

**PROGRAMACIÓN
HITO INDIVIDUAL
2º TRIMESTRE**



HTML



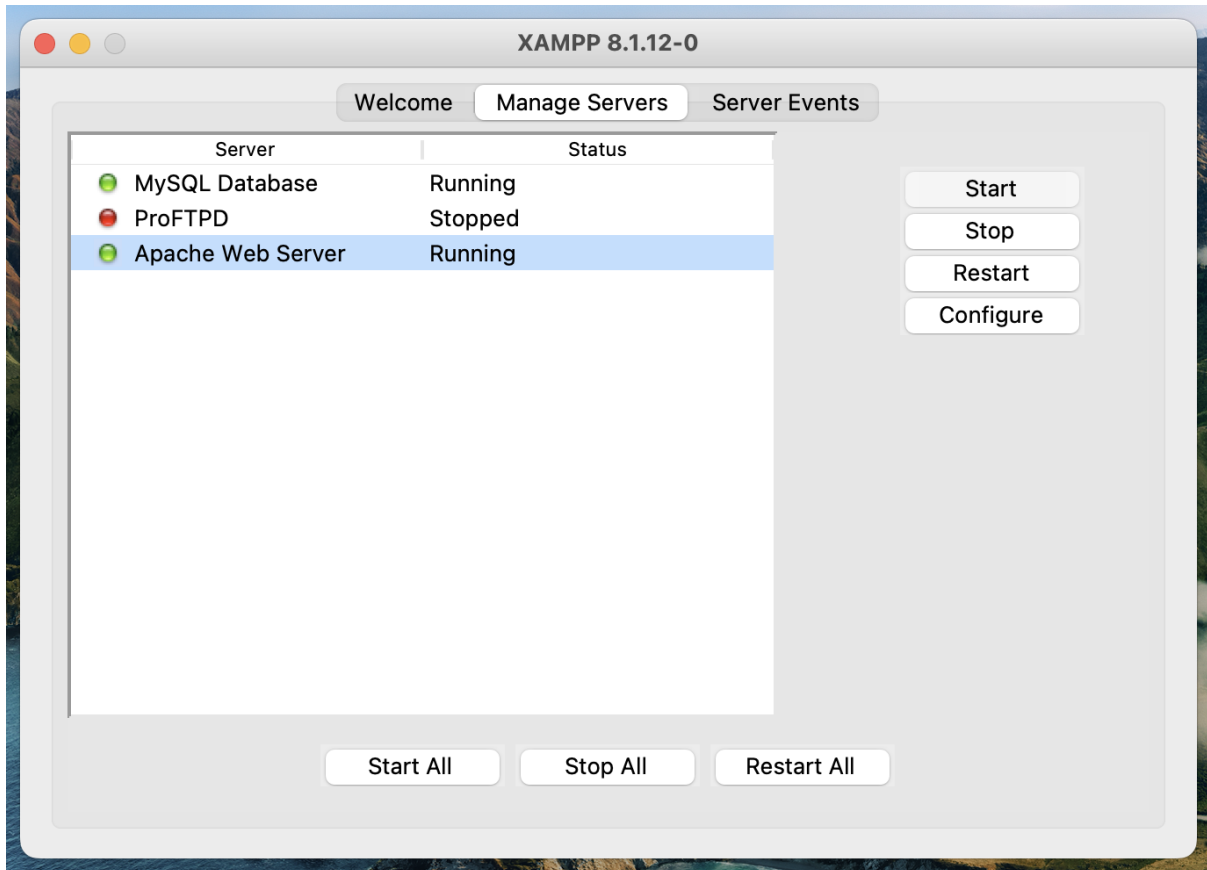
CSS



XAMPP

Paula Cubero
CampusFP Getafe

Ítem 1. Desplegada en un entorno de pruebas consistente en un servidor Apache local.



Ítem 2. 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.

