



TECNOLÓGICO NACIONAL DE MÉXICO  
INSTITUTO TECNOLÓGICO DE TLAXIACO

---

**SEGURIDAD Y VIRTUALIZACIÓN**

Practica 2

---

**Integrantes del Equipo:**

Arnol Jesus Cruz Ortiz

Amilkar Vladimir Reyes Reyes

Rael Gabriel Bautista

Sandra Gabriela Velasco Guzmán

**Docente:**

Edward Osorio Salinas

**Carrera:**

Ingeniera en Sistemas Computacionales

**Grupo:** 7US

**Semestre:** Agosto – diciembre 2024

09/Septiembre/2024

---

## 1.- Crea una aplicación web, mobile, escritorio que permita loguearse con un usuario y contraseña.

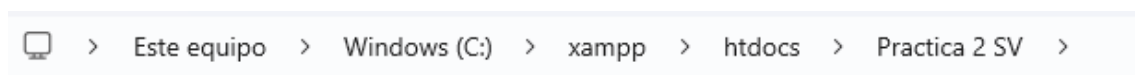
A continuación, explicaremos el proceso de desarrollo de nuestra aplicación, basado en los puntos específicos solicitados por el profesor.

Para desarrollar nuestra aplicación, seleccionamos Visual Studio Code como editor de código fuente, y utilizaremos MySQLi para el almacenamiento de datos.

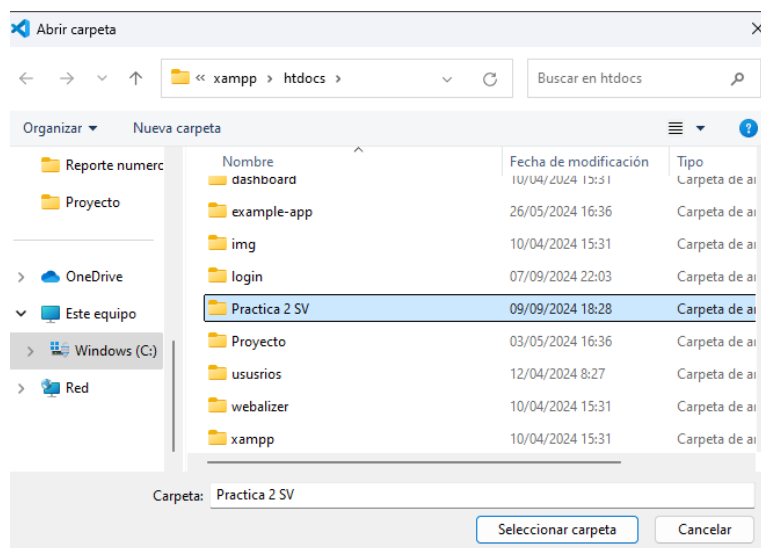
El objetivo principal es comprender y aplicar los conceptos de autenticación y autorización en la seguridad de la información.

### “INICIO DE LA APLICACIÓN”

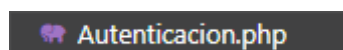
Para comenzar, creamos una carpeta llamada 'Practica 2 SV' dentro del directorio 'htdocs' de XAMPP.



Luego, abrimos Visual Studio Code y accedimos a la carpeta creada.



Primero, creamos un archivo denominado Autenticacion.php.



En el archivo Autenticacion.php, comenzamos definiendo el documento HTML5. En el código, especificamos la codificación de caracteres, configuramos el viewport para asegurar un diseño responsivo, establecimos el título de la página

