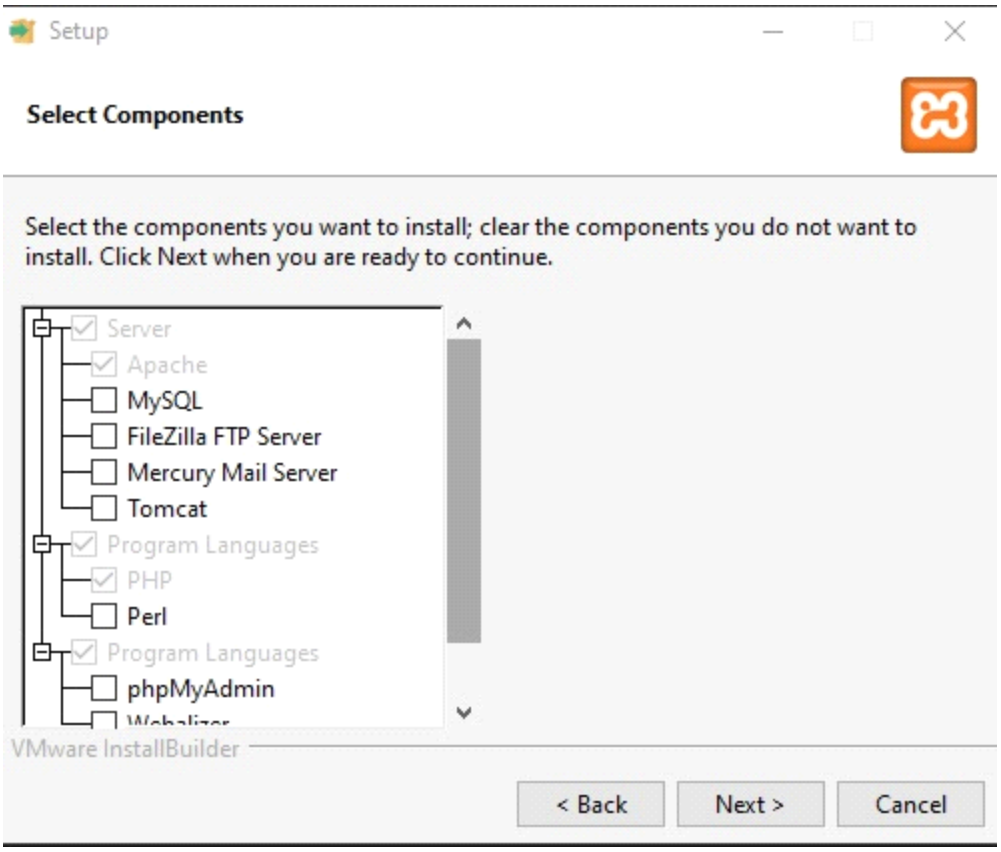
Tutorial: Formulario Paso a Paso

Paso 1: Preparando nuestro espacio de trabajo

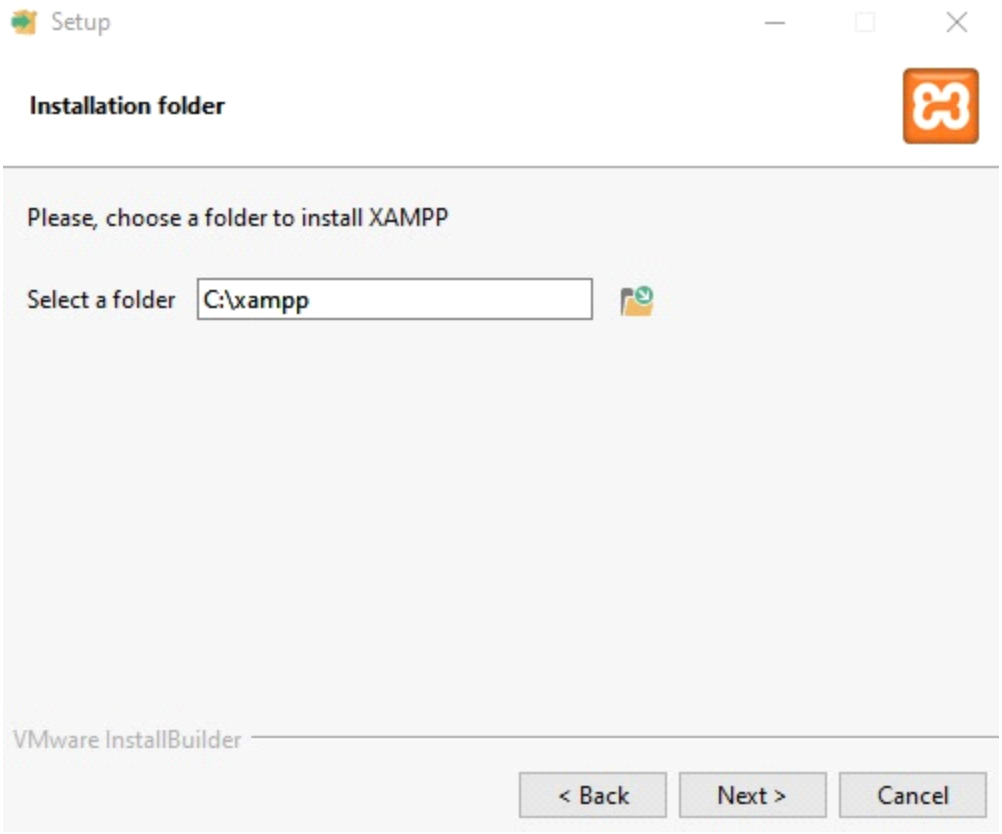
Instalaremos XAMPP para simular un entorno cliente-servidor de manera local en nuestra PC.

