HITO INDIVIDUAL

Programación

Ivan Guijarro Soto

Item1.

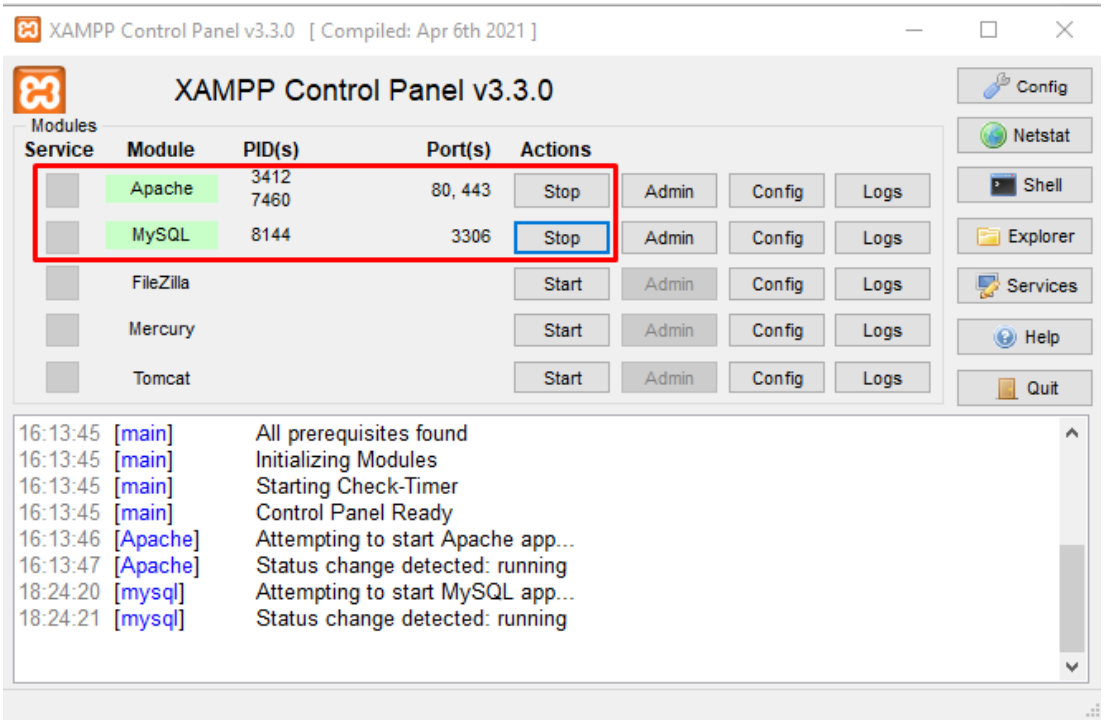
Desplegada en un entorno de pruebas consistente en un servidor Apache local

XAMPP es un paquete de software libre que se utiliza para crear y ejecutar aplicaciones web en un entorno local en un ordenador. La sigla XAMPP significa X (para cualquier sistema operativo), Apache, MySQL, PHP y Perl, que son los componentes principales del paquete.

XAMPP incluye un servidor web Apache, una base de datos MySQL y los lenguajes de programación PHP y Perl, entre otros componentes, lo que permite a los desarrolladores web crear y probar sus aplicaciones en su propio ordenador sin necesidad de tener una conexión a Internet.

La instalación de XAMPP es bastante sencilla y no requiere mucha configuración adicional. Una vez instalado, los desarrolladores pueden comenzar a crear y probar aplicaciones web en su propio ordenador, lo que les permite trabajar de forma más eficiente y sin tener que preocuparse por posibles errores o problemas de conectividad.

En resumen, XAMPP es una herramienta útil para los desarrolladores web que desean crear y probar sus aplicaciones en un entorno local, sin necesidad de una conexión a Internet y con un proceso de configuración relativamente sencillo.