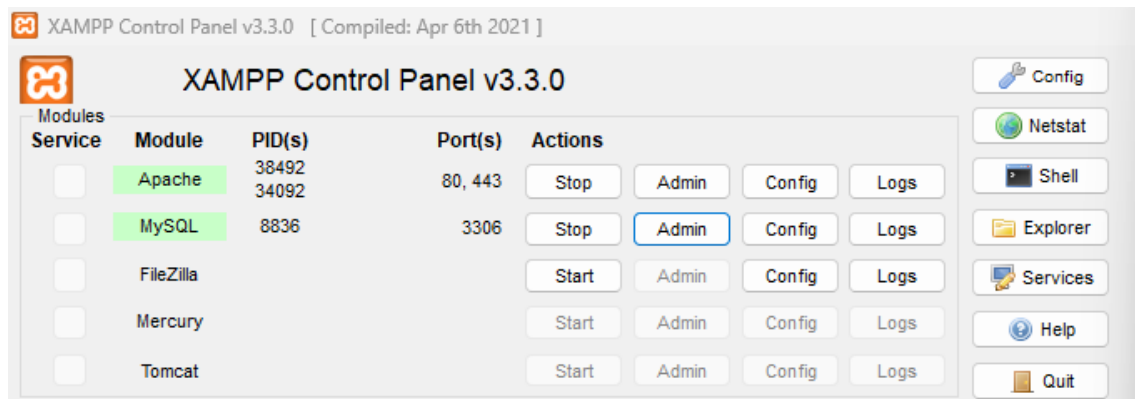
que se mostrará en la pestaña del navegador y añadimos dos enlaces a hojas de estilo CSS, que deben ser creadas para mejorar la apariencia de nuestra página.

```
Autenticacion.php X
Autenticacion.php
1 <!DOCTYPE html>
2 <html Lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Login</title>
7   <link rel="stylesheet" href="Estilos.css">
8   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@6.0.0-beta3/css/all.min.css">
9   <style>
10
11 </style>
12 </head>
```

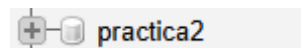
En el archivo Autenticacion.php, dentro del body, creamos un formulario utilizando el método POST para enviar datos a otro archivo PHP que desarrollaremos más adelante. El formulario incluye un título, un campo de texto para el nombre del usuario y un contenedor para el campo de contraseña. Añadimos un icono para mostrar y ocultar la contraseña, junto con un botón para cambiar entre mostrar y ocultar la contraseña. Además, se implementó un evento en el icono para alternar el tipo de campo entre password y text al hacer clic, permitiendo así mostrar u ocultar la contraseña, y cambiamos el icono a uno de ojo tachado cuando la contraseña está visible.

```
Autenticacion.php X
Autenticacion.php
9 <style>
10
11 </style>
12 </head>
13 <body>
14   <?php ?>
15   <form class="formulario" name="formulario" method="POST" action="configuracionLogin.php">
16     <h1>INICIAR SESIÓN</h1>
17     <h2></h2>
18
19     <input class="cajaUsuario" type="text" placeholder="Usuario" name="cajaUsuario">
20     <div class="password-container">
21       <input class="cajaContraseña" type="password" placeholder="Contraseña" name="cajaContraseña" id="password">
22       <i class="far fa-eye" id="togglePassword"></i>
23     </div>
24     <input type="submit" value="Iniciar Sesión">
25     <input type="button" value="Registrar" onclick="window.location.href = `Registrar.php`">
26   </form>
27   <script>
28     const togglePassword = document.querySelector('#togglePassword');
29     const password = document.querySelector('#password');
30
31     togglePassword.addEventListener('click', function () {
32       const type = password.getAttribute('type') === 'password' ? 'text' : 'password';
33       password.setAttribute('type', type);
34       this.classList.toggle('fa-eye-slash');
35     });
36   </script>
37 </body>
38 </html>
```

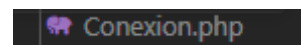
Después lo que se realizó fue la creación de la base de datos, para ello nos dirigimos a XAMPP y ahí en la parte del MySQL le damos en Admin.



Y estando adentro creamos la base de datos la cual nosotros la llamamos practica2.



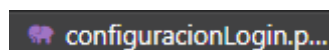
Ya creada la base de datos ahora la debemos de conectar a nuestro proyecto y para ello creamos un archivo llamado Conexión.php en donde realizamos la conexión.



