MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Web

Web personal con foro

Autor: David Fajardo Barral

Tutor: Laura Bermúdez Galbarriatu

Fecha de entrega: 02/12/2022

Convocatoria: 2S2122

Documentos del proyecto: https://drive.google.com/drive/folders/1FMIN_YpJDxf7Fv7tH9zxK4zCc61IVgN4?usp=sharing

Índice de contenidos

En la normativa de proyectos vigente encontrarás una breve descripción de cada uno de estos apartados para saber qué información debes incluir en ellos

1. Introducción

1.1. Motivación

Curiosamente, mi motivación para realizar este proyecto es aprender, ampliar mis conocimientos en desarrollo web. Es cierto que esta aplicación nació para publicitar las novelas que iba escribiendo, tanto a través de su lectura como de un foro con tintes de red social (*scroll* infinito, personalización de los colores por usuario…) para no parecer anticuado, pero una vez en el proceso, el mundo de la programación empezó a atraerme sobremanera y dejó de tratarse de simplemente publicitar nada para convertirse en pensar qué más podría añadirle al aplicativo para hacerlo más completo, a la par que practicaba y avanzaba en mi aprendizaje.

1.2. Abstract

This project consists of a web app with a reading area where users can read the (my) novels of the administrator, as well as write messages in a modern forum with various topics and open the possibility of meeting new people. Each user will have a public profile and a corresponding private control panel. The administrator will have several CRUDs to manage all aspects of the web app. In addition, through some optional data of each user control panel as their favourite colour, they will be able to customise the website for themselves, as some elements will be adapted to that colour. Compatibility with different screen sizes will be guaranteed by the use of Bootstrap. In addition, through the database used, the website will remember the reading position of each user in each novel. There will also be a dark mode so as not to damage the eyesight, an index... And the forum will have the latest innovations: a modern design inspired by Bootstrap elements and colours, an infinite scroll instead of pagination to look more like a social network and a quick system of buttons to edit your messages or reply to others.

1.3. Objetivos propuestos (generales y específicos)

- El objetivo general es atraer a la gente para que pueda leer las novelas y/o participar en el foro de la web. Lo primero se podrá hacer sin necesidad de crear una cuenta de usuario, lo cual debería ayudar a facilitar el objetivo.
- Para la completa realización de este proyecto será necesario pensar bien y elaborar la pertinente base de datos, realizando su diagrama entidad-relación y luego escribiendo el código. A continuación, llegará la hora de ponerse a escribir código

HTML y CSS para crear toda la estructura y estética de la web. Análogamente se irá usando JS y PHP para toda la capa interactiva y procesadora de la app. Conforme se vayan implementando funciones, será necesario probarlas, testeando cada elemento del proyecto y asegurándose de que cumplen su función. También se comprobarán e impedirán ciertas vulnerabilidades (inyección SQL, acceso público a zonas privadas… etc).

2. Metodología usada

Tras una seria valoración de las opciones disponibles, se ha optado por la metodología SCRUM, ya que se adapta perfectamente a la situación: un producto hecho partiendo de 0 absoluto donde inevitablemente surgirán errores que obligarán a retroceder para solventarlos e ir entregando el producto por partes, ya que debido a su complejidad es imposible entregarlo al completo en un solo intento. Y aunque así fuera, seguramente habría partes que se podrían mejorar, aunque solo fuera con una refactorización de código.

Las fases del diseño se verán con mayor detalle en el diagrama de Gantt más adelante en esta documentación, pero de momento se realizará una serie de comentarios sobre los motivos que obligan a usar SCRUM, por qué esta metodología se adapta tan bien a nuestras necesidades:

- Punto 1: se diseña la base de datos y se realizan una serie de diagramas (Entidad-Relación, en adelante **ER**, de clases, de casos de uso, de Gantt…), con lo que ya tenemos los planos, las reglas básicas para guiarnos en el desarrollo del proyecto.
- Punto 2: se diseña con HTML y CSS.
- Punto 3: empieza la programación de la interactividad con JavaScript.

Con suerte, hasta aquí habría sido posible no salirse del guión establecido, pero todo cambia al empezar con PHP.

- Punto 4: programar *backend* con PHP.

Si algunos aspectos del proyecto no estaban claros desde el principio y solo se tenía una idea teórica posiblemente inmadura, deberemos realizar saltos hacia atrás en el desarrollo, para lo que la metodología SCRUM es idónea. Ejemplo: la funcionalidad de mostrar el número de mensajes no leídos del foro cuando los haya para el usuario *logeado* requiere de una nueva tabla en la base de datos, lo que nos obliga a volver del punto 4 al punto 1 para hacer los cambios pertinentes. Y en este ejemplo quizá no habría que hacer más cambios, pero habrá otras ocasiones en las que se necesitarán cambios en los puntos 1, 2 y 3.

3. Tecnologías y herramientas utilizadas en el proyecto

Las tecnologías utilizadas serán las siguientes:

- **HTML** y **CSS**, que se dieron en Lenguajes de marcas. Necesarios al ser la base de toda aplicación web.

- **JavaScript**, que se vio en Entorno Cliente. Añadirá interactividad al proyecto.

- **PHP**, que se vio en Entorno Servidor. Será la base del *backend*, del trasfondo de la app. Desde ahí se realizarán todas las funcionalidades avanzadas, interacciones con la base de datos…

- La librería **jQuery**, que se aprendió en Interfaces. Agilizará y facilitará algunas partes de la programación en JavaScript, permitiendo hacer lo planeado escribiendo menos código.

- Una base de datos **MySQL**, aprendida en Bases de datos. Necesaria para la permanencia y toda la estructura de datos.

- **Bootstrap**, que está permitido según la normativa actual 1S2223. Aportará que la app web sea *responsive* y el diseño goce de modernidad, además de reducir nuestro CSS, según estimaciones personales, en un 80%.

Herramientas:

- **XAMPP**, por *Apache* para ejecutar la app y el *PhpMyAdmin* como gestor de BD.

- **Mailhog**, una herramienta para desarrolladores que, con tan solo abrirla y mantenerla minimizada para que moleste lo menos posible, nos permitirá recibir los emails que enviemos offline (a través de nuestro proyecto ubicado en *localhost*) y así poder testearlos. Concretamente, la app web los enviará cuando un usuario se registre para confirmar dicho registro, como medida de seguridad contra bots, y también si un usuario solicita restablecer su contraseña por olvido.

- **Visual Studio Code**, un editor de código que, gracias a sus múltiples extensiones, será una excelente herramienta para desarrollar el proyecto con la mayor comodidad posible.

- Los siguientes navegadores para testear la app web y maximizar su compatibilidad: **Microsoft Edge**, **Mozilla Firefox**, **Google Chrome** y **Opera**.

4. Estimación de recursos y planificación

Diagramas de Gantt (estimación y realidad)

Imagen incluida en los anexos.

5. Análisis del proyecto

Requisitos funcionales

- Los usuarios podrán registrarse con un nombre de usuario y un email, pero solo una vez, ya que ambas cosas son valores únicos en la base de datos.
- Cuando alguien se registre recibirá un email de confirmación para que verifique su cuenta y establezca una contraseña.
- La web contará con un modo oscuro que, con la ayuda de JavaScript, pasará los fondos blancos a negros y el color del texto negro a blanco a fin de no dañar la vista de lo usuarios en zonas con poca luz. Además, usará una cookie para recordar el último modo elegido en ese dispositivo, claro u oscuro.
- La zona de lectura recordará mediante la BD el último capítulo leído por usuario en cada novela (siempre que haya iniciado sesión, claro).
- El botón para acceder al foro y los propios temas del foro en la tabla principal que los muestra contarán con un contador de mensajes no leídos, indicando así de un rápido vistazo si el usuario *logeado* tiene mensajes pendientes de leer (si no está *logeado*, todos contarán como pendientes, señalando así el total de mensajes existentes en la BD).
- Sobre el punto anterior, cada usuario tendrá en la parte inferior de la tabla de temas foro un botón para marcar todos los mensajes como leídos, si así lo desea.
- Los mensajes del foro aparecerán en cada tema desde el más reciente hasta el más antiguo, como en las redes sociales actuales, pero eso sí, solo los 10 primeros. Si se quiere continuar viendo más mensajes antiguos, se deberá hacer *scroll* hacia abajo hasta llegar al fondo de la página, donde pulsando sobre el texto “Cargar más mensajes…” realizará una llamada AJAX y se cargarán otros 10. Esta técnica se denomina *scroll infinito*, otra característica de las redes sociales actuales.
- La app web usa *Bootstrap*, por lo que será *responsive* y siempre se adaptará a los diferentes tipos de dispositivos que existen.
- La posibilidad de escribir mensajes en el foro solo será posible si se ha iniciado sesión.
- Cada vez que se realiza una consulta a la base de datos, se utiliza un bloque try/catch, donde siempre se capturarán los posibles errores que puedan surgir, de forma que se podrán identificar (aparecerá un mensaje con el código de error si éste no se ha previsto) y actuar en consecuencia.

- Dentro de la clase **Utilities**, en el archivo **classes/Utilities.php** se encuentran las funciones **mensajeError()** y **mensajeInfo()**, que se usan constantemente en el resto de clases PHP para mostrar mensajes informativos para cada acción realizada por el usuario, salga bien o mal.
- La app web tendrá una página estática con una pequeña bibliografía sobre el administrador. Otra página estática será una que resuma las tecnologías empleadas para este proyecto.
- Los usuarios registrados tendrán un perfil público que podrá ser visitado por cualquiera, eso sí, siempre que hayan completado el proceso de registro verificando su cuenta y no estén bloqueados.
- Los usuarios logeados tendrán un panel de control desde el que poder cambiar los datos de su perfil, salvo su nombre. Si se desea cambiar el *nick* deberán hablar con el administrador (en la cabecera de la web habrá una serie de redes sociales y contacto).
- Para mayor repercusión de las novelas alojadas en la web, en el foro, los temas que traten sobre las novelas aparecerán de primeros en la parte superior de la tabla, todos con el mismo icono de un libro.
- El administrador tendrá la posibilidad de poner la web en modo de mantenimiento, por lo que solo quedarán accesibles los perfiles de usuario y una página de portada alternativa con un mensaje, imagen y fecha de retorno previamente establecidos. Esto conlleva que en la BD exista una tabla separada de todas las demás, puesto que no tiene nada que ver con los usuarios, novelas, mensajes…

Requisitos no funcionales

Restricciones:

- Todos los formularios (del registro de usuarios, panel de control de usuarios, los de los CRUDs…) tienen validación en el servidor, por lo que no se podrán hacer trampas en la inserción/modificación de datos.
- Todas las consultas a la base de datos se realizarán vía PDO, con consultas preparadas y el uso de la función **bindParam()**, lo que inmunizará la web contra inyecciones SQL.
- La zona de CRUDs para administrar todos los aspectos de la web solo estará visible y disponible para los usuarios cuyo nivel de permisos (columna en la tabla de usuarios

de la BD) sea *Administrador*, además de estar correctamente *logeado*. Si se intenta acceder a ella forzosamente sin cumplir con los requisitos se redirigirá a la portada.

- La sesión se guardará y comprobará mediante cookies y comparación con los datos que existen en la BD.
- Si la web se encuentra en modo de mantenimiento, solo los usuarios de tipo *Administrador* podrán iniciar sesión y acceder a la totalidad del contenido.
- Todos los usuarios tienen en la BD un código y un token, ambos alfanuméricos de más de 60 caracteres, que cambian cada vez que se inicia sesión, cierra sesión y se restablece la contraseña. Todo esto en pos de la seguridad e invalidar el viejo truco de clonar cookies, aparte de los propios códigos enviados por email en su día para verificar tu cuenta y restablecer tu contraseña. Si cambian en estos 3 procesos, se reduce drásticamente el peligro de que se roben y usen con malas intenciones.

Diagrama ER

Imagen incluida en los anexos.

Diagramas de casos de uso

Imagen incluida en los anexos.

Diagrama de clases

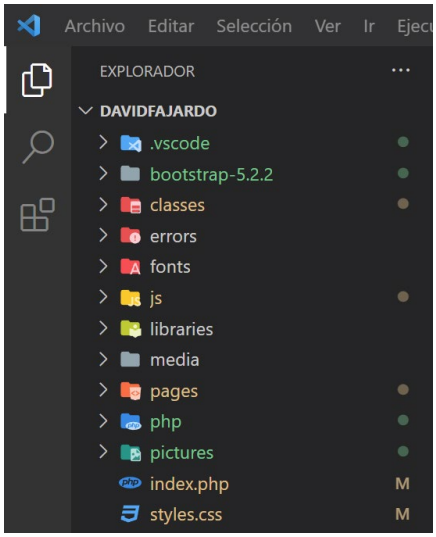
Imagen incluida en los anexos.

6. Diseño del proyecto

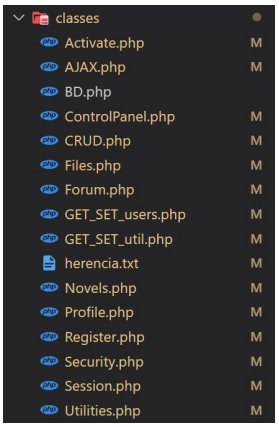
Wireframes

Imágenes incluidas en los anexos.

Primero de todo, mostremos la estructura raíz de archivos y carpetas:



Obviando la primera carpeta generada por el IDE, el resto contienen archivos acorde al nombre de la carpeta contenedora.



Sobre las clases, éstas están contenidas dentro de la carpeta **classes**, y cada una tiene un nombre descriptivo y hereda de otra, como se puede ver en el diagrama de clases en los anexos.

A continuación, algo importante que está en todas las páginas (debido a que se encuentra en el *header*, elemento presente siempre) es el menú de inicio de sesión que, sin estar *logeado*, muestra un botón para eso, pero si inicias sesión pasa a mostrar enlaces en forma de botones a tu perfil, panel de control de control de usuario y un botón para cerrar sesión. El inicio y cierre de sesión se realiza mediante la llamada a estas 2 funciones:

[Imagen de la función iniciarSesion() incluida en los anexos debido a su extenso tamaño.]

Función **cerrarSesion()** (presente en el archivo **Session.php** dentro de la carpeta **classes**):

```php
function cerrarSesion($force = false) {
    if (isset($_POST['cerrar']) || $force) {
        $this->cambiarToken();
        $this->cambiarCodigo();
        $this->borrarCookie("cod_usuario");
        $this->borrarCookie("usuario");
        $this->borrarCookie("token");
        header("location: ".$this->evitarXSS($_SERVER['PHP_SELF']));
    }
}
```

Esta función contiene el parámetro opcional **$force** simplemente para cuando la web se pone en modo de mantenimiento y por lo tanto solo queda disponible para administradores, así se le cierra la sesión a la fuerza a los demás usuarios si entran en la app, y como está programado, solo podrán acceder a sus perfiles y a la portada provisional.

Luego, al igual que la de iniciar sesión, ejecuta otras 2 funciones de seguridad, **cambiarToken()** y **cambiarCodigo()**, que realizan un cambio de estas 2 columnas de la tabla de usuarios de la BD, alterando estos 2 códigos alfanuméricos de más de 60 caracteres.

[Imágenes incluidas en el anexo]

Previamente se vio en la función de cierre de sesión que ejecuta la función **borrarCookie()**. La de iniciar sesión usa la contraria, **crearCookie()**. Éstas contienen el siguiente código:

```php
function crearCookie($nombre, $valor) {
    setcookie($nombre, $valor, time()+31536000, "/", "localhost", true, true);
}

function borrarCookie($nombre) {
    setcookie($nombre, "", time()-1, "/", "localhost", true, true);
}
```

No son especialmente largas, por lo que no ahorran escribir demasiado, pero ya que la mayoría de los parámetros se repetían al crear cookies, se optó por crear esta función para mayor comodidad.

Continuando con el tema de la seguridad, se han creado otra serie de funciones usadas constantemente para diversas operaciones:

```php
function encriptar($input) {
    return substr(password_hash($input, PASSWORD_DEFAULT), 7, 53);
}
```

encriptar() recoge el parámetro pasado y primero lo encripta con la función de PHP password_hash() y luego le quita los primeros 7 caracteres. ¿Por qué? Porque eso es la *salt*, el nivel de encriptación utilizado, lo cual podría dar pistas a

posibles crackers. Luego a la hora de comprobar estos datos ya se añade la salt en cada caso, pero así los datos están más seguros en la BD.

