



UNIVERSIDAD NACIONAL SAN AGUSTÍN

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

PROYECTO FINAL

WIKIPEDIA - MARKDOWN



PROGRAMACIÓN WEB I

AREQUIPA - PERÚ

2024

DOCENTE

Corrales Delgado Carlo Jose Luis

INTEGRANTES

Chipana Mamani, Andhy Brayan
Choque Sánchez, Alejandra Camila
Hatches Curo, José León Enrique
Huamaní Machaca, Gael Alexander
Pacheco Medina, Geisel Reymar

PROGRAMA DE ESTUDIOS

Escuela Profesional de Ingeniería de Sistemas



ÍNDICE

1. Introducción	2
2. Alcance del proyecto	3
a. Definición del proyecto	3
b. Objetivos del proyecto	3
c. Entregables principales	3
d. Requisitos específicos	3
3. Flujo del proyecto	4
a. Fase de implementación	4
b. Colaboración con Github	5
c. Fase de prueba y revisión	5
d. Fase de despliegue	5
4. Diagrama base de datos	6
5. Funcionalidades	6
a. Página de Inicio (index.html)	6
b. Página de listado (lista.pl)	7
c. Página de Nueva Página (nuevo.html)	7
d. Página de visualización (ver.pl)	8
e. Funcionalidades de registro e inicio de sesión	9
f. Gestión de nuevas páginas	10
g. Base de Datos y conexión	11
6. Código fuente	12 - 17
7. Conclusiones	18
8. Recomendaciones	18
9. Enlaces	19



1. Introducción

El curso de Programación Web I nos brinda la oportunidad de aprender y aplicar herramientas clave en el desarrollo de páginas web. Como parte de este aprendizaje, hemos emprendido el desarrollo de una página web inspirada en Wikipedia, diseñada para permitir la creación y edición de contenido para páginas utilizando Markdown. Esta herramienta es muy práctica, ya que su sintaxis es sencilla y fácil de entender, lo que hace que estructurar y dar formato al texto sea intuitivo. Además, Markdown permite transformar ese contenido en HTML de forma sencilla, facilitando su visualización en la web.

En este proyecto utilizaremos diversas tecnologías para construir la página web planteada. Por un lado, HTML, CSS y JavaScript nos permitirán diseñar una interfaz amigable e intuitiva, haciendo que la página sea atractiva y fácil de usar. Por otro lado, Perl será clave para implementar la lógica del servidor y gestionar las solicitudes, mientras que MariaDB será nuestra base de datos, donde almacenaremos todo el contenido generado, así como las ediciones realizadas por los usuarios.

Lo que se busca con el presente proyecto es crear una herramienta que incluya funciones como un editor de Markdown, la transformación automática de este formato a HTML y el almacenamiento organizado de la información en la base de datos. Este trabajo representa un reto que nos permitirá consolidar los conceptos aprendidos en el curso, poniendo en práctica nuestras habilidades y explorando nuevas formas de desarrollar páginas web dinámicas.



2. Alcance del proyecto

Definición del proyecto

El proyecto abarca el desarrollo de una página web funcional que simula una plataforma web similar a Wikipedia, utilizando el lenguaje de marcado Markdown como base para la creación y edición de páginas, los cuales serán convertidos a HTML para su visualización. Los usuarios podrán registrar, iniciar sesión, y gestionar artículos (crear, editar, eliminar, ver y listar) asociados a su cuenta. Todos los artículos estarán almacenados en una base de datos MariaDB y el sistema usará CGI para interactuar con el servidor, devolviendo respuestas en formato XML.

Objetivos del proyecto

Este proyecto tiene como objetivo principal crear una página web que permita gestionar contenido escrito en Markdown y transformarlo automáticamente en HTML para su visualización. La página debe ofrecer funcionalidades clave, como la creación y edición de contenido, su almacenamiento en el servidor y la visualización dinámica del listado de páginas creadas. El sistema utilizará expresiones regulares para realizar la conversión de Markdown a HTML.

