMensaje("Todos los campos son obligatorios");
            $error = true;
        }

        // Validar la longitud de los campos
        if (strlen($password) > 100) {
            MensajeFlash::guardarMensaje("La contraseña debe tener al menos 100 caracteres");
            $error = true;
        }
    }
}
```

La carpeta Utilidades sin embargo sirve para la creación de clases que no provengan de la misma BBDD , por ejemplo tenemos una clase llamada MensajeFlash que crea y devuelve mensajes a diferentes vistas desde los controladores a partir de variables de sesión.



The screenshot shows the PHP code for the `MensajeFlash.php` file. The code defines a class `MensajeFlash` with three static methods: `guardarMensaje()`, `imprimirMensajes()`, and `borrarMensajes()`. The `guardarMensaje()` method adds a message to the `$_SESSION['mensajesFlash']` array. The `imprimirMensajes()` method iterates through this array and prints each message as an `<h4>` element. The `borrarMensajes()` method removes the array from the session.

```
<?php
/*
 * @author Javier Hernando Diaz
 * @date 06/06/23
 * @version 2.0
 */

class MensajeFlash {
    static function guardarMensaje($mensaje) {
        self::borrarMensajes(); // Eliminar mensajes flash existentes
        $_SESSION['mensajesFlash'][] = $mensaje;
    }

    static function imprimirMensajes() {
        if (isset($_SESSION['mensajesFlash'])) {
            foreach ($_SESSION['mensajesFlash'] as $mensaje) {
                print '<h4 class="flash-message">' . $mensaje . '</h4>';
            }
            self::borrarMensajes(); // Eliminar mensajes flash después de imprimirlos
        }
    }

    static function borrarMensajes() {
        unset($_SESSION['mensajesFlash']);
    }
}
```

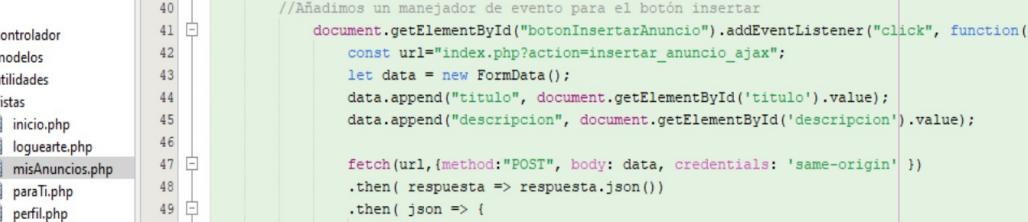
Por ultimo pero no menos importante tenemos la carpeta Vistas, que contiene todas y cada una de las vistas de la pagina web, hay que tener en cuenta que la pagina principal siempre sera inicio, que es una vista más, ya el HTML principal es plantilla.php, que es la que llamara a las distintas vistas por medio de PHP, en plantilla se encuentran todos los import y links con script que podremos utilizar automáticamente en todas las demás vistas.

The screenshot shows a code editor interface with the following details:

- Projects:** Limonitter
- Source Files:** Limonitter, app, vistas, web, index.php, config.php, Include Path, Remote Files.
- File:** plantilla.php
- Editor Tabs:** Source, History
- Code Content:** The code is an HTML template for a website. It includes meta tags for charset and viewport, a title "Inicio", and a script tag for a fontawesome kit. It links to various CSS files from CDNJS and Bootstrap. It also includes a script for jQuery and SweetAlert2. The main content area starts with a Bootstrap navbar.

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Inicio</title>
    <script src="https://kit.fontawesome.com/2aa0fc03.js" crossorigin="anonymous"></script>
    <link rel="icon" href="web/img/logo.jpg" type="image/jpg">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css" rel="stylesheet">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-kenU1KFdBI58tGxNvPnJ3jMa58tkgEZA2lZL6+tXoVZtOz9eZqZoqfD6W5t5n" data-bbox="10 108 976 488" data-label="Code-Block">
    <link rel="stylesheet" href="web/css/estilos.css"/>
    <script src="https://code.jquery.com/jquery-3.6.1.slim.js" integrity="sha256-tXm+salusbFnBt8GJsgI2Tw+m4BLGOf6eUPjb" data-bbox="10 108 976 488" data-label="Code-Block">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/sweetalert2@11.0.19/dist/sweetalert2.min.css">
    <script src="https://cdn.jsdelivr.net/npm/sweetalert2@11.0.19/dist/sweetalert2.all.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-kenU1KFdBI58tGxNvPnJ3jMa58tkgEZA2lZL6+tXoVZtOz9eZqZoqfD6W5t5n" data-bbox="10 108 976 488" data-label="Code-Block">
<nav class="navbar navbar-dark bg-dark fixed-top">
    <div class="container-fluid">
        <div class="d-flex align-items-center">
            <a class="navbar-brand" href="index.php?action=inicio">LIMONITTER
                
            </a>
        </div>
        <button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-bs-target="#offcanvasDarkNavbar" aria-label="Toggle navigation">
            <span class="visually-hidden">Toggle navigation
            <span class="navbar-toggler-icon"></span>
        </button>
    </div>
</nav>
<div class="container">
    <div class="row justify-content-center">
        <div class="col-12 col-md-8 col-lg-6">
            <div class="card border-0 shadow-sm" style="background-color: #fff; border-radius: 15px; padding: 15px; margin-bottom: 10px">
                <div class="card-body" style="text-align: center; background-color: #f0f0f0; border-radius: 15px; padding: 10px; border: none">
                    <h2 class="mb-2" style="font-size: 1.2em; font-weight: bold; color: #007bff; margin: 0">¡Bienvenido!</h2>
                    <p style="margin: 0">Por favor, elige una opción de navegación:</p>
                    <div style="margin-top: 10px; margin-bottom: 10px">
                        <div style="display: flex; justify-content: space-around; width: 100%">
                            <a href="index.php?action=inicio" style="color: #007bff; text-decoration: none; font-weight: bold; font-size: 0.9em; padding: 5px 15px; border-radius: 10px; border: 1px solid #007bff; transition: all 0.3s ease-in-out">Inicio</a>
                            <a href="index.php?action=registro" style="color: #007bff; text-decoration: none; font-weight: bold; font-size: 0.9em; padding: 5px 15px; border-radius: 10px; border: 1px solid #007bff; transition: all 0.3s ease-in-out">Registro</a>
                            <a href="index.php?action=login" style="color: #007bff; text-decoration: none; font-weight: bold; font-size: 0.9em; padding: 5px 15px; border-radius: 10px; border: 1px solid #007bff; transition: all 0.3s ease-in-out">Login</a>
                        </div>
                    </div>
                    <div style="text-align: right; margin-top: 10px">
                        <small style="font-size: 0.8em; color: #007bff; margin: 0">Limonitter - Desarrollado por Limonitter</small>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

Como ya habíamos hablado anteriormente , en muchas de las vistas encontraremos código javascript que es utilizado para las funciones con AJAX y que la pagina se sienta más conforme a ayudar al usuario a que todo se haga sin molestas recargas de pagina.