Entramos al link: https://www.apachefriends.org/es/index.html y descargamos la versión para nuestro sistema operativo.

Ejecutamos el instalador y para ésta practica podemos desactivar todos los servicios extras, con que estén marcados los servicios Apache y PHP es suficiente para este ejercicio:

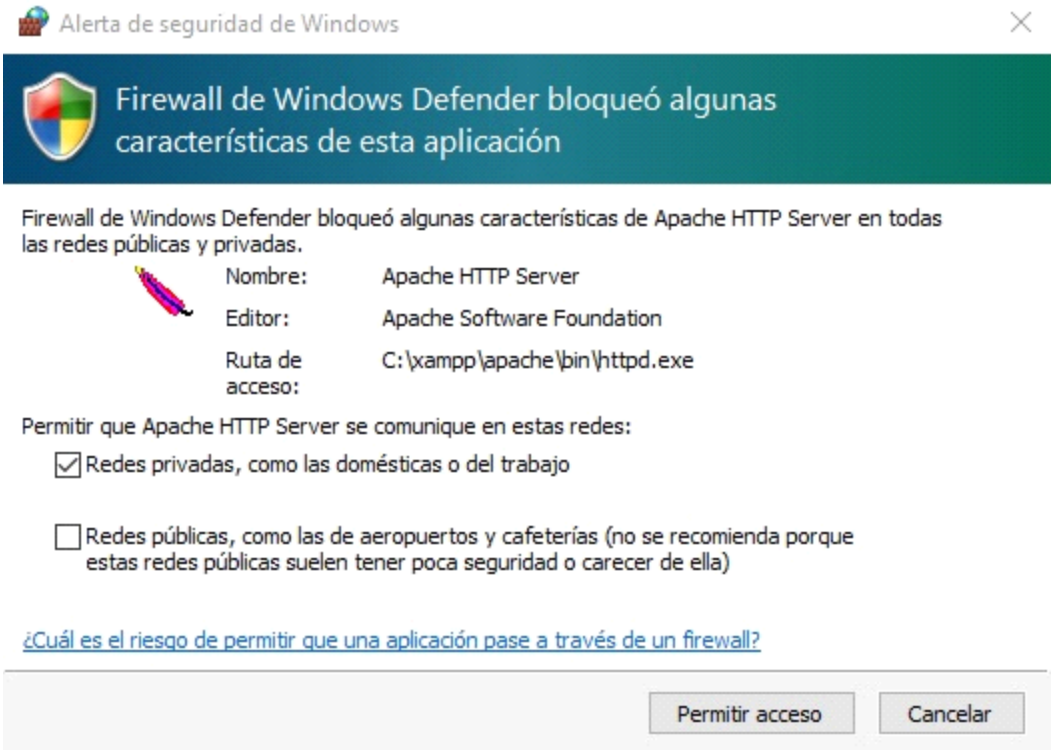


Para evitar inconvenientes a la hora de visualizar nuestro formulario web, recomiento instalarlo en la ruta raiz de nuestro disco rigido, en esta caso C:/