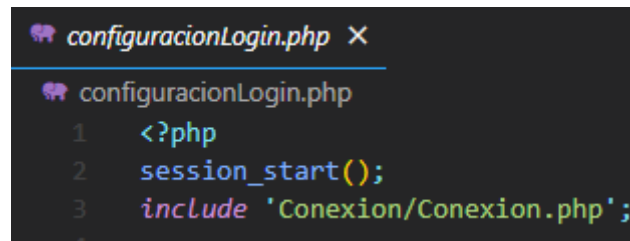
Y bueno para conectarla tenemos que configurarla diciéndole donde esta alojada la base de datos, cual es el nombre del usuario, ponerle contraseña si es que tenga en nuestro caso no tiene, se le pone el nombre de la base de datos en este caso es practica2 y quedo de la siguiente manera.



Ya que se hizo la conexión de la base de datos correctamente procedemos a realizar otro archivo llamado configuraciónLogin.php.

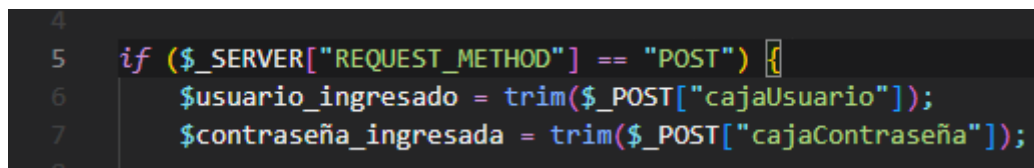


Bueno aquí en `session_start()`; se inicia una nueva sesión o reanuda una sesión existente. Esto es necesario para almacenar información sobre el usuario en la sesión. Ahora en `include 'Conexion/Conexion.php'`; lo que hace es que incluye el archivo de conexión a la base de datos. Asegurando de que la ruta al archivo `Conexion.php` sea correcta y que este archivo establezca una conexión a la base de datos y almacene la conexión en una variable, típicamente `$conexion`.



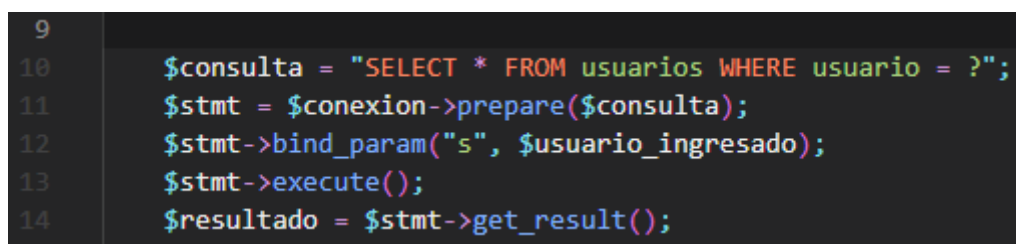
```
configuracionLogin.php X
configuracionLogin.php
1  <?php
2  session_start();
3  include 'Conexion/Conexion.php';
```

Después se utiliza `$_SERVER["REQUEST_METHOD"] == "POST"`: que verifica si el formulario se envió usando el método POST, también utilizamos `trim()` que se dedica a eliminar los espacios en blanco al principio y al final de las entradas del formulario.



```
4
5  if ($_SERVER["REQUEST_METHOD"] == "POST") {
6      $usuario_ingresado = trim($_POST["cajaUsuario"]);
7      $contraseña_ingresada = trim($_POST["cajaContraseña"]);
```

Ya que se realizó lo anterior se prepara la consulta para verificar si el usuario existe en la base de datos.



```
9
10  $consulta = "SELECT * FROM usuarios WHERE usuario = ?";
11  $stmt = $conexion->prepare($consulta);
12  $stmt->bind_param("s", $usuario_ingresado);
13  $stmt->execute();
14  $resultado = $stmt->get_result();
```