Requisitos específicos

- **Página de Inicio (index.html):** Muestra el nombre completo del estudiante, curso, grupo y una breve descripción. Incluye enlaces a la página de listados y a la página para crear nuevas páginas.
- **Página de Listado (lista.pl):** Muestra los nombres de las páginas almacenadas como hiperenlaces que dirigen a la visualización de su contenido.
- **Página de Nueva Página (nuevo.html):** Proporciona un formulario con campos para el nombre y el contenido de una nueva página, que se almacenará en el servidor.
- **Página de visualización (ver.pl):** Permite ver el contenido de un documento almacenado en el servidor, transformando el código Markdown en HTML.
- Scripts CGI para la autenticación, registro de usuarios, y gestión de artículos (login.pl, register.pl, lista.pl, ver.pl, editor.pl, conexion.pl, eliminar.pl).
- Base de datos MariaDB (wikiweb1) con las tablas para almacenar la información de usuarios y artículos.



3. Flujo del proyecto

El flujo del proyecto se desarrolla en varias etapas, interconectadas a través de las distintas páginas y scripts CGI, junto con la base de datos MariaDB. A continuación se describe cómo se ha desarrollado la página:

Fase de implementación

En esta fase del proyecto se define a profundidad las distintas tareas por trabajar dentro el proyecto, además su distribución para mayor una mayor facilidad al momento de trabajar asíncronamente.

1. Página de Inicio (index.html):

Crear la página principal con enlaces funcionales a otras partes del sistema.

- Uso de HTML y CSS para diseño de página.
- Agregar enlaces a lista.pl y nuevo.html.

2. Página de Listado (lista.pl):

- Implementar un script CGI que conecte con la base de datos para extraer y mostrar los títulos de los artículos en formato de hipervínculos.
- Gestionar las solicitudes GET y construir las respuestas dinámicamente en HTML.

3. Página de Nueva Página (nuevo.html) y Script (nuevo.pl):

- Diseñar un formulario en nuevo.html para capturar datos del usuario.
- Desarrollar el script CGI (nuevo.pl) para recibir los datos, validar entradas y realizar inserciones en la base de datos.
- Confirmar las operaciones exitosas y redirigir al listado.

4. Página de Visualización (ver.pl):

- Implementar un script CGI que transforme el contenido Markdown de un artículo en HTML.
- Utilizar un algoritmo de interpretación para generar etiquetas HTML equivalentes.

5. Scripts Adicionales:

- login.pl: Validar usuarios y gestionar sesiones.
- register.pl: Registrar nuevos usuarios en la base de datos.
- editor.pl: Proporcionar funcionalidad para modificar artículos existentes.
- eliminar.pl: Permitir la eliminación de artículos específicos.



Colaboración con GitHub

En el desarrollo del proyecto utilizamos GitHub como nuestra principal herramienta colaborativa, permitiendo la distribución del trabajo mediante la creación de ramas específicas para cada funcionalidad, corrección o mejora. Tras una revisión de código se realizó la integración de los códigos trabajados. Además, GitHub nos permite manejar un control de versiones mediante el uso de commits, que no permitirás dejar en registro las tareas realizadas.

- **Distribución del trabajo:** Crear ramas para cada funcionalidad principal, como frontend, backend, etc.
- **Control de Versiones:** Realizar commits de las ramas específicas al repositorio principal tras revisiones.

Repositorio: [Proyecto final PW1](#)

Fase de pruebas y revisión

En esta etapa se desarrollará la prueba y revisión por separado de cada parte que conforma el trabajo, esto con el fin de facilitar el proceso de prueba y poder detectar de manera más fácil y sencilla un error en caso exista.

- Probar individualmente cada script CGI y funcionalidad de la base de datos.
- Verificar que todas las páginas y scripts se comuniquen correctamente.
- Revisar y corregir errores en las ramas antes de fusionarlas al repositorio principal.