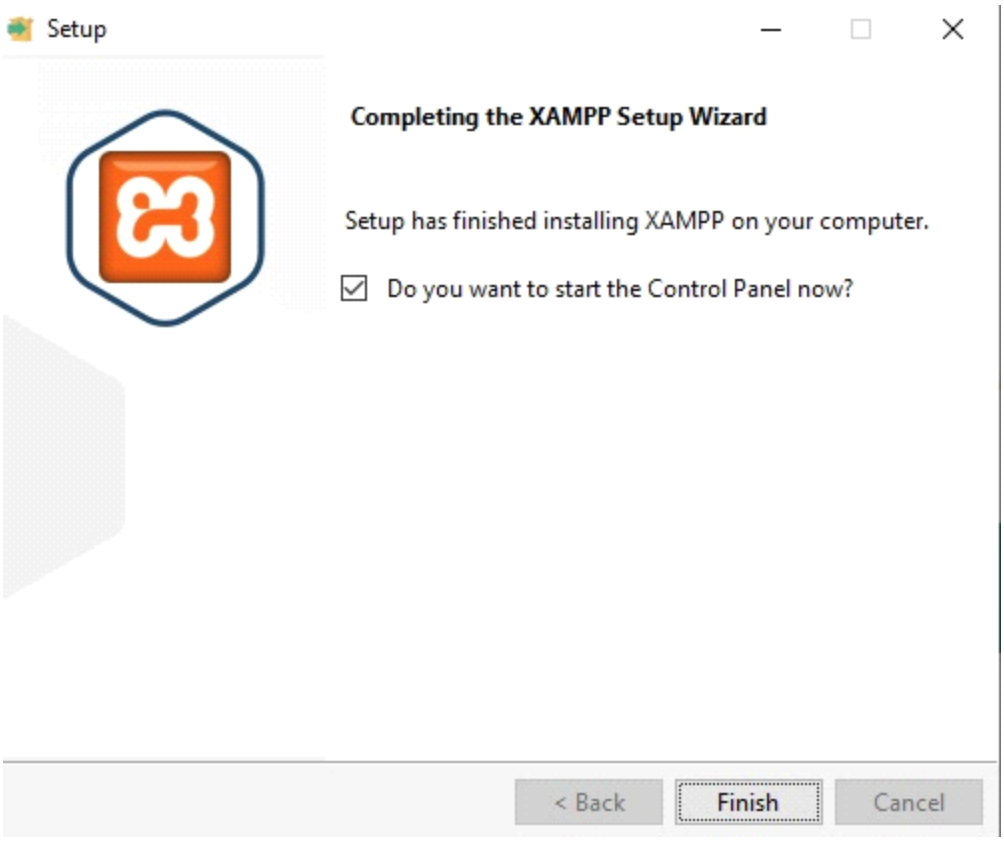


Le damos, siguiente -> finalizar y esperamos a que el entorno se instale.

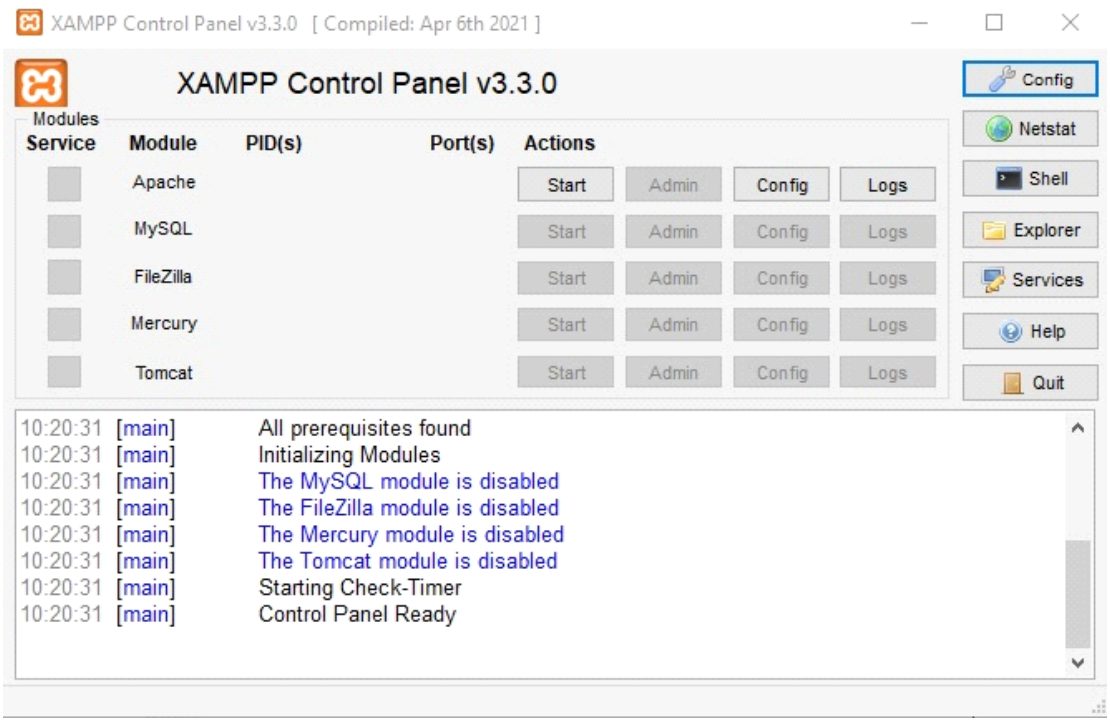
En caso de que nos salga un cartel del firewall, le vamos a permitir el acceso a las redes privadas.