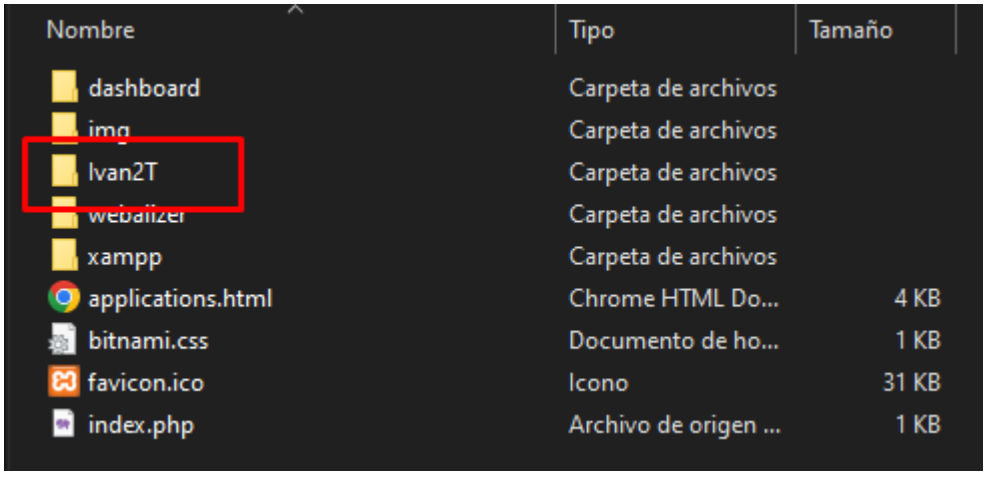
Para ver un proyecto en XAMPP, hay que seguir los siguientes pasos:

1. Coloca los archivos del proyecto en la carpeta htdocs: La carpeta htdocs es la carpeta raíz del servidor web Apache de XAMPP. Para poder visualizar un proyecto en XAMPP, debes colocar todos los archivos y carpetas relacionados con tu proyecto dentro de esta carpeta.

2. Inicia los servicios de XAMPP: Una vez que hayas colocado los archivos del proyecto en la carpeta htdocs, debes asegurarte de que los servicios de XAMPP estén en ejecución. Abre el panel de control de XAMPP y asegúrate de que el servidor Apache y MySQL estén activos.

3. Abre el navegador web: Abre el navegador web que prefieras y escribe la dirección "localhost/nombre-del-proyecto" en la barra de direcciones. Si el proyecto se llama "mi-proyecto", por ejemplo, debes escribir "localhost/mi-proyecto".

4. Comprueba que el proyecto se muestra correctamente: Si el proyecto está configurado correctamente, deberías poder verlo en tu navegador web. Si aparece algún error o problema, asegúrate de que los archivos estén en la ubicación correcta y de que los servicios de XAMPP estén en ejecución.



Item2.

En la página de arranque / inicio, el cliente nos pide que hagamos una explicación de las diferencias entre lenguajes de programación orientada a objetos, a eventos y lenguajes procedimentales. Esta explicación será texto con marcado HTML y diseño CSS.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="index.css">
    <title>Home Page</title>
</head>
<body>
</head>
<body>
    <header>
        <h1>Hito Individual</h1>
        <nav>
            <a href="index.html">Inicio</a>
            <a href="formulario.html">Publicar un Post</a>
            <a href="publicados.php">Publicaciones</a>
            <a href="login.html">Login</a>
            <a href="modificar.html">Modificar</a>
        </nav>
    </header>
    <div class="container">
        <br>
        <h2>Lenguajes de programación orientada a objetos (OOP)</h2>
        <p>Los lenguajes de programación orientada a objetos se basan en el concepto de "objetos", que son
        instancias de clases que contienen datos y funciones relacionadas. La programación orientada a objetos se
        centra en el encapsulamiento, la herencia y el polimorfismo. En este paradigma de programación, los
        programas se diseñan y construyen mediante la definición de clases y objetos, y las relaciones entre ellos.
            Algunos ejemplos de lenguajes de programación orientada a objetos son Java, C++, Python, Ruby y PHP.</p>
        <br>
        <br>
        <h2>Lenguajes de programación orientada a eventos (EOP)</h2>
        <p>Los lenguajes de programación orientada a eventos se basan en la idea de que los programas son reactivos
        a los eventos que ocurren en el entorno. Estos eventos pueden ser acciones del usuario, como hacer clic en
        un botón o mover el mouse, o eventos del sistema, como recibir una señal o recibir datos de una red. La
        programación orientada a eventos se centra en la creación de "manejadores de eventos" que responden a estos
        eventos.
            Algunos ejemplos de lenguajes de programación orientada a eventos son JavaScript, Visual Basic y C#.</p>
        <br>
        <br>
        <h2>Lenguajes de programación procedimentales</h2>
        <p>Los lenguajes de programación procedimentales se basan en la idea de que los programas son una secuencia
        de instrucciones que se ejecutan en orden. Los programas se dividen en pequeñas funciones que realizan
        tareas específicas y se combinan para crear programas más complejos. En este paradigma de programación, el
        énfasis está en la "proceduralidad" o en el control de flujo.
            Algunos ejemplos de lenguajes de programación procedimentales son C, Fortran y Pascal.</p>
    </div>
</body>
</html>