Fase de despliegue

Finalmente en la fase de despliegue realizaremos una “prueba final” en la que comprobaremos mediante el servidor que todas las partes funcionen correctamente en conjunto. Esta etapa será una simulación de la presentación final para la revisión del proyecto, permitirá conocer si el proyecto está listo para esta.

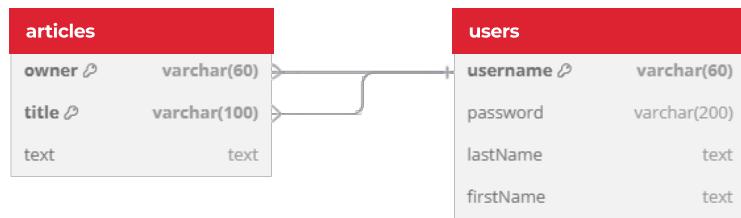
- Configuración del servidor: Configurar el servidor web dentro del entorno Docker.
- Pruebas finales: Realizar pruebas validación para comprobar el correcto funcionamiento de todas las partes desarrolladas dentro de la página.
- Entrega: Documentar el proyecto final.



4. Diagrama de base de datos

El propósito del diagrama de base de datos será proporcionar una visión clara y concisa de cómo es que se organizan y estructuran los datos en el sistema dentro del desarrollo de la página web con las funcionalidades Markdown ya vistas previamente.

Este diagrama ayudará a entender mejor cómo los datos se almacenan, cómo se interrelacionan y cómo las operaciones en la base de datos afectarán las distintas tablas.



Fuente: Elaboración propia con dbdiagram.io

5. Funcionalidades

1. Página de Inicio (index.html)

- Descripción:** La página principal que el usuario visualizará al ingresar a la página
- Funcionalidades:**
 - Muestra información de introducción: integrantes, breve descripción del curso y de la página
 - Incluye hipervínculos para ir a la página de listado de artículos y la página para crear nuevos artículos.

La captura de pantalla muestra la página de inicio 'index.html'. En la parte superior, hay un menú con los enlaces 'Nosotros', 'Ver Lista', 'Regístrate' y 'Iniciar Sesión'. El título principal es '¡Bienvenido a nuestra Wikipedia!'. Debajo del título, hay un texto que dice: 'En esta página podrás crear tu propia página basándote en el Markdown...'. A continuación, hay dos botones: 'Crear página' (en rojo) y 'Conocer más' (en negro). A la derecha, hay un ilustración de un teléfono móvil que muestra una interfaz de usuario con un editor de código. Alrededor del teléfono, hay tres burbujas de diálogo que contienen los nombres de tecnologías: 'HTML', 'PERL' y 'CSS'.

Fuente: Elaboración propia - index.html



2. Página de Nueva Página (nuevo.html)

- **Descripción:** Permite al usuario crear nuevos artículos.
- **Funcionalidades:**
 - Contiene un formulario con los siguientes campos:
 - Título del artículo.
 - Contenido (en formato Markdown).
 - Envío de datos al servidor para almacenarlos en la base de datos.
 - Validación de datos antes de enviarlos al servidor.
 - Confirmación visual de que el artículo fue agregado con éxito.
 - Enlace para regresar al listado o visualizar el nuevo artículo.

The screenshot shows a 'Crear Nueva Página' (Create New Page) form. The title field contains 'Prueba'. The content field contains the following Markdown code:
Encabezado 2
Encabezado 6
Texto negrita y cursiva
Textos negrita
Textos cursiva
[Texto](https://example.com)
Linea 1
Linea 2
Linea 3