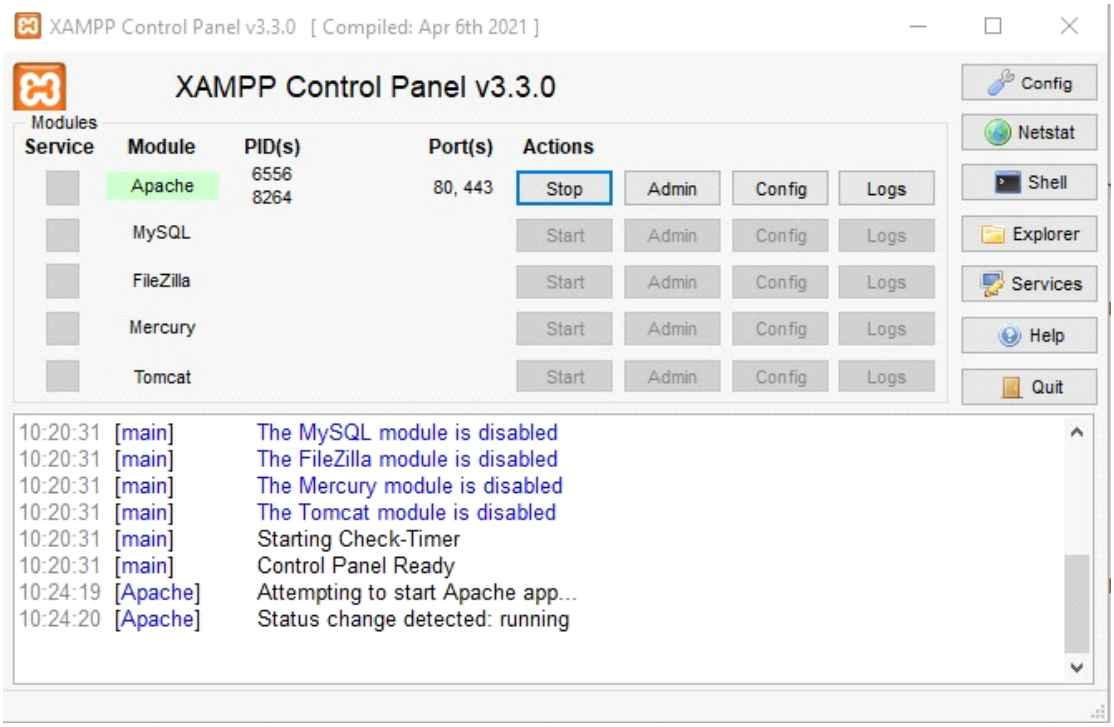
Una vez finalizado el instalador marcamos la casilla de Iniciar el Panel de Control y le damos a Finish.



Esperamos un poco y se nos abrirá nuestro panel de control:



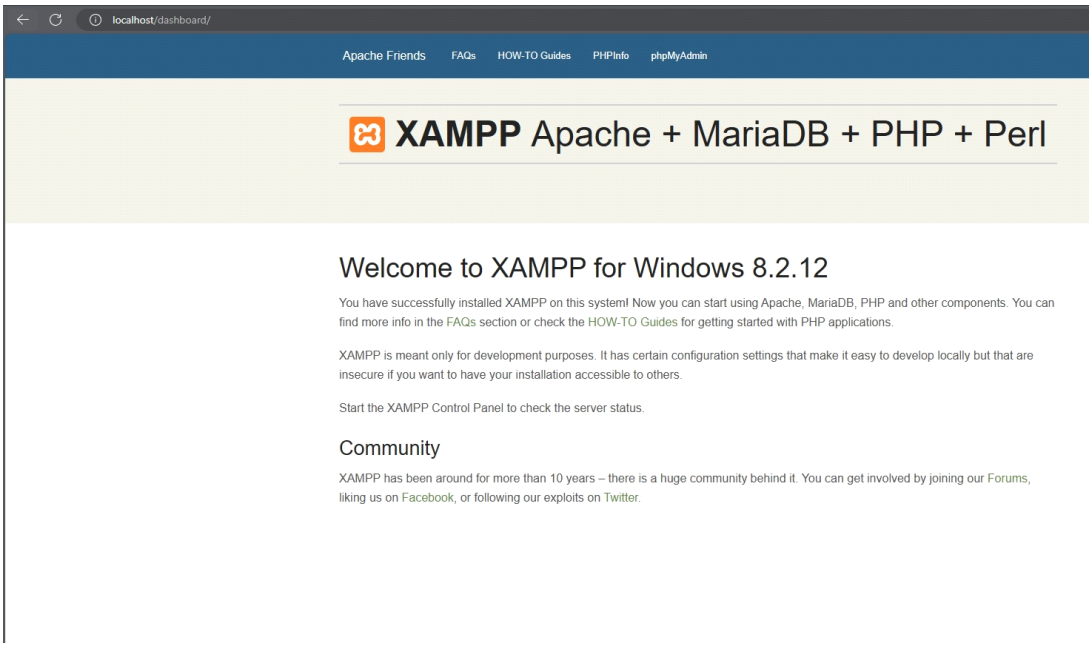
Le damos al botón de Start de nuestro servicio Apache:



Podemos ver que nuestro servidor Apache se encuentra corriendo en los puertos predeterminados 80 y 443.