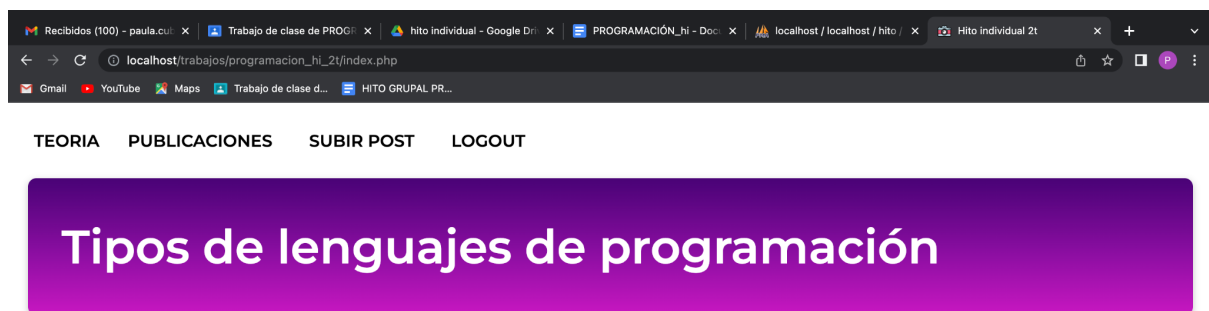
Programación Orientada a Objetos (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

Lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de este paradigma. Esto nos ayuda muchísimo en sistemas grandes, ya que en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema. La Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener.

Programación Orientada a Eventos (POE) La programación dirigida por eventos es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.

Se caracteriza por el desarrollo de aplicaciones en entornos gráficos que permiten crear formularios con botones de comandos, cuadros de texto y muchos otros tipos de controles, además de incluir las funciones propias de un lenguaje de programación de alto nivel para codificar todos los eventos e incluso, permitir enlazar a bases de datos.

Lenguajes procedimentales. Son aquellos que se basan en una estructura secuencial y jerárquica, y que ejecutan acciones de acuerdo con reglas precisas. Los lenguajes procedimentales están diseñados para dar instrucciones precisas a los sistemas sobre cómo deben comportarse ante ciertos estímulos o valores de entrada. Con este lenguaje se pueden construir herramientas digitales que procesan valores de entrada para aplicar ciertas reglas y arrojar valores de salida; siempre y cuando cumplan con los requisitos del sistema.



Programación Orientada a Objetos (POO)

Es un paradigma de programación que se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

Lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de este paradigma. Esto nos ayuda muchísimo en sistemas grandes, ya que en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

La Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener.

Programación Orientada a Eventos (POE)

La programación orientada a eventos es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.

Se caracteriza por el desarrollo de aplicaciones en entornos gráficos que permiten crear formularios con botones de comandos, cuadros de texto y muchos otros tipos de controles, además de incluir las funciones propias de un lenguaje de programación de alto nivel para codificar todos los eventos e incluso, permitir enlazar a bases de datos.



Programación Orientada a Objetos (POO)

Es un paradigma de programación que se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

Lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de este paradigma. Esto nos ayuda muchísimo en sistemas grandes, ya que en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

La Programación Orientada a objetos permite que el código sea reutilizable, organizado y fácil de mantener.

Programación Orientada a Eventos (POE)

La programación orientada a eventos es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.

Se caracteriza por el desarrollo de aplicaciones en entornos gráficos que permiten crear formularios con botones de comandos, cuadros de texto y muchos otros tipos de controles, además de incluir las funciones propias de un lenguaje de programación de alto nivel para codificar todos los eventos e incluso, permitir enlazar a bases de datos.

Lenguajes procedimentales

Son lenguajes imperativos, que se basan en una estructura secuencial y jerárquica. Están diseñados para dar instrucciones precisas a los sistemas sobre cómo deben comportarse ante ciertos estímulos o valores de entrada. Con este lenguaje se pueden construir herramientas digitales que procesen valores de entrada para aplicar ciertas reglas y arrojar valores de salida; siempre y cuando cumplan con los requisitos del sistema.

© 2023 Hito individual, Paula Cubero

Para el diseño de la página además de utilizar una hoja de estilos he utilizado bootstrap.

