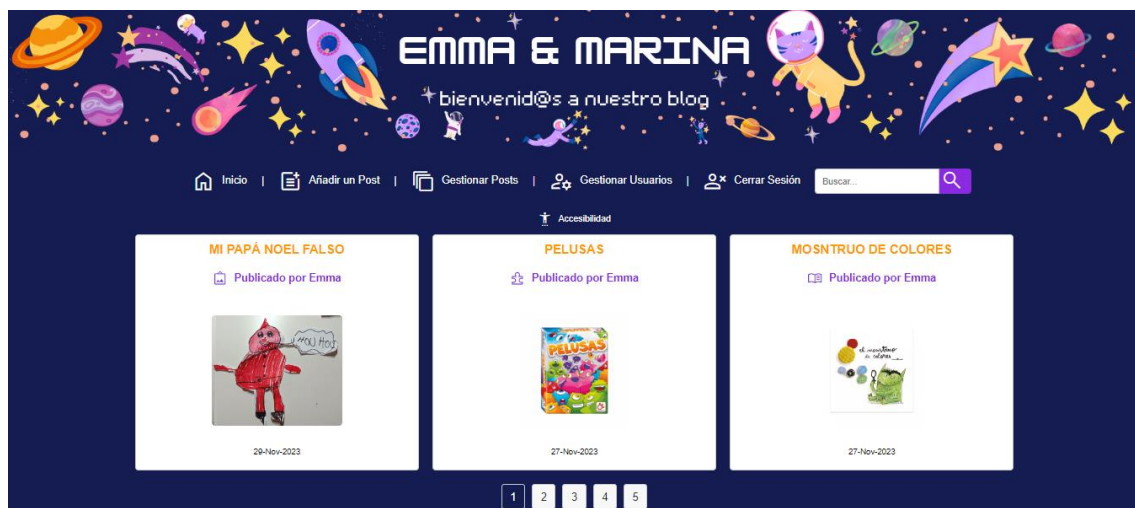


BLOG

INFANTIL



Índice:

1. Introducción y Objetivo.
2. Estudio de mercado.
3. Presupuesto.
4. Manual de usuario.
 - 4.1. Página de Inicio
 - 4.2. Inicio de Sesión
 - 4.3. Usuarios no Administradores
 - 4.4. Usuarios Administradores
 - 4.5. Agregar Post
 - 4.6. Gestión de Posts
 - 4.7. Gestión de Usuarios
 - 4.8 Diseño responsive
 - 4.9. Accesibilidad
5. Lenguajes, tecnologías y recursos utilizados
6. Estructura del Proyecto MVC (Modelo Vista Controlador)
 - 6.1. Controllers
 - 6.2. Images
 - 6.3. Models
 - 6.5. Uploads
 - 6.6. Views
 - 6.7. .htcaccses
 - 6.8. Accessibility.js y Accessibility.php
 - 6.9. Config.php
 - 6.10. Index.php
 - 6.11. Message.php
 - 6.12. Modal.js
 - 6.13. Styles.css
7. Base de Datos.
 - 7.1. Tablas
 - 7.2. Relaciones
8. Manual de instalación.
9. Proyecto en GitHub.
10. Validación de HTML, CSS y Accesibilidad
11. Problemas y Mejoras
12. Conclusiones Finales
13. Webgrafía.

1. Introducción y objetivo.

El proyecto consiste en el desarrollo de un blog infantil con un enfoque especial en la seguridad y el control parental. El objetivo es crear una plataforma que ofrezca un espacio seguro para crear y compartir contenido, mientras que proporciona herramientas efectivas para supervisar y moderar las actividades en línea.

Con el aumento de la participación infantil en plataformas digitales, se vuelve crucial abordar los desafíos relacionados con la seguridad en línea y garantizar que los niños tengan acceso a contenido adecuado para su edad. Este proyecto aborda la importancia crítica de ofrecer una solución confiable que combine aprendizaje, entretenimiento y seguridad.

Esta idea surge cuando mis hijas, incansables guardianas de sus creaciones artísticas, insistieron en que no desecháramos ninguno de sus dibujos. Por lo que tomé la decisión de crear un espacio digital dedicado a ellas en el que pudieran compartir y almacenar sus dibujos, los juegos y libros que les gustan o se inventan, y lo que van aprendiendo y diseñando en las clases de robótica.

2. Estudio de mercado:

2.1. Identificación del público objetivo: El uso de blog está destinado principalmente al público infantil entre los 6 y 12 años, dado su diseño, facilidad de uso y control parental, mientras que el contenido está abierto al público en general.

2.2. Análisis de competencia: Buscando otros blogs, he podido comprobar que la mayoría son escritos por adultos dirigiéndose a un público infantil, o niños en edades ya adolescentes, que escriben para un público en general. La mayoría de estos blogs, tienen una estructuración de post más compleja, que no resulta tan atractiva y de fácil lectura/escritura para el público infantil como el propuesto.

2.3. Evaluación de tendencias: Actualmente, los niños empiezan a controlar dispositivos móviles y a visualizar contenido digital desde muy pequeños, principalmente como consumidores. Por ello, ofertar una herramienta en la que puedan crear contenido, y no sólo ser meros espectadores, me parece interesante, para fomentar su creatividad y hacerlos participantes activos del mundo de la tecnología.

2.4. Conclusiones: el estudio del mercado actual confirma la necesidad de un blog creado por niños con un diseño y facilidad de uso adecuado a su edad, que sea seguro y que se adapte a dispositivos móviles. Para los tutores ofrece tranquilidad, y una herramienta educativa, que fomenta la independencia, la creatividad y una forma diferente de usar las nuevas tecnologías.

3. Presupuesto:

Partiendo de que este blog está inspirado y diseñado fundamentalmente para uso personal y que no se había estudiado la posibilidad de su comercialización hasta la realización de esta documentación, es difícil desarrollar un presupuesto de cara a ofertar el blog en el mercado. Por ello el presupuesto se basa en lo que realmente ha costado su realización.

3.1. Plantilla, diseño y desarrollo: No se ha hecho inversión económica en estos conceptos, ya que no se ha usado ninguna plantilla de terceros y el diseño y el desarrollo son propios en su totalidad.

Para las imágenes, se ha usado Canva, <https://www.canva.com/>, que tiene muchos recursos gratuitos.

Las fuentes e iconos son de Google, <https://fonts.google.com/> y <https://fonts.google.com/icons> que también son de uso libre.

3.2. Servidor y dominio: En este caso, se ha escogido 000webhost.com, <https://www.000webhost.com/> que ofrece servicios de alojamiento web gratuito y también un subdominio gratuito bajo su dominio principal (tudominio.000webhostapp.com). La duración del mismo sería de un año. Si más adelante quisiera mantener el alojamiento y el dominio, tienen planes de pago desde 1.49€/mes.

The screenshot shows the 000webhost website interface. At the top, there's a navigation bar with the logo, 'OFERTA Hosting Barato', 'AI Website Builder', 'Foro', a language selector (ES), 'INICIA SESIÓN', and 'PÁSATE A PREMIUM'. Below this, there are four main hosting plans displayed in a row:

- Hosting Gratis:** 'Hosting web gratuito para crear tu primer sitio web'. Price: €0.00/mes. Button: 'Registrate'. Note: 'Always free'.
- Single:** 'Solución ideal para principiantes'. Original price: 7.99€, Current price: €1.49/mes (SAVE 81%). Button: 'Obtener oferta'. Note: '€2.99/mo when you renew'.
- Premium:** 'Todo lo que necesitas para crear tu sitio web'. Original price: 11.99€, Current price: €2.59/mes (SAVE 78%). Button: 'Obtener oferta'. Note: '€5.99/mo when you renew'. This plan is highlighted with a purple border and a 'Más Vendido' badge.
- Business Web Hosting:** 'Sube de nivel con más potencia y funciones mejoradas'. Original price: 14.99€, Current price: €3.99/mes (SAVE 73%). Button: 'Obtener oferta'. Note: '€7.99/mo when you renew'.

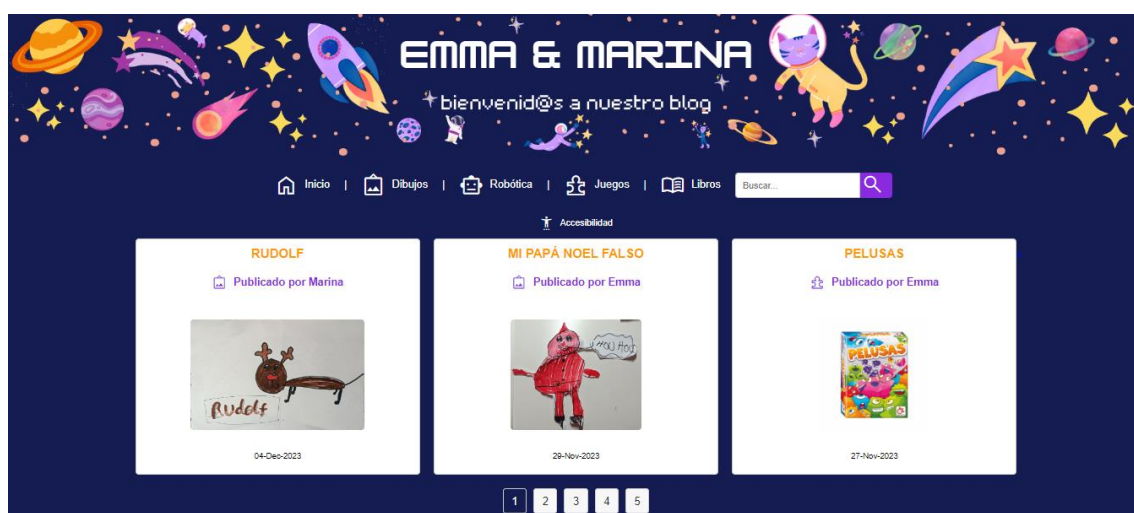
Each plan has a 'Comparación de funciones' link at the bottom. At the very bottom of the page, there is a small footer: 'Utilizamos cookies para personalizar el contenido proporcionado por socios publicitarios y para analíticas, con el fin de ofrecerte la mejor experiencia de servicio. Read our cookie policy.'