Vamos a nuestro navegador web y tipeamos en la barra de direcciones: localhost

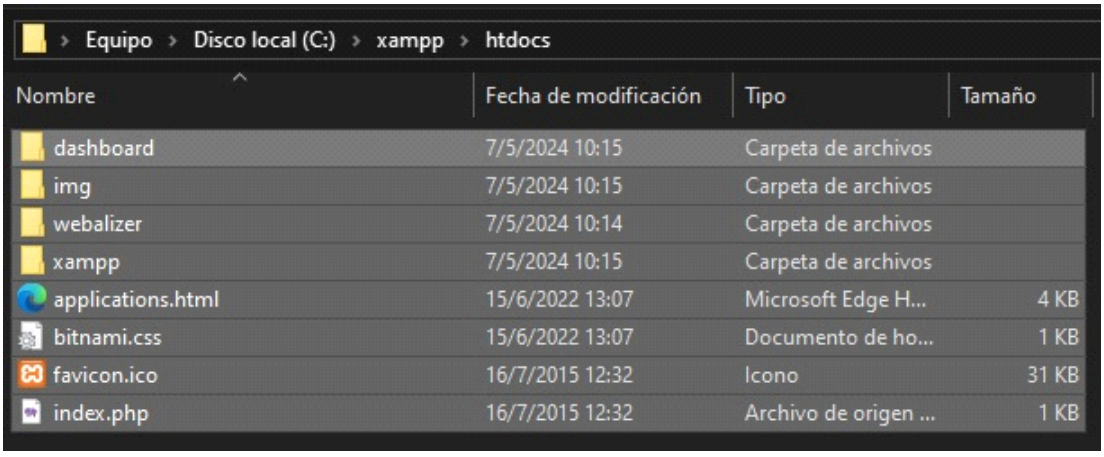
Si nos redirige a esta página es porque ya tenemos nuestro "servidor" corriendo y accesible desde nuestro entorno local.



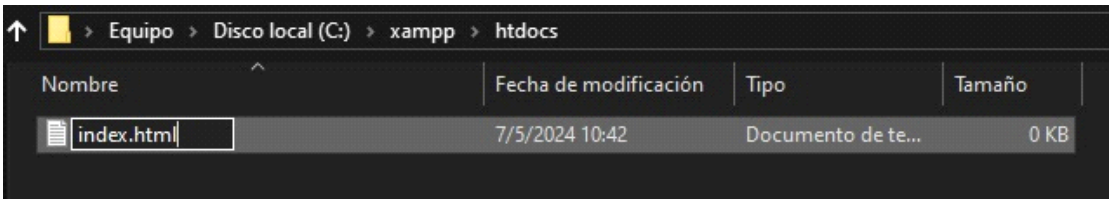
Paso 2: Creamos nuestro formulario

Procedemos a ingresar a la ruta de nuestra web. Si seguimos los pasos aquí descriptos la ruta es: C:\xampp\htdocs

Notar que la carpeta donde va el directorio web en Apache se llama "htdocs". Debemos eliminar todo el contenido que se encuentre en esta ruta:



Procedemos a crear nuestro formulario. Creamos un archivo de texto, le cambiamos el nombre a index y la extensión a .html, debe llamarse index.html



Lo abrimos con el editor de texto de nuestra preferencia y colocamos el código HTML del formulario que debemos securizar. En este caso trabajaremos sobre el siguiente código:

<!-- Inicio del código -->

<form action="login_form.php">

<input id="username" name="username"/>

<input id="password" name="password"/>

<input id="rememberMe" name="rememberMe"/>

<input id="login" name="login"/>

<input type="submit"/>

</form>

<!-- Final del código -->

Quedando así:

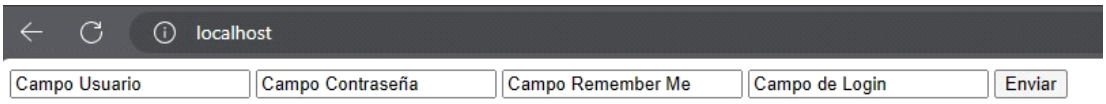
index.html

```
1   <!-- Inicio del código -->
2   <form action="login_form.php">
3   <input id="username" name="username"/>
4   <input id="password"  name="password"/>
5   <input id="rememberMe"  name="rememberMe"/>
6   <input id="login" name="login"/>
7   <input type="submit"/>
8   </form>
9   <!-- Final del código -->
10
```

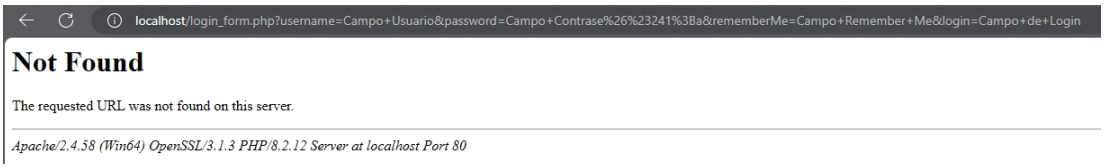
Guardamos los cambios en el archivo y volvemos a recargar localhost en nuestro navegador web.

Podemos observar que nuestro formulario carece de formato. No hay indicativos que identifiquen para

que sirve cada campo (a estos los llamaremos "labels"), el campo contraseña es completamente visible a las personas que tengamos alrededor y la casilla de "Remember Me" es otro campo donde se ingresa texto cuando no debería.



Y lo más grave del asunto, fíjense que pasa cuando le damos al botón "Enviar":



Podemos observar que toda la información que colocamos en los campos del formulario, es visible en nuestra URL.