Fuente: Elaboración propia - nuevo.html

3. Página de listado (lista.pl)

- **Descripción:** Muestra una lista de los artículos creados por el usuario actual. Genera una respuesta en formato XML a partir de datos almacenados en la base de datos
- **Funcionalidades:**
 - Utiliza el módulo CGI para recibir un parámetro llamado usuario enviado a través de una solicitud HTTP. Este parámetro identifica al propietario de los artículos que se desean consultar.
 - Si el parámetro usuario está definido, realiza una consulta SQL para buscar los artículos en la tabla wiki que pertenecen al propietario especificado (owner = ?).
 - Extrae dos columnas de la tabla: id (identificador único del artículo) y titulo (título del artículo).
 - Construye un archivo XML con los datos obtenidos de la base de datos. Cada artículo se representa como un elemento <article> con los subelementos <id> y <title>, que contienen los datos del artículo.



Wikipedia

Inicio

LISTA DE PÁGINAS

AUTOR	TÍTULO	ACCIONES
aaa	Prueba	V E X

[Volver al Inicio](#)

Fuente: *Elaboración propia - lista.html*

4. Página de visualización (ver.pl)

- **Descripción:** Permite al usuario visualizar el contenido de un artículo en formato HTML.
- **Funcionalidades:**
 - Consulta la base de datos para obtener el contenido del artículo seleccionado.
 - Convierte el contenido de Markdown a HTML de forma transparente para el usuario.
 - Muestra el contenido HTML procesado en la página.
 - Incluye un botón o enlace para regresar a “lista.pl”.

Wikipedia

Inicio

Prueba

Encabezado 2

Encabezado 6

Texto negrita y cursiva
Texto negrita
Texto cursiva
Texto
 Línea 1
 Línea 2
 Línea 3

Fuente: *Elaboración propia - ver.html*



5. Funcionalidades de registro e inicio de sesión

- **Inicio de sesión:**

- Permite a los usuarios iniciar sesión proporcionando nombre de usuario y contraseña.
- Valida las credenciales en la base de datos.
- Proporciona acceso al sistema si las credenciales son correctas.

- **Registro:**

- Permite a nuevos usuarios registrarse proporcionando nombre, apellido, nombre de usuario y contraseña.
- Almacena la información en la base de datos.
- Confirma el registro exitoso.

Wikipedia Inicio Ver lista

INICIAR SESIÓN

Usuario:

Contraseña:

Enviar

Wikipedia Inicio Ver lista

CREAR CUENTA

Nuevo Usuario:

Nueva Contraseña:

Nombres:

Apellidos:

Enviar

Fuente: Elaboración propia - register.html / login.html



6. Gestión de nuevas páginas

- **Crear artículo (nuevo.html):**

- Recibe los datos del formulario de nueva página.
- Valida y almacena el artículo en la base de datos.

- **Listar artículos (lista.pl):**

- Recupera y muestra los títulos de los artículos asociados a un usuario.

- **Visualizar artículo (ver.pl):**

- Convierte el contenido de Markdown a HTML para su presentación al usuario.

- **Editar artículo (editor.pl):**

- Permite modificar el contenido de un artículo existente.
- Guarda los cambios en la base de datos.

- **Eliminar artículo (eliminar.pl):**

- Elimina un artículo seleccionado de la base de datos.

Wikipedia Inicio Ver lista

EDITANDO LA PÁGINA

Título:

Contenido:

```
## Encabezado 2
##### Encabezado 6
***Texto negrita y cursiva***
**Texto negrita**
*Texto cursiva*
[Texto][https://example.com]
Linea 1
Linea 2
Linea 3
```

Enviar

Fuente: Elaboración propia - editor.html

Wikipedia Inicio

PÁGINA ELIMINADA EXITOSAMENTE

Usuario:aaa
Título:Prueba

Ver Mi Lista

Fuente: Elaboración propia - eliminar.html



ELEMENTO	MARKDOWN	HTML	RESULTADO
Encabezado o heading	# Encabezado 1 ## Encabezado 2 ##### Encabezado 6	<h1>Encabezado 1</h1> <h2>Encabezado 2</h2> <h6>Encabezado 6</h6>	Encabezado 1 Encabezado 2 Encabezado 6
Estilos de texto	***Texto negrita y cursiva*** **Texto negrita** *Texto cursiva*	Texto Texto Texto	Texto Texto Texto
Enlaces	[Texto](https://example.com)	Texto	<u>Texto</u>
Saltos de línea	Línea 1 Línea 2	Línea 1 Línea 2	Línea 1 Línea 2