</body>
</html>
```

Este código HTML representa una página web básica que tiene un encabezado (header), una barra de navegación (nav) y un contenedor (div) con tres secciones que describen diferentes paradigmas de programación: programación orientada a objetos (OOP), programación orientada a eventos (EOP) y programación procedimental.

En el encabezado, se encuentra un título "Hito Individual" y una barra de navegación con enlaces a diferentes páginas web: "Inicio", "Publicar un Post", "Publicaciones", "Login" y "Modificar".

Dentro del contenedor, se incluyen tres secciones: "Lenguajes de programación orientada a objetos", "Lenguajes de programación orientada a eventos" y "Lenguajes de programación procedimentales", cada una con una descripción breve sobre el concepto y algunos ejemplos de los lenguajes de programación que pertenecen a cada paradigma.

Además, se ha incluido un archivo CSS externo llamado "index.css" que se utiliza para dar estilo a la página.

```css
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
}
header {
    background-color: #333;
    color: white;
    padding: 10px;
    display: flex;
    justify-content: space-between;
}
header h1 {
    margin: 0;
}
nav {
    display: flex;
}
nav a {
    color: white;
    text-decoration: none;
    padding: 10px;
    display: block;
    text-align: center;
}
nav a:hover {
    background-color: white;
    color: #333;
    transition: background-color 0.3s, color 0.3s;
}
.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 50px;
    text-align: center;
}
```

La primera regla establece que se quiten los márgenes y rellenos (margin y padding) predeterminados del body y que se utilice la fuente "Arial" o cualquier fuente genérica sin serifa para todo el texto.

La segunda regla define el estilo para el encabezado (header) de la página, que incluye un fondo de color oscuro (#333), texto en blanco, un relleno (padding) de 10 píxeles y se establece como un contenedor flex con espacio entre sus elementos.

La tercera regla establece que el título dentro del encabezado (h1) no tenga márgenes.

La cuarta regla define el estilo para la barra de navegación (nav), que se establece como un contenedor flex.

La quinta regla define el estilo para los enlaces (a) dentro de la barra de navegación, que tienen texto blanco, no tienen subrayado (text-decoration: none), un relleno de 10 píxeles, se muestran como un bloque y se centran en el contenedor.