Es aquí donde debemos aplicar las primeras modificaciones de código en nuestro HTML para comenzar hacer más seguro nuestro formulario.

Volvemos abrir nuestro index.html con nuestro editor de texto preferido.

Aplicamos las siguientes correcciones:

Le indicaremos a nuestro formulario que utilice el método "post" para enviar la información de manera segura al servidor, ya que al no indicarle el método, utiliza por defecto el método "GET" que es el que hace que el texto ingresado se refleje en la URL.

Al input "password" le agregamos el parámetro type="password" . Ésto lo que hace es indicar que es un campo donde se escribirá una contraseña. De esta manera al escribir en éste campo, los caracteres aparecerán ocultos:

Al input "rememberMe", le indicaremos que es una casilla para marcar, añadiendo el parámetro type="checkbox" .

```
<!-- Inicio del código -->

<form action="login_form.php" method="post">

<input id="username" name="username"/>

<input id="password" type="password" name="password"/>

<input id="rememberMe" type="checkbox" name="rememberMe"/>

<input id="login" name="login"/>

<input type="submit"/>
```

```
</form>

<!-- Final del código -->
```

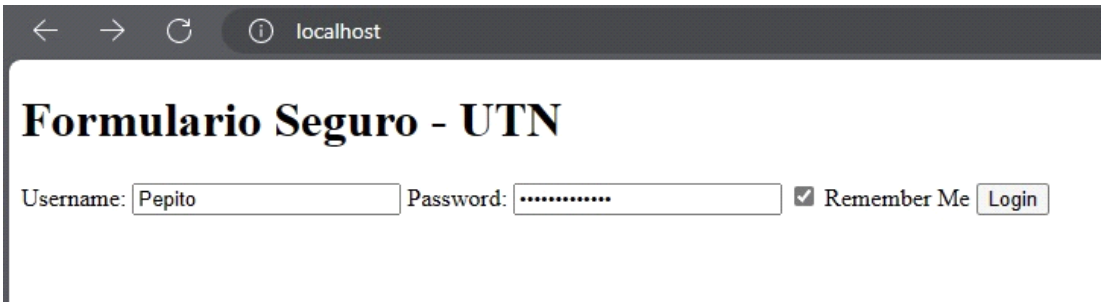
Ahora podemos observar que nuestro formulario ya tiene un poco más de sentido y es más seguro. La contraseña se encuentra oculta y si hacemos click en "Enviar" nos llevara a una página de error pero la contraseña ya no se refleja en la URL.

Vamos a darle algo de formato a gusto. Compartiré el código HTML para que sigamos trabajando sobre la misma base:

```
<html>

<head>

    <title>Tutorial Formulario Seguro - UTN</title>

</head>

<body>

    <h1>Formulario Seguro - UTN</h1>

<!-- Inicio del código -->

    <form action="login_form.php" method="post">

        <label for="username">Username:</label>

        <input type="text" id="username" name="username" required>


        <label for="password">Password:</label>

        <input type="password" id="password" name="password" required>


        <input type="checkbox" id="rememberMe" name="rememberMe">

        <label for="rememberMe">Remember Me</label>
```

```
            <button type="submit">Login</button>
        </form>
<!-- Final del código -->
</body>
</html>
```

Observemos los resultados:



Hemos incorporado:

- Segmentación de nuestro código en <head> y <body>.
- Etiqueta <h1> para hacer nuestro Titulo.
- Agregado etiquetas <label> a nuestros campos de Username y Password. Lo que hace esto es agregar un texto descriptivo para poder identificarlos mejor.
- Eliminamos el input "login" por no servirnos para nada, ya que tenemos esa característica como botón. Con el fragemento de codigo: <button type="submit">Login</button>

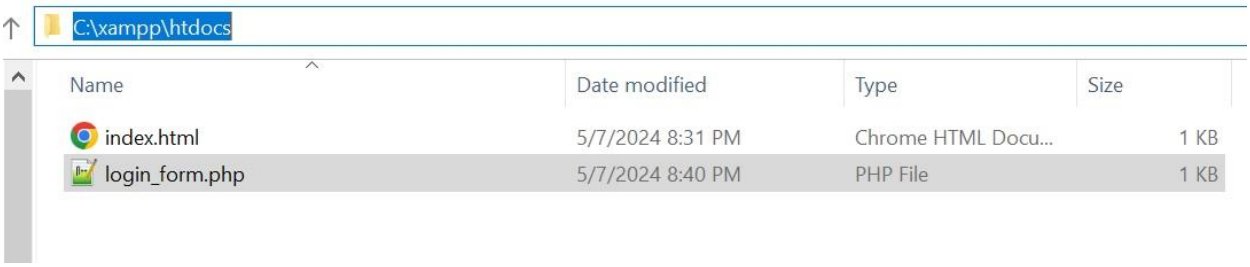
Ahora nuestro formulario es un poco más seguro a simple vista, aunque sigue siendo vulnerable a algunos tipos de ataques.

Paso 3: Agregando funciones PHP

Lo primero que vamos hacer es crear un registro donde se guarde la información introducida en nuestro formulario, en este caso Usuario y Contraseña en un archivo dentro del directorio.