Ahora realizaremos la validación del usuario y contraseña. Para eso utilizamos el `$resultado->num_rows > 0`: que verifica si se encontró un usuario con el nombre ingresado, la comparación `if ($usuario['contraseña'] == $contraseña_ingresada && $usuario['usuario'] == $usuario_ingresado)` valida si la contraseña ingresada coincide con la almacenada. Si nuestra autenticación es correcta nos almacenará el nombre del usuario. Luego, dirigimos al usuario a diferentes páginas dependiendo de si es administrador (`Es_admin == 1`) o un usuario

regular ( $\text{Es\_admin} == 0$ ). También en caso de que la contraseña es incorrecta o el usuario no se encuentra en la base de datos nos mandará un mensaje de error.

```
15
16     if ($resultado->num_rows > 0) {
17         $usuario = $resultado->fetch_assoc();
18
19         if ($usuario['contraseña'] == $contraseña_ingresada && $usuario['usuario'] == $usuario_ingresado) {
20
21             $_SESSION['usuario'] = $usuario['usuario'];
22             $_SESSION['id'] = $usuario['id'];
23
24             if ($usuario['Es_admin'] == 1) {
25                 header("Location: Administrador.html");
26                 exit();
27             }
28             else if ($usuario['Es_admin'] == 0) {
29                 header("Location: Usuario.html");
30                 exit();
31             }
32         } else {
33
34             echo "Contraseña incorrecta.";
35         }
36     } else {
37
38         echo "Usuario no encontrado. Por favor, verifica tu información.";
39     }
40 }
```

Ya por último hacemos el cierre con `close()`: esta función cierra la consulta y la conexión a la base de datos.

Y bueno en conclusión lo que hace el anterior código es poner seguridad. El código compara las contraseñas en texto plano, lo que no es seguro. Debemos utilizar funciones de hash para almacenar y comparar contraseñas, como `password_hash()` y `password_verify()` en PHP.

Validación de entrada: Aunque el código utiliza consultas preparadas, siempre es una buena práctica validar y sanitizar las entradas del usuario.

Redirección: Asegúrate de que `Administrador.html` y `Usuario.html` existan y estén correctamente configurados para manejar la redirección.

Ahora a lo que se realizó se le tiene que ponerle una interfaz gráfica agradable, ósea una hoja de estilos para los archivos de `Autenticacion.php` y `configuracionLogin.php` y para ello se crea una CSS llamada `Estilos.css`.

```
# Estilos.css
```

Lo que contiene mi CSS son estilo de fondo del cuerpo, estilo para el formulario, campos de entrada de texto y contraseña, botones de envío y acción,