4. Manual de usuario:

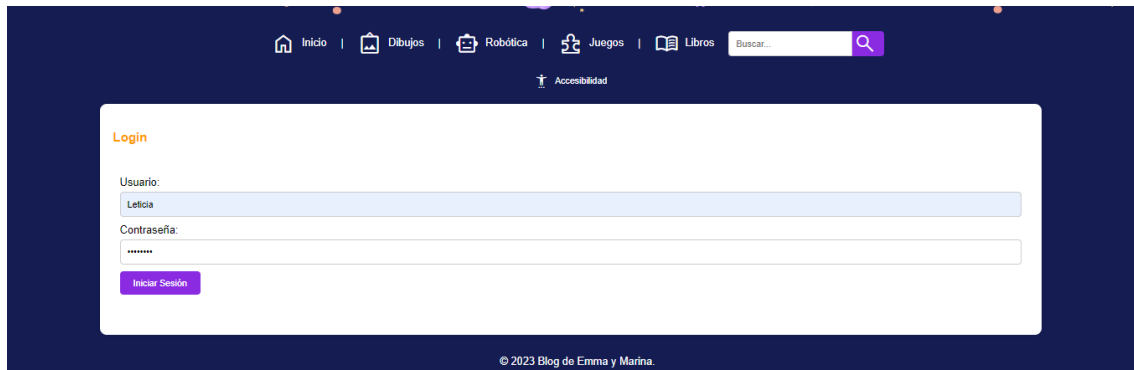
4.1. Página de inicio: desde <https://emmarinablog.000webhostapp.com/> los lectores podrán visualizar un listado de post ordenados por fecha. Se puede navegar a través de la paginación, seleccionar un post para visualizarlo a tamaño completo o buscar un post en concreto desde el buscador del navegador. Esta búsqueda puede hacerse por palabra clave que exista en el título o en el contenido.

Se visualizarán todas las categorías, dibujos, robótica, libros y juegos, y se podrá acceder a cada una individualmente desde el menú de navegación.

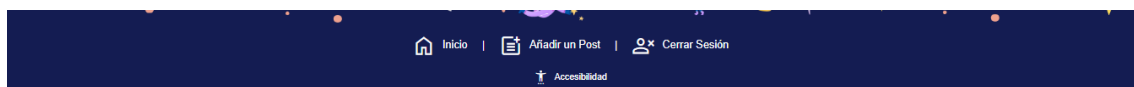
También se puede acceder al menú de accesibilidad para cambiar el tamaño de la letra, la tipografía de la fuente o el color a escala de grises según las necesidades de cada lector.



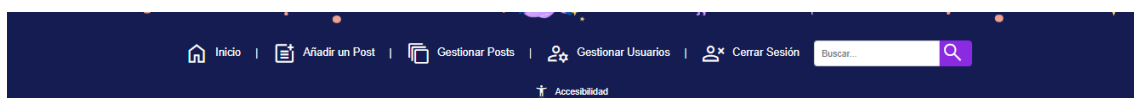
4.2. Inicio de sesión: los usuarios accederán al panel de login escribiendo en el navegador /login de esta forma: <https://emmarinablog.000webhostapp.com/login> En el formulario se introducirá el usuario y la contraseña. Si algún dato no es correcto se mostrará mensaje de error. Todas las contraseñas están cifradas en la base de datos.



4.3. Usuarios no administradores. Su panel de control permite visualizar todo el listado de post con paginación, agregar post y cerrar sesión.



4.4. Usuarios administradores: Su panel de control permite visualizar todo el listado de post, añadir un post, gestionar post, gestionar usuarios y cerrar sesión.



4.5. Agregar post: todos los usuarios pueden agregar post, el cual constará de un título personalizable con fuente y color, el contenido y una imagen, que se podrá cargar desde el equipo. Si hubiera algún error emergería el mensaje de error correspondiente.

Cuando un usuario no administrador pulsa en publicar post, este pasa a estado en revisión, para ser luego revisado y publicado por un administrador.

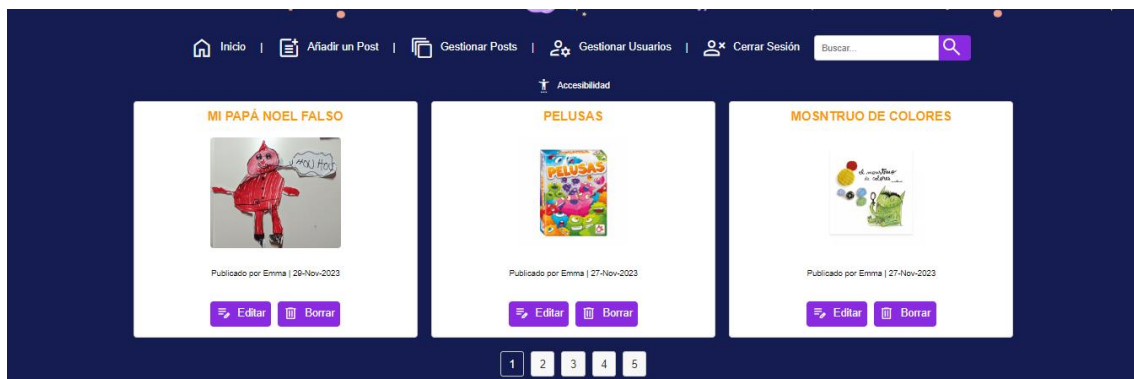
Es necesario cumplimentar todos los campos para poder publicar un post, y si alguno queda vacío saldrá un mensaje de error.

The screenshot shows the 'Añadir Post' form within a dark blue sidebar navigation menu. The form is white and contains the following fields and options:

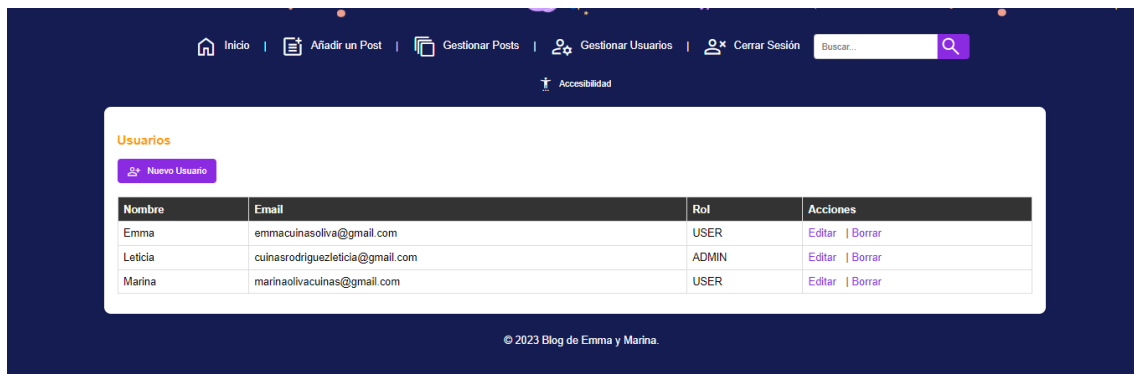
- Título:** A text input field.
- Color del Título:** A color selection box.
- Fuente del Título:** A dropdown menu with 'Verdana' selected.
- Categoría:** A dropdown menu with 'Dibujos' selected.
- Contenido:** A large text area for the post content.
- Imagen:** A section with a 'Seleccionar archivo' button and the text 'Ninguno archivo selec.'.
- Publicar:** A purple button to publish the post.

The sidebar menu includes links for 'Inicio', 'Añadir un Post', 'Gestionar Posts', 'Gestionar Usuarios', 'Cerrar Sesión', and 'Accesibilidad'.