Como vemos en la primera línea de nuestro código HTML el formulario hace un llamado a un archivo PHP: “<form action="login_form.php" method="post">”

Procedemos a crear en nuestro directorio el archivo login_form.php . Quedando de esta manera:



En el contenido del archivo pegamos el siguiente código PHP:

```
<?php

// Verificar si se ha enviado el formulario

if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // Obtener los datos del formulario

    $username = $_POST["username"];

    $password = $_POST["password"];


    // Crear una cadena con los datos a guardar en el archivo

    $datos = "Username: $username\nPassword: $password\n\n";


    // Ruta del archivo de texto donde se guardarán los datos

    $archivo = "datos.txt";


    // Guardar los datos en el archivo de texto

    if (file_put_contents($archivo, $datos, FILE_APPEND | LOCK_EX) !== false) {

        echo "Los datos se han guardado correctamente.";

    } else {

        echo "Ha ocurrido un error al intentar guardar los datos.";

    }
```

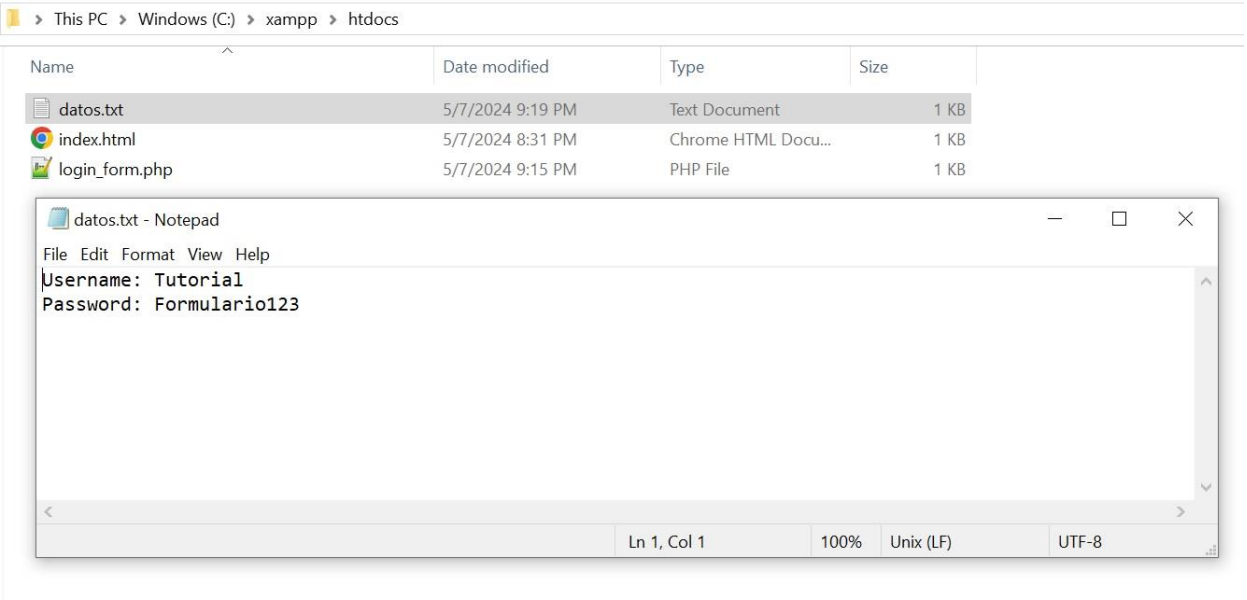
```
} else {
    // Si se intenta acceder directamente al script, redirigir al formulario
    header("Location: index.html");
    exit;
}
?>
```

El contenido debe quedarnos de la siguiente manera:

index.html | login_form.php

```php
<?php
// Verificar si se ha enviado el formulario
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Obtener los datos del formulario
    $username = $_POST["username"];
    $password = $_POST["password"];

    // Crear una cadena con los datos a guardar en el archivo
    $datos = "Username: $username\nPassword: $password\n\n";

    // Ruta del archivo de texto donde se guardarán los datos
    $archivo = "datos.txt";

    // Guardar los datos en el archivo de texto
    if (file_put_contents($archivo, $datos, FILE_APPEND | LOCK_EX) !== false) {
        echo "Los datos se han guardado correctamente.";
    } else {
        echo "Ha ocurrido un error al intentar guardar los datos.";
    }
} else {
    // Si se intenta acceder directamente al script, redirigir al formulario
    header("Location: index.html");
    exit;
}
?>
```

Ahora, al rellenar nuestro formulario podemos observar que los datos ingresados se guardan en un archivo que se crea automáticamente llamado “datos.txt”:



Ya tenemos nuestro formulario funcional.

Paso 4: Parámetros y encriptado de contraseñas

Como pudimos ver. Nuestros datos se guardan sin ningún tipo de codificación.