Otra función sencilla pero potente: **evitarXSS()**. Mediante el método de PHP htmlspecialchars() se eliminan del parámetro pasado los caracteres del ampersan (&), comillas dobles(“), simples (‘) y los símbolos menor y mayor que (<>). De esta forma se previenen los ataques XSS, que intentan inyectar y ejecutar código malicioso para dañar el servidor. Esta función se usa al procesar todos los parámetros que introduce un usuario en formularios.

```php
function evitarXSS($input) {
    if (is_array($input))
        $input = $input[0];

    return htmlspecialchars($input, ENT_QUOTES);
}
```

Función comprobarLogin()

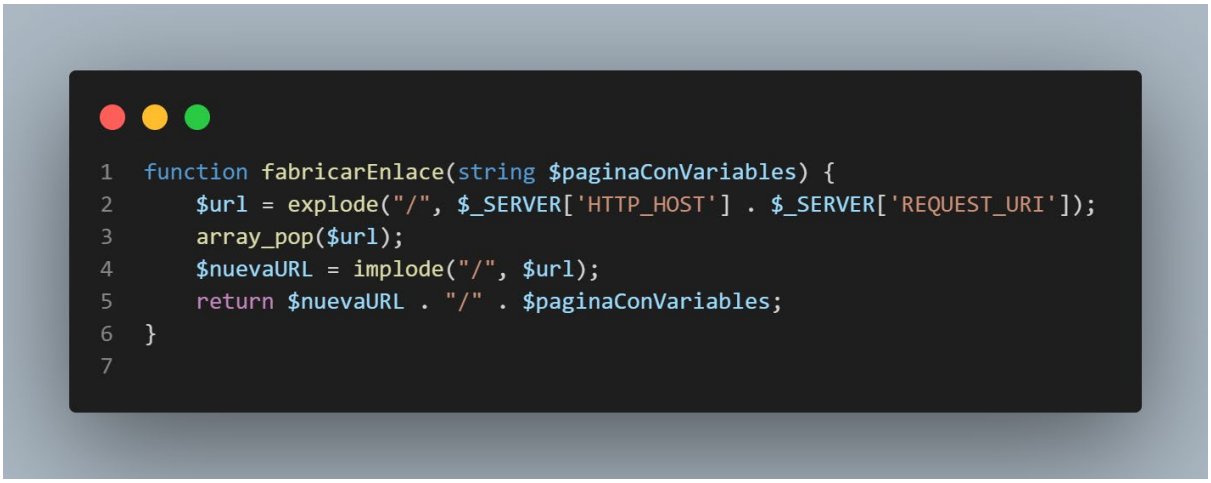
[Incluida en el anexo]

Función comprobarPermisos()

[Incluida en el anexo]

Otra clase útil y de la cual heredan todas las demás es la clase **Utilities**, que contiene funciones como las siguientes:

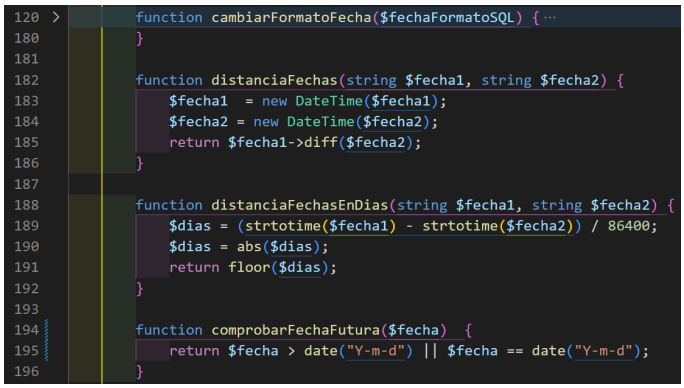
Función **fabricarEnlace()**



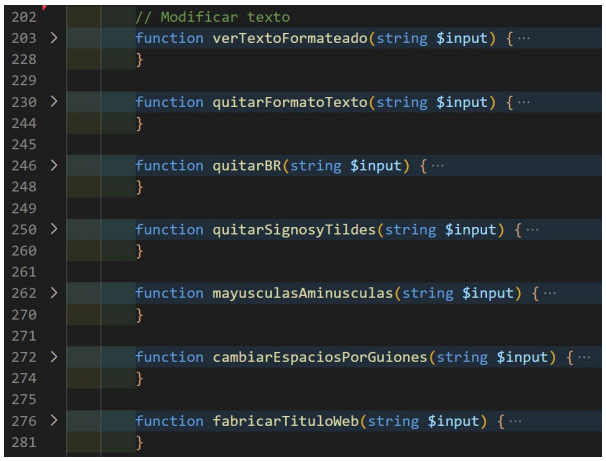
```php
function fabricarEnlace(string $paginaConVariables) {
    $url = explode("/", $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI']);
    array_pop($url);
    $nuevaURL = implode("/", $url);
    return $nuevaURL . "/" . $paginaConVariables;
}
```

Este método es necesario por precaución a la hora de ejecutar la web en distintas plataformas; es posible que la URL cambie ligeramente de aspecto a la hora de enviar emails de verificación de cuenta o restablecimiento de contraseña, así que para asegurarse siempre que los enlaces funcionen a la hora de pasar parámetros, esta función recibe el final de la URL que tú le mandes (una página con variables, p.e. activate.php?email=usuario@servidor.com&codigo=abc123.), convierte la URL actual en un array que se crea y divide por barras laterales (/), elimina el último elemento (de forma que literalmente nos queda la ubicación de la página actual, pero sin la página) y vuelve a pasar el array a un string. Luego lo devuelve seguido de una barra lateral y el parámetro pasado.

No se analizarán en profundidad ya que son demasiadas líneas, pero conviene señalar la importancia de las funciones para tratamiento de fechas (para mostrarlas con mejor aspecto en mensajes, verificar que no se hagan trampas al introducir fechas… etc).



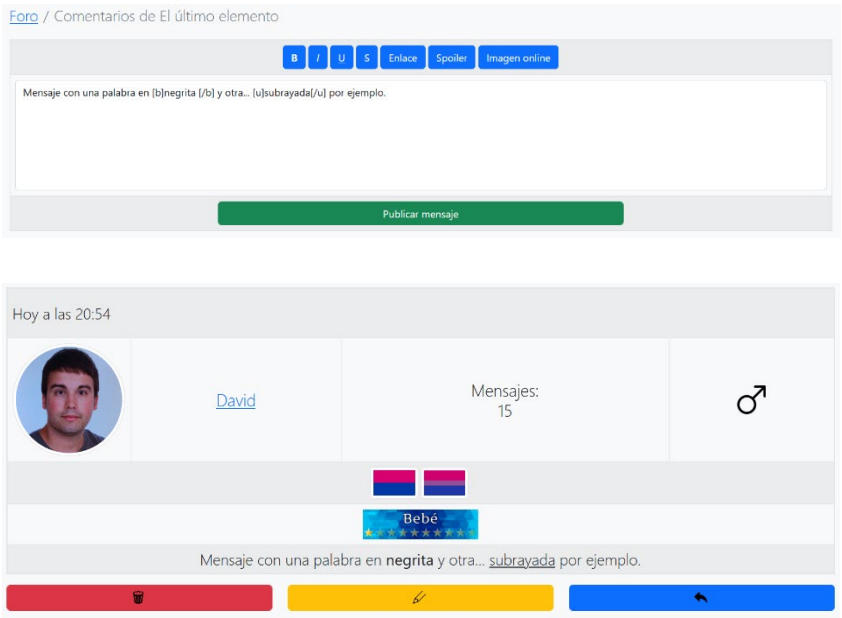
De nuevo, las siguientes funciones son demasiadas y sus nombres ya las describen, pero no por ello son menos importantes: permiten, entre otras cosas, que el foro permita formatear nuestros mensajes con negrita, cursiva, subrayado… Este aspecto sí se detallará a continuación debido a su complejidad, ya que conlleva el uso de estas funciones de PHP, pero también de JavaScript.



Las 2 primeras convierten etiquetas especiales (se explicarán a continuación), mientras que las otras ayudan a formatear nombres para mensajes, inserción en la BD…

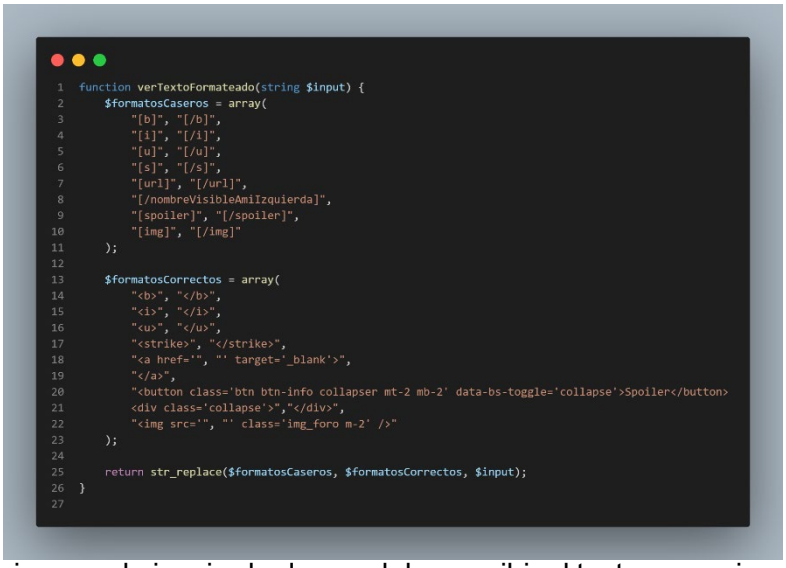
La última, **fabricarTituloWeb()**, coge el string que le pases y le aplica las 3 funciones anteriores a esa: elimina signos y cambia vocales acentuadas, pasa las mayúsculas a minúsculas y cambia los espacios entre palabras por guiones normales (-).

Ahora vamos con la inserción/edición de mensajes en el foro



El cuadro de texto posee botones para usar **negrita**, *cursiva*, subrayado, ~~tachado~~, insertar enlaces, spoilers e imágenes online. Se selecciona el texto a formatear y se pulsa el botón necesario, así se añaden a ambos lados de la(s) palabra(s) las etiquetas necesarias. Esto se realiza mediante una función de JavaScript, cuya imagen se podrá ver en el anexo.

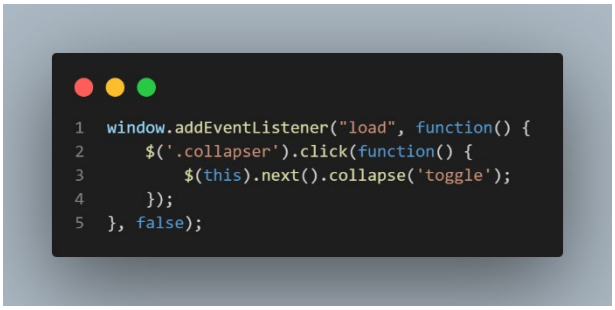
No obstante, sí podemos ver aquí la función **verTextoFormateado()**, que recoge el string pasado (en este caso los mensajes del foro) y los devuelve sustituyendo estas "etiquetas caseras" por las necesarias en HTML para que de esta forma el formateo de texto tenga éxito:



```php
function verTextoFormateado(string $input) {
    $formatosCaseros = array(
        "[b]", "[/b]",
        "[i]", "[/i]",
        "[u]", "[/u]",
        "[s]", "[/s]",
        "[url]", "[/url]",
        "[/nombreVisibleAmiIzquierda]",
        "[spoiler]", "[/spoiler]",
        "[img]", "[/img]"
    );

    $formatosCorrectos = array(
        "<b>", "</b>",
        "<i>", "</i>",
        "<u>", "</u>",
        "<strike>", "</strike>",
        "<a href='", "' target='_blank'>",
        "</a>",
        "<button class='btn btn-info collapser mt-2 mb-2' data-bs-toggle='collapse'>Spoiler</button>
        <div class='collapse'>","</div>",
        "<img src='", "' class='img_foro m-2' />"
    );

    return str_replace($formatosCaseros, $formatosCorrectos, $input);
}
```

Para el aspecto del texto es fácil, etiqueta al principio, al final y listo, pero para los enlaces hubo que esforzarse más, ya que un enlace lleva código a su izquierda, derecha, pero entre eso y la etiqueta de cierre (</a) debe llevar el texto visible del enlace, así que hubo que crear la etiqueta [/nombreVisibleAmiIzquierda], una forma *sutil* de decirle al usuario que a la izquierda de eso debe escribir el texto que quiera que se vea luego. Esto vendría a ser la etiqueta de cierre del enlace (</a).

Otro elemento medio complejo son los spoilers, por suerte Bootystrap ya viene preparado con sus funciones JavaScript para hacerlos funcionales escribiendo muy poco. ¿Problema? Se rigen por IDs, y no hay una forma fiable de crear IDs únicos, ni sacando números aleatorios tendríamos la certeza de que ninguno se repita, así que se introdujo a mayores el siguiente script en JS:



Es simple: si se hace clic en un elemento con clase **collapser**, el siguiente elemento se despliega/repliega. Si nos fijamos en la captura anterior se verá que se crea un botón con clase **collapser** que ataca a la clase **collapse**. Y a su vez una div con clase **collapse**.

Por último, algo importante de esta clase fueron las 2 funciones creadas con el objetivo de buscar fallos cuando algo no funciona:

Una ejecuta un alert() de JS con un mensaje en mayúsculas, así sabremos si el código llegó a la parte donde se llame a esta función. La otra muestra los datos que tú le pases de forma ordenada y muy legible, así ayuda a encontrar más rápido lo que se busque.

Vamos con una de las 2 funcionalidades principales de la app web: **leer novelas**. No se mostrará ni aquí ni el anexo la función de leer novelas por ocupar demasiado en vertical, pero las de cargar y guardar el progreso de lectura de los usuarios *logeados* sí:

Función **GET_ProgresoLecturaUsuario()**

```php
function GET_ProgresoLecturaUsuario() {
    if ($this->comprobarLogin())
        $usuario = $this->evitarXSS($_COOKIE['usuario']);
    else
        return 1;

    $tituloWeb = $this->evitarXSS($_GET['tituloWeb']);

    try {
        $this->__construct();
        $query = $this->db->prepare("SELECT `".$tituloWeb."` FROM novelas_leidas WHERE usuario = :usuario");
        $query->bindParam(":usuario", $usuario, PDO::PARAM_STR);
        $query->execute();
        $capitulo = $query->fetch();
        return $capitulo[0];
    } catch (Exception $e) {
        exit("Error: " . $e->GetMessage());
    } finally {
        $this->db = null;
    }
}
```

Lo primero que hace es verificar si el usuario está *logeado*, y si es el caso guarda su nick. Si no, devuelve un 1 y fin de la función (es decir, si no hay *login*, no se pude recordar la posición de lectura, así que todas las novelas se abrirán siempre por defecto en el capítulo 1).

Luego recoge de la URL el *título web* de la novela, que no es otra cosa que el título de la novela pasada por la anterior función **fabricarTituloWeb()**, que elimina espacios, signos, tildes y mayúsculas. Luego se conecta a la BD, recoge el número de capítulo del usuario en cuestión de la novela en cuestión y lo devuelve.