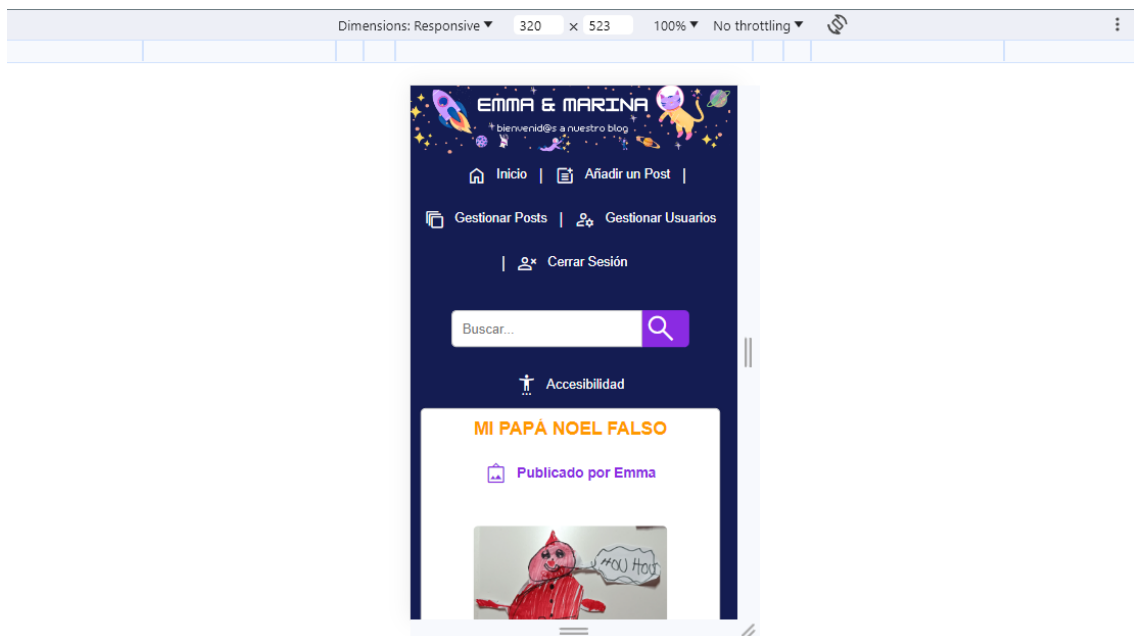
4.6. Gestión de post: permite a los usuarios administradores revisar un post que aún no se ha publicado y publicarlo, y editar y borrar post ya publicados. Se mostrará un listado de post por fecha de publicación con el estado correspondiente que podrá ser en revisión para editar y borrar. Un post con estado en revisión, una vez publicado cambia su estado ya sólo podrá ser editado o borrado.



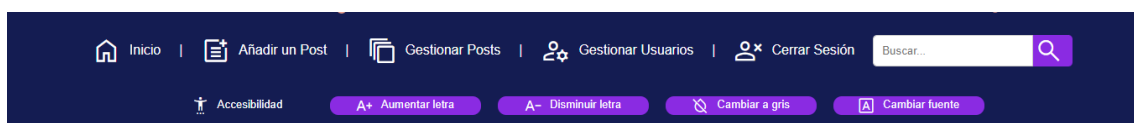
4.7. Gestión de usuarios: sólo accesible para el usuario administrador. Se visualizan los usuarios existentes con su rol, nombre y correo. Se pueden añadir nuevos usuarios, y modificar o borrar los existentes. El usuario administrador escoge el rol de los usuarios. Las contraseñas están protegidas con hash y se verifica que cumplan unos requisitos de seguridad.



4.8 Diseño responsive: Se han tenido en cuenta diferentes casos en cuanto a las opciones de visualización, adaptando las vistas a diferentes tamaños de pantalla. La vista móvil quedaría así:



4.9. Accesibilidad: el menú de accesibilidad está visible en cualquier vista del blog y permite a los usuarios cambiar el tamaño de la fuente, la tipografía y el color a escala de grises.



5. Lenguajes, tecnologías y recursos utilizados:



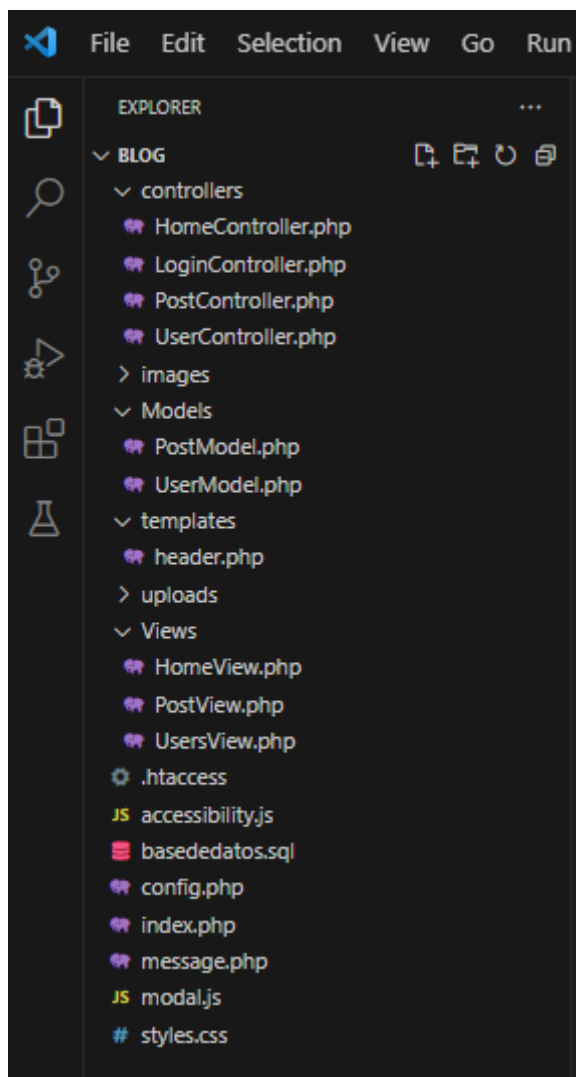
- **PHP:** Utilizado para el desarrollo del lado del servidor, gestionando la lógica del negocio y la interacción con la base de datos.
- **JavaScript (JS):** Utilizado para la creación de interactividad en el lado del cliente, como en el menú de accesibilidad, la gestión de los mensajes y la redirección de las acciones.
- **HTML:** Empleado para la estructura y presentación del contenido en la interfaz web.
- **CSS:** Utilizado para el diseño y estilo de la interfaz de usuario.
- **MySQL:** Como sistema de gestión de bases de datos, permitiendo el almacenamiento y recuperación eficiente de la información necesaria para el funcionamiento del blog.
- **XAMPP:** Como servidor local para facilitar el desarrollo y prueba del proyecto.
- **Filezilla:** Se utilizó como cliente FTP para la transferencia de archivos entre el entorno local y el servidor, facilitando la gestión de archivos estáticos y dinámicos.
- **000webhost.com:** Se seleccionó webhost.com como servidor de alojamiento debido a su opción gratuita y su facilidad de uso. Este servicio

permitió la transferencia eficiente del proyecto a un entorno de producción.

- **GitHub:** Como repositorio de versiones, proporcionando un sistema de control de versiones descentralizado para gestionar y colaborar en el desarrollo del proyecto.
- **Google Fonts y Google Icons:** Para las opciones de fuente y los iconos se ha optado por usar los de google, que tienen un acceso fácil y son reconocidos por la mayoría de los navegadores. Además, se pueden modificar ciertas características tanto de la fuente como de los iconos que permiten adaptarlos según las necesidades del proyecto.
- **Canva:** Las imágenes de la cabecera, el puntero, y cuando no hay publicaciones o imágenes disponibles, se han diseñado con Canva.

6. Estructura del proyecto MVC (Modelo Vista Controlador):

El proyecto se ha organizado siguiendo el patrón Modelo Vista Controlador, una arquitectura de software que separa las responsabilidades del desarrollo en tres componentes principales. Esta división clara de responsabilidades, mejora la legibilidad y comprensión del código y facilita el mantenimiento del mismo.