Fuente: Elaboración propia - Tabla de funcionalidades Markdown

7. Base de Datos y conexión

- **Funcionalidad principal:**

- Convierte contenido en formato Markdown a etiquetas HTML para presentarlo de manera visual.
- Compatible con elementos básicos de Markdown como:
 - Encabezados (#, ##, ###).
 - Texto en negrita y cursiva.

Field	Type	Null	Key	Default	Extra
userName	varchar(60)	NO	PRI	NULL	
password	varchar(200)	NO		NULL	
lastName	text	NO		NULL	
firstName	text	NO		NULL	

Fuente: Elaboración propia - Base de datos: Table users

Field	Type	Null	Key	Default	Extra
owner	varchar(60)	NO	PRI	NULL	
title	varchar(100)	NO	PRI	NULL	
text	text	NO		NULL	

Fuente: Elaboración propia - Base de datos: Table articles



4. Código fuente

register.pl

En el desarrollo del proyecto utilizamos GitHub como nuestra principal herramienta colaborativa, permitiendo la distribución del trabajo mediante la creación de ramas específicas para cada funcionalidad, corrección o mejora. Tras una revisión de código se realizó la integración de los códigos trabajados. Además, GitHub nos permite manejar un control de versiones mediante el uso de commits, que no permitirás dejar en registro las tareas realizadas.

```
register.pl  
Línea de código: 68 - 109  
68 sub crearCuenta {  
69  
70     my ($username, $passw, $firstName, $lastName) = @_;  
71     my @row;  
72  
73     my $consulta = "INSERT INTO Users (userName, password, firstName, lastName) VALUES (?, ?, ?, ?)";  
74     my $sth_INSERT = $dbh->prepare($consulta);  
75  
76     eval {  
77         $sth_INSERT->execute($username, $passw, $firstName, $lastName);  
78     };  
79  
80     if (!$@) {  
81  
82         $consulta = "SELECT userName, firstName, lastName FROM Users WHERE userName = ? AND firstName = ? AND lastName = ?";  
83         my $sth_SELECT = $dbh->prepare($consulta);  
84         $sth_SELECT->execute($username, $firstName, $lastName) or die;  
85  
86         @row = $sth_SELECT->fetchrow_array;  
87  
88         $sth_SELECT->finish;  
89         $sth_INSERT->finish;  
90  
91     }  
92  
93     $dbh->disconnect;  
94  
95     return @row;  
96 }  
97  
98 sub xmlCuenta {  
99     my @row = @_;  
100    my ($username, $firstName, $lastName) = (@row[0], $row[1], $row[2]);  
101  
102    print<<XML;  
103    <user>  
104    <owner>$username</owner>  
105    <firstName>$firstName</firstName>  
106    <lastName>$lastName</lastName>  
107    </user>  
108    XML  
109 }
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1

<https://github.com/LeonHatches/Proyecto-Final-PW1>



login.pl

La lógica de este código se basa en una consulta a la base de datos para devolver información si es que los parámetros ingresados son correctos. Comenzando con la consulta de selección ingresando un userName y password; si esta es correcta devolverá información (userName, firstName y lastName) en un XML, caso contrario, el arreglo que retorna la subrutina estará vacío y el script devolverá un XML vacío.