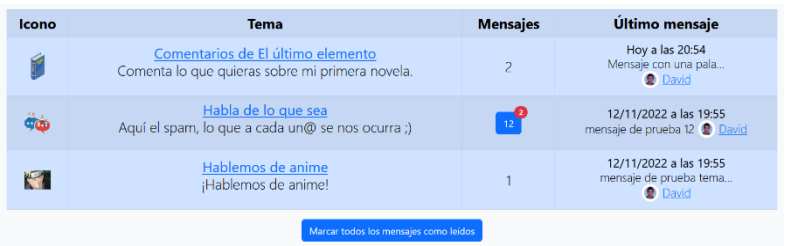
Análogamente, la función **SET_ProgresoLecturaUsuario()** se ejecuta en la lectura de novelas cada vez que se carga la página (solo si hay *login*, recordemos). Detecta el número del capítulo de la URL y lo envía a esta función como parámetro. Va en la URL con la variable cap (de capítulo) si es mayor que el 1 (si aún van por el capítulo 1, para qué guardar nada):

```php
if ($this->comprobarLogin() && isset($_GET['cap']))
    $this->SET_ProgresoLecturaUsuario($this->evitarXSS($_GET['cap']));
```

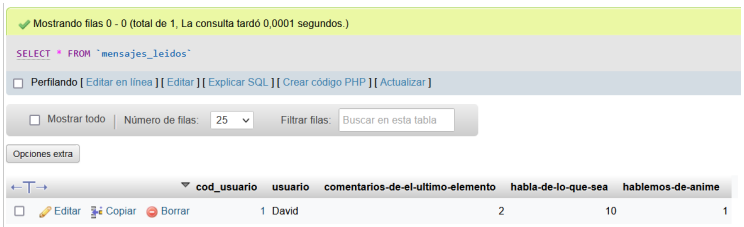
Se vuelven a recoger usuario y título de la novela y luego se actualiza la tabla novelas_leidas (la misma de antes) con el nuevo capítulo al que se ha accedido.

```php
function SET_ProgresoLecturaUsuario(int $nuevoCapitulo) {
    $tituloWeb  = $this->evitarXSS($_GET['tituloWeb']);
    $usuario    = $this->evitarXSS($_COOKIE['usuario']);

    try {
        $this->__construct();
        $query = $this->db->prepare(
            "UPDATE novelas_leidas SET `".$tituloWeb."` = ".$nuevoCapitulo."
            WHERE usuario = :usuario"
        );
        $query->bindParam(":usuario",   $usuario,   PDO::PARAM_STR);
        $query->execute();
    } catch (Exception $e) {
        exit("Error: " . $e->GetMessage());
    } finally {
        $this->db = null;
    }
}
```

Seguidamente vamos con la otra funcionalidad importante de esta app web: **el foro**. Para empezar, antes de entrar a cualquier tema existente, podemos apreciar que, si tenemos mensajes sin leer, se muestran, además de un botón de "marcar todos como leídos", como en otros foros. Usando además la misma filosofía que la memoria de progreso de lectura, una tabla en la BD llamada mensajes_leidos. A continuación, se mostrarán imágenes para ver el proceso de ver mensajes no leídos y que la BD se actualice:

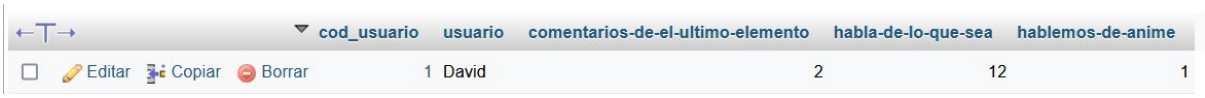


El segundo tema muestra que tiene un total de 12 mensajes, pero que 2 son nuevos para nosotros. Esto puede verse también reflejado en la tabla mensajes_leidos de la BD:



Como podemos ver, la cuarta columna corresponde a ese tema del foro, y aunque posee 12 mensajes, marca 10, que son los que hemos leído. Tenemos 2 opciones, o entramos en el tema o pulsamos el botón de marcar todos como leídos. Haremos eso, y observaremos como la página se recarga y ya no aparecerán contadores:

Y lógicamente, la tabla de la BD ya actualizó el 10 al 12:

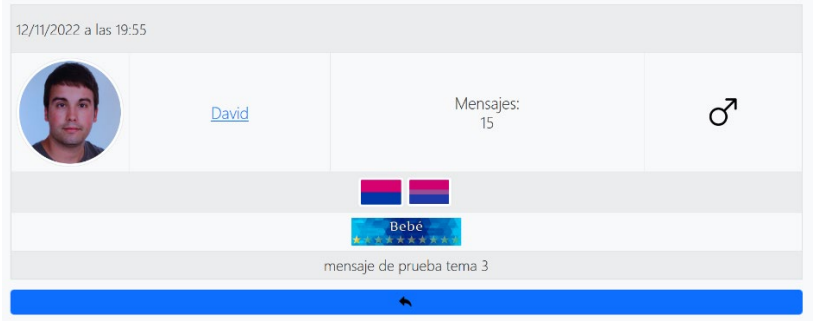


Ya que se ha mostrado lo que hace esta función podemos mostrar su código para verla por dentro:

El botón de antes envía la petición por POST, aquí se recoge y ejecuta una consulta para conseguir todos los temas existentes en el foro y los mensajes de cada uno. Luego las pasa por un bucle para ejecutar un UPDATE y actualizarle al usuario su número de mensajes leídos en todos los temas.

```php
function marcarTodosLeidos() {
    if (isset($_POST['todos_leidos'])) {
        $cod_usuario = $this->evitarXSS($_COOKIE['cod_usuario']);
        $this->__construct();
        try {
            $query = $this->db->prepare("SELECT tituloWeb, mensajes FROM temas ORDER BY cod_tema");
            $query->execute();

            foreach ($query as $temas) {
                $query = $this->db->prepare(
                    "UPDATE mensajes_leidos SET
                    `".$temas['tituloWeb']."` = ".$temas['mensajes']."
                    WHERE cod_usuario = :cod_usuario"
                );
                $query->bindParam(":cod_usuario", $cod_usuario, PDO::PARAM_INT);
                $query->execute();
            }
        } catch (Exception $e) {
            exit("Error: " . $e->GetMessage());
        } finally {
            $this->db = null;
        }
        header("location: ../pages/foro.php");
    }
}
```

Una vez dentro de un tema del foro observamos varias cosas (si estamos *logeados*, claro): el cuadro de texto para publicar un nuevo mensaje, el botón de Responder en los mensajes ajenos, ese y los de Editar y Borrar en los nuestros… En las siguientes imágenes podemos verlo desde la perspectiva de un usuario normal, sin ser Administrador.



El usuario externo puede responder a mi mensaje si así lo desea, y si es así se hará un *scroll* vertical hacia arriba, al cuadro de texto, pero éste ahora tendrá un *placeholder* que pondrá Respondiendo… de forma que ese cuadro ahora tiene por dentro el ID del mensaje al que se responde. Al darle a publicar podremos ver como el nuevo mensaje tiene en la parte superior una banda gris que otorga datos sobre el mensaje al que se responde: autor (que además es un enlace a su perfil), fecha, una flecha descendente en pos de la estética y los 50 primeros caracteres de ese mensaje.

El código de esta parte concreta para mostrar esta información en gris sobre el mensaje x puede apreciarse en el siguiente código:

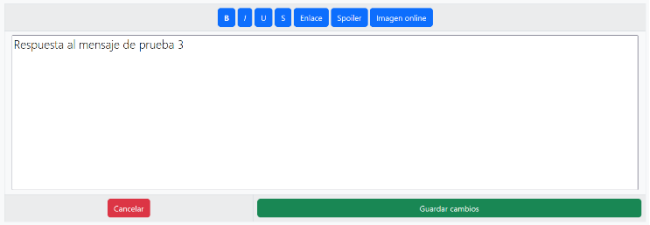
```
// Si este mensaje está respondiendo a otro
if ($datos['respondiendoA'] > 0) {
    print("
        <tr class='bg-secondary'>
            <td colspan='5' class='text-light fw-bold'>
                Respondiendo a
                <a href='../pages/perfil.php?usuario=".$this->mensajesConRespuesta($datos['respondiendoA'])['usuario']."' target='_blank'
                class='text-light me-2'> ".$this->mensajesConRespuesta($datos['respondiendoA'])['usuario']."</a>-
                ".$this->cambiarFormatoFecha(substr($datos['fecha'], 0, 10))."
                <img src='../pictures/foro/emisor.png' class='width20 ms-1' /><br />

                ".substr($this->quitarBR($this->quitarFormatoTexto($this->mensajesConRespuesta($datos['respondiendoA'])['contenido'])), 0, 50)."...
            </td>
        </tr>
    ");
}
```

Si lo que queremos es editar un mensaje, el cuadro será igual al de insertar mensajes con un par de botones a mayores para guardar o cancelar, claro. En el caso del borrado, se pedirá una confirmación hecha con un simple evento onclick de JS por precaución (imágenes disponibles en la próxima página).

Otra característica de nivel de este foro es el *scroll infinito*, nada de paginación; así se parecerá más a las redes sociales modernas. Por defecto los temas del foro cargan solo 10 mensajes. Si tiene más, debe hacerse *scroll* hasta abajo del todo y hacer clic en el texto Cargar más mensajes… Y se cargarán 10 más, o los que existan a mayores. En los anexos podrán verse imágenes del funcionamiento y del código (el cual son un archivo JS con una llamada AJAX y una función PHP, pero ésta útima no se enseñará debido a que la primera ya es muy grande y la segunda es una simple consulta a la BD con un simple **LIMIT ".$mensajesPresentes.", 10**). De esta forma cargará 10 mensajes (segundo parámetro del LIMIT) a partir del primero que toque cargar luego de los existentes (10, 20, 30…; primer parámetro pasado por AJAX a esta función).

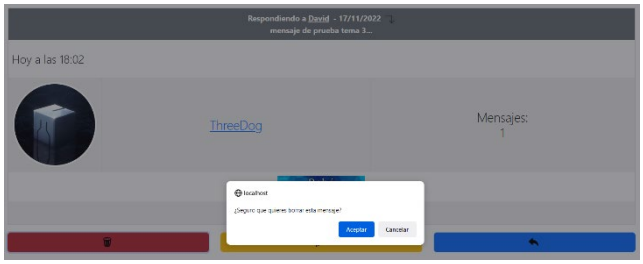
Otro elemento que tendremos en toda la app web son una serie de botones en la parte inferior derecha, uno para hacer scroll vertical hacia la parte superior (aparecerá solo si hemos hecho *scroll* hacia abajo gracia s JS) y otro para activar el modo oscuro. Dicho modo cambiará el color de fondo blanco por negro y el texto negro por blanco. Además tiene memoria gracias

a una cookie, por lo que aunque salgas de la web recordará tu elección de colores cuando vuelvas. Esta característica es importante siempre para la salud visual de los usuarios, aparte de que el apartado de lectura de las novelas está pensado para pasar cuanto más tiempo mejor frente a la pantalla. Si la web se encuentra en modo claro, aparecerá la imagen de ua luna. En caso contrario, la de un sol. Imágenes disponibles en el anexo.



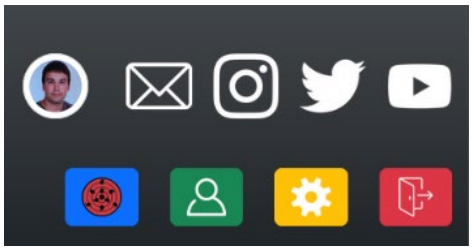
Edición de un mensaje a la izquierda.

Confirmación de la eliminación de un mensaje abajo.



Una aclaración para la imagen del anexo del código del scroll infinito: sobre la estructura HTML de los mensajes, cada mensaje es una tabla, seguida de un formulario con los botones de responder/editar/borrar y por último un *span* vacío con la clase **FIN_mensaje**. Este span es necesario para que funcione bien el *scroll* infinito: cada mensaje tiene que estar bien localizable en el código, sobre todo dónde termina. ¿Y no se podía poner esta clase al formulario de los botones, que es lo que hay al final? No, porque no siempre existe ese formulario: si no has iniciado sesión puedes entrar en el foro aunque no haya opciones de inserción ni botones, ergo no existiría esa localización. La llamada AJAX cargará esos 10 nuevos mensajes (o los que resten) a partir del último elemento con la clase FIN_mensaje.

Como última característica de las más importantes de esta app web tenemos el panel de Administración, solo accesible para administradores, valga la redundancia. El acceso se realiza a través de un botón con un círculo rojo múltiple con comillas visible en el *header* solo cuando el usuario cumple la condición. Si por un casual un usuario malintencionado conociera la URL de la página de administración e intentara acceder, se le expulsaría hacia la portada.



Al igual que en el foro, donde también hay seguridad contra la suplantación de identidad: si alguien consiguiera mandar por POST las variables y parámetros necesarios para editar o borrar un mensaje que no es suyo, la petición se anularía y se mostraría un mensaje informativo. Una muestra del código de la función para borrar mensajes del foro:

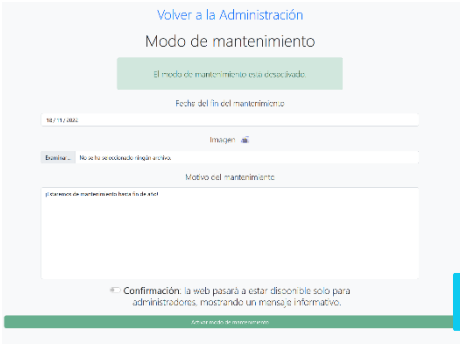
```php
function borrarMensaje() {
    if (isset($_POST['borrarMensaje'])) {

        // Si el mensaje es tuyo, se borra
        if ($this->comprobarPropiedadMensaje($_POST['cod_mensaje'])) {···
        }

        // Si eres un admin, se borra también, pero se le resta
        // un mensaje al usuario en cuestión y no a ti, obviamente
        else if ($this->comprobarPermisos("Administrador")) {···
        }

        else
            $this->mensajeError("Error de propiedad para borrar el mensaje.");
    }
}
```

El código de la función **ComprobarPropiedad Mensaje()** podrá verse en los anexos.

La tabla de administración de usuarios contendrá botones de edición y eliminación de cada uno (salvo la eliminación de tu propio perfil, donde el botón de editar estará desactivado). No se añade un formulario para añadir usuarios como es habitual en los CRUDs porque si alguien quiere registrarse, puede hacerlo fácilmente; es innecesario que el administrador añada a nadie. Y las opciones de edición solo incluyen el nombre del usuario y sus *strikes* o faltas de orden. Pueden ser entre 0 y 3. Si se establecen 3, ese usuario pasará a estar bloqueado, ya no podrá iniciar sesión y su perfil dejará de estar disponible, mostrando simplemente un mensaje de que ese usuario está bloqueado. A continuación, podemos ver imágenes del proceso: tabla de usuarios, edición del usuario ThreeDog cambiándole los *strikes* de 0 a 3, y yendo a su perfil: no está accesible. Además, fijémonos en el nombre de cada usuario en la tabla: si el usuario tiene 0 strikes, aparecerá sobre un fondo verde. Si tiene 1, amarillo; si son 2, rojo y si ya son 3, aparecerá una señal de prohibición. También puede observarse que el usuario que ha entrado a esta zona no puede eliminarse a sí mismo; el botón está desactivado.

A continuación, tenemos la sección para activar/desactivar el modo de mantenimiento, lo que nos permitirá dejar inaccesible la web (a excepción de los administradores) si surge alguna emergencia o simplemente queremos hacerle un mantenimiento, pruebas… Podemos verlo a continuación, tanto la configuración como la portada provisional, a través de un simple

formulario estableceremos una fecha de fin del mantenimiento, una imagen y un motivo, datos que luego se mostrarán en esta portada:



En esta pantalla pasará a predominar el color rojo en vez del verde para indicar que la web está de reformas. Y mientras esté así, aparecerá un banner rojo en la parte superior permanente en todas las páginas salvo la portada provisional con los datos anteriormente establecidos.

Las otras 2 secciones de gestión de novelas y del foro son iguales en esencia: CRUDs para insertar, editar y borrar, por lo que se enseñarán meras imágenes de las tablas en la página siguiente. Al contrario que en los formularios tradicionales, donde si se sube una imagen se incluye un *input* de tipo *file* y listo, junto con todos los demás campos, aquí para las portadas, los capítulos, las sinopsis de las novelas y los iconos de los temas del foro, son formularios aparte, un simple par de botones para subir (el *input* tipo *file*) y otro para aplicar tipo *submit*. En la siguiente página podremos ver ambos CRUDs.