6.1. Controllers: Gestionan el funcionamiento de las rutas haciendo de intermediario entre el modelo y la vista.

- **HomeController.php:** Gestiona la página de inicio y las funcionalidades relacionadas con la presentación de contenido principal, las categorías y la búsqueda, así como de la paginación de cada una de ellas.
- **LoginController.php:** Maneja la autenticación, el inicio de sesión de usuarios y el cierre de sesión.
- **PostController.php:** Controla las operaciones relacionadas con las publicaciones, como el publicar post, editar, revisar y borrar.
- **UserController.php:** Encargado de la gestión de usuarios y sus perfiles.

```

// HomeController.php
class HomeController {
    //Muestra la página principal con una 1
    public function index($page = 1) {
        $postsPerPage = 3;
        // Instancia el modelo
        $postModel = new PostModel();
        // Obtiene el listado de posts desc
        $posts = $postModel->getAllPosts($page);
        // Instancia la vista
        $homeView = new HomeView();
        // Renderiza los posts en la vista
        $homeView->renderView($posts, $page);
    }
    //Obtiene el número de página de la URL
    public function getPage() {
        $url_segments = explode('/', $_SERVER['REQUEST_URI']);
        $page = end($url_segments);
        $this->index(intval($page));
        exit();
    }
    // Obtiene la categoría de la URL y mue
    public function getCategory() {
        $url_segments = explode('/', $_SERVER['REQUEST_URI']);
        $category = $url_segments[2];
        if (count($url_segments) > 3) {
            $page = intval($url_segments[3]);
        }
    }
}

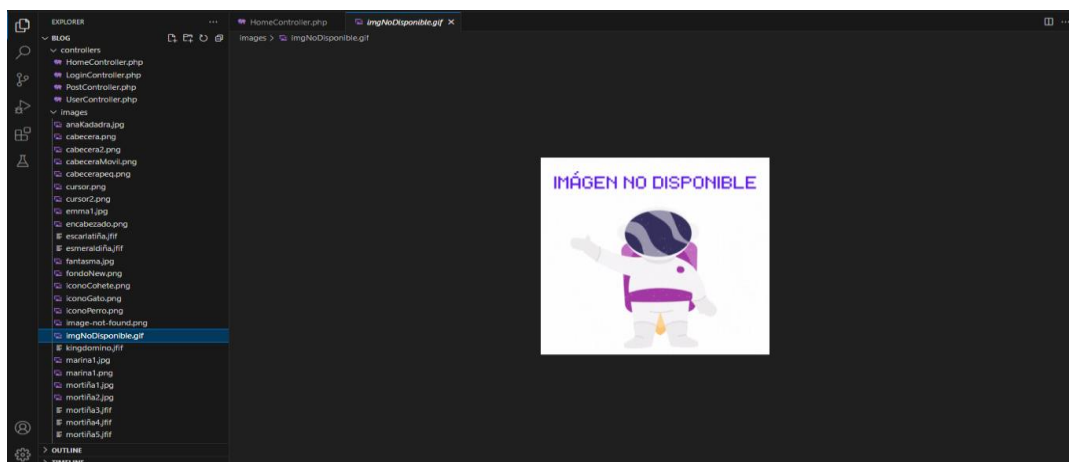
// LoginController.php
class LoginController {
    //Muestra la página de inicio de sesión
    public function index() {
        // Verifica si se ha enviado el formulario
        if ($_SERVER['REQUEST_METHOD'] == 'POST') {
            // Obtiene las credenciales del usuario
            $username = $_POST['usuario'];
            $password = $_POST['contraseña'];
            // Realiza la lógica de autenticación
            $userModel = new UserModel();
            $authenticated = $userModel->login($username, $password);
            if ($authenticated) {
                // Inicio de sesión exitoso
                //Header('Location: /');
                echo '<script type="text/javascript">window.location.href="/";</script>';
                echo '<meta http-equiv="refresh" content="0;url=/";</meta>'; exit();
            } else {
                // Inicio de sesión fallido
                $msg = array('tipo' => 'error', 'mensaje' => 'Usuario o contraseña incorrectos');
                //Header('Location: /login');
                echo '<script type="text/javascript">window.location.href="/login";</script>';
                echo '<meta http-equiv="refresh" content="0;url=/login";</meta>'; exit();
            }
        }
    }
}

// PostController.php
class PostController {
    //Muestra la página para crear un nuevo post
    public function index() {
        $this->checkAdmin();
        $postsPerPage = 3;
        $url_segments = explode('/', $_SERVER['REQUEST_URI']);
        $send = intval(end($url_segments));
        $page = $send > 0 ? $send : 1;
        // Instancia el modelo
        $postModel = new PostModel();
        // Obtiene el listado de posts desc
        $posts = $postModel->getAllPosts($page);
        // Instancia la vista
        $postView = new PostView();
        // Renderiza los posts en la vista
        $postView->renderView($posts, $page);
    }
    //Muestra la página para crear un nuevo post
    public function getNewPost() {
        $this->checkAdmin();
        // Instancia la vista
        $postModel = new PostModel();
        $postView = new PostView();
        $categorias = $postModel->getAllCategories();
        $postView->renderView($categorias);
    }
    //Procesa el formulario para crear un nuevo post
    public function postNewPost() {
        $this->checkAdmin();
        $titulo = $_POST['titulo'];
        $contenido = $_POST['contenido'];
        // Instancia el modelo
        $postModel = new PostModel();
        $postModel->create($titulo, $contenido);
    }
}

// UserController.php
class UserController {
    //Muestra el formulario para crear un nuevo usuario
    public function index() {
        $this->checkAdmin();
        $usersView = new UsersView();
        $usersView->renderForm();
    }
    //Muestra la lista de usuarios.
    public function getUsers() {
        $this->checkAdmin();
        $userModel = new UserModel();
        $users = $userModel->getAllUsers();
        $usersView = new UsersView();
        $usersView->renderView($users);
    }
    //Procesa el formulario para crear un nuevo usuario
    public function postNewUser() {
        $this->checkAdmin();
        $username = $_POST['nombre'];
        $password = $_POST['contraseña'];
        $email = $_POST['email'];
        $rol = $_POST['rol'];
        $userModel = new UserModel();
        $userModel->addUser($username, $password, $email, $rol);
    }
    //Procesa el formulario para actualizar un usuario
    public function postUpdateUser() {
        $this->checkAdmin();
        $url_segments = explode('/', $_SERVER['REQUEST_URI']);
        $userId = end($url_segments);
        $username = $_POST['nombre'];
    }
}

```

6.2. Images: Almacena recursos estáticos.



6.3. Models: Son los modelos de datos, encargados de hacer las consultas a la base de datos y devolvérselos al controlador.

- **PostModel.php:** Modelo de datos para la gestión de publicaciones.
- **UserModel.php:** Modelo encargado de las operaciones relacionadas con la gestión de usuarios.

```

// PostModel.php
38 // Consulta para obtener las publicaciones con paginación y búsqueda
39 $sql = "SELECT publicaciones.publicacion_id, publicaciones.titulo, publicaciones.c
40 FROM publicaciones
41 JOIN usuarios ON publicaciones.autor_id = usuarios.usuario_id JOIN categorias on c
42 if ($onlyPublished == true) {
43     $sql = $sql . " WHERE publicado = 1";
44 }
45 if ($category) {
46     $sql = $sql . " AND categorias.nombre = '$category'";
47 }
48 $sql = $sql . " ORDER BY fecha_publicacion DESC LIMIT ?, ?";
49 $stmt = $conn->prepare($sql);
50 $stmt->bind_param("ii", $limit, $postsPerPage);
51 $stmt->execute();
52 $result = $stmt->get_result();
53
54 $stmt->close();
55 $conn->close();
56 return ['content' => $result, 'totalPages' => $total_pages];
57
58 // Agrega un nuevo post a la base de datos.
59 1 referencia [0] overrida
60 public function addPost($titulo, $contenido, $colorFuenteTitulo, $fuenteTitulo, $imagen
61     $dbHost = DB_HOST;
62     $dbUser = DB_USER;
63     $dbPassword = DB_PASSWORD;
64     $dbName = DB_NAME;
65     $conn = new mysqli($dbHost, $dbUser, $dbPassword, $dbName);
66     move_uploaded_file($_FILES['imagen']['tmp_name'], $imagen_path);
67
68     // Insertar la información en la base de datos
69     $usuario = $_SESSION['usuario'];
70     $sql = "INSERT INTO publicaciones (titulo, contenido, imagen_url, autor_id, color_t
71     VALUES ('$titulo', '$contenido', '$imagen_path', (SELECT usuario_id FROM us
72     $result = $conn->query($sql);
73     $conn->close();
74     if ($result == TRUE) {
75         echo "Se ha creado una nueva publicación exitosamente. Redirigiendo a la
76
// UserModel.php
39 // Obtiene todos los usuarios de la base de datos.
40 1 referencia [0] overrida
41 public function getAllUsers() {
42     $dbHost = DB_HOST;
43     $dbUser = DB_USER;
44     $dbPassword = DB_PASSWORD;
45     $dbName = DB_NAME;
46     $conn = new mysqli($dbHost, $dbUser, $dbPassword, $dbName);
47
48     if ($conn->connect_error) {
49         die("Error en la conexión a la base de datos: " . $conn->connect_error);
50     }
51
52     $sql = "SELECT * FROM usuarios ORDER BY nombre ASC";
53     $result = $conn->query($sql);
54     $conn->close();
55
56     return ['content' => $result ];
57
58 //Añade el usuario
59 1 referencia [0] overrida
60 public function addUser($username, $password, $email, $rol) {
61     // Hash de la contraseña antes de almacenarla en la base de datos
62     $hashedPassword = password_hash($password, PASSWORD_DEFAULT);
63
64     $dbHost = DB_HOST;
65     $dbUser = DB_USER;
66     $dbPassword = DB_PASSWORD;
67     $dbName = DB_NAME;
68     $conn = new mysqli($dbHost, $dbUser, $dbPassword, $dbName);
69
70     // Validar el formato del correo electrónico
71     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
72         $_SESSION['mensaje'] = array('tipo' => 'error', 'contenido' => 'Formato de correo elec
73         //header("Location: /newUser");
74         echo "<script type='text/javascript'>";
75         echo "window.location.href='/newUser'";
76         echo "</script>";
    
```

6.4. Templates: Se encuentra el header que es el elemento común a todas las páginas y en el que se define la barra de navegación, el menú de accesibilidad y el fondo de la cabecera del blog.