```
<script>
    //Añadimos un manejador de evento para el botón insertar
    document.getElementById("botonInsertarAnuncio").addEventListener("click", function(){
        const url = "index.php?action=insertar_anuncio_ajax";
        let data = new FormData();
        data.append("titulo", document.getElementById('titulo').value);
        data.append("descripcion", document.getElementById('descripcion').value);

        fetch(url, {method: "POST", body: data, credentials: 'same-origin'})
            .then( respuesta => respuesta.json())
            .then( json => {
                //Código a ejecutar cuando reciba la respuesta
                console.log(json);
                //Crear capas
                let capa_anuncio = document.createElement("div");
                let capa_titulo = document.createElement("div");
                let capa_descripcion = document.createElement("div");
                //Asigno clases a las capas
                capa_anuncio.className = 'anuncio container';
                capa_titulo.className = 'titulo';
                capa_descripcion.className = 'descripcion';
                //Rellen el contenido o html de cada capa
                capa_titulo.innerHTML = json.anuncio.titulo;
                capa_descripcion.innerHTML = json.anuncio.descripcion;
            })
    })
}
```

Se que esto es un resumen muy resumido de la web y sus métodos junto archivos y funciones, pero creo que es aburrido leer estéticamente un documento, así que si lo deseas tienes un video explicativo más dinámico y divertido de ver y escuchar.

b. Pruebas unitarias y de integración

Vamos a realizar un análisis de 2 funciones importantes con pruebas unitarias:

A. function registrar()

1. **Prueba de registro exitoso:** Envías una solicitud POST con todos los campos válidos y únicos. Verifica que no se muestre ningún mensaje de error y que el usuario se redirija correctamente a la página de inicio (`index.php`).
2. **Prueba de campos vacíos:** Envías una solicitud POST con uno o más campos vacíos. Verifica que se muestre el mensaje "Todos los campos son obligatorios".
3. **Prueba de longitud de contraseña:** Envías una solicitud POST con una contraseña que excede los 100 caracteres. Verifica que se muestre el mensaje "La contraseña no puede tener más de 100 caracteres".
4. **Prueba de longitud de email:** Envías una solicitud POST con un email que excede los 60 caracteres. Verifica que se muestre el mensaje "El email no puede tener más de 60 caracteres".
5. **Prueba de longitud de nombre:** Envías una solicitud POST con un nombre que excede los 20 caracteres. Verifica que se muestre el mensaje "El nombre no puede tener más de 20 caracteres".
6. **Prueba de email duplicado:** Envías una solicitud POST con un email que ya existe en la base de datos. Verifica que se muestre el mensaje "Email repetido".
7. **Prueba de contraseñas diferentes:** Envías una solicitud POST con dos contraseñas que no coinciden. Verifica que se muestre el mensaje "Las contraseñas no son iguales".

B. function cambiarContra()

1. **Prueba de registro exitoso:** Verifica que, al enviar una solicitud POST con todos los campos válidos y únicos, no se muestre ningún mensaje de error y el usuario se redirija correctamente a la página de inicio (`index.php`).
2. **Prueba de campos vacíos:** Verifica que, al enviar una solicitud POST con uno o más campos vacíos, se muestre el mensaje "Todos los campos son obligatorios".
3. **Prueba de longitud de contraseña:** Verifica que, al enviar una solicitud POST con una contraseña que excede los 100 caracteres, se muestre el mensaje "La contraseña no puede tener más de 100 caracteres".
4. **Prueba de longitud de email:** Verifica que, al enviar una solicitud POST con un email que excede los 60 caracteres, se muestre el mensaje "El email no puede tener más de 60 caracteres".
5. **Prueba de longitud de nombre:** Verifica que, al enviar una solicitud POST con un nombre que excede los 20 caracteres, se muestre el mensaje "El nombre no puede tener más de 20 caracteres".
6. **Prueba de email duplicado:** Verifica que, al enviar una solicitud POST con un email que ya existe en la base de datos, se muestre el mensaje "Email repetido".

7. **Prueba de contraseñas diferentes:** Verifica que, al enviar una solicitud POST con dos contraseñas que no coinciden, se muestre el mensaje "Las contraseñas no son iguales".

c. Corrección de errores y optimización del rendimiento

En mi proyecto Limonitter, es importante corregir errores y mejorar el rendimiento para brindar a los usuarios una experiencia de uso óptima. Algunas acciones que se pueden realizar incluyen:

1. **Corrección de errores:** Identificar y solucionar cualquier problema o comportamiento inesperado en la aplicación, como errores de funcionalidad o problemas de seguridad.
2. **Mejora de la velocidad:** Optimizar la carga de las páginas para que se carguen rápidamente, lo cual implica mejorar la eficiencia del código, reducir el tamaño de los archivos y minimizar el número de solicitudes al servidor.
3. **Optimización de la base de datos:** Mejorar el rendimiento de las consultas de la base de datos, como creando índices adecuados y optimizando las consultas complejas.
4. **Optimización del código:** Revisar y mejorar el código existente para eliminar cualquier redundancia o código ineficiente, siguiendo buenas prácticas de desarrollo.
5. **Escalabilidad y manejo de la carga:** Evaluar la capacidad de la aplicación para manejar un aumento en el número de usuarios y realizar ajustes necesarios para garantizar un funcionamiento efectivo.
6. **Mejora de la seguridad:** Fortalecer la seguridad de la aplicación implementando medidas adicionales, como autenticación y autorización robustas, protección contra ataques y cifrado de datos.



V. Documentación

a. Documentación técnica

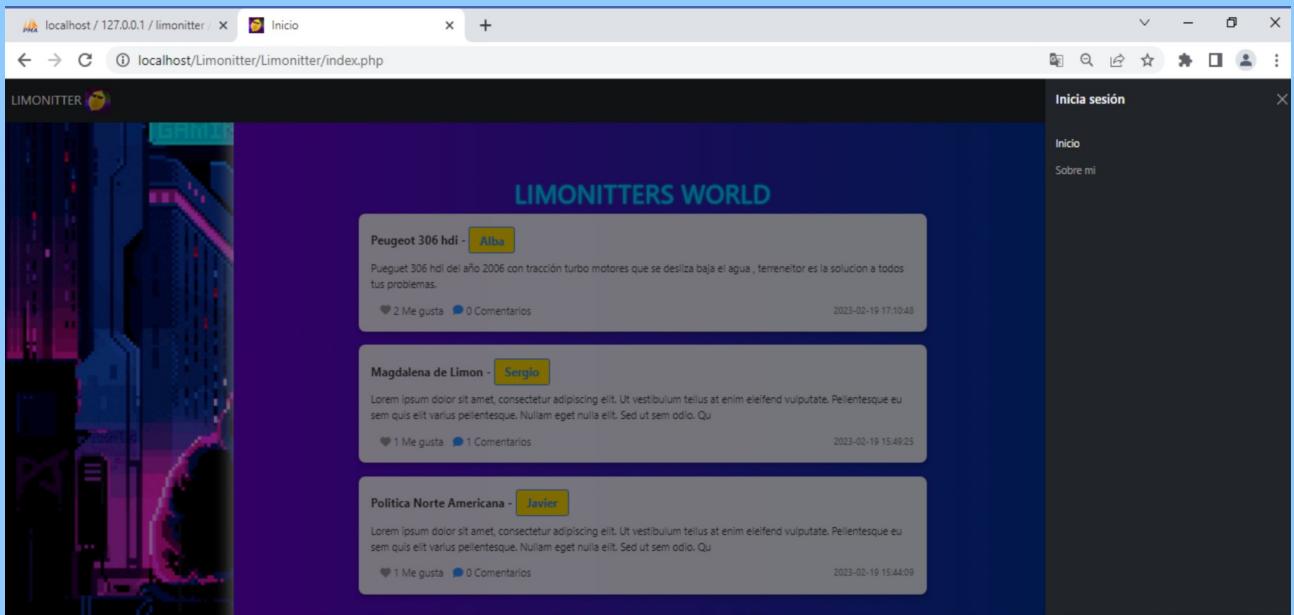
En el proyecto Limonitter, he utilizado varias tecnologías para desarrollar y poner en funcionamiento la aplicación. Estas tecnologías incluyen:

1. **Servidor XAMPP:** He utilizado XAMPP como servidor local para ejecutar y probar la aplicación en mi entorno de desarrollo. XAMPP proporciona un conjunto completo de herramientas que incluye Apache, MySQL, PHP y phpMyAdmin, lo que facilita la configuración y gestión del entorno de desarrollo.