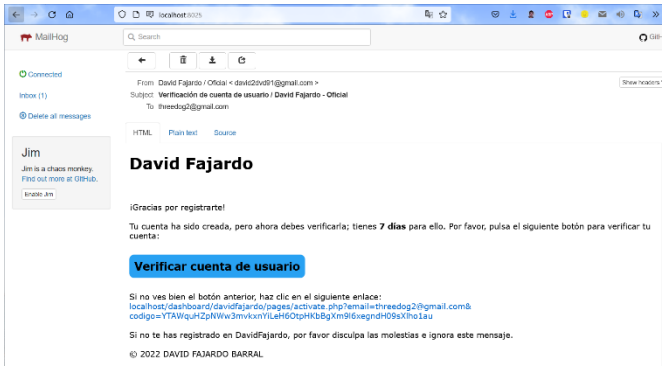
CRUD de las novelas. Nótese que los botones amarillos de Cambiar… la portada no son *input* tipo *file*, sino que tienen un **getElementById().click()** hacia el verdadero *input* tipo *file*, que está oculto. Así al clicar sobre él se clica el de verdad. Se hizo así porque claro, los *input* tipo *file* ocupan una barbaridad horizontalmente.

En el CRUD del foro nada especial, el mismo truco para los iconos de los temas.

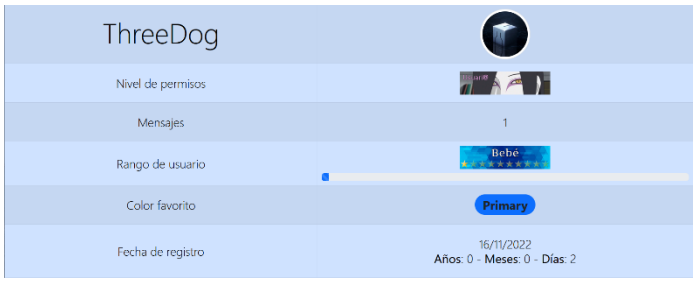
Sobre el **registro de usuarios**, el enlace solo será visible si no se está *logeado*, aunque se podrá acceder igualmente sin problemas si se introduce manualmente la dirección; no se vio la necesidad de proteger este proceso (el aplicativo está preparado: tras registrarse, se envía un email para que el usuario verifique su cuenta. Si en 7 días no se hace, esa cuenta de usuario será eliminada). Esta eliminación no es automática debido a una carencia de funcionalidad en la vida real: se pensó en usar un evento en el gestor de bases de datos, pero si se sube la app a un hosting, nunca te darán el permiso SUPER para el gestor de bases de datos ya que tendrías acceso a todo su servidor, y dicho permiso es necesario para usar los eventos del gestor de BD. Aún así esta función de eliminación se ejecuta en 2 ocasiones: cuando alguien accede al registro de usuarios y cuando el administrador accede al CRUD de usuarios. Cada nuevo registro aportará la limpieza de la BD, y cuando el administrador quiera consultar la tabla de usuarios de la web, directamente no llegará a ver nunca los que deban ser eliminados; tabla más limpia de filas inútiles. Esta función podrá verse en los anexos.

Y he aquí el registro: un formulario de lo más simple, pide usuario, email y 3 respuestas de seguridad para cuando se olvide la contraseña de acceso. Además los cuadros del usuario y el email hacen una llamada AJAX a cada tecla pulsada para comprobar que estén disponibles esos datos (el nombre de usuario es clave primaria de la tabla de la BD y el email es UNIQUE, es decir, candidata a clave primaria; tampoco se puede repetir). Es cierto que si se usa el autocompletado del navegador la llamada no funciona, pero es igual; a la hora de ejecutarse el INSERT INTO saltará una excepción cogida mediante un bloque try/catch, el proceso se cancelará y saldrá un mensaje informativo.

Gracias a la aplicación de pruebas Mailhog se ha podido testear los emails cómodamente, ya que se envían localmente gracias a esta herramienta. Funcionan y llevan código CSS en línea (única opción o no se aplicaría) para ser mejores estéticamente. A continuación, se ofrecen capturas del email y el proceso de verificación; cada usuario lleva en la BD un código de 63 dígitos que ya se mencionó anteriormente, y dicho código y el email se usan como parámetro GET para enviárselos al usuario por email y luego activar y establecer una nueva contraseña para su cuenta. La función utilizada para mandar emails vía PHP se verá en el anexo.



Vamos ahora con los **perfiles de usuario**. Son públicos (pueden verse aunque no se esté *logeado*) y enseñan los datos que el usuario haya querido aportar en su panel de control de usuario (apartado que veremos después de este). Por defecto, un usuario al registrarse tiene el avatar de un cubo, se le asigna el nivel de permisos "Usuario@" y como color favorito un azul oscuro (concretamente el color *Primary* de Bootstrap). En base a su número de mensajes aparecerá un banner del rango establecido mediante las cifras programadas el código y justo debajo una barra de progreso sobre el mismo hasta el siguiente rango; dicha barra obtiene su valor marcado mediante una serie de operaciones matemáticas. El parámetro que marca cómo de llena estará es su *width*, y debe ser un porcentaje, así que se tiene que saber el número de mensajes que se tiene, los necesarios para el próximo rango y el porcentaje que representa la primera cifra sobre la segunda (luego veremos el código de ese cálculo). Aquí podemos ver un perfil de usuario por defecto y otro ya con algunos datos opcionales cubiertos:



```
<div class='progress mt-2'>
    <div class='progress-bar progress-bar-striped progress-bar-animated bg-".$datos['color']."' role='progressbar'
        style='width: ".$this->GET_PorcentajeSobreCifra($this->GET_RangoUsuario($datos['cod_usuario'])['mensajes'],
        $this->GET_RangoUsuario($datos['cod_usuario'])['mensajesNecesarios']).'%;'
        title='Progreso hasta el siguiente rango'
        data-bs-toggle='popover' data-bs-trigger='hover focus' data-bs-content='" .
        $this->GET_RangoUsuario($datos['cod_usuario'])['mensajes'] . "/" .
        $this->GET_RangoUsuario($datos['cod_usuario'])['mensajesNecesarios'] . " mensajes'>
    </div>
</div>
```

Este código genera esa barra de progreso. Debajo podemos ver la función **GET_PorcentajeSobre Cifra()**

```
function GET_PorcentajeSobreCifra(int $cifra1, int $cifra2) {
    // Fórmula de calcular qué porcentaje es una cifra de otra: (a/b) x 100
    return round(($cifra1 / $cifra2) * 100);
}
```

Sobre el **panel de control** de cada usuario, lógicamente hay que hacer *login* y clicar sobre la rueda dentada del botón amarillo vista anteriormente.

Allí podremos cambiar nuestros datos, añadir otros… salvo nuestro nombre. Por costumbre vista en otras webs, se ha decidido que solo los administradores puedan cambiar nombres de usuarios (recordemos que la web posee un enlace *mailto* para que cualquiera pueda ponerse en contacto). Por supuesto dicho cambio es comprobado para evitar duplicidades.

Todos los cambios de todas las zonas son comprobados siempre como se dijo. Aquí además a la hora de subir un avatar se comprueba que de verdad sea una imagen (para evitar archivos maliciosos), que no pese más de 2 MB, el nombre no supere los 100 caracteres… etc. Luego se comprueba también que tu fecha de nacimiento no sea una fecha futura con respecto a hoy, que tu altura (si ya la habías establecido y quieres cambiarla) no se cambie a una cifra menor…

Y cómo no, entre otras cosas, el color favorito. He aquí el toque de personalización personal para cada usuario: como se usan los colores de Bootstrap, luego pueden usarse con sus clases para darle color a tablas, botones… Es decir, se fabrica una función que simplemente haga un SELECT color del usuario *logeado*, lo devuelve y se pone una llamada a la función al escribir la clase con el color elegido para el elemento HTML.

Como podemos ver no se introduce la edad sino la fecha de nacimiento; la edad visible en los perfiles se calcula mediante la función **GET_Edad()**:

```php
function GET_Edad(string $fecha_nac) {
    if (!empty($fecha_nac)) {
        $dia_actual     = date("d");
        $mes_actual     = date("m");
        $anho_actual    = date("Y");
        $dia_nac        = substr ($fecha_nac, 8, 2);
        $mes_nac        = substr ($fecha_nac, 5, 2);
        $anho_nac       = substr ($fecha_nac, 0, 4);

        if (($mes_nac == $mes_actual) && ($dia_nac > $dia_actual))
            $anho_actual--;

        if ($mes_nac > $mes_actual)
            $anho_actual--;

        return ($anho_actual-$anho_nac);
    }
    else
        return "0";
}

```

Para terminar, sería interesante ver el código del contador inverso que se muestra en la portada provisional cunado se activa el modo de mantenimiento, y podrá verse en los anexos.

7. Despliegue y pruebas

CU-01	INICIO DE SESIÓN	
Versión	1.0 (17/11/2022)	
Autores	David Fajardo Barral	
Descripción	El sistema permitirá interactuar con el foro, recordar posiciones de lectura de novelas, modificar sus datos de usuario...	
Precondición	El usuario está registrado	
Secuencia normal	Paso	Acción
	1	La app web pide usuario y contraseña
	2	El usuario introduce ambos datos
	3	Se comprueba la BD y ifnaliza el caso de uso
Postcondición	El usuario está dentro de la aplicación	
Excepciones	Paso	Acción
	3	Si los datos son incorrectos, la app web no concede acceso, y finaliza el caso de uso.
	3	Si el usuario no está verificado, la app web no concede acceso, y finaliza el caso de uso.
	3	Si el usuario está bloqueado, la app web no concede acceso, y finaliza el caso de uso.
	3	Si la web tiene el modo de mantenimiento activado y el usuario no es un administrador, la app web no concede acceso, y finaliza el caso de uso.

CU-01	ACCESO A LA ADMINISTRACIÓN	
Versión	1.0 (18/11/2022)	
Autores	David Fajardo Barral	
Descripción	Acceso a la zona de administración	
Precondición	El usuario está registrado y es Administrador	
Secuencia normal	Paso	Acción
	1	Haces clic en el enlace a la zona de administración
	2	Se comprueba que seas un administrador
	3	Accedes a la zona de administración
Postcondición	El usuario está dentro de la zona de administración	
Excepciones	Paso	Acción
	2	Si el usuario no es un administrador, el enlace a la zona de administración no será visible.
	2	Si se conoce la URL y se intenta entrar a la fuerza sin ser Administrador, se redirigirá a la portada de la web.

CU-01	MODO DE MANTENIMIENTO	
Versión	1.0 (17/11/2022)	
Autores	David Fajardo Barral	
Descripción	El sistema pasará a estar disponible solo para administradores, mostrando una portada provisional.	
Precondición	El usuario es un Administrador y ha iniciado sesión	
Secuencia normal	Paso	Acción
	1	Se accede a la zona de administración
	2	Se activa el modo de mantenimiento
	3	Se aplican los cambios a la BD
	4	Se aplican los cambios a la web según su programación
Postcondición	La web solo queda disponible para administradores	

CU-01	MEMORIA EN LECTURA DE NOVELAS	
Versión	1.0 (18/11/2022)	
Autores	David Fajardo Barral	
Descripción	La BD se actualizará guardando el último capítulo visitado por cada novela de cada usuario para reanudar lecturas	
Precondición	El usuario se ha registrado y ha iniciado sesión	
Secuencia normal	Paso	Acción
	1	Se accede a la zona de lectura de la novela deseada
	2	Si se cambia de capítulo, se guardará en la BD
	3	Se sale de la zona de lectura
	4	Al volver a la zona de lectura, se redigirá directamente al último capítulo visitado

8. Contexto laboral

El aplicativo trata de una zona de lectura de novelas y, aparte, un foro; todo ello realizado con el máximo cuidado en comodidad para el usuario, seguridad, estética… No es la aplicación más adecuada posible para obtener un trabajo, pero sí puede servir para darse a conocer. Podría crearse una cuenta de usuario de prueba, incluso con un rango especial de permisos para poder entrar a la zona de administración, y, aunque no permitiera editar nada, que sí dejase ver todas las páginas. Así podrían vislumbrarse todos los aspectos del aplicativo, incluso éste podría subirse a un portfolio previamente creado, subir el código a plataformas como Github… y de esta forma darse a conocer enseñando una muestra de los conocimientos de desarrollo obtenidos.

No se eligió este tema porque en el contexto laboral actual estén de moda este tipo de aplicaciones, de hecho, como se mencionó en más de una ocasión, los foros están de capa caída, pero por ello mismo se emplearon técnicas como el *scroll* infinito. Aún así se considera que es una aplicación de lo más completa y detallada para darse a conocer y ver que lo ha hecho una persona perfeccionista.

Con todo esto se pretendería entrar en el mercado laboral como desarrollador web, principalmente para el *backend*.

9. Instalación y configuración

Se trata de una aplicación web, así que requerirá de un servidor como por ejemplo Apache (incluido en el software **XAMPP** p.e.) y, si se requiere entrar a la administración de la BD, un gestor de bases de datos (XAMPP mismamente incluye el **PhpMyAdmin**). A mayores y, por último, si se quisiera probar el envío de emails como se ha hecho, sería necesario el software **Mailhog**, gratuito y ejecutable (sin necesidad de instalación) y de rapidísima configuración (descomentar 2 líneas en el archivo del XAMPP php.ini). Pueden descargarse desde sus enlaces oficiales:

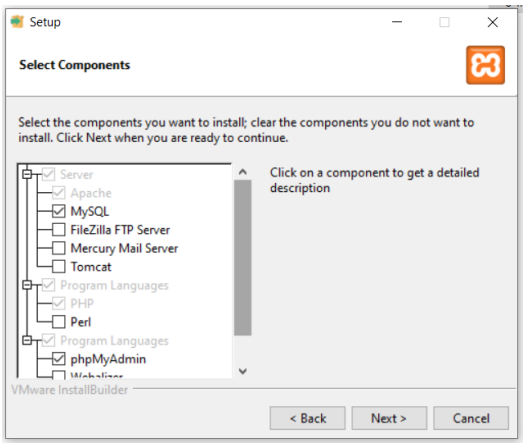
XAMPP:

https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/8.1.10/xampp-windows-x64-8.1.10-0-VS16-installer.exe

Mailhog:

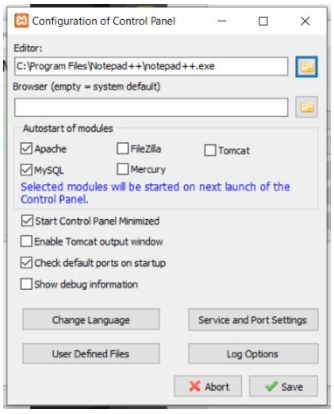
https://github.com/mailhog/MailHog/releases/download/v1.0.1/MailHog_windows_amd64.exe

La instalación del XAMPP no tiene complejidad, solo destacar que no son necesarias todas sus características, sino que bastará con Apache y PHP (ya marcados a la fuerza), MySQL y phpMyAdmin como muestra la siguiente imagen:

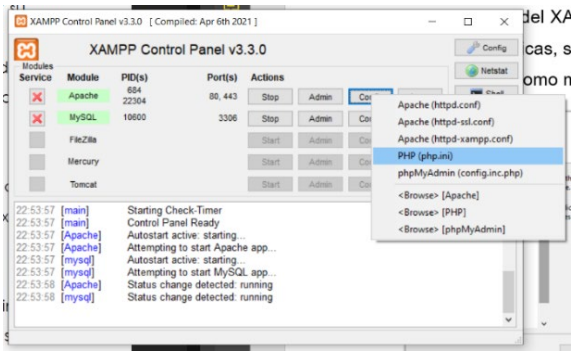


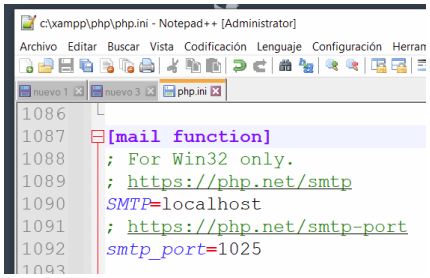
Una vez instalado (se recomienda hacerlo en la raíz de C:/ como se te comunica al principio), debe accederse al archivo php.ini. Puede hacerse manualmente yendo a C:\xampp\php o abriendo el panel de control del XAMPP con el icono que muestra en la barra de tareas al ejecutarse. Allí debemos activar los 2 primeros módulos (Apache y MySQL), aunque lo mejor es ir a las opciones y marcarlos para que se autoinicien. Acto seguido, se clica en **Config** del Apache > **php.ini**

En las opciones podemos marcar las casillas de los módulos del **Apache** y **MySQL** y la casilla de **Start Control Panel minimized**.