- **Header.php**

```

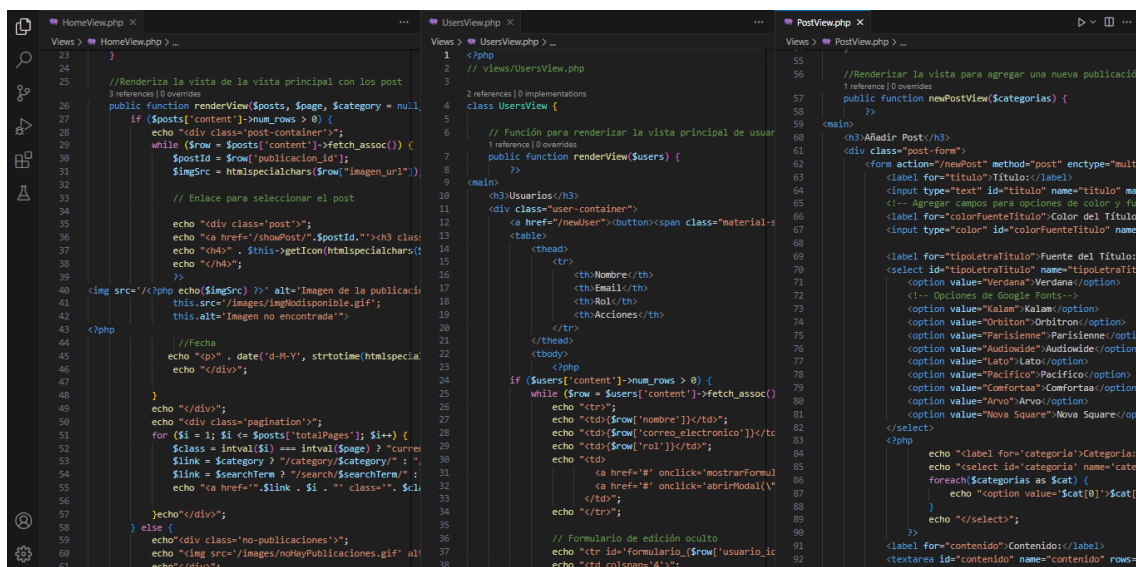
1 header.php
2
31 </head>
32
33 <body>
34
35 <header>
36     
38     
40 </header>
41
42 <nav>
43     <php
44     if (isset($_SESSION['usuario'])) {
45         if ($_SESSION['rol'] == "ADMIN") {
46             //Sección de navegación para usuarios administradores
47             echo "<a href='/'><span class='material-symbols-outlined'>home/</span>Inicio</a> |
48             <a href='/newPost'><span class='material-symbols-outlined'>post_add/</span>Añadir un Post</a> |
49             <a href='/posts'><span class='material-symbols-outlined'>auto_awesome_motion/</span>Gestionar Posts</a> |
50             <a href='/users'><span class='material-symbols-outlined'>manage_accounts/</span>Gestionar Usuarios</a> |
51             <a href='/logout'><span class='material-symbols-outlined'>person_cancel/</span>Cerrar Sesión</a>;
52         }
53     }
54     <form class='form-container'>
55         <input type='text' placeholder='Buscar...' name='searchTerm' id='searchTerm'>
56         <button type='button' onclick='submitForm("searchPosts");'><span class='material-symbols-outlined'>
57             search
58         </span></button>
59     </form>
60
61     <php
62     } else {
63         //Sección de navegación para usuarios no administradores
64         echo "<a href='/'><span class='material-symbols-outlined'>home/</span>Inicio</a> |
65         <a href='/newPost'><span class='material-symbols-outlined'>post_add/</span>Añadir un Post</a> |
66         <a href='/logout'><span class='material-symbols-outlined'>person_cancel/</span>Cerrar Sesión</a>;
67     }
68 } else {
69     echo "<a href='/'><span class='material-symbols-outlined'>home/</span>Inicio</a> |
70     <a href='/category/dibujos'><span class='material-symbols-outlined'>hallway/</span>Dibujos</a> |
71     <a href='/category/robotica'><span class='material-symbols-outlined'>smart_toy/</span>Robótica</a> |
72     <a href='/category/imagenes'><span class='material-symbols-outlined'>image/</span>Imágenes</a>;
    
```

6.5. Uploads: Para guardar las imágenes subidas.

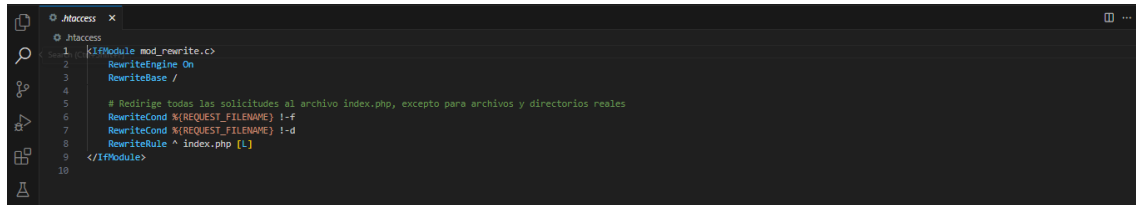


6.6. Views: Las vistas, que se encargarán de pintar el contenido de las páginas.

- **HomeView.php:** Vista principal para la página de inicio.
- **PostView.php:** Vista para la visualización y gestión de publicaciones.
- **UsersView.php:** Vista para la gestión de usuarios y perfiles.

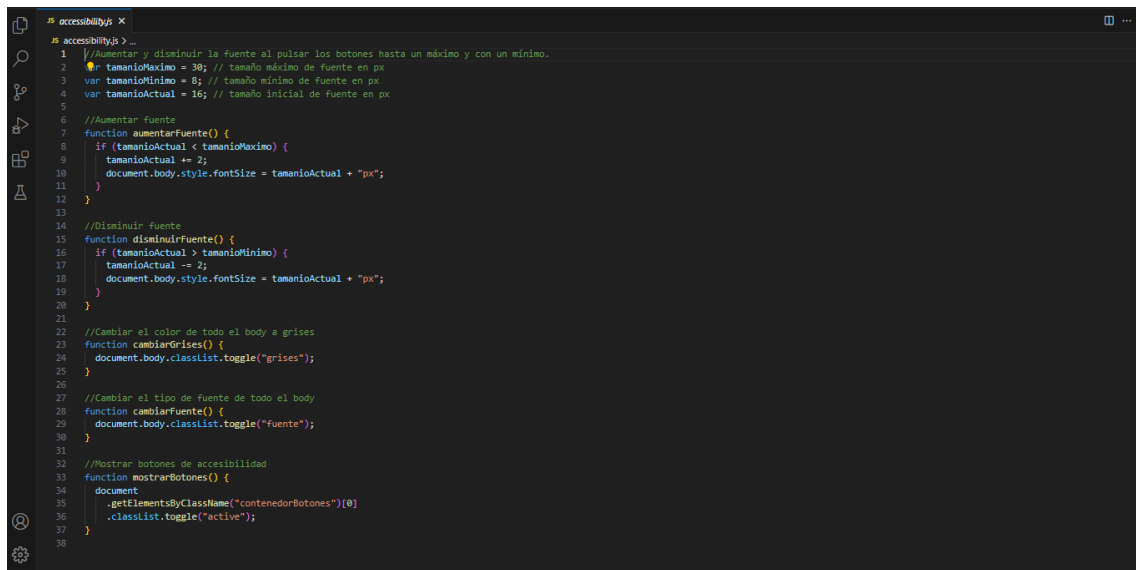


6.7. .htcaces: Este archivo es esencial para la configuración del servidor web. Contiene las reglas de reescritura de URL, configuraciones de seguridad y otras directivas que afectan la forma en que el servidor web maneja las solicitudes.



```
1 <IfModule mod_rewrite.c>
2 RewriteEngine On
3 RewriteBase /
4
5 # Redirige todas las solicitudes al archivo index.php, excepto para archivos y directorios reales
6 RewriteCond %{REQUEST_FILENAME} !-f
7 RewriteCond %{REQUEST_FILENAME} !-d
8 RewriteRule ^ index.php [L]
9 </IfModule>
10
```

6.8. Accesibility.js: Contiene funciones, lógica y diseño, relacionados con la accesibilidad. Es posible aumentar o disminuir el tamaño de la fuente del documento, cambiar el color a escala de grises y cambiar la tipografía de la fuente de forma puntual, para adaptarlo a necesidades visuales específicas.



```
1 //Aumentar y disminuir la fuente al pulsar los botones hasta un máximo y con un mínimo.
2 var tamañoMaximo = 30; // tamaño máximo de fuente en px
3 var tamañoMinimo = 8; // tamaño mínimo de fuente en px
4 var tamañoActual = 16; // tamaño inicial de fuente en px
5
6 //Aumentar fuente
7 function aumentarFuente() {
8   if (tamañoActual < tamañoMaximo) {
9     tamañoActual += 2;
10    document.body.style.fontSize = tamañoActual + "px";
11  }
12 }
13
14 //Disminuir fuente
15 function disminuirFuente() {
16   if (tamañoActual > tamañoMinimo) {
17     tamañoActual -= 2;
18     document.body.style.fontSize = tamañoActual + "px";
19   }
20 }
21
22 //Cambiar el color de todo el body a grises
23 function cambiarGrises() {
24   document.body.classList.toggle("grises");
25 }
26
27 //Cambiar el tipo de fuente de todo el body
28 function cambiarFuente() {
29   document.body.classList.toggle("fuente");
30 }
31
32 //Mostrar botones de accesibilidad
33 function mostrarBotones() {
34   document
35     .getElementsByClassName("contenedorBotones")[0]
36     .classList.toggle("active");
37 }
38
```

6.9. Config.php: Centraliza la configuración del proyecto. Se incluyen las credenciales de base de datos y las variables de entorno.