2. **phpMyAdmin:** es una herramienta de administración de bases de datos que me ha permitido crear, administrar y manipular la base de datos utilizada en Limonitter. Con phpMyAdmin, puedo realizar tareas como crear tablas, definir relaciones, ejecutar consultas SQL y gestionar los datos almacenados en la base de datos de manera fácil y conveniente.
3. **HTML:** (HyperText Markup Language) es el lenguaje de marcado utilizado para estructurar y presentar el contenido en la aplicación web. Con HTML, he creado la estructura básica de las páginas y he definido los elementos y su disposición en la interfaz de usuario.
4. **CSS:** (Cascading Style Sheets) se utiliza para definir el aspecto y el diseño visual de la aplicación. Con CSS, he aplicado estilos personalizados a los elementos HTML, como colores, fuentes, márgenes y tamaños, para lograr una apariencia coherente y atractiva en toda la aplicación.
5. **JavaScript:** es un lenguaje de programación que se utiliza para agregar interactividad y funcionalidad dinámica a la aplicación. He utilizado JavaScript para implementar características como validación de formularios, animaciones, manipulación del DOM y llamadas asíncronas al servidor utilizando AJAX.
6. **PHP:** (Hypertext Preprocessor) es un lenguaje de programación del lado del servidor que se utiliza para desarrollar la lógica de negocio y el procesamiento de datos en el proyecto Limonitter. Con PHP, puedo interactuar con la base de datos, realizar cálculos, gestionar sesiones de usuarios, generar contenido dinámico y mucho más.
7. **AJAX:** (Asynchronous JavaScript and XML) es una técnica de desarrollo web que permite realizar solicitudes asíncronas al servidor sin necesidad de recargar toda la página. En Limonitter, he utilizado AJAX para mejorar la experiencia del usuario al cargar y enviar datos de forma dinámica, sin interrupciones en la navegación.

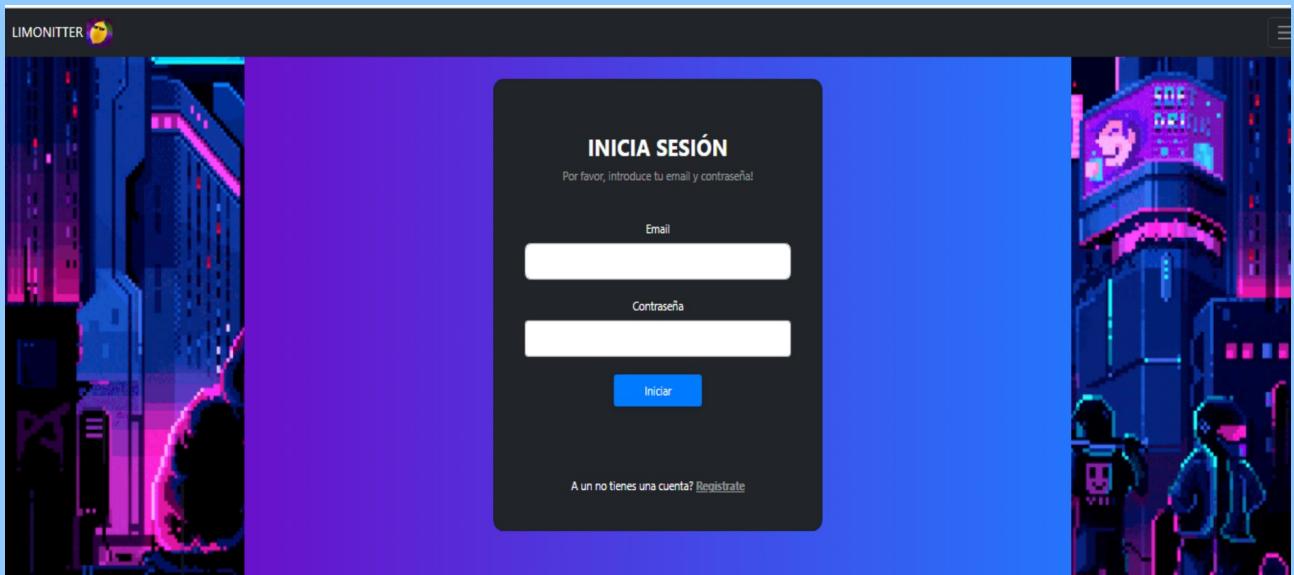
b. Documentación de usuario

Aquí explicaremos un poco el manual de uso de Limonnitter para Usuarios primerizos, quiero recalcar que existe un tutorial de uso y explicativo del código en el video que dejo junto al proyecto, pero en este caso haremos una breve documentación de ello con capturas.