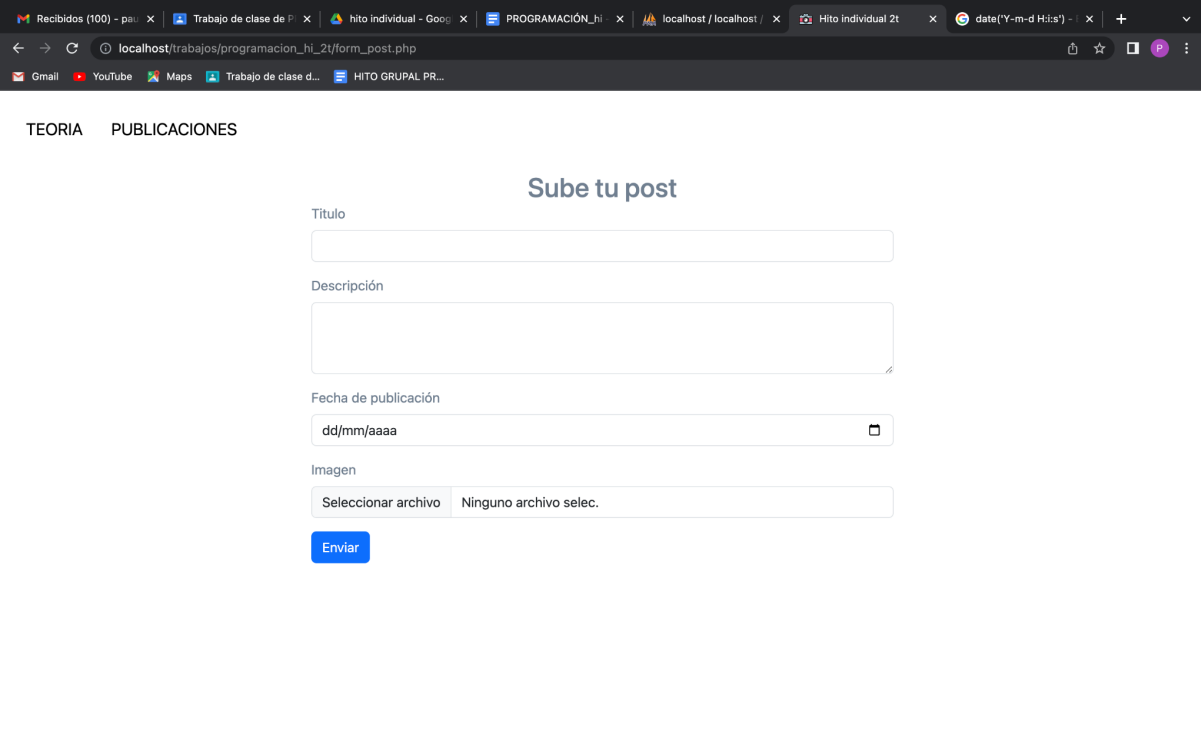
Al acceder a esta página se almacena en una cookie la IP del equipo al que está accediendo y la fecha de acceso. Aunque este asunto se puede realizar con Javascript, en esta ocasión lo hacemos con PHP.

Lo primero que hago es iniciar sesión, a continuación creó dos variables, una llamada ip, donde gracias a una simple llamada `$_SERVER['REMOTE_ADDR']` es posible obtener la ip del usuario que entra a la aplicación. La otra variable la he llamado fecha, y con el método `date('Y-m-d H:i:s')` se obtiene la fecha actual en la que el usuario ha entrado en la aplicación, además muestro la hora minutos y segundos. Por último se guarda en el método `setcookie`.

```
programacion_hi_2t > index.php
```

```
1  <?php
2  session_start();?>
3
4  <?php
5  // cookie
6  $ip=$_SERVER['REMOTE_ADDR']; //optiene la ip del equipo
7  $fecha=date('Y-m-d H:i:s'); //optiene la fecha actual
8  // cookie con ip y fecha
9  setcookie('ip',$ip);
10 setcookie('fecha_acceso',$fecha);
11 ?>
```

Ítem 3. Este ítem consiste en diseñar una página que permita al usuario escribir un post del blog del cliente. Se realizará un formulario que nos pedirá, email del autor, título, contenido, fecha de publicación e imagen. La imagen puede ser considerada para almacenar en la base de datos como texto o como objeto. Una vez enviado el formulario, al autor le confirmamos el correcto funcionamiento de la tarea.