6.10. Index.php: Inicia el flujo de control del programa con la lógica para enrutar las solicitudes a los controladores correspondientes y cargar las vistas adecuadas.

```
index.php
9
10 // Define las rutas y los controladores correspondientes
11 $routes = [
12     '/' => 'HomeController',
13     '/page' => 'HomeController',
14     '/login' => 'LoginController',
15     '/logout' => 'LoginController',
16     '/users' => 'UserController',
17     '/newUser' => 'UserController',
18     '/deleteUser' => 'UserController',
19     '/updateUser' => 'UserController',
20     '/posts' => 'PostController',
21     '/searchPosts' => 'PostController',
22     '/newPost' => 'PostController',
23     '/editPost' => 'PostController',
24     '/deletePost' => 'PostController',
25     '/showPost' => 'PostController',
26     '/category' => 'HomeController',
27     '/search' => 'HomeController',
28 ];
29
30 // Verifica si la ruta solicitada existe en la lista de rutas
31 $request_url_without_prefix = '/' . explode('/', $_SERVER['REQUEST_URI'])[1];
32 if (array_key_exists($request_url_without_prefix, $routes)) {
33     // Obtiene el nombre del controlador
34     $controllerName = $routes[$request_url_without_prefix];
35
36     // Incluye el archivo del controlador
37     include_once("../controllers/{$controllerName}.php");
38 }
```

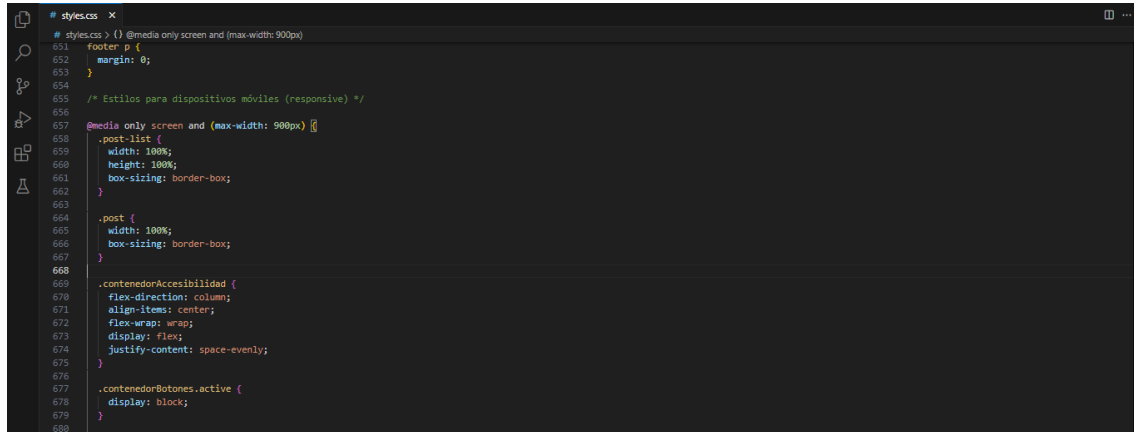
6.11. Message.php: Se encarga de la gestión de mensajes y notificaciones en el blog.

```
message.php
1 <?php
2 // Verificar si hay un mensaje en la sesión
3 if (isset($_SESSION['mensaje'])) {
4     $mensaje = $_SESSION['mensaje'];
5     // Mostrar el mensaje con el tipo correspondiente
6     echo "código";
7     setTimeout(function() {
8         document.getElementById('mensaje').style.display = 'none';
9     }, 5000);
10 }
11
12 echo "<div id='mensaje' class='\" . $mensaje['tipo'] . \"'>";
13     <p>\" . $mensaje['contenido'] . \"</p>";
14 </div>";
15
16 // Eliminar el mensaje de la sesión para que no se muestre en futuras visitas
17 unset($_SESSION['mensaje']);
18 }
```

6.12. Modal.js: Implementa la ventana de confirmación para las acciones de borrado.

```
modal.js
1 // script.js
2 modal = document.getElementById("confirmationModal");
3 var actionToPerform = null;
4
5 // Mostrar el modal con la acción configurada
6 function showModal(action) {
7     actionToPerform = action;
8     modal.style.display = "block";
9 }
10
11 // Ocultar el modal
12 function hideModal() {
13     modal.style.display = "none";
14 }
15
16 // Lógica para confirmar la acción
17 function confirmAction() {
18     if (actionToPerform) {
19         // Redirigir a la acción configurada
20         window.location.href = actionToPerform;
21     }
22     hideModal();
23 }
24
25 // Lógica para cancelar la acción
26 function cancelAction() {
27     hideModal();
28 }
29
30 // Cerrar el modal si se hace clic fuera de él
31 window.addEventListener("click", function (event) {
32     if (event.target == modal) {
33         hideModal();
34     }
35 });
36 }
```

6.13. Styles.css: La hoja de estilos del proyecto define la apariencia visual del blog. Contiene reglas CSS que personalizan la presentación de las páginas, asegurando una experiencia coherente y atractiva para los visitantes.



7. Base de datos:

7.1. Tablas: Consta de tres tablas: "usuarios", "publicaciones" y "categorías". Estas tablas están interconectadas para facilitar la gestión de usuarios y la organización de publicaciones en categorías específicas.

- **Usuarios:** almacena información sobre los usuarios del blog, como su identificación única (usuario_id), nombre, correo electrónico y roles asignados. Se han usado hash para las contraseñas.
- **Publicaciones:** gestiona las publicaciones en el blog. Cada publicación tiene un identificador único (publicacion_id) y contiene información esencial como título, contenido, fecha de publicación, y la URL opcional de la imagen asociada.
- **Categorías:** se utiliza para organizar las publicaciones en categorías específicas. Cada categoría tiene un identificador único (categoria_id) y un nombre descriptivo.

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> categorías	Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> publicaciones	Examinar Estructura Buscar Insertar Vaciar Eliminar	14	InnoDB	utf8mb4_general_ci	48.0 KB	-
<input type="checkbox"/> usuarios	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	32.0 KB	-

7.2. Relaciones:

- **Relación Usuario-Publicación:** Cada publicación está asociada con un autor a través del campo `autor_id`, que es una clave foránea que se conecta con la tabla "usuarios". Esto establece el autor de la publicación.
- **Relación Publicación-Categoría:** Cada publicación está asignada a una categoría específica a través del campo `categoria_id`, que se relaciona con la tabla "categorías". Esto organiza las publicaciones según temas o temas específicos.



8. Manual de instalación:

Se ha elegido **000webhost.com** por contar con alojamiento gratuito y por la facilidad de uso. Los pasos a seguir son los siguientes:

1. Registro en 000webhost.com: Proporcionar la información necesaria para el registro y elegir el plan de alojamiento, en este caso, gratuito.

2. Configuración del Dominio: Crear un dominio nuevo con 000webhost.com.

3. Transferencia de Archivos: Utilizar un cliente de FTP (File Transfer Protocol) o el administrador de archivos del panel de control para subir los archivos al servidor. Se ha usado FileZilla para realizar la transferencia mediante FTP.

4. Directorio Raíz: Subir los archivos a la carpeta correcta en el servidor. Esto suele ser el directorio raíz o una carpeta específica designada para los archivos.

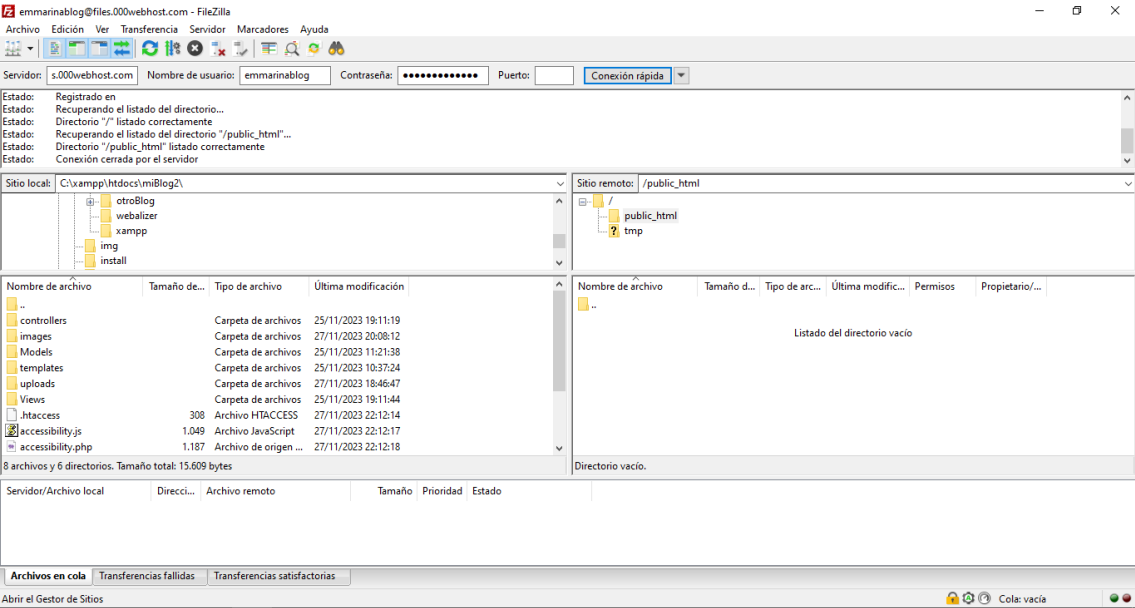
5. Base de Datos: Configurar la base de datos en el panel de control y ajustar la configuración del proyecto para que se conecte a esta base de datos.

7. Verificación: Verificar que los archivos se hayan transferido correctamente y que el sitio web esté funcionando según lo esperado.