La sexta regla establece que cuando el cursor del mouse se sitúa sobre un enlace de la barra de navegación, su color de fondo se convierte en blanco y el texto en negro (#333), y se produce una transición de 0,3 segundos para el cambio de color de fondo y texto.

La última regla define el estilo para el contenedor principal (div.container) que contiene los párrafos de texto. Se establece un ancho máximo de 800 píxeles, se centra horizontalmente en la página (margin: 0 auto), tiene un relleno de 50 píxeles y se centra el texto en el contenedor.

Item3.

Este item consiste en diseñar una página que permita al usuario escribir un post del blog de cliente. Se realizará un formulario que nos pida, email del autor, título, contenido, fecha de publicación imagen. La imagen puede ser considerada para almacenar en la base de datos como texto o como objeto.

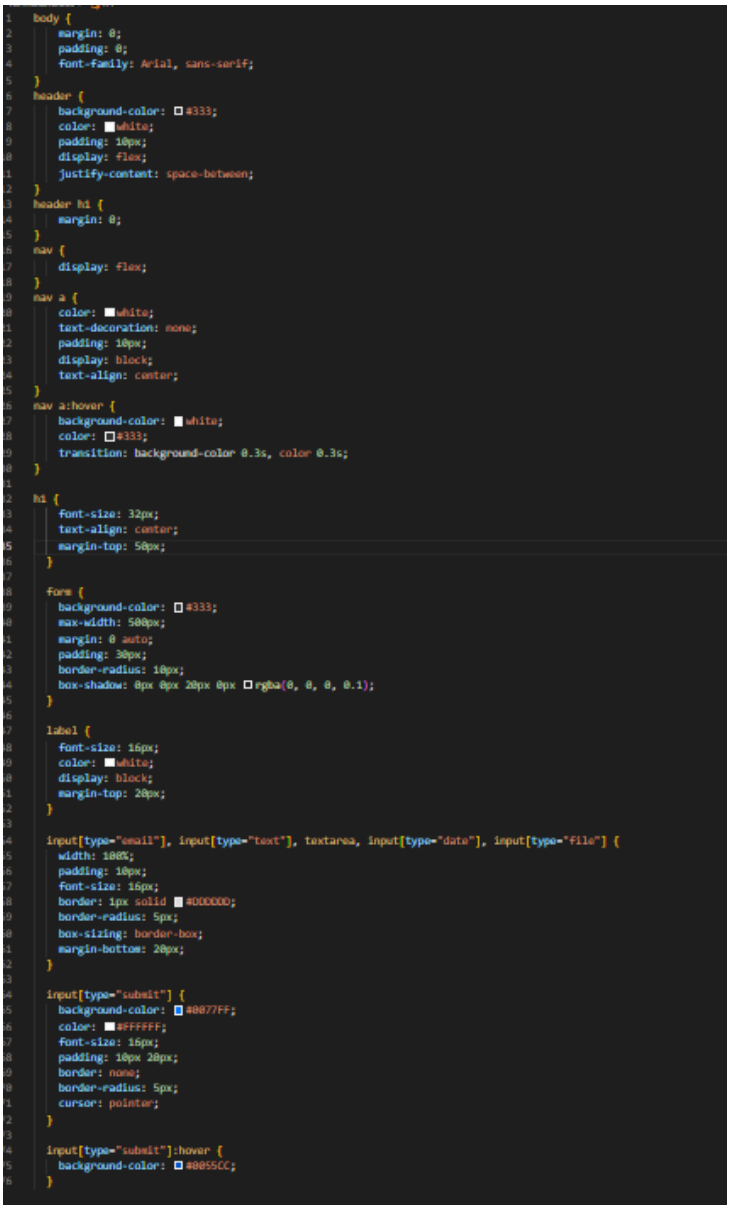
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="formulario.css">
    <title>Document</title>
</head>
<header>
    <h1>Hito Individual</h1>
    <nav>
      <a href="index.html">Inicio</a>
      <a href="formulario.html">Publicar un Post</a>
      <a href="publicados.php">Publicaciones</a>
      <a href="login.html">Login</a>
      <a href="modificar.html">Modificar</a>
    </nav>
</header>
<body>
<h1>Nuevo post</h1>
    <form method="post" action="formulario.php">
      <label for="email">Email:</label>
      <input type="email" id="Email" name="Email" required><br>

      <label for="titulo">Título:</label>
      <input type="text" id="Titulo" name="Titulo" required><br>

      <label for="contenido">Contenido:</label>
      <textarea id="Contenido" name="Contenido" required></textarea><br>

      <label for="fecha">Fecha de publicación:</label>
      <input type="date" id="Fecha" name="Fecha" required><br>

      <label for="imagen">Imagen:</label>
      <input type="file" id="Imagen" name="Imagen"><br>

      <input type="submit" value="Registrar" name="enviar">">
    </form>
</body>
</html></htML>
```

El encabezado del sitio web incluye un título "Hito Individual" y una barra de navegación con enlaces a diferentes páginas. El cuerpo del documento HTML contiene un formulario con campos para ingresar información sobre el post a publicar. Los campos incluyen el correo electrónico del autor, el título del post, el contenido del post, la fecha de publicación y una opción para cargar una imagen.