TEORIA PUBLICATIONES

Sube tu post

Título

Descripción

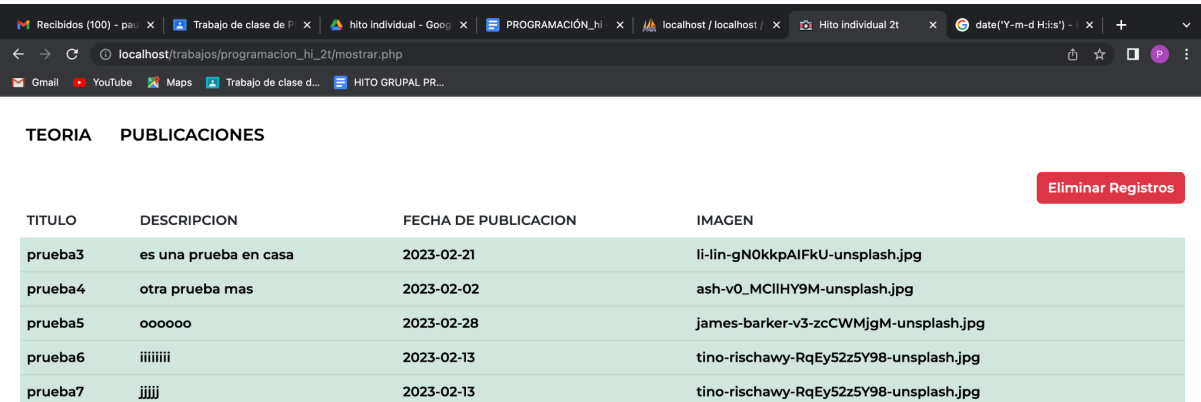
Fecha de publicación

Imagen

Seleccionar archivo Ninguno archivo selec.

Enviar

Ítem 4. En esta sección, se muestra una página con todas las entradas publicadas por todos los usuarios. Deberemos utilizar un diseño atractivo y usable.



TEORIA PUBLICATIONES

Eliminar Registros

TITULO	DESCRIPCION	FECHA DE PUBLICACION	IMAGEN
prueba3	es una prueba en casa	2023-02-21	li-lin-gN0kcpAIFkU-unsplash.jpg
prueba4	otra prueba mas	2023-02-02	ash-v0_MCIHY9M-unsplash.jpg
prueba5	oooooooo	2023-02-28	james-barker-v3-zcCWMjgM-unsplash.jpg
prueba6	iiiiiiii	2023-02-13	tino-rischawy-RqEy52z5Y98-unsplash.jpg
prueba7	jjjjj	2023-02-13	tino-rischawy-RqEy52z5Y98-unsplash.jpg

El botón de eliminar no tiene funcionalidad.

Ítem 5. 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. Considera utilizar un método de autenticación basado en sesiones, cookies, persistencia... y aplica el que consideres más oportuno.

Si el usuario se ha logueado o no le saldrán más o menos opciones en el menú de navegación.

```
<nav class="barra_navegacion"> <!-- menu se navegación -->
<a href="index.php" class="menu">TEORIA</a>
<a href="mostrar.php" class="menu">PUBLICACIONES</a><br>
<?php
    if(isset($_SESSION['token'])){ //si el token coincide aparece en el menu la opción de subir un post
        echo "<a href='form_post.php' class='menu'>SUBIR POST</a><br>";
    } else {
        echo "<a href='form_register.php' class='menu'>REGISTRARSE</a><br>";
        //si el token no coincide, aparece en el menu la opción de registrarse
    }
?>
<?php
    if(isset($_SESSION['token'])){ //si el token coincide aparece en el menu la opción de cerrar sesión
        echo "<a href='logout.php' class='menu'>LOGOUT</a>";
    }
?>
</nav>
```

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. Tras definir el algoritmo, puedes utilizar un caso de uso para dejar más clara la funcionalidad.