```

login.pl                                         Línea de código: 72 - 99

72  sub verificarCuenta {
73      my $username = $_[0];
74      my $passw = $_[1];
75
76      my $consulta = "SELECT userName, firstName, lastName FROM Users WHERE userName = ? AND password = ?";
77      my $sth = $dbh->prepare($consulta);
78      $sth->execute($username, $passw);
79
80      my @row = $sth->fetchrow_array;
81
82      $sth->finish;
83      $dbh->disconnect;
84
85      return @row;
86  }
87
88  sub xmlCuenta {
89      my @row = @_;
90      my ($username, $firstName, $lastName) = (@row[0], $row[1], $row[2]);
91
92      print<<XML;
93      <user>
94          <owner>$username</owner>
95          <firstName>$firstName</firstName>
96          <lastName>$lastName</lastName>
97      </user>
98      XML
99  }

```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1
<https://github.com/LeonHatches/Proyecto-Final-PW1>

nuevo.pl

Aquí se inserta un nuevo artículo y se verifica que no haya otro con el mismo título y propietario de la siguiente forma:

Verifica si owner, title y text estan definidos (linea 49), los cuales representan al propietario, el título y el texto del artículo, si faltase alguno llama a la subrutina XMLvacio.

Si los parámetros están bien, hace una consulta SELECT para buscar si ya existe un artículo con el mismo titulo y propietario en la base de datos, si es que si se encuentra, llama a la subrutina XMLvacio, si no, ejecuta una consulta INSERT INTO para añadir el nuevo artículo (52-57)

La subrutina sucessLogin funciona como una respuesta de que la operación se realizó satisfactoriamente y genera un XML con el título y el texto del artículo recién creado (76).

**nuevo.pl**

Línea de código: 48 - 73

```

48 # Verificar que se recibieron los datos necesarios
49 if (defined $owner and defined $title and defined $text) {
50
51     # Comprobar si ya existe un artículo con el mismo título y propietario
52     my $check_sql = "SELECT * FROM Articles WHERE title = ? AND owner = ?";
53     my $check_sth = $dbh->prepare($check_sql);
54     $check_sth->execute($title, $owner);
55
56     if ($check_sth->fetchrow_array) {
57         XMLvacio();
58     } else {
59         # Insertar un nuevo artículo
60         my $sql = "INSERT INTO Articles (owner, title, text) VALUES (?, ?, ?)";
61         my $sth = $dbh->prepare($sql);
62         $sth->execute($owner, $title, $text);
63
64         # Finalizar la operación
65         $sth->finish;
66         $dbh->disconnect;
67
68         # Mostrar éxito
69         successLogin($title, $text);
70     }
71 } else {
72     XMLvacio();
73 }
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1

<https://github.com/LeonHatches/Proyecto-Final-PW1>

lista.pl

Genera una respuesta XML que va tener información de los artículos almacenados en la bd. Se guía según la variable dirección, si es que es login, busca los artículos de un usuario específico, si es dirección, consulta todos los artículos disponibles. El resultado de la consulta se pasa a la subrutina mostrar que los recorre y construye el XML para cada artículo, con el dueño owner y el título del artículo (línea 66)

lista.pl

Línea de código: 44 - 57

```

44 # Generar respuesta XML
45 print "<articles>\n";
46
47 if (defined $direccion && $direccion eq "login") {
48
49     my $userName = $cgi->param('owner');
50
51     # Consultar artículos según el propietario
52     my $sql = "SELECT owner, title FROM Articles WHERE owner = ?";
53     my $sth = $dbh->prepare($sql);
54     $sth->execute($userName);
55
56     mostrar($sth);
57     $sth->finish();
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1

<https://github.com/LeonHatches/Proyecto-Final-PW1>

**lista.pl**

Línea de código: 58 - 85