El formulario utiliza el método POST para enviar la información ingresada por el usuario a una página llamada "formulario.php". Cuando el usuario presiona el botón "Registrar", los datos se envían a esa página para ser procesados.

En resumen, este código HTML y CSS crea un formulario básico para que los usuarios ingresen información y publiquen un post en un sitio web.

Este el CSS del HTML del formulario :



```css
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
}
header {
    background-color: #333;
    color: white;
    padding: 10px;
    display: flex;
    justify-content: space-between;
}
header h1 {
    margin: 0;
}
nav {
    display: flex;
}
nav a {
    color: white;
    text-decoration: none;
    padding: 10px;
    display: block;
    text-align: center;
}
nav a:hover {
    background-color: white;
    color: #333;
    transition: background-color 0.3s, color 0.3s;
}

h1 {
    font-size: 32px;
    text-align: center;
    margin-top: 50px;
}

form {
    background-color: #333;
    max-width: 500px;
    margin: 0 auto;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0px 0px 20px 0px rgba(0, 0, 0, 0.1);
}

label {
    font-size: 16px;
    color: white;
    display: block;
    margin-top: 20px;
}

input[type="email"], input[type="text"], textarea, input[type="date"], input[type="file"] {
    width: 100%;
    padding: 10px;
    font-size: 16px;
    border: 1px solid #DDDDDD;
    border-radius: 5px;
    box-sizing: border-box;
    margin-bottom: 20px;
}

input[type="submit"] {
    background-color: #0077FF;
    color: #FFFFFF;
    font-size: 16px;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

input[type="submit"]:hover {
    background-color: #0055CC;
}
```

```php
<?php
require_once('registro.php');

if(isset($_POST["enviar"])) {

  echo "El Formulario se ha enviado con éxito";

} else {
  echo "No se ha enviado el formulario";
}
```

Este código PHP comprueba si el formulario ha sido enviado o no. Si el botón de envío del formulario se ha pulsado y se ha recibido la información del formulario, es decir, si la variable $_POST["enviar"] está definida, entonces el código imprimirá "El Formulario se ha enviado con éxito". De lo contrario, si la variable $_POST["enviar"] no está definida, se imprimirá "No se ha enviado el formulario".

Es importante tener en cuenta que este código es solo una prueba básica para comprobar si el formulario se ha enviado correctamente, y no realiza ninguna acción adicional con la información recibida. Si se quiere procesar la información recibida del formulario, se deben incluir instrucciones adicionales para manejar los datos del formulario recibidos.

Y este es el registro.php que contiene lo siguiente:

```php
<?php
$Email = $_POST['Email'];
$Titulo = $_POST['Titulo'];
$Contenido = $_POST['Contenido'];
$Fecha = $_POST['Fecha'];


$conexion = new mysqli('localhost', 'root','','test' );
$consulta = "INSERT INTO `hito` (`Email`, `Titulo`, `Contenido`, `Fecha`) VALUES (?, ?,
$insertar = $conexion->prepare($consulta);
$resultado = $insertar->execute([$Email, $Titulo, $Contenido, $Fecha]);
```

Primero, se obtienen los valores enviados por el formulario mediante la superglobal $_POST. Luego, se establece una conexión con la base de datos utilizando la clase mysqli y se prepara una consulta SQL con la sentencia INSERT INTO para insertar los valores en la tabla "hito".

La consulta preparada se construye utilizando placeholders (?), que son sustituidos por los valores reales mediante el método execute. De esta forma, se evita que los valores ingresados por el usuario puedan ser interpretados como código SQL y manipular la base de datos de forma malintencionada.