Uno de los primeros requisitos es mostrar la explicación de las diferencias entre lenguajes de programación orientada a objetos, a eventos y lenguajes procedimentales. Por ello lo primero que hago es hacer una página index, donde voy a mostrar dicha información con marcado html, la diseñé a mi gusto, es un diseño simple, colorido e intuitivo para el usuario. En esta página index también hay que incluir las cookies como indica en uno de los requisitos: “Al acceder a esta página se almacena en una cookie la IP del equipo al que está accediendo y la fecha de acceso” por lo tanto al html que ya tengo creado con la información le tengo que incrustar php, para crear variables que almacenen la ip y la fecha, como he explicado en el ítem 2.

Continuando con los requisitos de los ítems se que voy a tener que hacer varios formularios, de hecho en el ítem 3 el cliente pide que cree un formulario para poder subir post. Pero más adelante también voy a tener que crear dos formularios más para el tema de permisos, uno para que el usuario se registre si nunca se ha

registrado antes en la aplicación y otro de login para poder acceder a la aplicación y ya poder ver contenido de esta e interactuar con ella.

En el ítem 4 se pide que se muestre una tabla de los post del usuario, por lo tanto mediante código php llamo al archivo donde tengo mi conexión a la base de datos y creó dos variable la primera que almacena el select de mi base de datos y la segunda que muestra el resultado en forma de query, como lo tengo que mostrar en un tabla abro un echo para pintar una tabla con los campos que quiero mostrar.

Para que lea y muestre todos los campos del formulario creo un bucle while donde creó la variable row que va a pintar el resultado de la query, gracias al método fetch, va a buscar en los valores de mi tabla post la información que tiene en los campos que le indico y dicha información la pinta en los td de mi tabla.

```
28 <?php
29 include('conexion.php');
30 $sql= "SELECT * FROM post";
31 $resultado= $conn->query($sql);
32 //tabla
33 echo"<table class='table'>
34 <thead><td>TITULO</td><td>DESCRIPCION</td><td>FECHA DE PUBLICACION</td><td>IMAGEN</td></thead>";
35 while ($row=$resultado->fetch()) {
36     echo("<tr class='table-success'>");
37     echo("<td>".$row['TITULO']. "</td>");
38     echo("<td>".$row['DESCRIPCION']. "</td>");
39     echo("<td>".$row['FECHA_PUBLICACION']. "</td>");
40     echo("<td>".$row['IMAGEN']. "</td>");
41     echo("<td></td>");
42     echo("</tr>");
43 } // cierra bucle
44
45 ?>
46 <div class="contenedor_eliminar">
47     <button type="button" class="btn btn-danger">Eliminar Registros</button>
48 </div>
```

En el ítem 5 comenta el tema de los permisos que es lo que he comentado anteriormente con los formularios de login y de registro.

Explicación del registro.

Si la funcionalidad de registro se va a mostrar mediante un formulario lo primero que hay que hacer es crear el formulario. Es muy importante que todo los campos quiera almacenar tengan el name.

```
<div>
  <h2 class="form_titulo">Sing up</h2>
  <form class="form_login" action='register.php' method='post'>
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-label">Nombre</label>
      <input type="text" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" name="nombre">
      <div id="emailHelp" class="form-text"></div>
    </div>
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-label">Email</label>
      <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" name="email">
      <div id="emailHelp" class="form-text" ></div>
    </div>
    <div class="mb-3">
      <label for="exampleInputPassword1" class="form-label">Password</label>
      <input type="password" class="form-control" id="exampleInputPassword1" name="pass">
    </div>
    <button type="submit" class="btn btn-primary" name="enviar">Enviar</button>
  </form>
</div>
<?php
require_once('register.php');
?>
```

Como se puede ver es un formulario simple con tres input y que lleva los estilos de bootstrap. Hay que destacar la penúltima línea donde hay un require_once donde está incluyendo la página 'register.php' y la está evaluando.