```

58 } else {
59
60     # Consultar todos los artículos
61     my $sql = "SELECT owner, title FROM Articles";
62     my $sth = $dbh->prepare($sql);
63     $sth->execute();
64
65     mostrar($sth);
66     $sth->finish();
67 }
68
69 print "</articles>";
70
71 $dbh->disconnect();
72
73
74 sub mostrar {
75     my ($sth) = @_;
76
77     while (my ($owner, $title) = $sth->fetchrow_array) {
78         print<<XML;
79         <article>
80             <owner>$owner</owner>
81             <title>$title</title>
82         </article>
83         XML
84     }
85 }
```

Fuente: *Elaboración propia - Repositorio Github: Proyecto final PWEB1*

<https://github.com/LeonHatches/Proyecto-Final-PW1>

ver.pl

Esta subrutina utiliza expresiones regulares para transformar texto en formato Markdown a HTML. Por ejemplo, las líneas que inician con #, ##, o ##### son convertidas en encabezados <h1>, <h2>, y <h6>, respectivamente. Los estilos como negrita, cursiva y tachado son identificados mediante combinaciones de asteriscos (*) y tildes (~) y luego reemplazados por etiquetas , y . Los bloques de código, delimitados por triple acento grave (```), se convierten en etiquetas <pre> y <code>, manteniendo el formato original.

Además, los enlaces en formato Markdown [texto](url) se transforman en etiquetas <a> para crear hipervínculos funcionales.

Finalmente, los saltos de línea (\n) son sustituidos por la etiqueta
, garantizando que el texto se muestre con el formato esperado en el navegador.

**ver.pl**

Línea de código: 73 - 98

```

73 # Subrutina para convertir Markdown a HTML
74 sub convertir_markdown_a_html {
75     my ($markdown) = @_;
76
77     # Convertir encabezados
78     $markdown =~ s/^##### (.+)$/<h6>$1</h6>/gm;
79     $markdown =~ s/^## (.+)$/<h2>$1</h2>/gm;
80     $markdown =~ s/^# (.+)$/<h1>$1</h1>/gm;
81
82     # Convertir estilos de texto
83     $markdown =~ s/\*\*(.+?)\*\*/<strong><em>$1</em></strong>/g;
84     $markdown =~ s/\*\*(.+?)\*\*/<strong>$1</strong>/g;
85     $markdown =~ s/\*(.+?)\*/<em>$1</em>/g;
86     $markdown =~ s/~~(.+?)~~/<del>$1</del>/g;
87
88     # Convertir bloques de código
89     $markdown =~ s/```(.+?)```<pre><code>$1</code></pre>/gs;
90
91     # Convertir enlaces
92     $markdown =~ s/[(.+?)]((.+?))<a href="$2">$1</a>/g;
93
94     # Convertir saltos de línea a <br>
95     $markdown =~ s/\n/<br>\n/g;
96
97     return $markdown;
98 }
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1

<https://github.com/LeonHatches/Proyecto-Final-PW1>

editor.pl

El código actualiza el contenido de un artículo ya existente, con los parámetros owner, title, y text ejecuta una consulta UPDATE para editar y reemplazar el contenido del texto del artículo con el título y propietario mandados (línea 53,54). Después llama a la subrutina editar que crea un xml con el título y el texto del artículo editado (57). Si no se proporcionan los parámetros necesarios, se llama a la subrutina XMLvacio, que retorna un XML vacío sin contenido, que sirve como un manejo de errores.

editor.pl

Línea de código: 49 - 62

```

49 # Verificar que se recibieron los datos necesarios
50 if (defined $owner and defined $title and defined $text) {
51
52     # Comprobar si ya existe un artículo con el mismo título y propietario
53     my $check_sql = "UPDATE Articles SET text = ? WHERE title = ? AND owner = ?";
54     my $check_sth = $dbh->prepare($check_sql);
55     $check_sth->execute($text, $title, $owner);
56
57     editar($title, $text);
58     $check_sth->finish;
59
60 } else {
61     XMLvacio();
62 }
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1



eliminar.pl

Si el artículo existe según su propietario y título, se procede a eliminarlo mediante una sentencia DELETE. Después de una eliminación exitosa, se llama a la subrutina mostrarEliminar, que genera un XML confirmando la acción, mostrando el owner y el title del artículo eliminado. Si no existe, se llama a la subrutina XMLvacío, que devuelve un XML con campos vacíos.

eliminar.pl

Línea de código: 48 - 71