Sobre el php.ini:





Ya entro del archivo, pulsamos **Ctrl+F** para abrir el cuadro de búsqueda e introducimos **smtp**. Allí descomentamos las líneas 1090 y 1092 (o en su carencia las creamos) e introducimos los valores **localhost** y el puerto **1025**. Luego ejecutamos también el Mailhog; aparecerá una ventana de comandos que podemos minimizar y olvidarnos de ella. Ahora, si accedemos s localhost en el navegador comprobaremos que el XAMPP está arrancado y funcionando, y si vamos a **localhost:8025**, podremos acceder a la interfaz gráfica del Mailhog. Para acceder a la app web deberemos escribir en la URL el directorio donde la descomprimimos, generalmente suele ser en la carpeta dashbaord del XAMPP, y si es así, la dirección sería **localhost/dashboard/davidfajardo**.

Por último, debemos importar la base de datos, para lo cual debemos ir a la URL http://localhost/phpmyadmin/ e ir al apartado **Importar**, seleccionamos el archivo SQL y clicamos más abajo en **Continuar**.

NOTA: la BD incluye una cuenta con privilegios de administrador para hacer todo tipo de pruebas; el usuario es **David** y la contraseña **Abc123.** (punto incluido).

10. Posicionamiento SEO

Para esta tarea se han realizado búsquedas en Internet en pos de informarse sobre el posicionamiento SEO, y se ha llegado a la conclusión de que incluir las siguientes etiquetas sería beneficioso para la posición de esta página en buscadores.

```
<link rel="apple-touch-icon" sizes="57x57" href="../pictures/touch-icon-iphone.ico" />
<link rel="apple-touch-icon" sizes="114x114" href="../pictures/touch-icon-ipad.ico" />
<meta name="robots"                content="index, follow, max-snippet:-1, max-image-preview:large, max-video-preview:-1" />
<meta name="description"           content="Sitio web oficial de David Fajardo Barral. Aquí encontrarás mis novelas, biografía y un foro para hablar de diversos temas." />
<meta property="og:locale"         content="es_ES" />
<meta property="og:type"           content="website" />
<meta property="og:title"          content="David Fajardo - Oficial" />
<meta property="og:site_name"      content="David Fajardo - Oficial" />
<meta property="article:publisher" content="https://twitter.com/david_dvd91" />
<meta property="og:image"          content="../pictures/alas_negras.jpg" />
<meta property="og:image:secure_url" content="../pictures/alas_negras.jpg" />
<meta property="og:image:width"    content="1471" />
<meta property="og:image:height"   content="1362" />
<meta property="og:description"    content="Sitio web oficial de David Fajardo Barral." />
<meta name="twitter:card"          content="summary" />
<meta name="twitter:site"          content="@david_dvd91" />
<meta name="twitter:creator"       content="@david_dvd91" />
<meta name="twitter:image"         content="../pictures/alas_negras.jpg" />
<meta name="twitter:title"         content="David Fajardo - Oficial" />
<meta name="twitter:description"   content="Sitio web oficial de David Fajardo Barral." />
<meta name="keywords"              content="david fajardo, david fajardo barral, fajardo barral">
<meta name="author"                content="David Fajardo Barral">
```

Las etiquetas <link> ofrecerán un icono en dispositivos de Apple, los <meta> con *property* ofrecerán información a buscadores, al igual que las 2 últimas, y las de Twitter servirán para que, a la hora de mencionar la URL en Twitter, ésta gane una previsualización.

11. Conclusiones

Se han logrado realizar prácticamente todos los objetivos propuestos al principio de este proyecto; se darán más detalles en el siguiente apartado. Por supuesto surgieron dificultades constantes, sobre todo al llegar al *backend* con PHP, pero con esfuerzo e investigación en Internet (más detalles en la webgrafía) todo se acabó resolviendo.

En el siguiente apartado veremos con más detalle qué cosas se podrían haber hecho mejor y posibles mejoras futuras.

12. Vías futuras

Se han logrado la inmensa mayoría de objetivos, a excepción de algunos pequeños detalles que se han dejado de lado por falta de tiempo. He aquí una lista de fallos menores y posibles mejoras:

- Algunos campos de tablas de la BD fueron **ENUM** cuando hubiera sido más óptimo usar **BOOL** (TINYINT) ya que sus valores posibles son 2.
- La tabla del **CRUD** de usuarios no tiene implementado el **scroll infinito**, de forma que se muestran siempre todos los usuarios existentes, ya sean 5 ó 5,000 p.e.
- A la hora de subir un nuevo **avatar** en tu panel de control, si la **imagen** no es **cuadrada**, luego en miniaturas y demás visualizaciones se verá estirado. Se advierte a la hora de subirlo en el panel de control, pero nada más.
- En un principio se implantó la mecánica de que al Administrador pudiera **anclar mensajes** en los temas del foro, de forma que aparezcan siempre de primeros, útil para comunicados importantes a los usuarios de la plataforma, pero por antiestética (falta de sitio donde los botones de responder, editar y borrar mensajes) se decidió descartarla.
- No se implementó la función para que los usuarios puedan **darse de baja** y eliminar su cuenta de usuario si quieren.
- Análogamente, el administrador sí puede eliminar usuarios, pero solo si no han escrito mensajes en el foro. Habría que implementar el borrado de mensajes (que tal como está montado todo, solo habría que añadir la palabra “mensajes” en la función de borrar usuarios, ya que así se llama la tabla de la BD donde se guardan), pero eso es la parte fácil. También sería necesaria la consecuente actualización del número de mensajes de los temas del foro, a los mensajes que respondieran a los eliminados habría que quitarles esa vinculación… y un gran etc.
- El proceso por el cual **se eliminan cuentas de usuario no verificadas** pasados 7 días desde su registro no es automático; sino que la función se ejecuta si se entra en la página de registro o en la administración de usuarios.
- Para averiguar si ciertos datos de la BD están vacíos, se ha usado mucho (que no siempre) una comparación de strings (== “”) en vez de la función **empty()** de PHP, que habría sido mejor.

- Se podría haber implementado un botón junto al CRUD de usuarios de resetearle la contraseña a un usuario en caso de que nos comunicase que no recuerda sus 3 **respuestas de seguridad**.
- Se podría haber implementado una **puerta trasera**, un acceso especial de superadmin únicamente para un único usuario, en este caso nosotros, el Administrador y desarrollador, por si en un futuro le diéramos a alguien más permisos de administración y por lo tanto la posibilidad de quitarnos los permisos. Así podríamos entrar por ahí, con una contraseña secreta maestra y quitar a quien sea los permisos oportunos.
- Para que la **función del panel de control** no sea tan grande, se debería haber separado en 2 funciones: una para el formulario visible para el usuario y otra para el proceso de hacer la modificación en la BD.
- Para las 3 **preguntas/respuestas** de seguridad no hay ningún tipo de comprobación (salvo prevenir ataques XSS), por lo que podrías repetir preguntas y respuestas, ya que es un mal menor; se avisa al usuario en grande y en rojo de que las guarde bien.

13. Glosario

- BD: base de datos
- Login / logeado: inicio de sesión / usuario con la sesión iniciada
- XSS: Cross-Site Scripting
- JS: JavaScript
- Nick: nombre de usuario
- Backend: la parte de la app web que no se ve, que no genera vistas de ningún tipo; la programación que está por detrás que lleva la lógica del programa y hace que todo funcione.
- Scroll: desplazamiento.

14. Bibliografía/Webgrafía

Ayuda para documentar los requisitos funcionales y no funcionales:

https://repositorio.konradlorenz.edu.co/micrositios/001-1527/requerimientos_no_funcionales.html

https://visuresolutions.com/es/blog/functional-requirements/

Ayuda para documentar los diagramas de casos de uso:

https://www.abiztar.com.mx/articulos/casos-a-incluir-casos-a-extender.html

Documentación de PHP:

https://www.php.net/manual/es/intro-whatis.php

Documentación de Bootstrap:

https://getbootstrap.com/docs/5.2/getting-started/introduction/

Ayudas varias en problemas concretos:

https://stackoverflow.com/

Posicionamiento SEO

https://www.arsys.es/blog/etiquetas-head

Curso de PHP de 30 horas:

https://www.youtube.com/playlist?list=PLU8oAlHdN5BkinrODGXToK9oPAlnJxmW_

Curso de JavaScript de 25 horas:

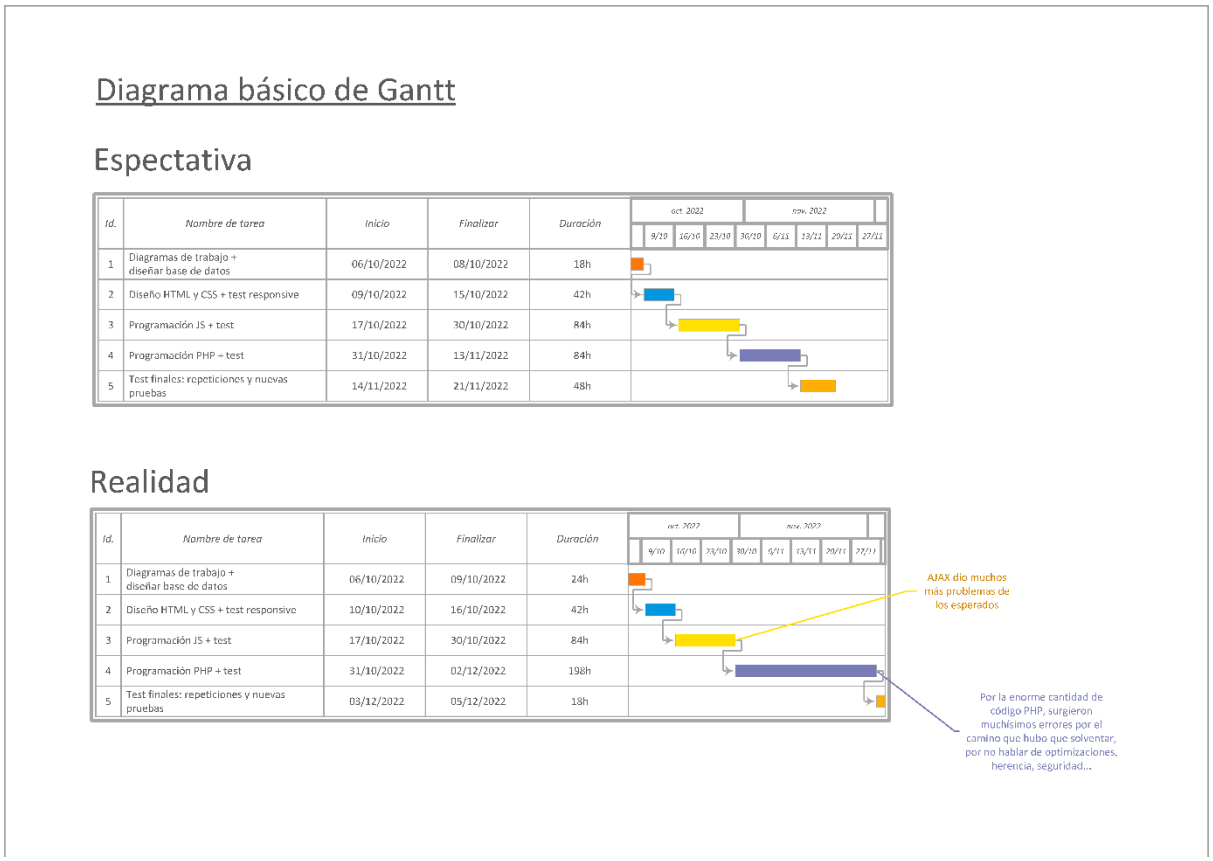
https://www.youtube.com/playlist?list=PLU8oAlHdN5BmpobVmj1IlneKlVLJ84TID

15. Anexos

12.1 Anexo I

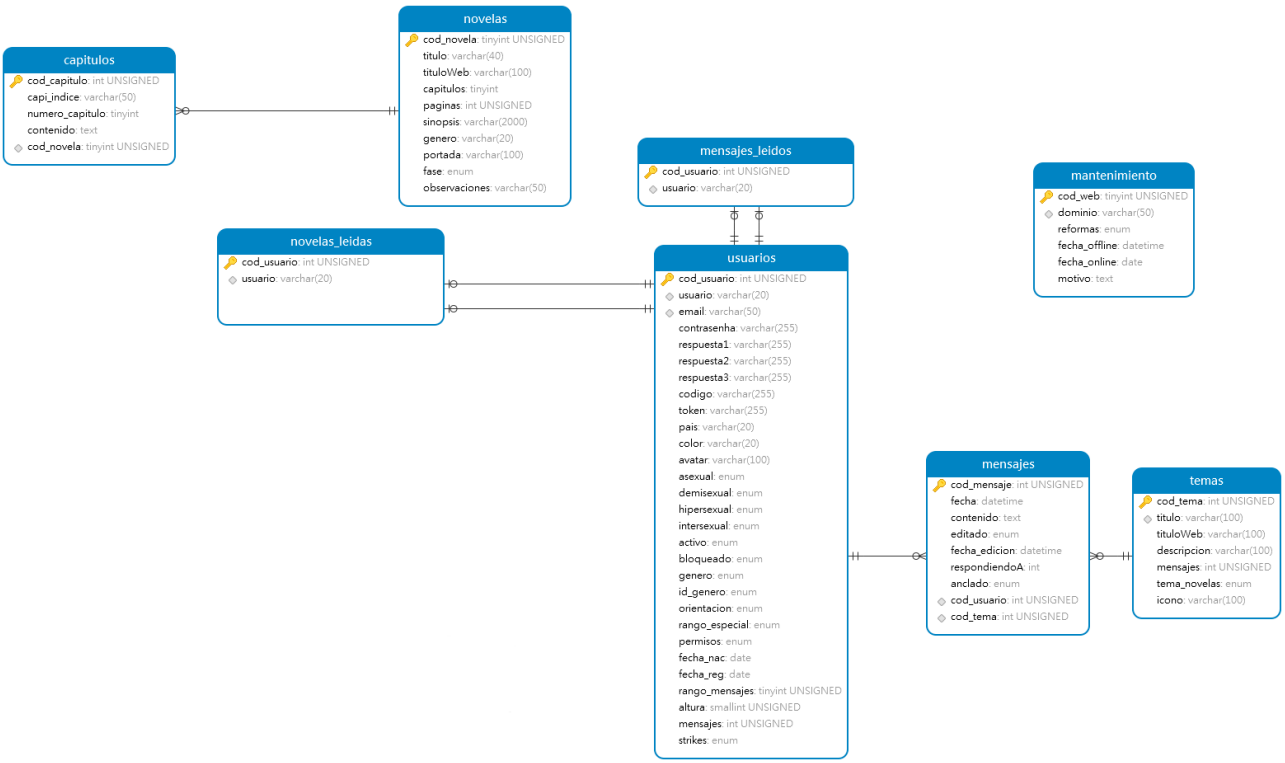
I - Diagramas

I.I - Diagrama de Gantt

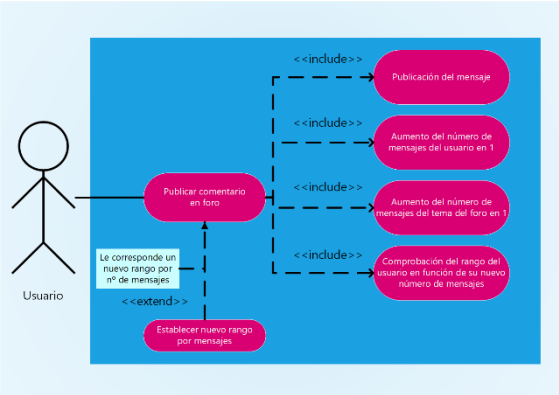


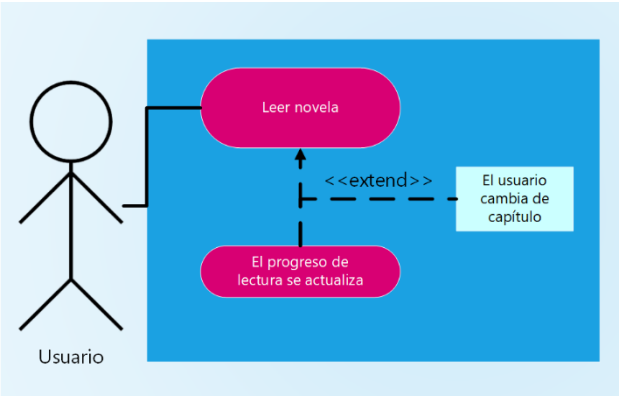
I.II - Diagrama ER (Entidad-Relación)

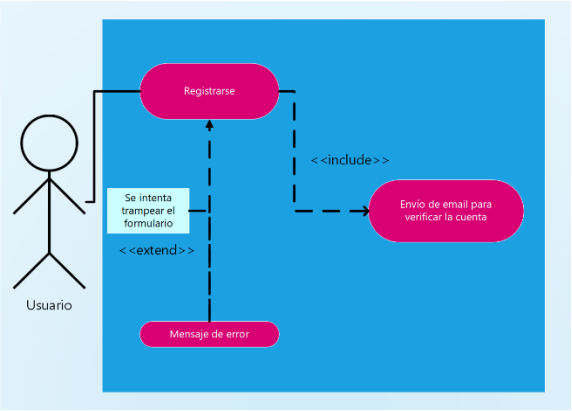
Recordemos el modo de mantenimiento: nada que ver con usuarios, novelas, mensajes… ergo su tabla no está relacionada con ninguna otra; no es necesario.