En esta pagina que acabo de nombrar se encuentra lo siguiente:

```
programacion_hi_2t > 🐞 register.php
1  <?php
2      require_once('conexion.php'); // se conecta a la bdd
3      if (isset($_POST['enviar'])) { // si se pulsado enviar hace lo siguiente
4          $email=$_POST['email']; // almacena el email
5          $pass=$_POST['pass']; // almacena la contraseña
6
7          $insert= "INSERT INTO registro (email,contraseña) values ('$email','$pass)";
8          // mediante un insert se guarda la información de estos valores en mi bdd
9          $result= $conn->query($insert); // se crea una query
10
11      header('location:form_login.php'); //rederije a la pagina del formulario login
12  }
13  ?>
```


Una vez el usuario ya se ha registrado le lleva al formulario de login para que acceda con su usuario y contraseña y ahora si poder visualizar el contenido de la página.

Así he creado el formulario de login:

```
<!-- el usuario se logea -->
<div class="contorno_form">
  <h2 class="form_titulo">Login</h2>
  <form class="form_login" action='login.php' method='post'>
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-label">Email</label>
      <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" name="email">
      <div id="emailHelp" class="form-text" ></div>
    </div>
    <div class="mb-3">
      <label for="exampleInputPassword1" class="form-label">Password</label>
      <input type="password" class="form-control" id="exampleInputPassword1" name="pass">
    </div>
    <button type="submit" class="btn btn-primary" name="enviar">Enviar</button>
  </form>
</div>
<?php
require_once('login.php');
?>
```

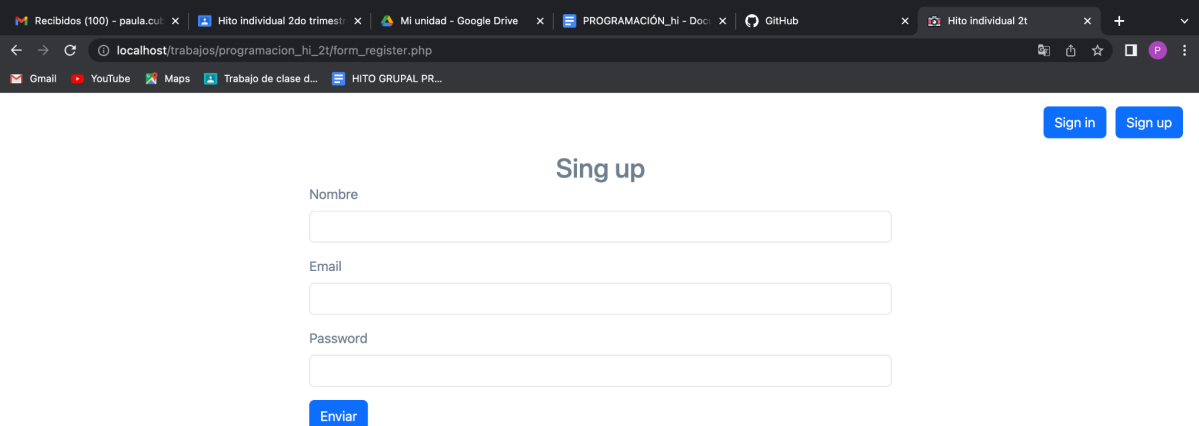
Es igual que el anterior formulario solo que esta vez solo he puesto dos input ya que solo quiero su usuario y contraseña.

Así es el la interacción con la base de datos del archivo login:

```
1 <?php
2 require_once('conexion.php');
3 if (isset($_POST['enviar'])){
4     $email=$_POST['email'];
5     $pass=$_POST['pass'];
6
7     $select= "select * from registro where email='".$email."'";
8     $result= $conn->query($select);
9     $row=$result->fetch();/*fetch permite meter la tabla en un array y leerlo*/
10 if ($email==$row['email'] and $pass==$row['contraseña']){
11     header('location:index.php');
12     session_start();
13     $_SESSION['token']='user';
14 }
15 else{
16     echo('Login fallido');
17 }
18 }
19 ?>
```

Funciona prácticamente igual que el anterior pero con alguna cosa más como la utilización de fetch.