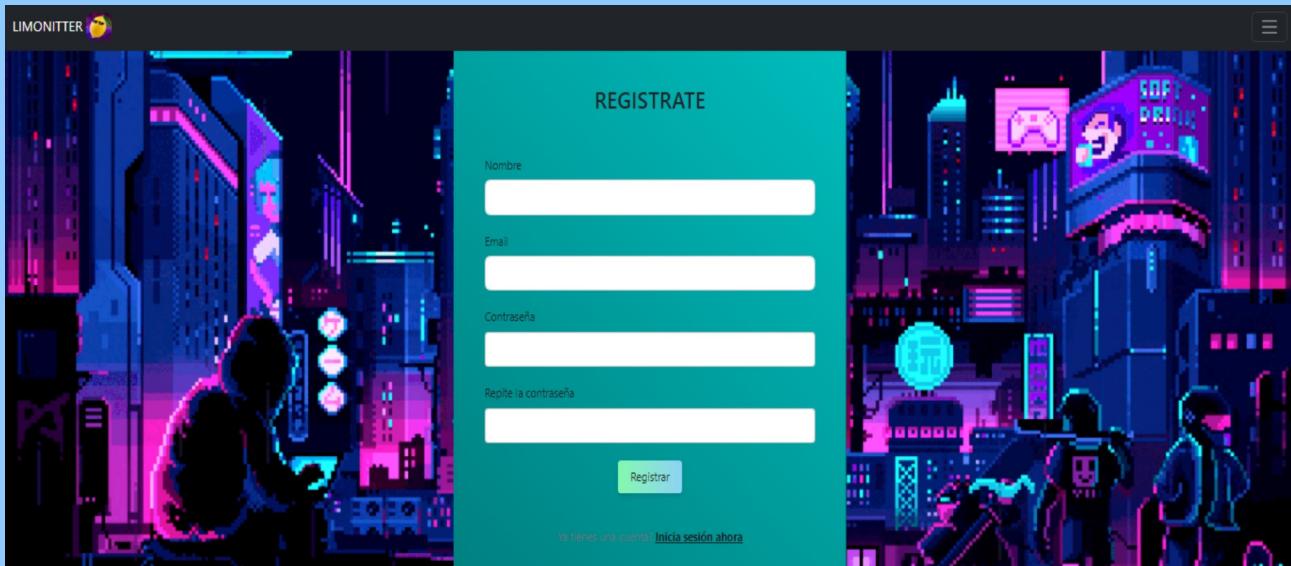
En el Inicio nos encontramos todas las publicaciones existentes en Limonnitter, junto a una barra de navegación que también despliega varios factores de uso sin estar iniciado como usuario del mismo.



Si pulsamos login, podremos acceder al Login de Limonnitter con nuestro email y contraseña:

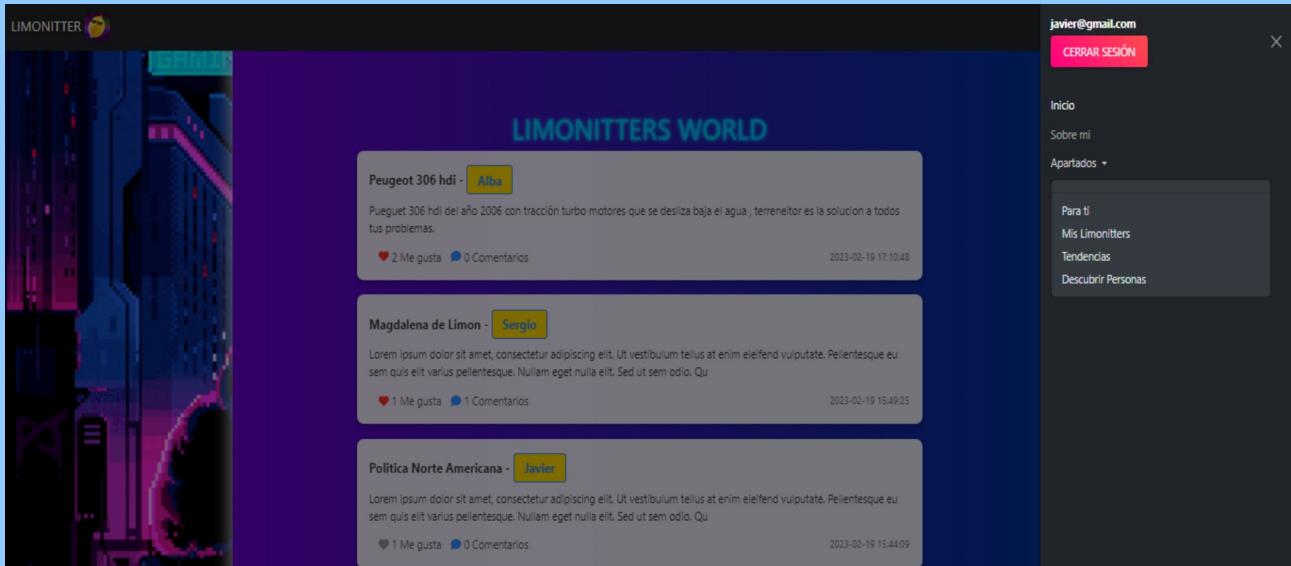


Si no tenemos cuenta propia y queremos ver y interactuar con las publicaciones podemos hacer un registro fácil y sencillo sin datos privados innecesarios.



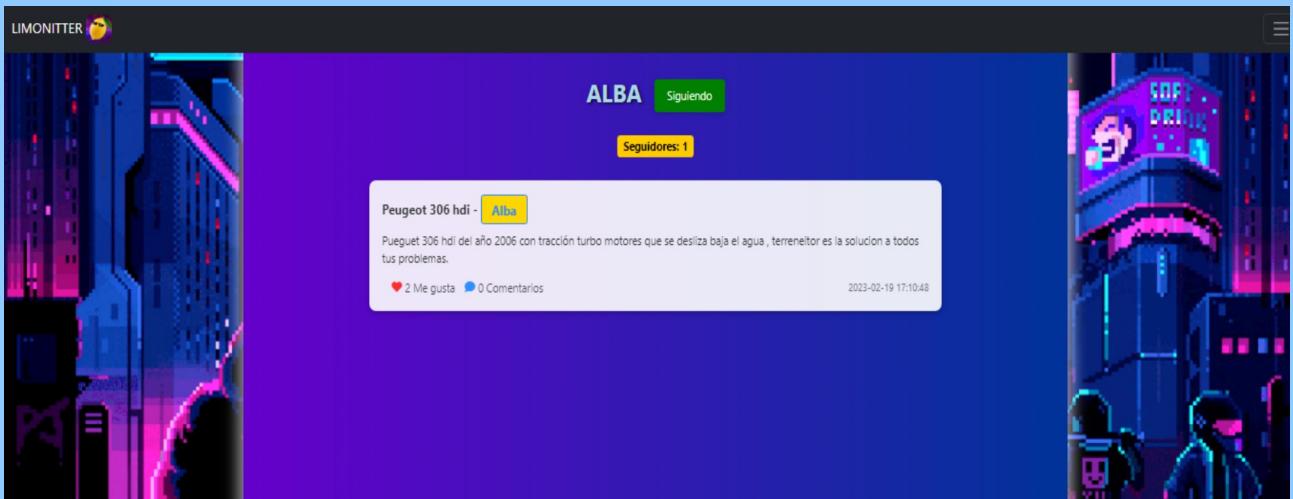
Todos los campos deben ser correctos, si ya te registraste con un Email ya no podrás hacerlo de nuevo con el mismo.