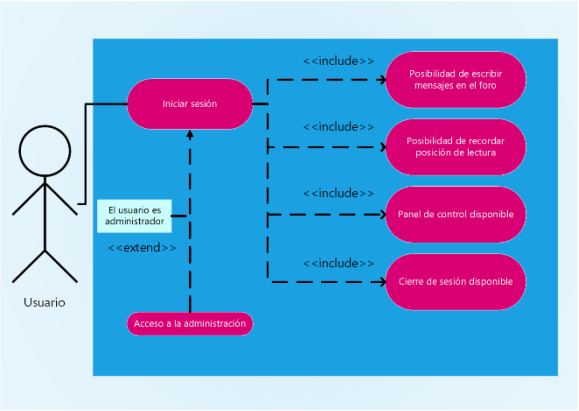


I.III - Diagramas de casos de uso:





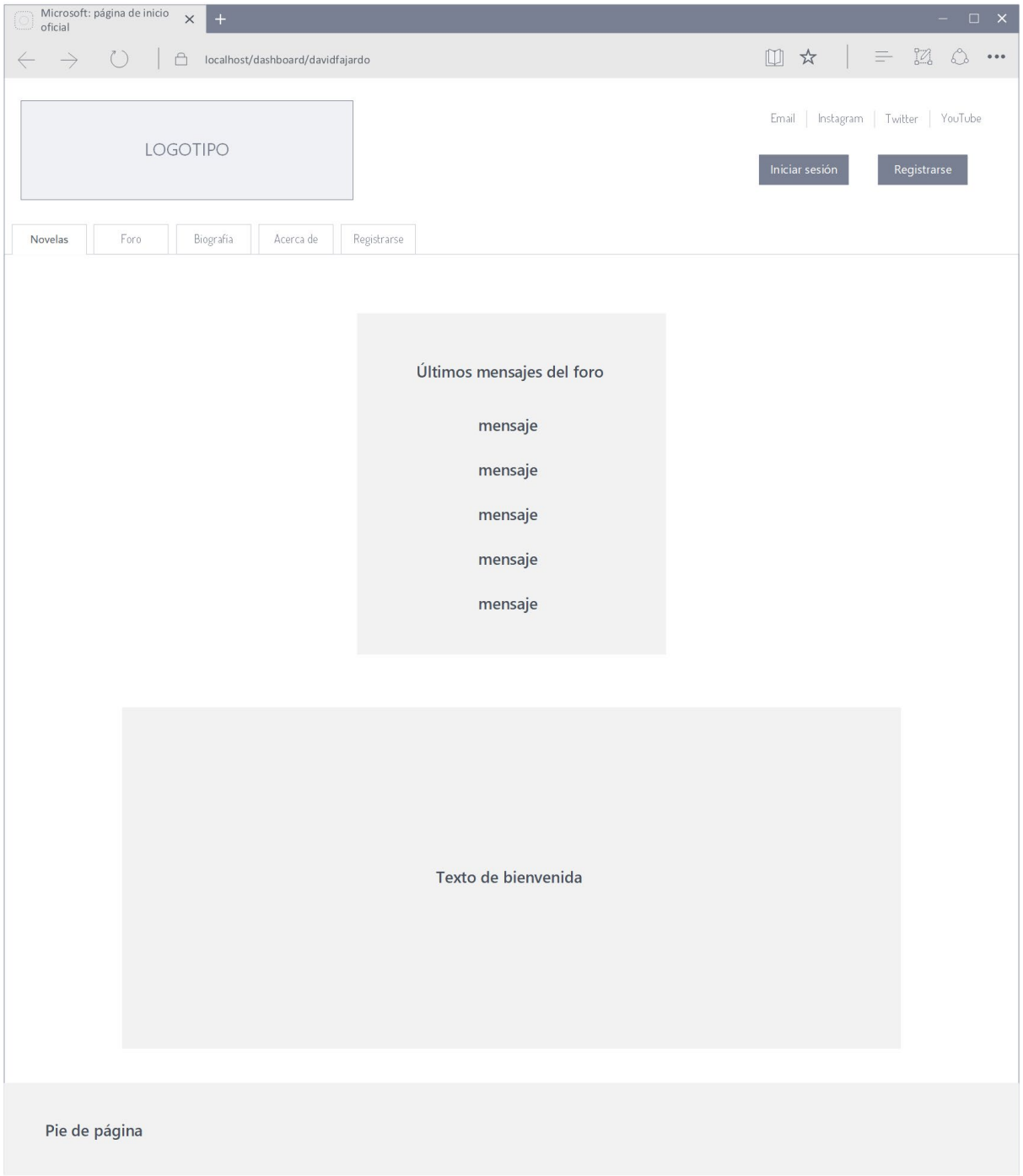




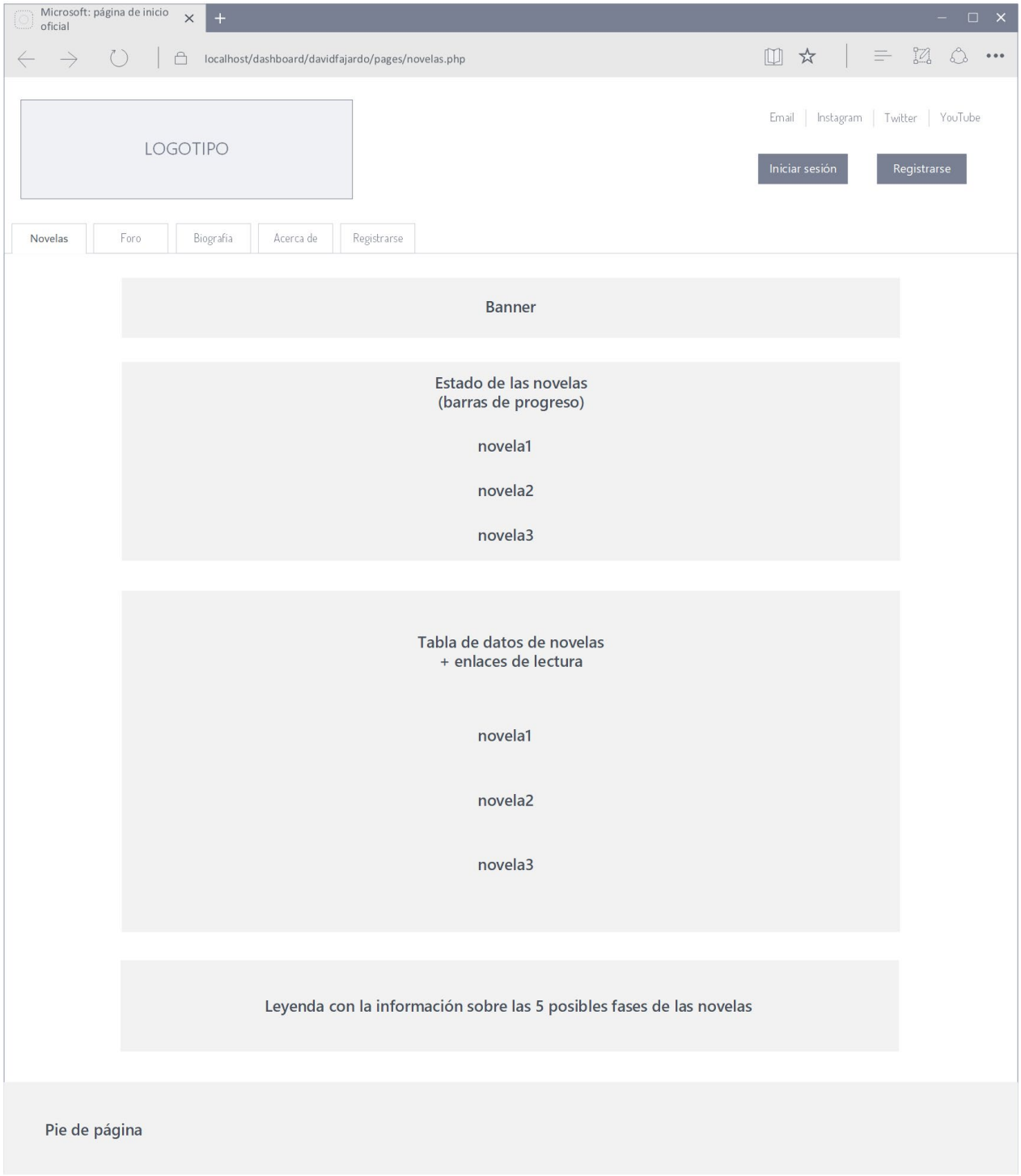
I.IV - Diagrama de clases:

II - Wireframes

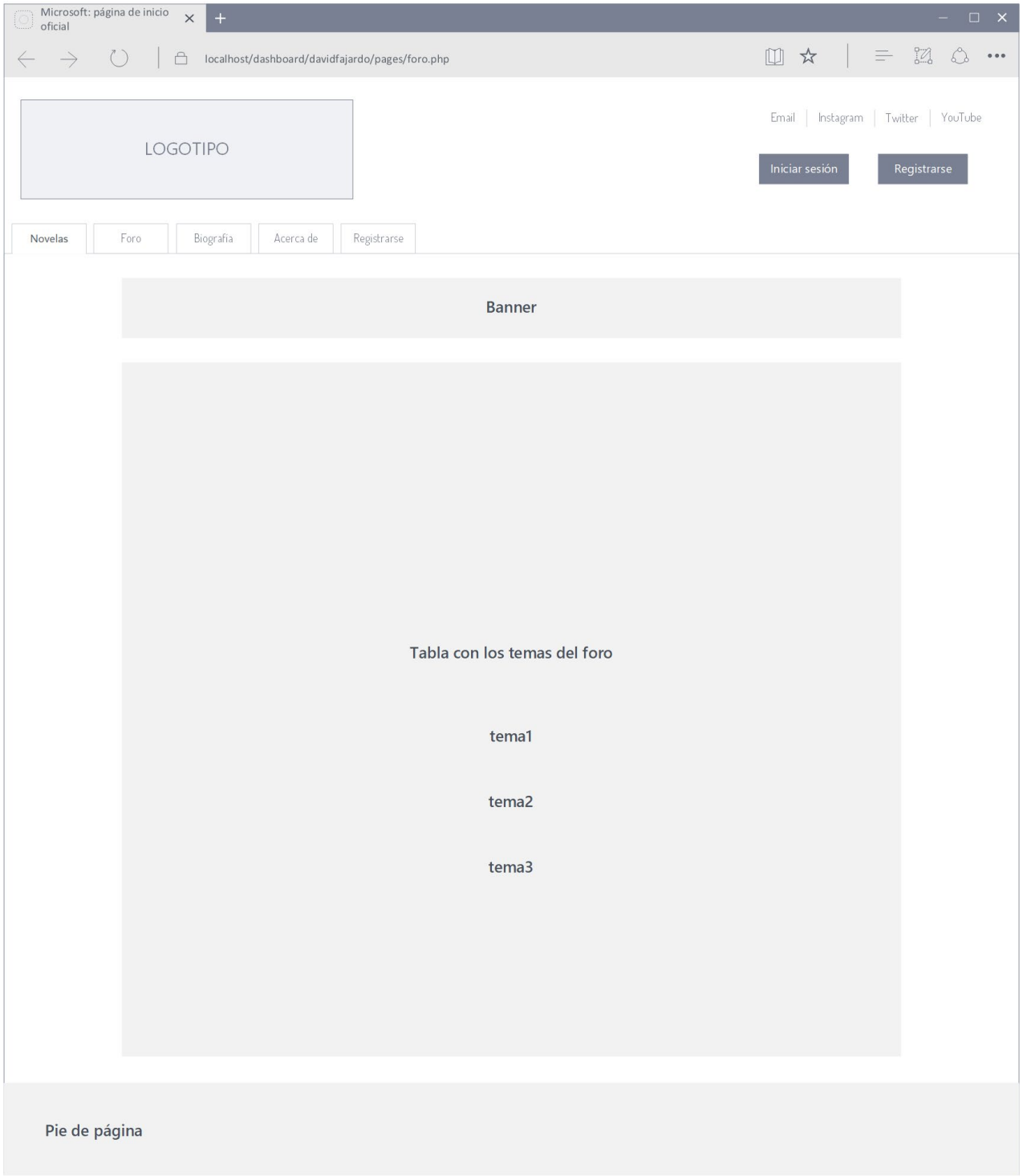
II.I - Portada de la web (sin iniciar sesión)



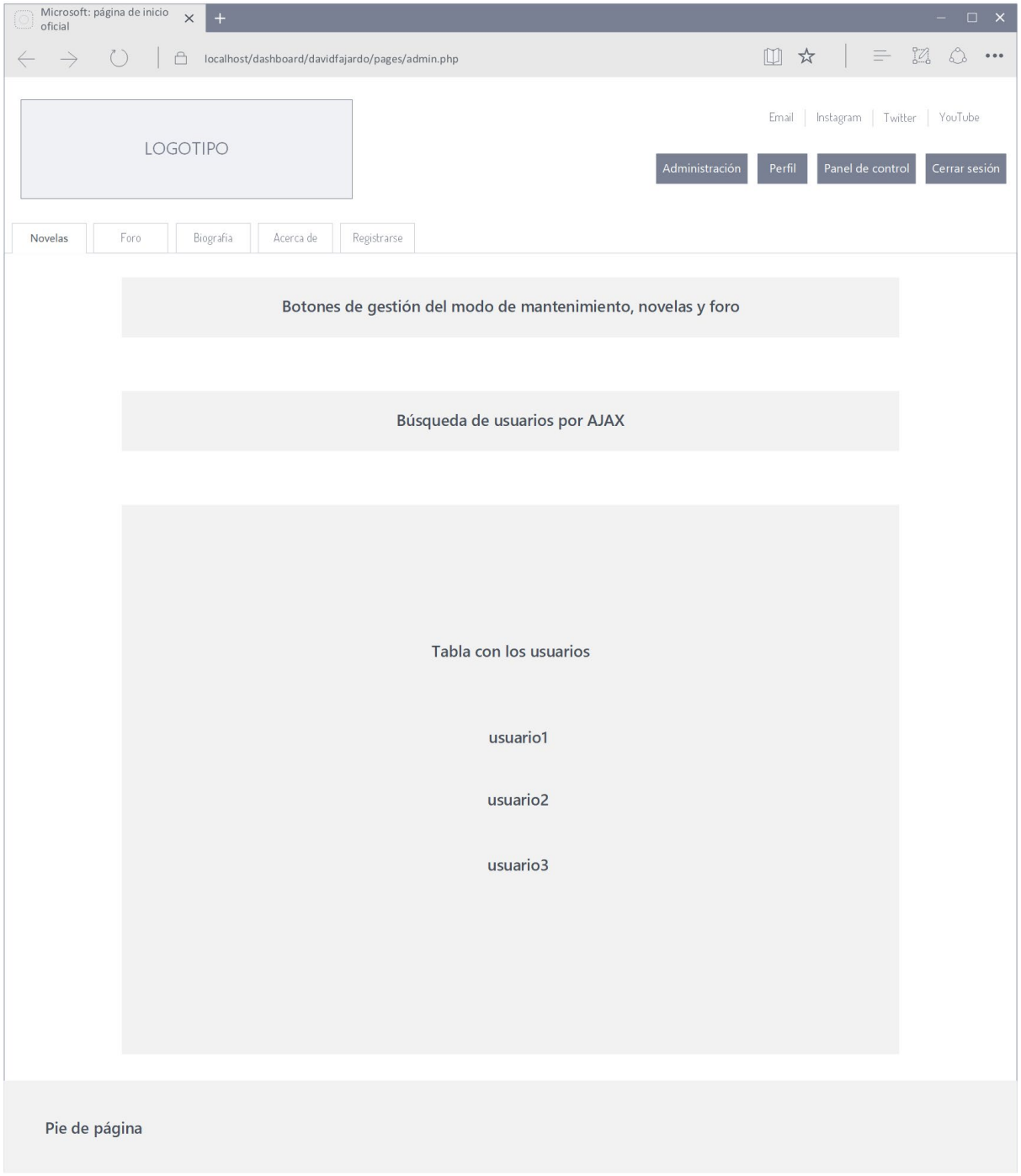
II.II – Sección de novelas (sin iniciar sesión)