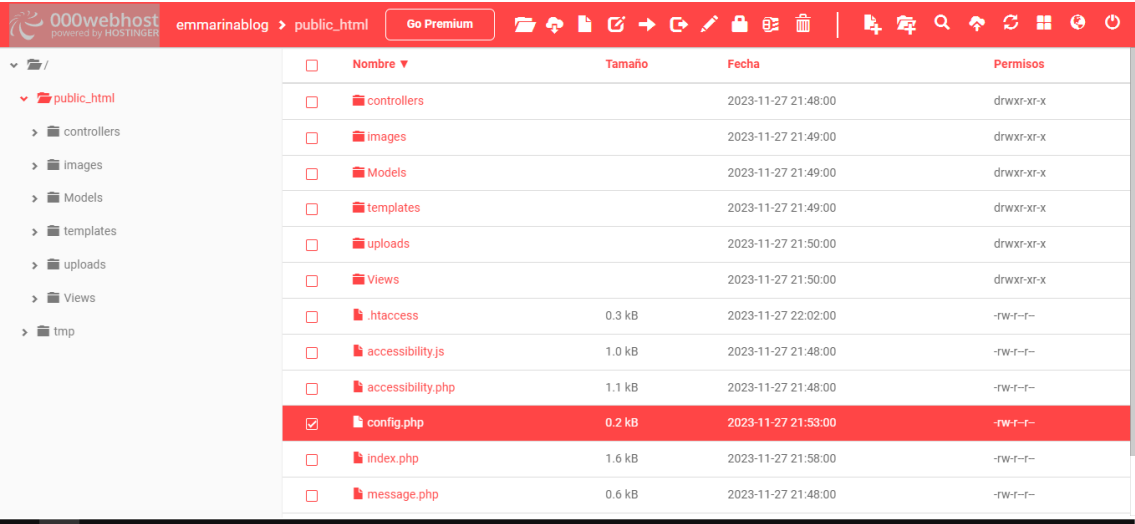
8. Configuraciones Adicionales: Se han realizado ajustes de seguridad y configuración de SSL.

9. Prueba: Realizar pruebas del sitio web para asegurarse de que todas las funcionalidades estén operativas y que no haya problemas de visualización ni de rendimiento. En este punto se han detectado diversos errores con las rutas, y se ha tenido que modificar todo el código para que funcionasen correctamente.

FileZilla: vista de los datos subidos mediante FTP.



000webhost.com: vista de las carpetas del proyecto desde la web del servidor.



9. Proyecto en GitHub:

Se ha subido el proyecto a GitHub para tener acceso al repositorio descentralizado. Los pasos seguidos son:

1. Registrarse en GitHub o crear una cuenta.

2. Crear un repositorio público nuevo.

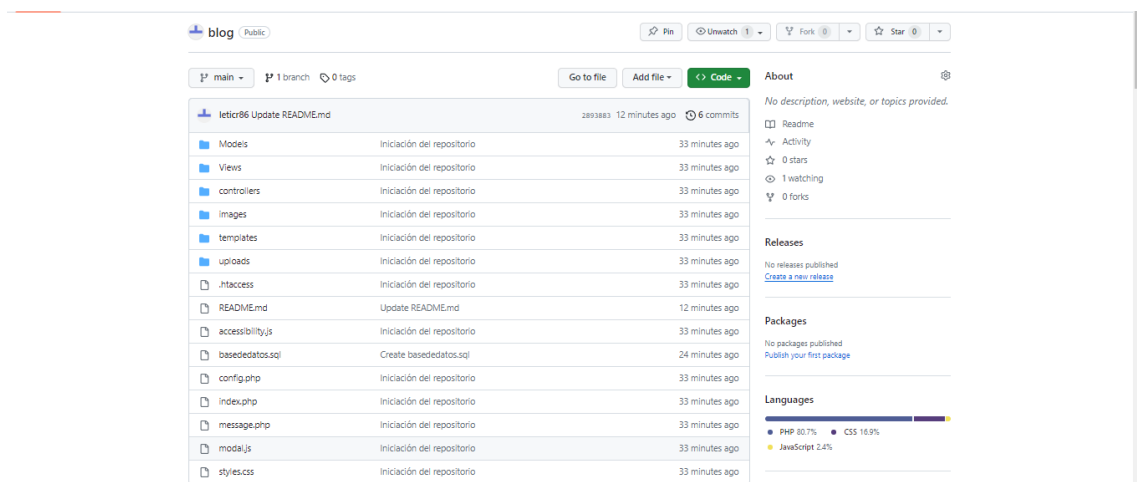


3. Clonar el repositorio en local: `git clone https://github.com/tu-usuario/tu-proyecto.git`

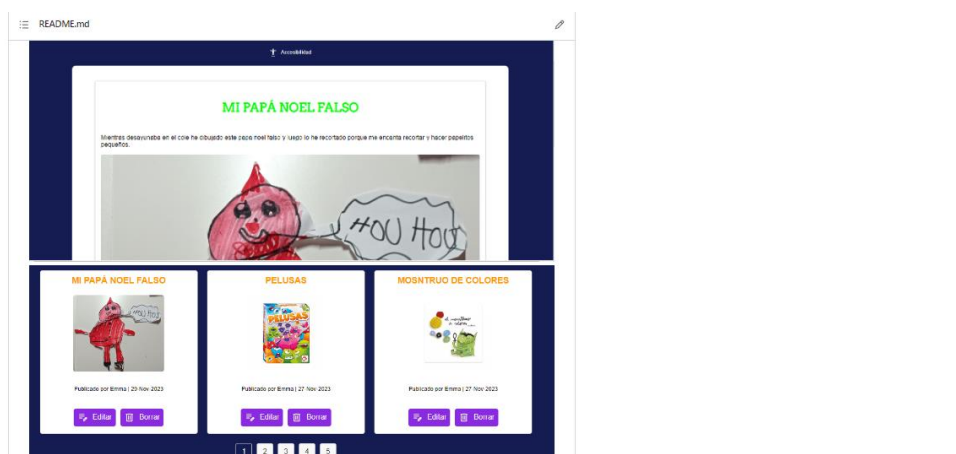
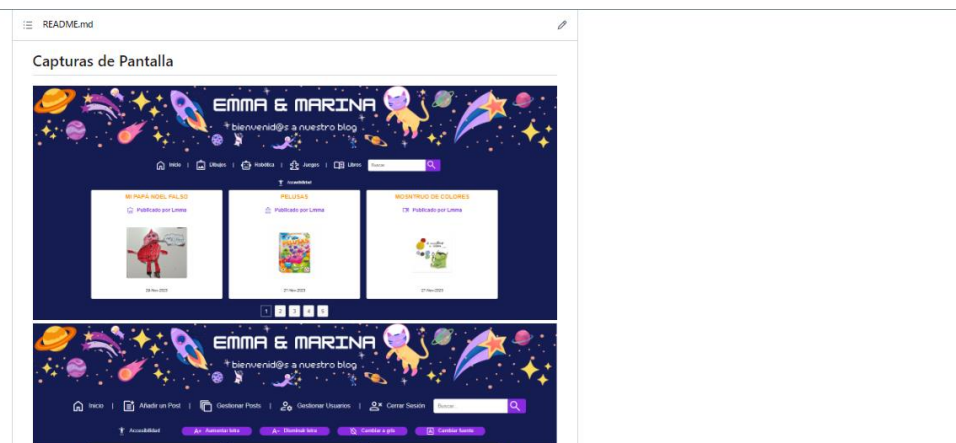
4. Copiar en local los ficheros del proyecto para luego añadirlos al repositorio: configurar la base de datos e importar la creación de tablas con el archivo SQL del proyecto.

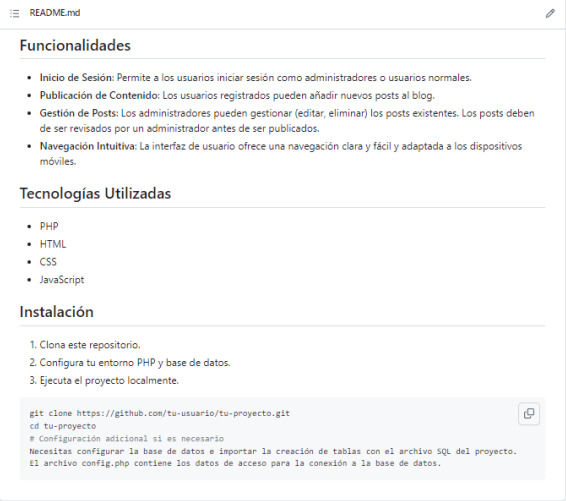
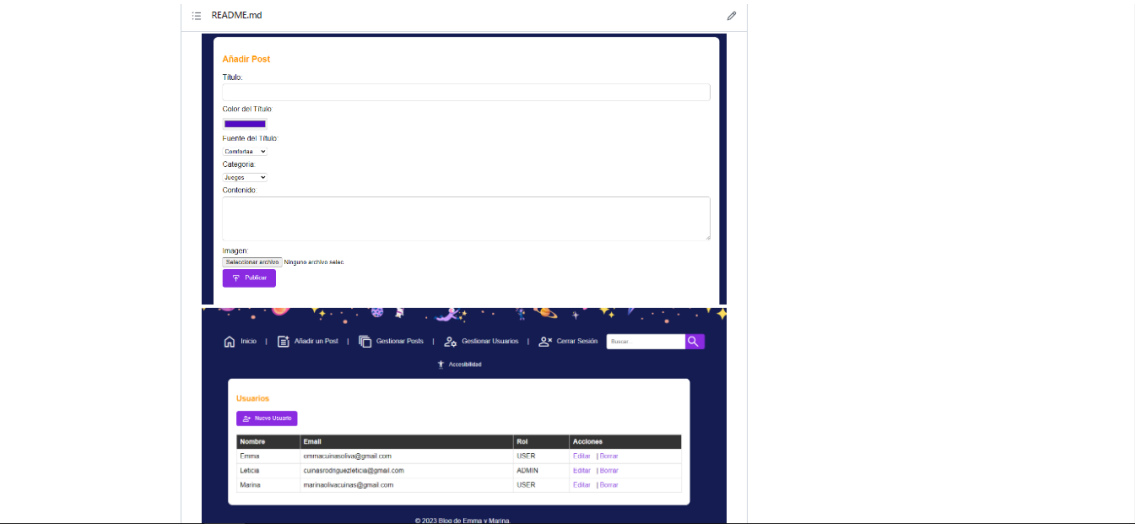
El archivo `config.php` contiene los datos de acceso para la conexión a la base de datos

5. Hacer un commit y un push de los cambios para guardarlos en el repositorio.



6. Crear el README: Se ha configurado con el nombre del proyecto, la descripción, capturas de pantalla, las funcionalidades, tecnologías utilizadas y el proceso de instalación.