Contenedor de contraseña con ícono, y se le dio la acción de que sea responsivo también.


```
# Estilos.css X
# Estilos.css > {} @media (max-width: 768px) > .formulario input[type="text"]
1  *{
2      margin: 0px;
3      box-sizing: border-box;
4  }
5  body {
6      background-image: url('imagen.jpg');
7      background-size: cover;
8      background-position: center;
9      background-repeat: no-repeat;
10     height: 100vh;
11     display: flex;
12     justify-content: center;
13     align-items: center;
14     margin: 0;
15 }
16
17 .formulario {
18     background-color: rgba(224, 212, 212, 0.8);
19     padding: 20px;
20     border-radius: 30px;
21     width: 80%;
22     max-width: 500px;
23     text-align: center;
24     box-shadow: 0px 0px 10px rgba(226, 10, 10, 0.1);
25 }
26 .formulario h1 {
27     margin-top: 0;
28     color: rgb(0, 15, 44);
29     font-size: 25px;
30     margin-bottom: 10px;
31 }
32 .formulario h2{
33
34 .formulario p {
35     color: #0c5ebd;
36     font-size: 20px;
37     font-weight: bold;
38     margin-bottom: 10px;
39 }
40
41 .formulario input[type="text"],
42 .formulario input[type="password"]{
43     width: calc(100% - 40px);
44     padding: 15px;
45     margin: 10px 0;
46     border: 1px solid rgb(0, 61, 135);;
47     border-radius: 5px;
48     box-sizing: border-box;
49     font-size: 16px;
50 }
51 .password-container {
52     position: relative;
53 }
54 .password-container i {
55     position: absolute;
56     right: 10px;
57     top: 50%;
58     transform: translateY(-50%);
59     cursor: pointer;
60 }
61 .formulario input[type="submit"],[type="button"]{
62     width: calc(100% - 40px);
63     padding: 10px;
64     margin: 8px 0;
65     border: none;
66     border-radius: 5px;
```

```

# Estilos.css X
# Estilos.css > {} @media (max-width: 768px) > .formulario input[type="text"]
61 .formulario input[type="submit"],[type="button"]{
71     transition: background-color 0.3s;
72     font-size: 16px;
73 }
74 p>a{
75     text-decoration: none;
76 }
77
78 .formulario input[type="submit"]:hover,[type="button"]:hover{
79     background-color: #007bff;
80     color: #fff;
81 }
82
83 .formulario input[type="submit"]:focus {
84     outline: none;
85 }
86
87 /* Media queries para responsividad */
88 @media (max-width: 768px) {
89     .formulario {
90         width: 90%;
91         margin-top: 0;
92         padding: 15px;
93     }
94     .formulario input[type="text"],
95     .formulario input[type="password"] {
96         padding: 10px;
97         font-size: 15px;
98     }
99     .formulario input[type="submit"] {
100         padding: 10px;
101         font-size: 15px;

```

Después creamos otro archivo llamado Registrar.php

 Registrar.php

Lo que nuestro archivo contiene es la página de registro de usuarios que incluye validaciones de contraseñas tanto del lado del servidor (PHP) como del lado del cliente (JavaScript).

En esta parte se comprueba si la solicitud HTTP es de tipo POST, lo que indica que el formulario fue enviado. Los valores de usuario, contraseña y confirmarContraseña son obtenidos desde el formulario enviado mediante \$\_POST.

```

Registrar.php X
Registrar.php
1 <?php
2 include 'Conexion/Conexion.php';
3
4 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
5     $usuario = $_POST['usuario'];
6     $contraseña = $_POST['contraseña'];
7     $confirmarContraseña = $_POST['confirmarContraseña'];
8

```



Después se realizaron las validaciones que las contraseñas deben coincidir, la contraseña debe tener al menos 8 caracteres, la contraseña debe incluir al menos una letra mayúscula, una letra minúscula, un número y un carácter especial.

```
if ($contraseña != $confirmarContraseña) {  
    echo "<script>alert('Las contraseñas no coinciden.');} elseif (strlen($contraseña) < 8) {  
    echo "<script>alert('La contraseña debe tener al menos 8 caracteres.');} elseif (!preg_match('/[A-Z]/', $contraseña) || !preg_match('/[a-z]/', $contraseña) ||  
    !preg_match('/[0-9]/', $contraseña) || !preg_match('/[\W]/', $contraseña)) {  
    echo "<script>alert('La contraseña debe contener al menos una letra mayúscula, una letra  
} else {
```

Después tenemos la inserción de la base de datos, la consulta SQL prepara una instrucción para insertar el nombre de usuario y la contraseña en la base de datos, bind\_param vincula los valores del usuario y la contraseña a la consulta preparada, si la inserción es exitosa, muestra un mensaje de éxito; si falla, muestra un mensaje de error.