Si nos logueamos en la barra de despliegue tendremos más opciones de uso:

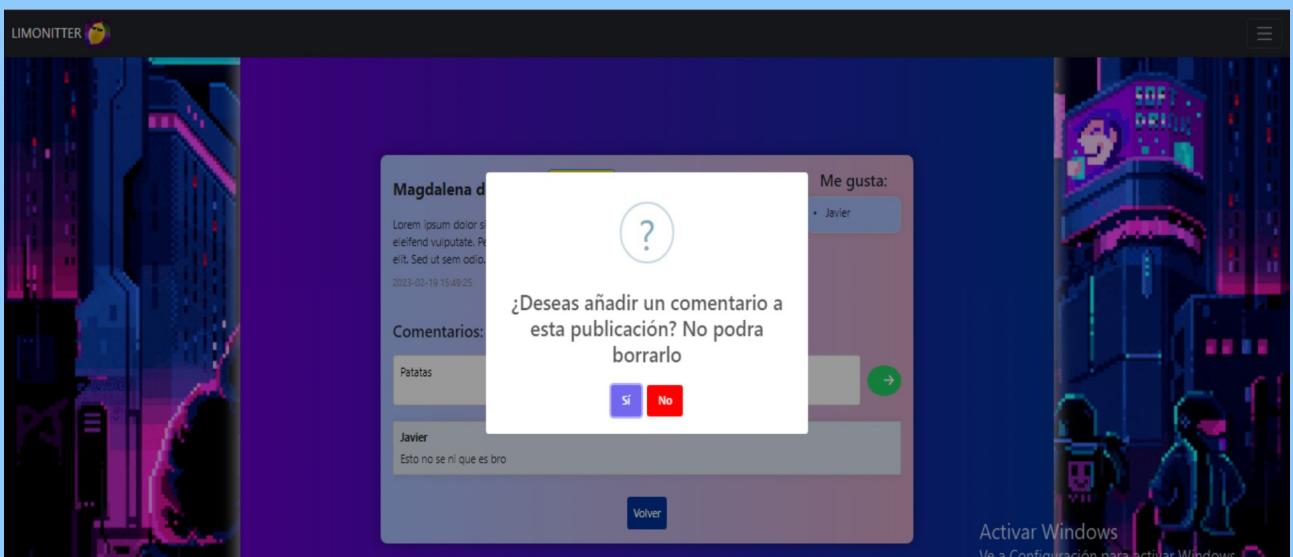


Tenemos las opciones Inicio: Todas las publicaciones, Sobre Mi: Mostrara un apartado con mi información y CV, Para ti : Aparecerán todas las publicaciones de los usuarios a los que sigues, Mis Limonitters : Mostrara tu perfil junto a tus publicaciones y numero de seguidores que tienes, si pulsas el nombre de cualquier usuario en su publicación llegarás a su perfil de la misma manera,

Tendencias: En el encontrarás las publicaciones con más me gusta del momento, Descubrir personas : Hará aparecer una lista de usuarios que podrás seguir o dejar de seguir a tu antojo, en cada publicación podrás acceder a sus comentarios y Me Gustas ademas de poder comentar lo que deseas con el inconveniente de no poder borrar ese comentario.



Habrá un aviso antes de introducir un comentario en una publicación:



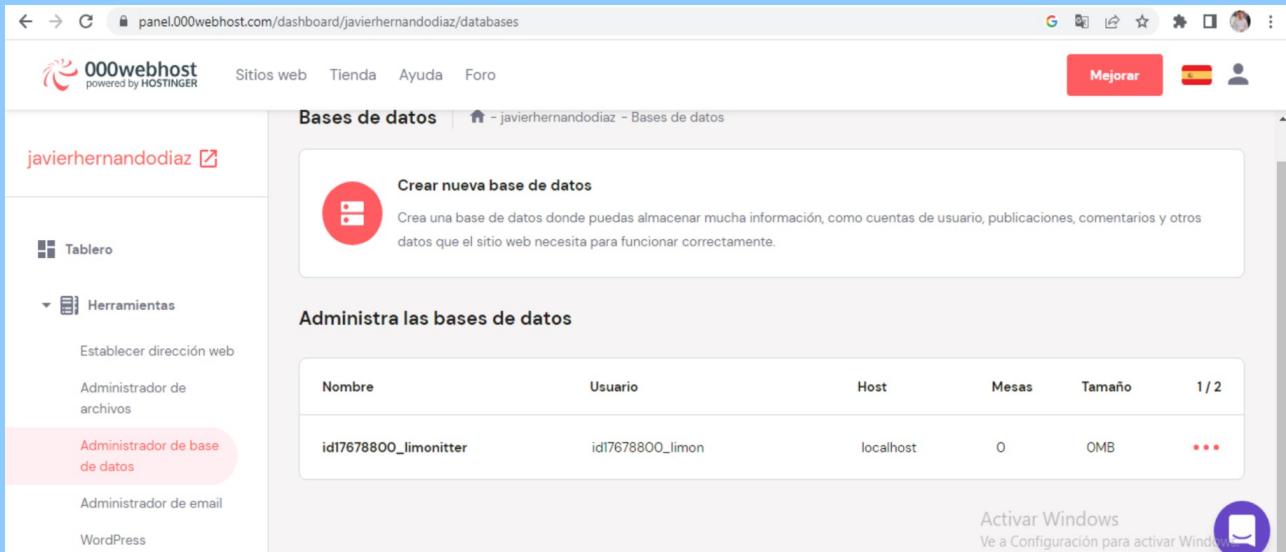
Vuelvo a repetir que en el video se muestra mejor todas las funciones de la propia web, no es documentar menos, ya que me parece más dinámico un video explicativo que unas simples capturas.

A screenshot of a LinkedIn profile for Javier Hernando Diaz. The profile includes sections for "EXPERIENCIA" (Experience), "CAMPAÑA VENDIMIA" (Felix Solis | Puebla de Almoradiel), "PRACTICAS GRADO MEDIO SMR" (Hospital General Mancha Centro | Alcázar de San Juan), and "OBJETIVO PROFESIONAL" (A brief statement about his professional goals and preparation for the programming world).

c. Manual de instalación y configuración

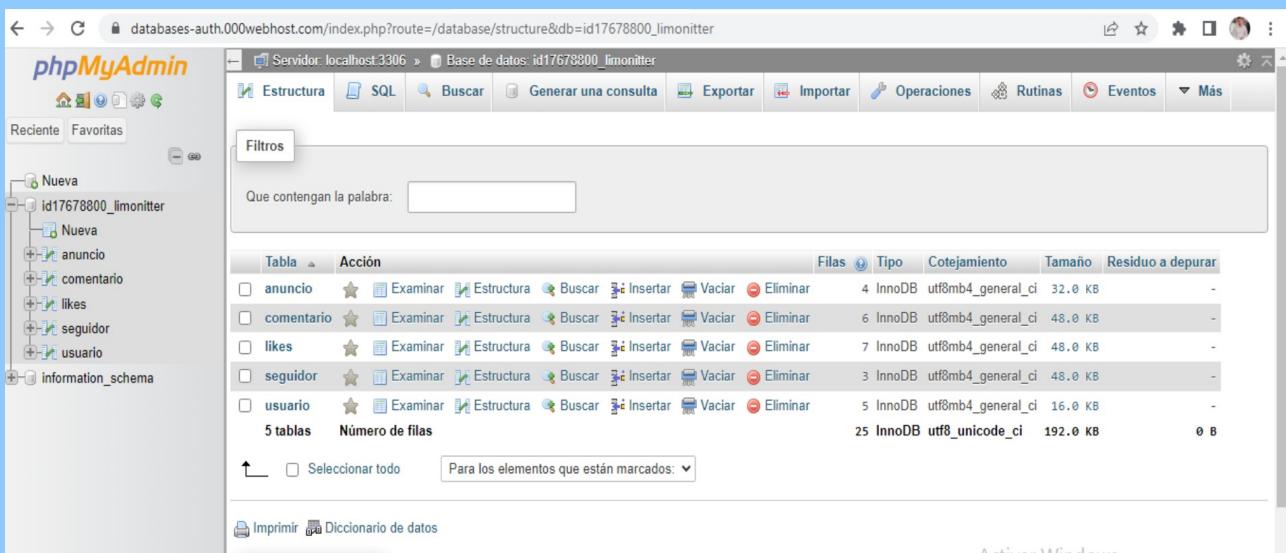
Para el hosting he utilizado 000WebHost , mostrare los pasos que he seguido para la producción y implementación de la pagina web en el hosting.