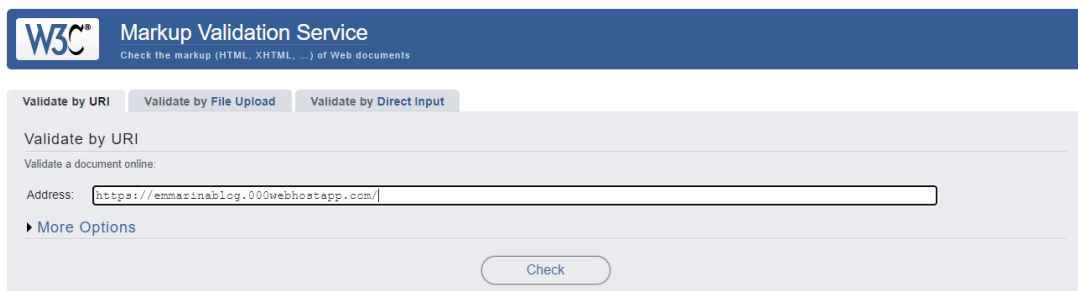
10. Validación de html, CSS y accesibilidad.

Durante el proceso de validación, se identificaron varios errores en la accesibilidad, HTML y CSS, los cuales fueron corregidos posteriormente. Se utilizaron las siguientes herramientas:

- **Tawdis:** para comprobar la accesibilidad WCAG.
<https://www.tawdis.net/>
- **Validator w3** para validar el html.
<https://validator.w3.org/>
- **Jigsaw w3** para la validación del CSS.
<https://jigsaw.w3.org/css-validator/>

Estas herramientas de validación fueron fundamentales para asegurar que el proyecto cumpliera con los estándares de accesibilidad y las mejores prácticas de codificación, garantizando una experiencia de usuario optimizada y libre de errores en la presentación y estructura del contenido web.

- **Validación de html con Validator W3C:** Los errores surgidos se corrigieron en su totalidad.



The image shows the W3C Markup Validation Service interface. At the top, there is a blue header with the W3C logo and the text 'Markup Validation Service' and 'Check the markup (HTML, XHTML, ...) of Web documents'. Below the header, there are three tabs: 'Validate by URI' (selected), 'Validate by File Upload', and 'Validate by Direct Input'. Under the 'Validate by URI' tab, there is a section titled 'Validate a document online:' with a text input field labeled 'Address:' containing the URL 'https://emmarinablog.000webhostapp.com/'. Below the input field, there is a link 'More Options' and a 'Check' button.

Info Trailing slash on void elements **has no effect** and **interacts badly with unquoted attribute values**.
From line 6, column 5; to line 7, column 115
`<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@48,400,0,0" />`

Error End of file seen and there were open elements.
From line 84, column 1114; to line 84, column 1117
`<class="">`

Error Unclosed element `div`.
From line 84, column 927; to line 84, column 950
`<div><div class="pagination"><a href=`

- **Validación de CSS con Jigsaw W3:** tanto las advertencias como errores mostrados pudieron ser solventados.

W3C El Servicio de Validación de CSS del W3C
Resultados del Validador CSS del W3C para <https://emmarinablog.000webhostapp.com/> (CSS versión 3 + SVG)

Ir a: [Los Errores \(2\)](#) [Las Advertencias \(11\)](#) [Su Hoja de Estilo validada](#)

Resultados del Validador CSS del W3C para <https://emmarinablog.000webhostapp.com/> (CSS versión 3 + SVG)

Disculpas! Hemos encontrado las siguientes errores (2)

URI : https://emmarinablog.000webhostapp.com/styles.css	
281	.grises
Error de análisis sintáctico: <code>gray()</code>	
334	.contenedorEnlaces li:hover
Propiedad no válida : font-weight dimensión desconocida <code>600m</code>	

Interested in understanding what new technologies are coming out of W3C? Follow [@w3cdevs on X](#) to keep track of what the future looks like!

[W3Cdevs logo](#)

[Donate](#) and help us build better tools for a better web.

- **Tawdis para comprobar la accesibilidad WCAG:** Se corrigieron varios problemas que fueron fácilmente detectables y que estaban acordes al proyecto, como alguna imagen que se mostraba sin título alternativo, el tamaño de la fuente o el color.

Tawdis | [ESI] | [ENI] | [IPT]

Resumen

7 Problemas en 6 criterios de éxito Son necesarias correcciones [?] Perceptible 3 [?] Operable 0 [?] Comprensible 3 [?] Robusto 1	19 Advertencias en 8 criterios de éxito Es necesario revisar manualmente [?] Perceptible 5 [?] Operable 8 [?] Comprensible 6 [?] Robusto 0	16 No verificados en 16 criterios de éxito Comprobación completamente manual [?] Perceptible 4 [?] Operable 8 [?] Comprensible 4 [?] Robusto 0
---	--	--

Recurso: <https://emmarinablog.000webhostapp.com/> Fecha: 28/11/2023 08:13 Pautas WCAG 2.1 Nivel del análisis: AA Tecnologías: HTML, CSS

Acceda al **informe detallado** para obtener más información sobre las incidencias detectadas.

@ email

11. Problemas y Mejoras:

En el desarrollo y validación del proyecto, se identificaron algunos problemas y se proponen mejoras para su consideración:

11.1. Problemas Detectados:

- Errores de rutas en la configuración del servidor que se solucionaron ajustando el archivo .htaccess y redirecciones en JavaScript.
- Necesidad de ajustes en el código para adaptarse al entorno de producción corrigiendo las rutas absolutas y relativas.
- Correcciones de validación HTML, CSS y accesibilidad.
- Uso de cliente FTP para subir el proyecto al servidor, que al ser gratuito no tiene la opción de sincronizarlo con GitHub.

11.2. Mejoras Propuestas:

- Personalizar las vistas de cada usuario.
- Ampliar las opciones de personalización al publicar post.
- Estudiar la posibilidad de incluir vídeos guardados en la nube.

12. Conclusiones Finales:

En el proceso de desarrollo de este proyecto, he tomado la decisión consciente de construir todo el código de manera manual, prescindiendo del uso de cualquier framework o plantilla preexistente. Esta elección se fundamenta en mi deseo de experimentar y abordar cada aspecto del desarrollo web de forma personal.

12.1. Razones para Desarrollar Manualmente:

- **Aprendizaje Integral:** Al escribir cada línea de código desde cero, he tenido la oportunidad de comprender en profundidad cómo funcionan cada uno de los componentes del proyecto. Esto incluye la estructura del modelo-vista-controlador (MVC), la interacción con la base de datos, y la gestión de las vistas y controladores.
- **Control Total sobre el Proyecto:** Desarrollar sin depender de frameworks o plantillas preconstruidas me ha otorgado un control completo sobre la arquitectura y la lógica del proyecto. Puedo adaptar cada aspecto según las necesidades específicas del blog infantil, sin limitaciones impuestas por terceros.
- **Desarrollo Personalizado:** Al prescindir de frameworks, he podido personalizar cada funcionalidad y diseño según mis preferencias y requisitos específicos del proyecto. Esto me permite crear una experiencia única y adaptada a las necesidades del público objetivo.
- **Desafío y Crecimiento Profesional:** Abordar el desarrollo sin la ayuda de frameworks representa un desafío que impulsa mi crecimiento profesional. Afrontar las complejidades del desarrollo web manualmente fortalece mis habilidades y conocimientos en programación y diseño.

12.2. Consideraciones Futuras:

Si bien esta decisión ha proporcionado una experiencia valiosa, es importante tener en cuenta que, en algunos proyectos o entornos de desarrollo, el uso de frameworks puede ofrecer eficiencia y ventajas considerables. En futuros proyectos, evaluaré la conveniencia de utilizar frameworks según los requisitos específicos y las metas del desarrollo.

13. Webgrafía:

- **Enlace a presentación en Canva:**
https://www.canva.com/design/DAF2f1BnCbo/BxfcRI24Cz5LXio0xkt56A/view?utm_content=DAF2f1BnCbo&utm_campaign=designshare&utm_medium=link&utm_source=editor
- **Canva:** <https://www.canva.com/>
- **Google icons:** <https://fonts.google.com/icons>
- **Google fonts:** <https://fonts.google.com/>
- **Tawdis:** <https://www.tawdis.net/>
- **Jigsaw W3:** <https://jigsaw.w3.org/css-validator/>
- **Validator W3:** <https://validator.w3.org/>
- **Xampp:** <https://www.apachefriends.org/es/index.html>
- **GitHub:** <https://github.com/>
- **MySql:** <https://www.mysql.com/>
- **Filezilla:** <https://filezilla-project.org/>
- **000Wenhost:** <https://www.000webhost.com/>