Vamos agregarle un poco mas de seguridad estableciendo parámetros de longitud de caracteres y encriptando las contraseñas almacenadas. Probemos y analicemos el siguiente código de nuestro login_form.php . Lo resaltado en amarillo es el código añadido a nuestro archivo original.

```
<?php

// Verificar si se ha enviado el formulario

if ($_SERVER["REQUEST_METHOD"] == "POST") {

    // Obtener los datos del formulario

    $username = $_POST["username"];

    $password = $_POST["password"];


    // Validar el nombre de usuario

    if (empty($username) || strlen($username) < 5 || strlen($username) > 20) {

        echo "Error: El nombre de usuario debe tener entre 5 y 20 caracteres.";
```

```
        exit;
    }

    // Validar la contraseña
    if (empty($password) || strlen($password) < 8) {
        echo "Error: La contraseña debe tener al menos 8 caracteres.";
        exit;
    }

    // Función para hashear contraseñas
    function hashPassword($password) {
        // Utilizar una función de hashing segura como bcrypt o Argon2
        return password_hash($password, PASSWORD_DEFAULT);
    }

    // Almacenar la contraseña hasheada en el archivo
    $datos = "Username: $username\nPassword: " . hashPassword($password) . "\n\n";

    // Ruta del archivo de texto donde se guardarán los datos
    $archivo = "datos.txt";

    // Guardar los datos en el archivo de texto
    if (file_put_contents($archivo, $datos, FILE_APPEND | LOCK_EX) !== false) {
        echo "Los datos se han guardado correctamente.";
    } else {
```

```
        echo "Ha ocurrido un error al intentar guardar los datos.";
    }
} else {
    // Si se intenta acceder directamente al script, redirigir al formulario
    header("Location: index.html");
    exit;
}
?>
```

Desmenucemos el código:

En la sección **Validar el nombre de usuario** podemos especificar la cantidad de caracteres mínimos y máximos que se pueden ingresar en el campo. **Esto nos ayuda a prevenir inyección de código malicioso**:

```
// Validar el nombre de usuario
    if (empty($username) || strlen($username) < 5 || strlen($username) > 20) {
        echo "Error: El nombre de usuario debe tener entre 5 y 20 caracteres.";
        exit;
    }
```

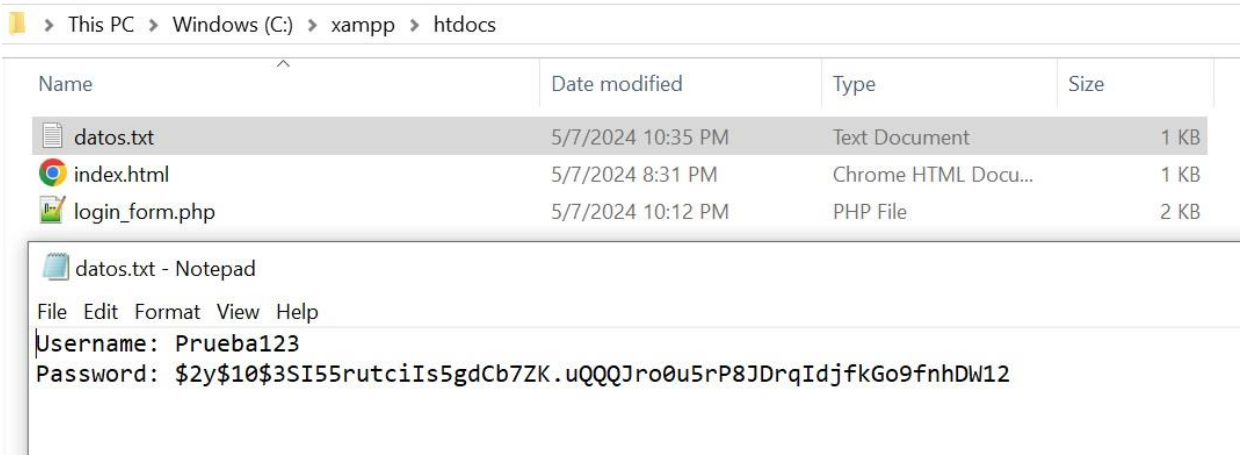
En la sección **Validar contraseña** estamos especificando que la contraseña tenga un mínimo de caracteres de longitud, para **evitar contraseñas genéricas que pueden ser vulneradas con ataques de fuerza bruta** del estilo “admin”, “1234”, etc.

```
// Validar la contraseña
    if (empty($password) || strlen($password) < 8) {
        echo "Error: La contraseña debe tener al menos 8 caracteres.";
        exit;
    }
```

Y aquí añadimos la función para encriptar nuestra contraseña y que no sea fácil de leer si alguien accede a nuestro archivo datos.txt:

```
// Función para hashear contraseñas
function hashPassword($password) {
    // Utilizar una función de hashing segura como bcrypt o Argon2
    return password_hash($password, PASSWORD_DEFAULT);
}
```

Ahora, si probamos nuestro formulario podemos comprobar que las contraseñas se encuentran cifradas:



Y hasta aquí hemos llegado con el Tutorial de formularios seguros.

Podemos implementar aun más prácticas para seguir haciendo el formulario cada vez más seguro.

¿Qué le agregarías?