II.III – Sección del foro (sin iniciar sesión)



II.IV – Administración de usuarios



III - Funciones y demás fragmentos de código

III.I - Función PHP iniciarSesion()

```php
function iniciarSesion($reform = false) {
    if (isset($_POST['login'])) {
        try {
            $usuario       = trim($this->evitarXSS($_POST['usuario']));
            $contrasenha   = trim($this->evitarXSS($_POST['contrasenha']));

            $this->__construct();
            $query = $this->db->prepare(
                "SELECT cod_usuario, usuario, contrasenha, permisos, activo, bloqueado, token
                FROM usuarios WHERE usuario = :usuario"
            );
            $query->bindParam(":usuario", $usuario, PDO::PARAM_STR);
            $query->execute();
            $num_filas = $query->rowCount();

            if ($num_filas == 1) {

                foreach ($query as $usuarios) {

                    // Si la contraseña es correcta, continuamos
                    if (password_verify($contrasenha, "$2y$10$".$usuarios['contrasenha'])) {

                        // Si la cuenta de usuario no está activada
                        if ($usuarios['activo'] == "No")
                            $this->mensajeError("Tu cuenta no está activada. ¿Recibiste el email?");

                        // Si el usuario está bloqueado
                        else if ($usuarios['bloqueado'] == "Si")
                            $this->mensajeError("Tu cuenta está bloqueada.");

                        // Si no lo está
                        else {
                            // Si la web está de reformas y el usuario no es Administrador
                            if ($reform && $usuarios['permisos'] != "Administrador")
                                $this->mensajeError("Con la web en reformas solo el Administrador puede iniciar sesión.");

                            // O si la web está de reformas pero se logea un admin
                            // o si no hay reformas, que entre todo dios que no esté bloqueado
                            else if (($reform && $usuarios['permisos'] == "Administrador") || !$reform) {
                                $this->crearCookie("cod_usuario", $usuarios['cod_usuario']);
                                $this->crearCookie("usuario", $usuarios['usuario']);

                                // Cambios de token y código puestos a la machada sin funciones porque no iban
                                // Igual porque haría falta actualizar la página por las cookies (?)
                                $codigo = $this->encriptar(rand(1000,5000));
                                $token = $this->encriptar(rand(1000,5000));
                                $this->crearCookie("token", $token);
                                $query = $this->db->prepare("UPDATE usuarios SET codigo = :codigo WHERE usuario = :usuario");
                                $query->bindParam(":usuario",   $usuario,   PDO::PARAM_STR);
                                $query->bindParam(":codigo",    $codigo,    PDO::PARAM_STR);
                                $query->execute();
                                $query = $this->db->prepare("UPDATE usuarios SET token = :token WHERE usuario = :usuario");
                                $query->bindParam(":usuario",   $usuario,   PDO::PARAM_STR);
                                $query->bindParam(":token",     $token,     PDO::PARAM_STR);
                                $query->execute();

                                header("location: ".$this->evitarXSS($_SERVER['PHP_SELF']));
                            }
                        }
                    }
                    else
                        $this->mensajeError("Contraseña incorrecta.");
                }
            }

            else
                $this->mensajeError("El usuario especificado no existe.");

        } catch (Exception $e) {
            exit("Error: " . $e->GetMessage());
        } finally {
            $this->db = null;
        }
    }
}

```

III.II Función PHP cambiarToken()

```php
function cambiarToken($email = false) {
    $token = $this->encriptar(rand(1000,5000));

    $this->__construct();
    try {
        if (!$email) {
            $usuario = $this->evitarXSS($_COOKIE['usuario']);
            $query = $this->db->prepare("UPDATE usuarios SET token = :token WHERE usuario = :usuario");
            $query->bindParam(":usuario", $usuario, PDO::PARAM_STR);
        }
        else {
            $query = $this->db->prepare("UPDATE usuarios SET token = :token WHERE email = :email");
            $query->bindParam(":email", $email, PDO::PARAM_STR);
        }
        $query->bindParam(":token", $token, PDO::PARAM_STR);
        $query->execute();
    } catch (Exception $e) {
        exit("Error: " . $e->GetMessage());
    } finally {
        $this->db = null;
    }
}
```

III.III Función PHP cambiarCódigo()

```php
function cambiarCodigo($email = false) {
    $codigo = $this->encriptar(rand(1000,5000));

    try {
        $this->__construct();
        if (!$email) {
            $usuario = $this->evitarXSS($_COOKIE['usuario']);
            $query = $this->db->prepare("UPDATE usuarios SET codigo = :codigo WHERE usuario = :usuario");
            $query->bindParam(":usuario", $usuario, PDO::PARAM_STR);
        }
        else {
            $query = $this->db->prepare("UPDATE usuarios SET codigo = :codigo WHERE email = :email");
            $query->bindParam(":email", $email, PDO::PARAM_STR);
        }
        $query->bindParam(":codigo", $codigo, PDO::PARAM_STR);
        $query->execute();
    } catch (Exception $e) {
        exit("Error: " . $e->GetMessage());
    } finally {
        $this->db = null;
    }
}
```

III.IV Función PHP comprobarLogin()

```php
function comprobarLogin() {
    // Medida 1: si falta alguna cookie, se manda FALSE
    if (!isset($_COOKIE['cod_usuario']) || !isset($_COOKIE['usuario']) ||
        !isset($_COOKIE['token']))
        return false;

    // Medida 2: comparar valores de las cookies con la BD de ese usuario
    try {
        $this->__construct();

        // Se asignan las cookies guardadas a variables para usarlas más cómodamente
        $cod_usuario        = $this->evitarXSS($_COOKIE['cod_usuario']);
        $usuario            = $this->evitarXSS($_COOKIE['usuario']);
        $token              = $this->evitarXSS($_COOKIE['token']);

        $query = $this->db->prepare (
            "SELECT cod_usuario, usuario, codigo, token, activo, bloqueado
            FROM usuarios WHERE
                cod_usuario = :cod_usuario  AND
                usuario     = :usuario      AND
                token       = :token        AND
                activo      = 'Si'          AND
                bloqueado   = 'No'
        ");

        $query->bindParam(":cod_usuario", $cod_usuario, PDO::PARAM_INT);
        $query->bindParam(":usuario", $usuario, PDO::PARAM_STR);
        $query->bindParam(":token", $token, PDO::PARAM_STR);
        $query->execute();
        $total_filas = $query->rowCount();

        // Si aparece el usuario, se comprueba que todo esté bien
        if ($total_filas == 1) {
            foreach ($query as $comprobaciones) {

                // Si el ID de usuario de la BD no es el almacenado en la cookie o
                // si el nombre de usuario de la BD no es el almacenado en la cookie o
                // si el token de la cookie no se corresponde con el de al BD
                // se manda FALSE
                if ($comprobaciones['cod_usuario']  != $cod_usuario ||
                    $comprobaciones['usuario']      != $usuario ||
                    $comprobaciones['token']        != $token) {
                    $this->borrarCookie("cod_usuario");
                    $this->borrarCookie("usuario");
                    $this->borrarCookie("token");
                    return false;
                }

                // Si está todo correcto, se manda TRUE
                else
                    return true;
            }
        }
    } catch (Exception $e) {
        exit("Error: " . $e->GetMessage());
    } finally {
        $this->db = null;
    }
}
```

III.V Función PHP comprobarPermisos()

```php
function comprobarPermisos(string $permisos) {
    if ($this->comprobarLogin()) {
        $cod_usuario    = $this->evitarXSS($_COOKIE['cod_usuario']);
        $usuario        = $this->evitarXSS($_COOKIE['usuario']);
        $token          = $this->evitarXSS($_COOKIE['token']);
        try {
            $this->__construct();

            if ($permisos == "Administrador") {
                $query = $this->db->prepare (
                    "SELECT COUNT(cod_usuario) FROM usuarios WHERE
                        cod_usuario = :cod_usuario  AND
                        usuario     = :usuario      AND
                        token       = :token        AND
                        cod_usuario = 1                         AND
                        usuario     = 'David'                   AND
                        email       = 'david2dvd91@gmail.com'   AND
                        permisos    = 'Administrador'           AND
                        activo = 'Si' AND bloqueado = 'No'"
                );
            }

            else if ($permisos == "Moderador") {
                $query = $this->db->prepare (
                    "SELECT COUNT(cod_usuario) FROM usuarios WHERE
                    cod_usuario = :cod_usuario  AND
                    usuario     = :usuario      AND
                    token       = :token        AND
                    permisos    = 'Moderador'   AND
                    activo = 'Si' AND bloqueado = 'No'"
                );
            }

            $query->bindParam(":cod_usuario",   $cod_usuario,   PDO::PARAM_INT);
            $query->bindParam(":usuario",       $usuario,       PDO::PARAM_STR);
            $query->bindParam(":token",         $token,         PDO::PARAM_STR);
            $query->execute();
            $total_filas = $query->fetch();

            if ($total_filas[0] == 1)
                return true;
            else
                return false;
        } catch (Exception $e) {
            exit("Error: " . $e->GetMessage());
        } finally {
            $this->db = null;
        }
    }
    else
        return false;
}

```

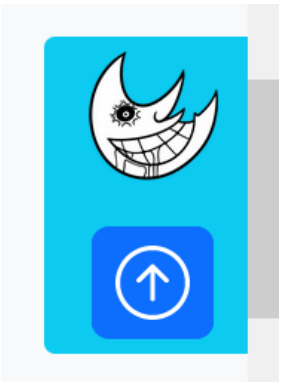
III.VI Función JS editorTexto()

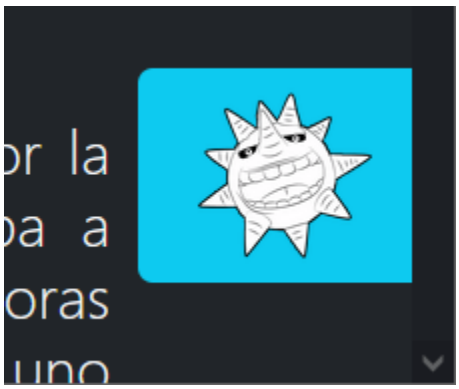
```javascript
function editorTexto(parametro, cuadroDeTexto) {
    var comentario      = document.getElementById(cuadroDeTexto);
    var inicioSeleccion = comentario.selectionStart;
    var finSeleccion    = comentario.selectionEnd;
    var seleccion       = comentario.value.substring(inicioSeleccion, finSeleccion);

    const etiquetas             = ["B", "I", "U", "S", "url", "spoiler", "img"];
    const etiquetasPrincipio    = ["[b]", "[i]", "[u]", "[s]", "[url]", "[spoiler]", "[img]"];
    const etiquetasFinal        = ["[/b]", "[/i]", "[/u]", "[/s]", "[/url]TEXTO VISIBLE DEL ENLACE[/nombreVisibleAmiIzquierda]", "[/spoiler]", "[/img]"];

    for (let i = 0; i < etiquetas.length; i++) {
        switch (parametro) {
            case etiquetas[i]:
                seleccion = etiquetasPrincipio[i] + seleccion + etiquetasFinal[i];
                break;
        }
    }

    comentario.value = comentario.value.substring(0, inicioSeleccion) + seleccion + comentario.value.substring(finSeleccion, comentario.length);
}
```

III.VII Función PHP comprobarPropiedadMensaje()

```php
function comprobarPropiedadMensaje($cod_mensaje, $mensajeAjeno = false) {
    try {
        $this->__construct();
        $query = $this->db->prepare(
            "SELECT cod_usuario FROM mensajes WHERE cod_mensaje = :cod_mensaje"
        );
        $query->bindParam(":cod_mensaje", $cod_mensaje, PDO::PARAM_INT);
        $query->execute();
        $cod_usuario = $query->fetch();

        if (!$mensajeAjeno) {
            if ($cod_usuario[0] == $_COOKIE['cod_usuario'])
                return true;
            else
                return false;
        }
        else
            return $cod_usuario[0];

    } catch (Exception $e) {
        exit("Error: " . $e->GetMessage());
    } finally {
        $this->db = null;
    }
}
```

III.VII Aspecto y código del modo oscuro y el botón del *scroll* hacia arriba






```
<script type="text/javascript">
        window.addEventListener("load", function() {
            <?php
                // Si existe la cookie por haber seleccionado el modo oscuro,
                // se cambia a ese modo
                if (isset($_COOKIE['modo_oscuro'])) {
                    if ($_COOKIE['modo_oscuro'] == "si")
                        print("cambiarColores();");
                }
            ?>

            // De primeras que el botón de volver arriba sea invisible
            $(".boton-arriba").hide();

            // Efecto de deslizamiento hacia arriba
            $(".boton-arriba").click(function(e) {
                e.preventDefault();
                $('html, body').animate({
                    scrollTop: '0px'
                }, 500);
            });

            // Que el botón de volver arriba no aparezca hasta que hayas bajado algo
            $(window).scroll(function(){
                if ($(this).scrollTop() > 0)
                    $(".boton_oculto").fadeIn(300);
                else
                    $(".boton_oculto").fadeOut(300);
            });

            // Para que en móviles también se puedan mostrar los botones mediante click táctil
            $(document).on("touchstart",function (e) {
                var botones = $("#botones_darktop");
                if (!botones.is(e.target) && $(e.target).closest(botones).length == 0)
                    botones.css("margin-right", "0px");
                else
                    botones.css("margin-right", "90px");
            });
        }, false);

        function cambiarColores() {
            var imagenBoton = $(".boton_oscuro").attr("src");

            // Si se activa el modo oscuro
            if (imagenBoton == "../pictures/luna.png") {
                $("body").removeClass("bg-light text-dark").addClass("bg-dark text-light");
                $(".navbar-brand, .textoForo").addClass("text-light");
                $(".navbar-toggler").removeClass("border-dark").addClass("border-light");
                $("a.nav-link").attr("style", "color: #1FF2F3 !important;");
                $(".card").removeClass("text-dark bg-light border-dark").addClass("text-light bg-dark border-light");
                $(".table:not(.tabla_per+ti, .tabla_foro)").addClass("text-light").removeClass("table-striped");
                $(".accordion-button, .accordion-body, table textarea, input:text").addClass("bg-dark text-light");
                $(".boton_oscuro").prop("src", "../pictures/sol.png").prop("title", "Modo claro");
                document.cookie = "modo_oscuro=si; max-age=31536000; path=/";
            }

            // Si se reactiva el modo claro
            else {
                $("body").removeClass("bg-dark text-light").addClass("bg-light text-dark");
                $(".navbar-brand, .textoForo").removeClass("text-light");
                $(".navbar-toggler").removeClass("border-light").addClass("border-dark");
                $("a.nav-link").attr("style", "color: black !important;");
                $(".card").removeClass("text-light bg-dark border-light").addClass("text-dark bg-light border-dark");
                $(".table:not(.tabla_per+ti, .tabla_foro)").removeClass("text-light").addClass("table-striped");
                $(".accordion-button, .accordion-body, table textarea, input:text").removeClass("bg-dark text-light");
                $(".boton_oscuro").prop("src", "../pictures/luna.png").prop("title", "Modo oscuro");
                document.cookie = "modo_oscuro=no; max-age=31536000; path=/";
            }
        }
    </script>

    <div class="bg-info rounded-start pt-2 pe-3 pb-2 ps-4" id="botones_darktop"
        style="position: fixed; right: -90px; bottom: 40px;">

        <!-- Botón del modo oscuro -->
        <img src="../pictures/luna.png" alt="boton_modo_oscuro" title="Modo oscuro"
        class="boton_oscuro img-fluid" width="70px" onclick="cambiarColores()" />

        <!-- Botón de volver arriba -->
        <div>
            <button class="boton-arriba boton_oculto fw-bold btn btn-primary display-1 boton-ripple m1-4" style="font-size: 28px !important; border-radius: 15% !important;">
                <svg xmlns="http://www.w3.org/2000/svg" width="40" height="40" fill="currentColor" class="bi bi-arrow-up-circle mb-1" viewBox="0 0 16 16">
                    <path fill-rule="evenodd" d="M1 8a7 7 0 1 0 14 0A7 7 0 0 0 1 8zm15 0A8 8 0 1 1 0 8a8 8 0 0 1 16 0zm-7.5 3.5a.5.5 0 0 1-1 0V5.707L5.354 7.854a.5.5 0 1 1-.708-.708l3-3a.5.5 0 0 1 .708 0l3 3a.5.5 0 0 1-.708.708L8.5 5.707V11.5z"/>
                </svg>
            </button>
        </div>
    </div><?php print("\n"); ?>
```