Primero deberemos introducir la Base de Datos en PHPMyAdmin de 000WebHost.



The screenshot shows the 000WebHost dashboard with the user 'javierhernandodiaz'. In the sidebar, under 'Herramientas', the 'Administrador de base de datos' option is highlighted with a red box. The main area shows a table of databases with one entry:

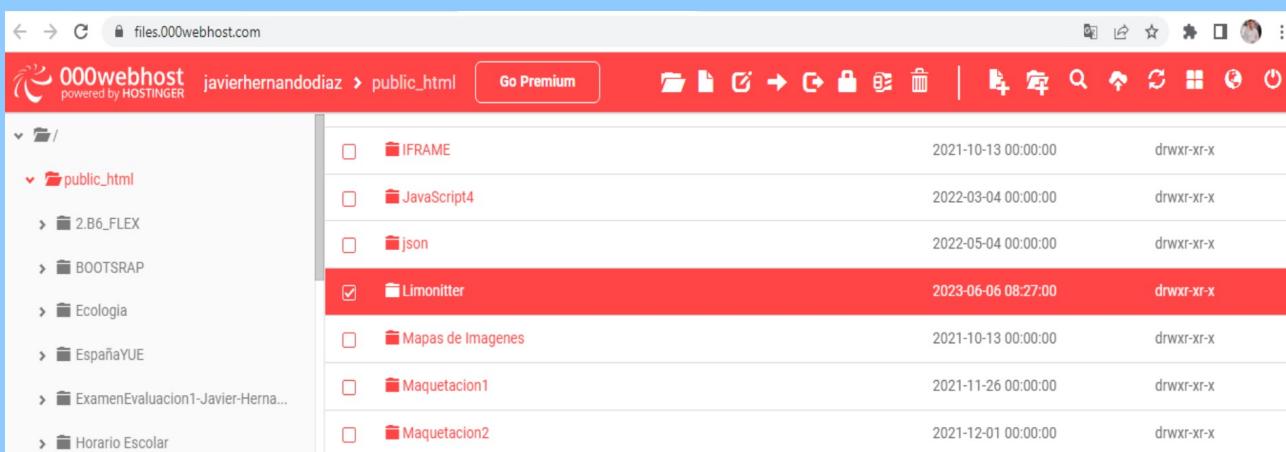
Nombre	Usuario	Host	Mesas	Tamaño	Opciones
id17678800_limonitter	id17678800_limon	localhost	0	0MB	...



The screenshot shows the phpMyAdmin interface for the database 'id17678800_limonitter'. The left sidebar shows the structure of the database with tables: anuncio, comentario, likes, seguidor, usuario, and Nueva. The main panel shows a list of tables with their details:

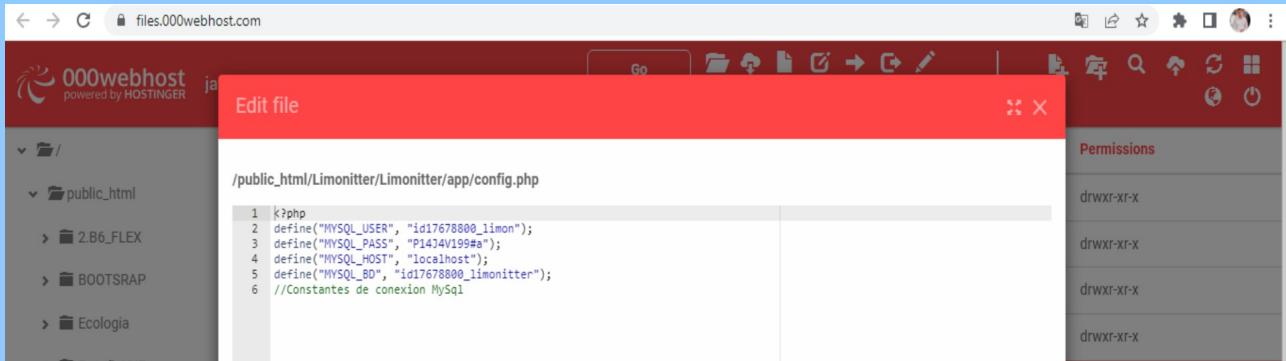
Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
anuncio	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	32.0 KB	-
comentario	Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	utf8mb4_general_ci	48.0 KB	-
likes	Examinar Estructura Buscar Insertar Vaciar Eliminar	7	InnoDB	utf8mb4_general_ci	48.0 KB	-
seguidor	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	48.0 KB	-
usuario	Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8mb4_general_ci	16.0 KB	-
5 tablas	Número de filas	25	InnoDB	utf8_unicode_ci	192.0 KB	0 B

Dejaremos el proyecto en el public_html de 000WebHost:



The screenshot shows the 000WebHost file manager. The left sidebar shows the directory structure with 'public_html' expanded. The right pane lists files in the 'public_html' directory, with 'Limonitter' selected and highlighted with a red box. Other files listed include IFRAME, JavaScript4, json, Mapas de Imagenes, Maquetacion1, and Maquetacion2.