También se creó un html que es una página de bienvenida para un panel de administrador. En ella se implementó un cierre de sesión para el administrador.

```
<> Administrador.html > ...  
1  
2 <!DOCTYPE html>  
3 <html Lang="es">  
4 <head>  
5     <meta charset="UTF-8">  
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">  
7     <title>Panel de Administrador</title>  
8 </head>  
9 <body>  
10     <h1>Bienvenido a la pagina del Administrador</h1>  
11     <p>Esta es la página principal del administrador en  
12         en donde puedes hacer varias configuraciones.</p>  
13  
14     <a href="logout.php">Cerrar sesión</a>  
15 </body>  
16 </html>
```

Al igual se tuvo que crear una para el usuario. Pero en este caso tuvimos que implementar un cierre de sesión en caso de que no allá actividad por un determinado tiempo que en este caso un minuto.

```

<> Usuario.html > html > body > h1
2   <html lang="en">
3   <head>
7   <script>
9       const inactividadMax = 60000; // 1 minuto
10      let tiempoInactividad;
11
12      // Función para redirigir al logout.php cuando el tiempo de inactividad se agote
13      function cerrarSesion() {
14          window.location.href = "logout.php"; // Redirige a la página de logout
15      }
16
17      // Función para reiniciar el temporizador de inactividad
18      function reiniciarTemporizador() {
19          clearTimeout(tiempoInactividad); // Limpia el temporizador existente
20          tiempoInactividad = setTimeout(cerrarSesion, inactividadMax); // Establece el temporizador de nuevo
21      }
22
23      // Detectar actividad del usuario
24      window.onload = reiniciarTemporizador; // Cuando la página se carga
25      window.onmousemove = reiniciarTemporizador; // Cuando se mueve el mouse
26      window.onmousedown = reiniciarTemporizador; // Cuando se hace clic
27      window.ontouchstart = reiniciarTemporizador; // Cuando se toca la pantalla (en dispositivos móviles)
28      window.onclick = reiniciarTemporizador; // Cuando se hace clic
29      window.onkeydown = reiniciarTemporizador; // Cuando se presiona una tecla
30      window.addEventListener('scroll', reiniciarTemporizador); // Cuando se desplaza la página
31  </script>
32  </head>
33  <body>
34      <h1>Usuario</h1>
35
36      <a href="logout.php">Cerrar sesión</a>

```

Para poder cerrar sesión se tuvo que crear otro archivo llamado logout.php, lo que hace este archivo es destruir todas las sesiones y que nos dirige al login principal.