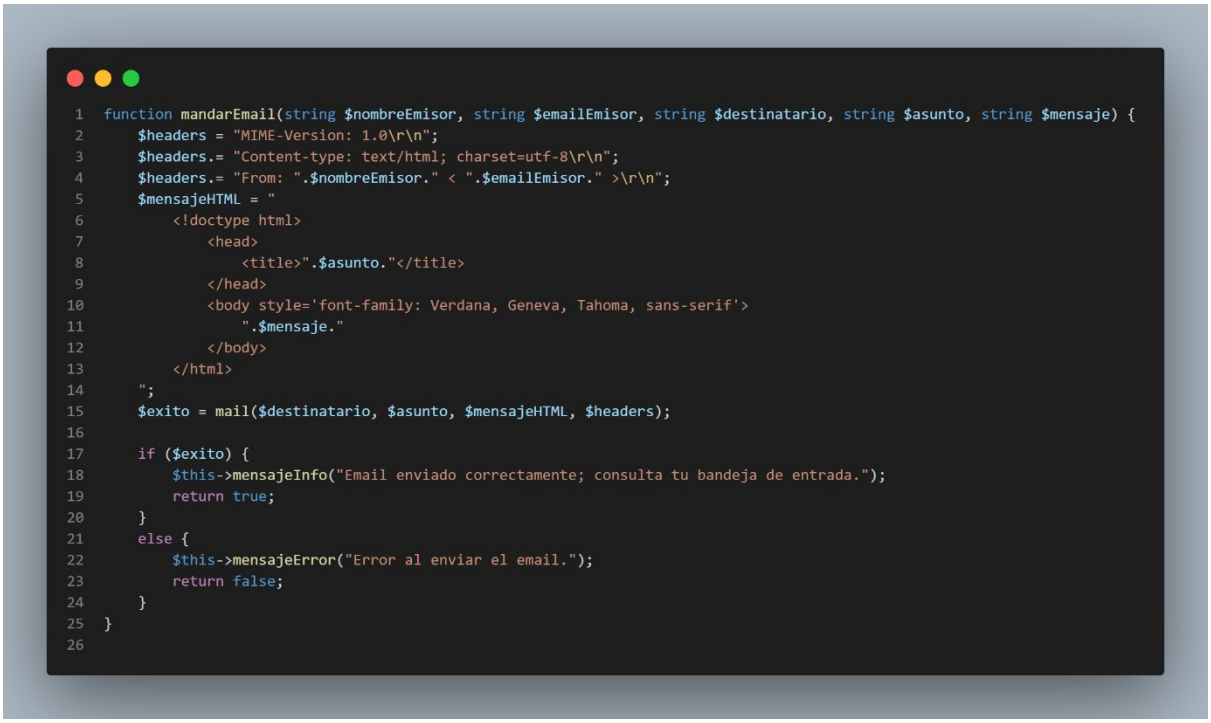
III.VIII Código JS del scroll infinito

```js
// Acción necesaria en función para repetirla y refrescar el dato
function contarElementos() {
    var elementos = document.getElementsByClassName("FIN_mensaje");
    var numElementos = elementos.length;
    return numElementos;
}

function eliminarBotonAJAX() {
    if(contarElementos() >= mensajesTema.value)
        $("#botonCargarMas").remove();
}

// Obtenemos la URL y la pasamos a un array dividio por barras laterales
var url = window.location.toString().split("tituloWeb=");

// Si el tema tiene 10 mensajes o menos y por lo tanto están todos, se elimina el botón
window.addEventListener("load", eliminarBotonAJAX, false);

cargarMasMensajes.addEventListener("click", function() {

    $.ajax({
        type: "POST",
        url: "../classes/AJAX.php",
        data: "mensajesPresentes=" + contarElementos() + "&tituloWeb=" + url[1],
        dataType: "html",
        beforeSend: function() {
            $(".FIN_mensaje:last").after("<div class='spinner-border text-primary mb-5' role='status'></div>");
        },
        error: function() {
            alert("Error al cargar más mensajes.");
        },
        success: function(data) {
            // Se eliminan los cargadores residuales
            $(".spinner-border").remove();

            // Se cargan los mensajes tras el último mensaje
            $(".FIN_mensaje:last").after(data);


            // Cosas de AJAX: hace falta recargar los eventos de los botones, que invoca las cosas sin eventos:

            // Botones de spoiler
            $(".collapser").click(function() {
                $(this).next().collapse('toggle');
            });


            // Cuando haya tantos mensajes como totales tiene el tema, se elimina el botón del AJAX
            eliminarBotonAJAX();
        }
    });
}, false);
```

III.IX Función PHP borrarUsuariosNoVerificados()

```php
function borrarUsuariosNoVerificados() {
    try {
        $this->__construct();
        $query = $this->db->prepare(
            "SELECT fecha_reg, usuario FROM usuarios WHERE activo = 'No'"
        );
        $query->execute();
        $fechas = $query->fetchAll();
        $usuariosNoVerificados = $query->rowCount();

        if ($usuariosNoVerificados == 0)
            return;

        $usuariosAborrar = array();
        for ($i = 0; $i < $usuariosNoVerificados; $i++) {
            if ($this->distanciaFechasEnDias($fechas[$i][0], date("Y-m-d")) > 7)
                array_push($usuariosAborrar, $fechas[$i][1]);
        }

        $tablas = array("mensajes_leidos", "novelas_leidas");

        $this->__construct();

        for ($i = 0; $i < count($usuariosAborrar); $i++) {
            for ($j = 0; $j < count($tablas); $j++) {
                $query = $this->db->prepare(
                    "DELETE FROM " . $tablas[$j] . " WHERE usuario = :usuario;");
                $query->bindParam(":usuario", $usuariosAborrar[$i], PDO::PARAM_STR);
                $query->execute();
            }
        }

        for ($i = 0; $i < count($usuariosAborrar); $i++) {
            $query = $this->db->prepare(
                "DELETE FROM usuarios WHERE usuario = :usuario AND activo = 'No'"
            );
            $query->bindParam(":usuario", $usuariosAborrar[$i], PDO::PARAM_STR);
            $query->execute();
        }
    } catch (Exception $e) {
        exit("Error: " . $e->GetMessage());
    } finally {
        $this->db = null;
    }
}
```

III.X Función PHP mandarEmail()



```php
function mandarEmail(string $nombreEmisor, string $emailEmisor, string $destinatario, string $asunto, string $mensaje) {
    $headers = "MIME-Version: 1.0\r\n";
    $headers.= "Content-type: text/html; charset=utf-8\r\n";
    $headers.= "From: ".$nombreEmisor." < ".$emailEmisor." >\r\n";
    $mensajeHTML = "
        <!doctype html>
            <head>
                <title>".$asunto."</title>
            </head>
            <body style='font-family: Verdana, Geneva, Tahoma, sans-serif'>
                ".$mensaje."
            </body>
        </html>
    ";
    $exito = mail($destinatario, $asunto, $mensajeHTML, $headers);

    if ($exito) {
        $this->mensajeInfo("Email enviado correctamente; consulta tu bandeja de entrada.");
        return true;
    }
    else {
        $this->mensajeError("Error al enviar el email.");
        return false;
    }
}

```

III.XI Código HTML y JS del temporizador en la portada en modo de mantenimiento

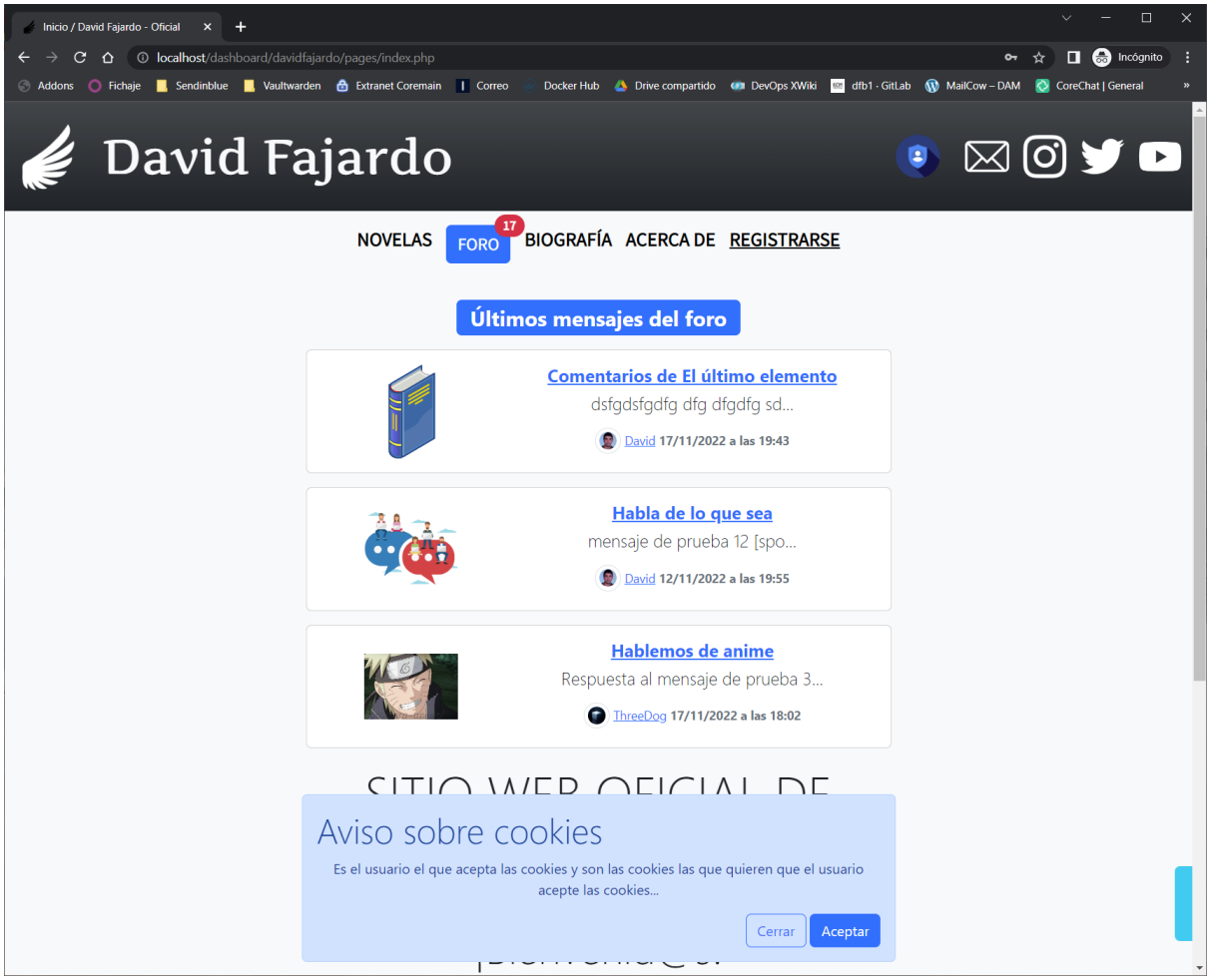
```html
<div class="row display-6 justify-content-center fw-bold">
    <div class="col-auto">
        <span id="dias"></span><br />
        día(s)
    </div>
    <div class="col-auto">
        <span id="horas"></span><br />
        hora(s)
    </div>
    <div class="col-auto">
        <span id="minutos"></span><br />
        minuto(s)
    </div>
    <div class="col-auto">
        <span id="segundos"></span><br />
        segundo(s)
    </div>
</div>
```

```javascript
<script type="text/javascript">
    window.addEventListener('load', function() {
        const SPAN_DIAS             = document.querySelector('span#dias');
        const SPAN_HORAS            = document.querySelector('span#horas');
        const SPAN_MINUTOS          = document.querySelector('span#minutos');
        const SPAN_SEGUNDOS         = document.querySelector('span#segundos');
        const MILISEGUNDOS_SEGUNDO  = 1000;
        const MILISEGUNDOS_MINUTO   = MILISEGUNDOS_SEGUNDO * 60;
        const MILISEGUNDOS_HORA     = MILISEGUNDOS_MINUTO * 60;
        const MILISEGUNDOS_DIA      = MILISEGUNDOS_HORA * 24;

        function actualizarTemporizador() {
            const FECHA_BUSCADA         = new Date(<?php print(str_replace("-", ",", $instancia->GET_Motivo_Mantenimiento()['fecha_online'])); ?>);
            const AHORA                 = new Date();
            const DURACION              = FECHA_BUSCADA - AHORA;
            // Al total de dias hay que restarle 1 mes; no me preguntes porqué
            const DIAS_RESTANTES        = Math.floor((DURACION / MILISEGUNDOS_DIA) - 30);
            const HORAS_RESTANTES       = Math.floor((DURACION % MILISEGUNDOS_DIA) / MILISEGUNDOS_HORA);
            const MINUTOS_RESTANTES     = Math.floor((DURACION % MILISEGUNDOS_HORA) / MILISEGUNDOS_MINUTO);
            const SEGUNDOS_RESTANTES    = Math.floor((DURACION % MILISEGUNDOS_MINUTO) / MILISEGUNDOS_SEGUNDO);

            SPAN_DIAS.innerHTML = DIAS_RESTANTES;

            let restantes   = [HORAS_RESTANTES, MINUTOS_RESTANTES, SEGUNDOS_RESTANTES];
            let spans       = [SPAN_HORAS, SPAN_MINUTOS, SPAN_SEGUNDOS];

            for (let i = 0; i < spans.length; i++) {
                if (restantes[i] > 9)
                    spans[i].innerHTML = restantes[i];
                else
                    spans[i].innerHTML = '0' + restantes[i];
            }
        }

        setInterval(actualizarTemporizador, 1000);
    }, false);
</script>
```

IV - Pruebas en diversos navegadores

IV.I – Mozilla Firefox

IV.II – Google Chrome



IV.III – Opera


Inicio / David Fajardo - Ofi
localhost/dashboard/davidfajardo/pages/index.php
David Fajardo
NOVELAS
FORO
17
BIOGRAFÍA
ACERCA DE
REGISTRARSE
Últimos mensajes del foro
Comentarios de El último elemento
dsfgdsfgdfg dfg dfgdfg sd...
David 17/11/2022 a las 19:43
Habla de lo que sea
mensaje de prueba 12 [spo...
David 12/11/2022 a las 19:55
Hablemos de anime
Respuesta al mensaje de prueba 3...
ThreeDog 17/11/2022 a las 18:02
SITIO WEB OFICIAL DE
Aviso sobre cookies
Es el usuario el que acepta las cookies y son las cookies las que quieren que el usuario acepte las cookies...
Cerrar
Aceptar

IV.IV – Microsoft Edge

ILERNA Online

12.2 Anexo II