El resultado de la inserción se almacena en la variable $resultado, que puede ser utilizada para comprobar si la operación fue exitosa o no.

Email	Titulo	Contenido	Fecha
ivan@gmail.com	Prueba	oacfnqweocfwrhfwergwevrvgveg	2023-02-28
ivan@gmail.com	Prueba	oacfnqweocfwrhfwergwevrvgveg	2023-02-28
ivan@gmail.com	Prueba	oacfnqweocfwrhfwergwevrvgveg	2023-02-28
ivan@gmail.com	Prueba	q2435342523452345	2023-02-28
ivan@gmail.com	Prueba231	sisisisiissisis	2023-02-28

Item4.

En esta sección, se muestra una página con todas las entradas publicadas por todos los usuarios. Deberemos utilizar un diseño atractivo y usable.

```php
<?php
$consulta = "select * from hito";
$conexion = new mysqli('localhost', 'root','','test' );

$insertar = $conexion->prepare($consulta);
$resultado = $conexion->query($consulta);

echo("<table border='1'>");
    echo("<tr>");
        echo("<th>Correo</th>");
        echo("<th>Título</th>");
        echo("<th>Contenido</th>");
        echo("<th>Fecha</th>");
    echo("</tr>");
    while($row=$resultado->fetch_assoc()){
        echo("<tr>");
            echo("<td>".$row["Email"]."</td>");
            echo("<td>".$row["Titulo"]."</td>");
            echo("<td>".$row["Contenido"]."</td>");
            echo("<td>".$row["Fecha"]."</td>");
        echo("</tr>");
    }
echo("</table>");
```

Primero se define la consulta SQL utilizando la variable $consulta. Luego se establece una conexión a la base de datos utilizando la función mysqli().

A continuación, se prepara la consulta utilizando la función prepare() y se ejecuta con la función query(). Los resultados de la consulta se guardan en la variable $resultado.

Se crea una tabla HTML con encabezados de columna para "Correo", "Título", "Contenido" y "Fecha". Luego, en un bucle while, se recorren los resultados de la consulta y se muestran en filas de la tabla. Finalmente, se cierra la tabla HTML.

Este código es útil para mostrar los registros de la base de datos en una página web, por ejemplo, en la sección "Publicaciones" de un blog.

Item5

Tendremos una sección para eliminar / actualizar entradas. No es necesario implementar esta funcionalidad, pero sí que necesitamos que el usuario deba autenticarse para poder acceder.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Botones de Eliminar y Actualizar</title>
    <link rel="stylesheet" type="text/css" href="modificar.css">
</head>
<body>
    <header>
        <h1>Hito Individual</h1>
        <nav>
            <a href="index.html">Inicio</a>
            <a href="formulario.html">Publicar un Post</a>
            <a href="publicados.php">Publicaciones</a>
            <a href="login.html">Login</a>
            <a href="modificar.html">Modificar</a>
        </nav>
    </header>
    <div class="contenedor">
        <button class="Eliminar">Eliminar</button>
        <button class="Actualizar">Actualizar</button>
    </div>
</body>
</html>
```

La sección <head> contiene información meta sobre la página, incluyendo el título de la página y la referencia a un archivo CSS externo llamado "modificar.css".

La sección <body> contiene el contenido principal de la página web. Hay un encabezado (<header>) que incluye un logotipo y un menú de navegación (<nav>) con varios enlaces a otras páginas.

Dentro del cuerpo hay un contenedor ( ) que contiene dos botones (<button>) etiquetados como "Eliminar" y "Actualizar".

El archivo también incluye una referencia a otro archivo HTML (modificar.html) que probablemente contenga el formulario y la funcionalidad necesaria para eliminar y actualizar publicaciones de la base de datos. El archivo CSS asociado, "modificar.css", probablemente contiene estilos visuales para la página web.