```

48 if (defined $owner && defined $title) {
49
50     # Comprobar si ya existe un artículo con el mismo título y propietario
51     my $check_sql = "SELECT * FROM Articles WHERE title = ? AND owner = ?";
52     my $check_sth = $dbh->prepare($check_sql);
53     $check_sth->execute($title, $owner);
54
55     if ($check_sth->fetchrow_array) {
56
57         # eliminar el artículo
58         my $sql = "DELETE FROM Articles WHERE owner = ? AND title = ?";
59         my $sth = $dbh->prepare($sql);
60         $sth->execute($owner, $title);
61
62         mostrarEliminar($owner, $title);
63
64         $sth->finish;
65
66     } else {
67         XMLvacío();
68     }
69 } else {
70     XMLvacío();
71 }
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1

<https://github.com/LeonHatches/Proyecto-Final-PW1>

article.pl

En esta sección, queremos obtener el contenido de un artículo en la base de datos usando el owner (propietario) y el title (título) como criterios de búsqueda. Si la consulta devuelve resultados válidos, se extrae el contenido del texto del artículo y se imprime en formato XML con las etiquetas <owner>, <title> y <text>. Si no hay artículo que coincida, se llama a una subrutina XMLvacío, que devuelve un XML con campos vacíos.

article.pl

Línea de código: 55 - 57

```

55 # Consultar el artículo
56 my $sth = $dbh->prepare("SELECT text FROM Articles WHERE owner = ? AND title = ?");
57 $sth->execute($owner, $title);
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1

<https://github.com/LeonHatches/Proyecto-Final-PW1>



article.pl

Línea de código: 58 - 71

```
58     # Obtener resultados y generar XML
59     if (my $row = $sth->fetchrow_hashref) {
60         my $text = $row->{text};
61         print<<XML;
62         <article>
63             <owner>$owner</owner>
64             <title>$title</title>
65             <text>$text</text>
66         </article>
67     XML
68
69     } else {
70         XMLvacio();
71 }
```

Fuente: Elaboración propia - Repositorio Github: Proyecto final PWEB1

<https://github.com/LeonHatches/Proyecto-Final-PW1>

7. Conclusiones

Dentro del proyecto se ha logrado integrar múltiples tecnologías como HTML, CSS, JavaScript, CGI, y MariaDB para crear una página que genera nuevas páginas en base al lenguaje de marcado Markdown. Los usuarios pueden gestionar su contenido, además de registrarse e iniciar sesión. Además, la implementación de variables de sesión permite la personalización de la página, aumentando la seguridad en un supuesto en el que se desarrolle una página que tenga estos requerimientos.

GitHub como herramienta colaborativa nos ha permitido la distribución de tareas y control de versiones, facilitando la integración de las partes trabajadas.

8. Recomendaciones

Recomendamos trabajar en el proyecto de manera organizada, avanzando por etapas y enfocándose en funcionalidades específicas, como el inicio de sesión o la gestión de artículos. Es fundamental utilizar GitHub como herramienta principal para la colaboración, creando ramas para cada funcionalidad y fusionándolas regularmente para mantener la organización dentro del proyecto.

Además, se sugiere realizar pruebas constantes después de desarrollar cada parte, asegurando su correcto funcionamiento antes de continuar. Documentar tanto el proceso como las configuraciones que se establezcan permitirá facilitar las mejoras que se requieran.



9. Enlaces

GitHub del proyecto:

<https://github.com/LeonHatches/Proyecto-Final-PWI>

Avance del Proyecto:

<https://youtu.be/lxLr4oa3lEc?feature=shared>

Proyecto Finalizado:

https://youtu.be/QxYt_Wz85Qw