Deberemos hacer los cambios oportunos en el archivo config.php de nuestro proyecto para que se conecte a la BBDD de 000WebHost:

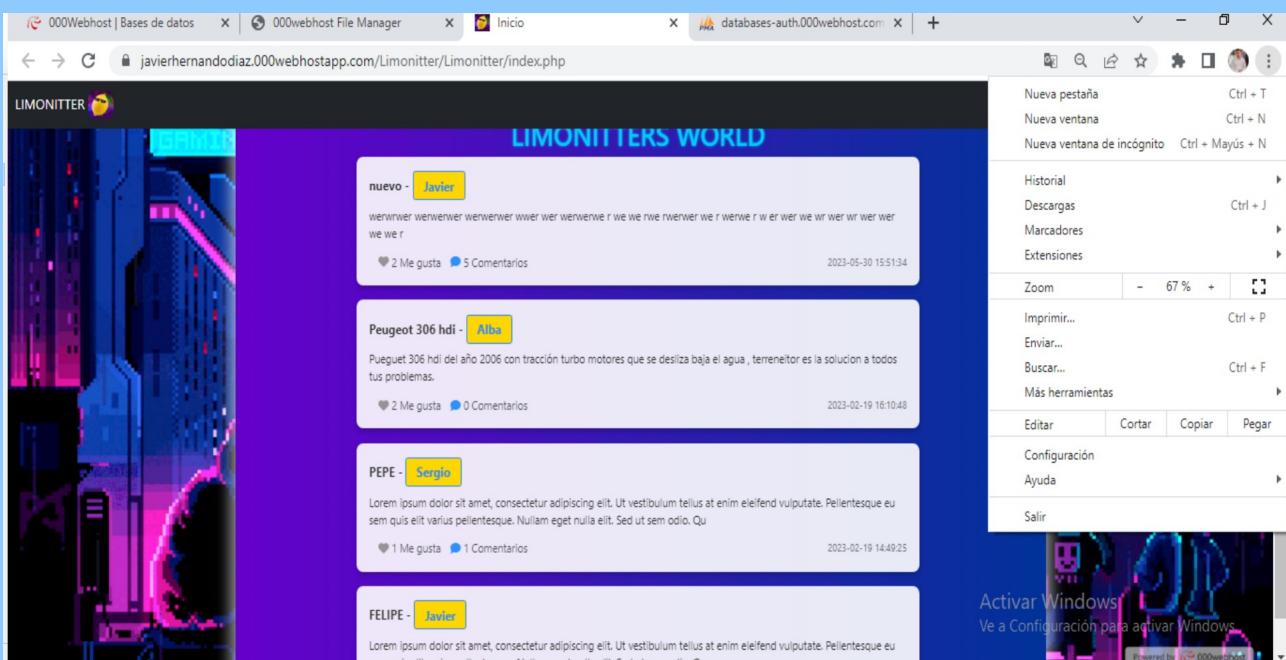


The screenshot shows the 000WebHost File Manager interface. On the left, there's a sidebar with a tree view showing the project structure: /, public_html, 2.B6_FLEX, BOOTSRAP, and Ecologia. The main area is titled "Edit file" and shows the contents of the file "/public_html/Limonitter/Limonitter/app/config.php". The code in the file is as follows:

```
1 <?php
2 define("MYSQL_USER", "id17678800_limon");
3 define("MYSQL_PASS", "P1434V199#");
4 define("MYSQL_HOST", "localhost");
5 define("MYSQL_80", "id17678800_limonitter");
6 //Constantes de conexión MySql
```

To the right of the editor, there's a "Permissions" panel showing the current file permissions.

Con el enlace que voy a dejar podrás acceder a mi proyecto web desde google con el Hosting, ten en cuenta que puede que se vea mejor si haces más pequeño el porcentaje de zoom dependiendo de tu pantalla:



Hosting Link : <https://javierhernandodiaz.000webhostapp.com/Limonitter/Limonitter/index.php>

o <https://javierhernandodiaz.000webhostapp.com/Limonitter/Limonitter/>

VI. Mantenimiento y evolución

a. Plan de mantenimiento y soporte

1. **Actualizaciones de software:** Se realizarán actualizaciones periódicas de las tecnologías utilizadas en la aplicación, como el servidor web, el lenguaje de programación y los frameworks o librerías externas. Estas actualizaciones garantizarán la compatibilidad, seguridad y eficiencia de la aplicación.
2. **Respaldo regular de la base de datos:** Se programarán tareas automáticas para realizar copias de seguridad de la base de datos de Limonitter de forma regular. Estos respaldos se almacenarán en un lugar seguro para asegurar la integridad de los datos y permitir una rápida recuperación en caso de pérdida de información.
3. **Monitoreo y rendimiento:** Se implementarán herramientas de monitoreo para supervisar el rendimiento de la aplicación, incluyendo el tiempo de respuesta, la carga del servidor y el uso de recursos. Estos datos nos permitirán identificar posibles cuellos de botella y optimizar el rendimiento de la aplicación.
4. **Corrección de errores:** Se establecerá un proceso para reportar y solucionar errores en la aplicación. Los usuarios podrán informar problemas a través de un sistema de tickets o formulario de contacto, y se asignará un equipo encargado de revisar y solucionar estos errores de manera oportuna.
5. **Soporte técnico:** Se proporcionará un canal de soporte técnico para que los usuarios puedan realizar consultas, obtener ayuda o reportar problemas relacionados con el uso de la aplicación. Este soporte estará disponible a través de un sistema de tickets, correo electrónico o chat en línea, y se establecerán tiempos de respuesta adecuados para cada tipo de consulta.
6. **Mejoras y actualizaciones de funcionalidades:** Se evaluarán regularmente las necesidades y feedback de los usuarios para identificar oportunidades de mejora y nuevas funcionalidades. Estas mejoras se planificarán y se implementarán en nuevas versiones de la aplicación, manteniendo un ciclo de desarrollo continuo.
7. **Capacitación y documentación:** Se brindará capacitación y documentación actualizada para los administradores y usuarios de la aplicación. Esto incluirá manuales de usuario, guías de administración y recursos de aprendizaje en línea para garantizar un uso eficiente y efectivo de Limonitter.

b. Identificación de posibles mejoras y evolución del proyecto

En cuanto a la identificación de posibles mejoras y la evolución del proyecto Limonitter, se seguirán las siguientes estrategias:

1. **Investigación de mercado:** Se llevará a cabo una investigación exhaustiva del mercado de las redes sociales y las tendencias en el comportamiento de los usuarios. Esto nos permitirá identificar nuevas oportunidades, necesidades no satisfechas y cambios en los patrones de uso que puedan influir en el desarrollo de Limonitter.
2. **Análisis de la competencia:** Se realizará un análisis detallado de la competencia, estudiando las características y funcionalidades de otras aplicaciones similares. Esto nos permitirá identificar brechas o áreas de mejora en comparación con los competidores y generar ideas para nuevas funcionalidades o mejoras en Limonitter.
3. **Evaluación continua del feedback de los usuarios:** Se establecerá un sistema para recopilar y analizar el feedback de los usuarios de Limonitter. Esto incluirá la revisión de comentarios, sugerencias y solicitudes de funcionalidades por parte de los usuarios. Este feedback será considerado para priorizar las mejoras y ajustes necesarios en futuras versiones de la aplicación.
4. **Exploración de nuevas tecnologías y tendencias:** Se mantendrá un monitoreo constante de las nuevas tecnologías y tendencias en el desarrollo de aplicaciones web. Esto nos permitirá evaluar la adopción de tecnologías emergentes como inteligencia artificial, realidad aumentada, blockchain, entre otras, que puedan potenciar la funcionalidad y experiencia de usuario de Limonitter.
5. **Planificación de actualizaciones graduales:** Se adoptará una estrategia de actualizaciones graduales, en lugar de cambios drásticos y disruptivos. Esto nos permitirá implementar mejoras y nuevas funcionalidades de manera incremental, minimizando los riesgos y garantizando una transición suave para los usuarios.
6. **Colaboración con la comunidad de usuarios:** Se fomentará la participación de la comunidad de usuarios de Limonitter en el proceso de evolución del proyecto. Se podrán realizar encuestas, foros de discusión o grupos de pruebas beta para involucrar a los usuarios en la toma de decisiones y obtener ideas valiosas para el desarrollo futuro de la aplicación.

c. Actualizaciones y mejoras futuras

1. **Mejora de la interfaz de usuario:** Se realizarán actualizaciones en la interfaz de usuario para mejorar la experiencia del usuario y hacerla más intuitiva y atractiva. Esto incluirá mejoras en la navegación, diseño visual, usabilidad y accesibilidad, con el objetivo de ofrecer una experiencia de usuario fluida y agradable.