```php
<?php
$conexion = new mysqli('localhost', 'root','','test' );
$Email = $_POST['Email'];
$Password = $_POST['Password'];


$stmt = $conexion->query("SELECT * FROM `user`");
if($Email=="ivan@gmail.com" and $Password=='1234'){
    header('location:modificar.html');
    session_start();
}else{
    echo('No puedes acceder. Vuelve a intentarlo');
}
```

En él, se establece una conexión a una base de datos MySQL utilizando la clase mysqli. Luego, se obtienen los valores de correo electrónico y contraseña que fueron enviados a través de un formulario HTML utilizando el método POST.

El archivo consulta la tabla user en la base de datos para comparar las credenciales ingresadas con las almacenadas en la base de datos. Si el correo electrónico y la contraseña coinciden con las de un usuario registrado, se inicia una sesión y se redirige al usuario a la página de modificación (modificar.html). Si las credenciales no coinciden, se muestra un mensaje de error.

Es importante tener en cuenta que este código no es seguro y no debe utilizarse en un entorno de producción sin realizar cambios significativos para proteger contra ataques maliciosos.

Fase 1

Esta primera fase de diseño y análisis nos debe permitir abordar la implementación de la aplicación web con garantías. Para ello, proponemos explicar con un algoritmo la tarea de publicar una entrada.

Un algoritmo es un conjunto ordenado de instrucciones o reglas que se siguen para llevar a cabo una tarea o resolver un problema en un número finito de pasos. En informática, los algoritmos son utilizados para resolver problemas mediante la escritura de programas informáticos que implementen esos algoritmos. Los algoritmos pueden ser expresados en lenguaje natural, gráficos, pseudocódigo o en un lenguaje de programación específico. La correcta definición y diseño de un algoritmo es fundamental para lograr una solución eficiente y correcta de un problema.

Algoritmo para el desarrollo de la aplicación web corporativa:

1. Definir los requisitos del cliente: Se deben cumplir los requisitos mencionados en la descripción del proyecto.
2. Elegir las herramientas y tecnologías: Se utilizarán HTML5, CSS3, JavaScript y PHP, además de una base de datos MySQL. También se elegirán las librerías y herramientas que sean necesarias para cumplir con los requisitos.
3. Configurar un entorno de pruebas: Se utilizará un servidor Apache local para desplegar la aplicación y realizar pruebas.
4. Crear la página de inicio: Se diseñará una página de inicio que explique las diferencias entre lenguajes de programación orientada a objetos, a eventos y lenguajes procedimentales. Se almacenará la IP del equipo que está accediendo y la fecha de acceso en una cookie mediante PHP.
5. Diseñar el formulario de publicación de entradas: Se creará un formulario que solicite al usuario su email, título, contenido, fecha de publicación e imagen (opcional). La información será almacenada en la base de datos MySQL.
6. Confirmar la publicación de la entrada: Al enviar el formulario, se confirmará al autor que la entrada ha sido publicada correctamente.
7. Diseñar la página de visualización de entradas: Se creará una página que muestre todas las entradas publicadas por todos los usuarios. Se utilizará un diseño atractivo y usable.
8. Diseñar la sección de autenticación: Se creará una sección para eliminar o actualizar entradas. Para acceder a esta sección, el usuario deberá autenticarse mediante un nombre de usuario y contraseña válidos.
9. Implementar la autenticación de usuarios: Se verificará si el usuario ha iniciado sesión con un nombre de usuario y contraseña válidos. Si no es así, se redirigirá

al usuario a la página de inicio de sesión. Si el usuario está autenticado, se mostrará la página de eliminación o actualización de entradas.

10. Implementar la funcionalidad de eliminación de entradas: Se permitirá al usuario seleccionar la entrada que desea eliminar y confirmar la acción. Si la acción es confirmada, se eliminará la entrada de la base de datos y se mostrará un mensaje de confirmación al usuario.
11. Implementar la funcionalidad de actualización de entradas: Se permitirá al usuario seleccionar la entrada que desea actualizar y ser redirigido a un formulario pre-populado con la información actual de la entrada. El usuario podrá editar la información de la entrada en el formulario y enviarlo para actualizar la entrada en la base de datos. Si la actualización es exitosa, se mostrará un mensaje de confirmación al usuario. Si hay algún error, se mostrará un mensaje de error al usuario.