Así se ve el formulario de registro en la web:



Recibidos (100) - paula.cui x Hito Individual 2do trimestre x Mi unidad - Google Drive x PROGRAMACIÓN_hi - Doc x GitHub x Hito individual 2t x +

localhost/trabajos/programacion_hi_2t/form_register.php

Gmail YouTube Maps Trabajo de clase d... HITO GRUPAL PR...

Sign in Sign up

Sing up

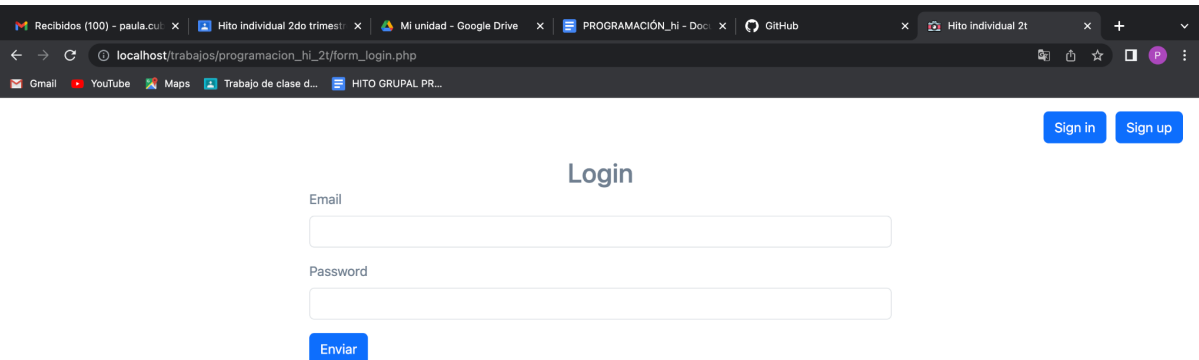
Nombre

Email

Password

Enviar

Y así se ve el formulario de login en la web:



Recibidos (100) - paula.cui x Hito Individual 2do trimestre x Mi unidad - Google Drive x PROGRAMACIÓN_hi - Doc x GitHub x Hito individual 2t x +

localhost/trabajos/programacion_hi_2t/form_login.php

Gmail YouTube Maps Trabajo de clase d... HITO GRUPAL PR...

Sign in Sign up

Login

Email

Password

Enviar

como se puede ver también le he añadido dos botones, para ir a ambas páginas según le convenga al usuario.

Fase 3. Para finalizar, realizamos las pruebas pertinentes de la aplicación y control de excepciones. Validación de campos y añadir seguridad en las comunicaciones. Debemos evaluar la implementación de la aplicación en base a la petición recibida inicialmente por el cliente. Es muy importante realizar un “feedback” de qué problemas nos han surgido a lo largo del desarrollo y cómo los hemos resuelto.

He tenido problemas a la hora de mostrar la imagen, de hecho no he sido capaz únicamente he podido mostrar en texto el nombre de la imagen que sube el usuario. También en el tema de permisos se me ha complicado bastante he conseguido hacer lo de la comprobación del token en el menú pero no se porque no comprueba el token en todas las páginas y de

hecho cuando accedo a la página de publicaciones ya estando registrado cuando se me abre dicha página se debe de borrar la sesión, porque no la guarda. Debo de tener algun header mal o no tengo claro el qué pero tampoco he sido capaz de solucionarlo. Aunque en la página dex si me funciona. Porque dependiendo si estás logueado o no aparece 'registrate' o 'subir post' en el menú de navegación.

Webgrafía

http://agrega.juntadeandalucia.es/repositorio/21012019/89/es-an_2019012112_9123621/5_programacin_orientada_a_eventos.html

[https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/#:~:text=La%20Programaci%C3%B3n%20Orientada%20a%20Objetos%20\(POO\)%20es%20un%20paradigma%20de,concepto%20de%20clases%20y%20objetos.](https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/#:~:text=La%20Programaci%C3%B3n%20Orientada%20a%20Objetos%20(POO)%20es%20un%20paradigma%20de,concepto%20de%20clases%20y%20objetos.)

<https://blog.hubspot.es/website/tipos-lenguaje-programacion>