2. **Ampliación de funcionalidades:** Se agregarán nuevas funcionalidades y características a Limonitter para ampliar su utilidad y atraer a más usuarios. Esto puede incluir la implementación de funciones avanzadas de búsqueda y filtrado, integración con otras redes sociales, soporte para multimedia (imágenes y videos), opciones de personalización del perfil, entre otras mejoras.
3. **Optimización del rendimiento:** Se realizarán mejoras en el rendimiento de Limonitter para garantizar una experiencia rápida y eficiente. Esto incluirá optimizaciones en la velocidad de carga de la aplicación, el manejo de la base de datos, el uso de memoria y recursos, y la minimización de tiempos de respuesta.
4. **Seguridad y privacidad:** Se fortalecerán las medidas de seguridad y privacidad en Limonitter para proteger la información de los usuarios y prevenir posibles brechas de seguridad. Esto incluirá la implementación de protocolos de encriptación, autenticación de usuarios, gestión de permisos y políticas de privacidad claras.
5. **Adaptación a dispositivos móviles:** Se realizará una adaptación completa de Limonitter para dispositivos móviles, como smartphones y tablets. Esto permitirá a los usuarios acceder y utilizar la aplicación de manera óptima desde cualquier dispositivo móvil, ampliando así su alcance y accesibilidad.
6. **Integración de herramientas de análisis:** Se integrarán herramientas de análisis y seguimiento para recopilar datos sobre el uso de Limonitter y el comportamiento de los usuarios. Esto ayudará a comprender mejor las necesidades y preferencias de los usuarios, así como a tomar decisiones informadas para futuras actualizaciones y mejoras.
7. **Colaboración con la comunidad:** Se fomentará la participación activa de la comunidad de usuarios y desarrolladores de Limonitter. Se podrán realizar eventos, hackathons o foros de discusión para involucrar a la comunidad en la generación de ideas y contribuciones al proyecto.

VII. Conclusiones

a. Evaluación del proyecto

La evaluación de mi proyecto Limonitter ha sido un proceso fundamental para comprender su funcionamiento y determinar su calidad y eficacia. Durante esta evaluación, me he enfocado en varios aspectos clave.

En primer lugar, me he asegurado de que Limonitter cumpla con todos los requisitos funcionales y no funcionales que se habían establecido inicialmente. Esto implica verificar que todas las funcionalidades planificadas estén implementadas de manera efectiva y que el sistema cumpla con las necesidades y expectativas de los usuarios.

Además, he prestado especial atención a la usabilidad y experiencia del usuario. Me he asegurado de que la interfaz sea intuitiva, las acciones sean claras y fáciles de realizar, y la navegación sea fluida. También he considerado la retroalimentación de los usuarios para identificar áreas de mejora y optimizar la experiencia general.

Otro aspecto fundamental ha sido el rendimiento y la escalabilidad del sistema. He realizado pruebas exhaustivas para evaluar la velocidad de carga, la respuesta a las solicitudes y la capacidad de manejar un alto volumen de usuarios y datos. Esto me ha permitido identificar posibles cuellos de botella y garantizar que Limonitter pueda escalar de manera eficiente.

b. Cumplimiento de objetivos y requisitos

Durante el desarrollo de mi proyecto Limonitter, me he enfocado en cumplir los objetivos y requisitos establecidos. He trabajado arduamente para implementar todas las funcionalidades planificadas y asegurarme de que el sistema se ajuste a las necesidades y expectativas de los usuarios. Además, he realizado pruebas exhaustivas para verificar que todas las características funcionen correctamente y cumplan con los estándares de calidad. En general, puedo decir con confianza que Limonitter cumple con los objetivos y requisitos establecidos, brindando una plataforma efectiva y satisfactoria para los usuarios.

c. Lecciones aprendidas y recomendaciones para futuros proyectos

Pienso que he aprendido bastante con el funcionamiento de modelos y funciones con AJAX Y Javascript, ademas recomiendo en un futuro un mayor estilo y personalización con la web y las funciones ya son usuarios, publicaciones, comentarios...etc. Si tuviera el tiempo necesario mejoraria notablemente muchas de las funciones utilizadas y Responsive de la pagina, aunque estoy contento sabiendo que en 1 mes he podido recuperar un proyecto perdido que al principio era con Framework de Symfony.

VIII. Bibliografía y referencias

a. Fuentes utilizadas en el proyecto

En el desarrollo de mi proyecto, he utilizado diversas fuentes y recursos que han sido fundamentales para su implementación exitosa. A continuación, se mencionan algunas de las principales fuentes utilizadas:

1. **Google Fonts:** He aprovechado la amplia variedad de fuentes disponibles en Google Fonts para mejorar la apariencia y legibilidad de los textos en el proyecto. Esta plataforma ofrece una amplia gama de opciones de fuentes de alta calidad y de fácil integración.
2. **SweetAlert2 (sweetalert2.github.io):** He utilizado la biblioteca SweetAlert2 para crear notificaciones y alertas personalizadas en el proyecto. Esta herramienta proporciona una forma sencilla y elegante de mostrar mensajes al usuario, mejorando la experiencia de usuario y la interactividad.
3. **PHP:** He utilizado PHP como lenguaje de programación principal en el proyecto. PHP es un lenguaje de scripting de servidor ampliamente utilizado en el desarrollo web. Me ha permitido crear y manipular datos, interactuar con la base de datos y gestionar la lógica del negocio.
4. **AJAX:** He utilizado la tecnología AJAX (Asynchronous JavaScript and XML) para realizar solicitudes asíncronas al servidor sin tener que recargar toda la página. Esto ha permitido mejorar la velocidad y la interactividad del proyecto al actualizar dinámicamente el contenido y realizar acciones sin interrupciones para el usuario.
5. **HTML, CSS y JavaScript:** Estos son los lenguajes fundamentales para el desarrollo web. He utilizado HTML para estructurar y organizar el contenido, CSS para estilizar y dar formato a los elementos visuales, y JavaScript para añadir interactividad y funcionalidad dinámica al proyecto.

Además de las fuentes mencionadas anteriormente, también he utilizado el **framework Bootstrap** en mi proyecto Limonitter. Bootstrap es una herramienta popular que proporciona una amplia gama de componentes y estilos predefinidos para agilizar el desarrollo de interfaces web responsivas y atractivas.

Al aprovechar **Bootstrap**, he podido crear una estructura de diseño consistente y adaptable para mi proyecto. Esto ha facilitado la creación de un diseño receptivo que se adapta a diferentes dispositivos y tamaños de pantalla, brindando una experiencia de usuario optimizada.

b. Referencias y enlaces de interés

Es importante destacar que las fuentes mencionadas han sido utilizadas con el objetivo de mejorar la apariencia, la interactividad y la eficiencia del proyecto Limonitter. Todas estas fuentes se consideran recursos confiables y ampliamente utilizados en la comunidad de desarrollo web.

Fuentes:

- Google Fonts: <https://fonts.google.com/>
- SweetAlert2: <https://sweetalert2.github.io/>
- Bootstrap: <https://getbootstrap.com/>