Algoritmo para autenticación de usuario en la sección de eliminación/actualización de entradas:

1. Verificar si se ha iniciado sesión con un usuario válido. Si no, redirigir a la página de inicio de sesión.
2. Si el usuario está autenticado, mostrar la página de eliminación/actualización de entradas.
3. Para eliminar una entrada, el usuario debe seleccionar la entrada que desea eliminar y confirmar la acción.
4. Si la acción es confirmada, eliminar la entrada de la base de datos y mostrar un mensaje de confirmación al usuario.
5. Para actualizar una entrada, el usuario debe seleccionar la entrada que desea actualizar y ser redirigido a un formulario pre-populado con la información actual de la entrada.
6. El usuario puede editar la información de la entrada en el formulario y enviarlo para actualizar la entrada en la base de datos.
7. Si la actualización es exitosa, mostrar un mensaje de confirmación al usuario. Si hay algún error, mostrar un mensaje de error al usuario.

Nombre del caso de uso: Administrar publicaciones del blog

Actor principal: Administrador

Objetivo: Permitir al administrador acceder a las publicaciones del blog y realizar acciones de gestión, como eliminar o actualizar las entradas.

Flujo básico:

1. El administrador accede a la sección de gestión de publicaciones.
2. La aplicación muestra una lista con todas las entradas publicadas por los usuarios.
3. El administrador selecciona una entrada de la lista para eliminar o actualizar.
4. Si el administrador elige eliminar la entrada, la aplicación solicita una confirmación antes de eliminarla de la base de datos.
5. Si el administrador elige actualizar la entrada, la aplicación muestra el formulario de actualización con los datos de la entrada seleccionada.
6. El administrador modifica los campos que desee y envía el formulario.
7. La aplicación valida los datos y actualiza la entrada en la base de datos.
8. La aplicación muestra un mensaje de confirmación de la acción realizada.

Flujos alternativos:

- Si el administrador no ha iniciado sesión, la aplicación le solicita que lo haga antes de acceder a la sección de gestión de publicaciones.
- Si no existen publicaciones previamente creadas, la aplicación muestra un mensaje indicando que no hay entradas para gestionar.
- Si el administrador decide cancelar la acción de eliminación o actualización, la aplicación muestra un mensaje indicando que la acción ha sido cancelada.
- Si la entrada no puede ser eliminada o actualizada por alguna razón (por ejemplo, porque está siendo utilizada por otro usuario), la aplicación muestra un mensaje de error indicando el problema.

Algoritmo, caso de uso y desarrollo de código son conceptos diferentes relacionados con el proceso de desarrollo de software. A continuación, se describen brevemente sus diferencias:

- **Algoritmo**: Es una secuencia finita de pasos lógicos que permiten solucionar un problema específico. En el contexto del desarrollo de software, un algoritmo puede utilizarse para diseñar la lógica de un programa, estableciendo las instrucciones que deben seguirse para alcanzar un objetivo determinado. El algoritmo no está limitado a un lenguaje de programación específico y puede representarse en forma de diagramas de flujo, pseudocódigo, entre otros.

- **Caso de uso**: Es una técnica de modelado que se utiliza para describir las interacciones entre un sistema y sus actores, es decir, las personas o sistemas externos que interactúan con él. Un caso de uso describe un escenario específico en el que un usuario interactúa con el sistema para lograr un objetivo específico. Los casos de uso se utilizan para definir los requisitos funcionales del sistema, es decir, qué debe hacer el sistema para satisfacer las necesidades del usuario.

- **Desarrollo de código**: Es el proceso de traducir los requisitos y especificaciones del sistema en código fuente que pueda ser interpretado por una máquina. El código se escribe en un lenguaje de programación específico y sigue una sintaxis determinada. El desarrollo de código implica escribir el código, probarlo, depurarlo y finalmente integrarlo en el sistema para que pueda ser utilizado.