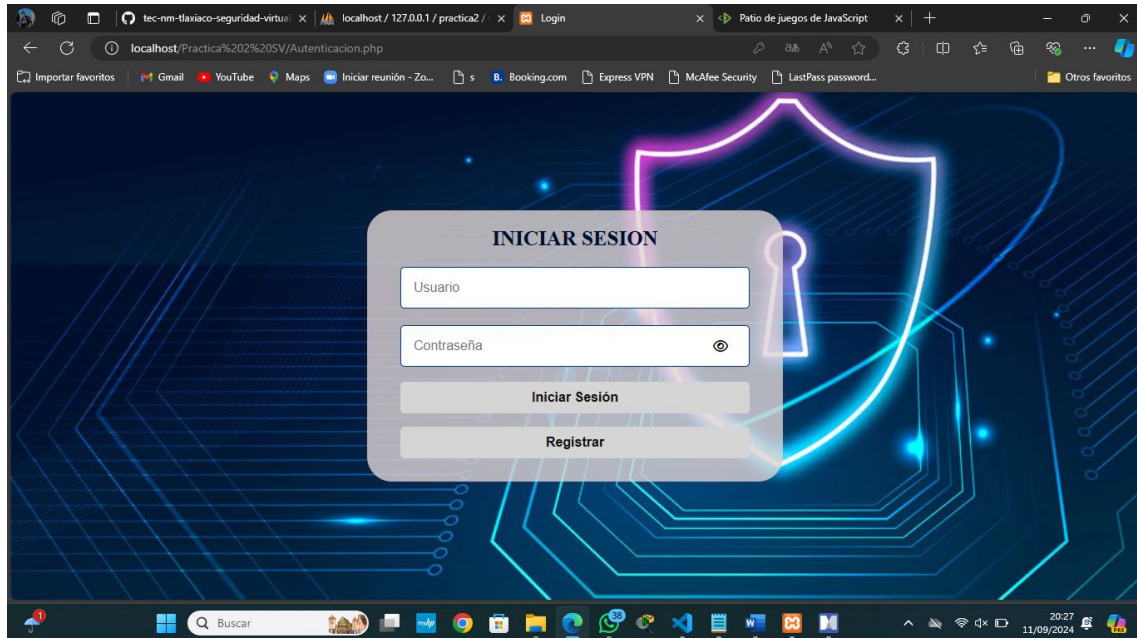
```

logout.php X
C:\xampp\htdocs\Practica 2 SW\logout.php
1   <?php
2   session_start();
3   // Destruir todas las sesiones.
4   session_destroy();
5   // Redirigir a la página de inicio o login
6   header("Location: Autenticacion.php");
7   exit();
8   ?>

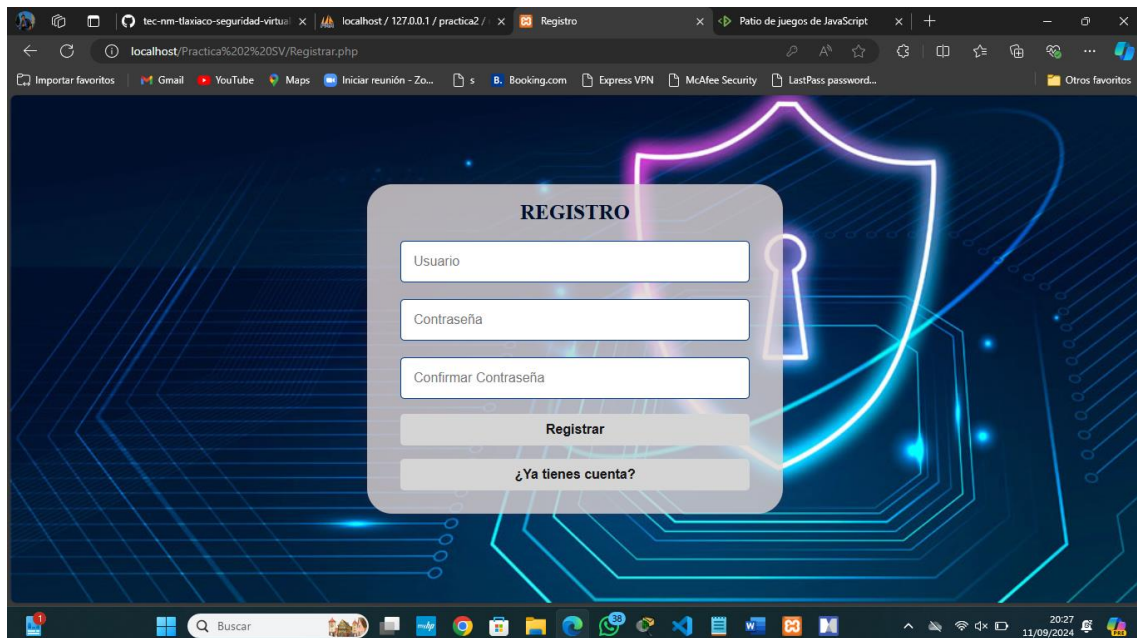
```

## Resultados

La aplicación debe tener un formulario de login con los campos de usuario y contraseña.



La aplicación debe tener un formulario de registro con los campos de usuario, contraseña y confirmación de contraseña.



La aplicación debe decirme si la contraseña es segura o no (extra).

localhost dice

La contraseña debe tener al menos 8 caracteres.  
La contraseña debe contener al menos una letra mayúscula.  
La contraseña debe contener al menos una letra minúscula.  
La contraseña debe contener al menos un carácter especial.

Aceptar

Chali

...

...

Registrar

¿Ya tienes cuenta?

La aplicación debe tener una página de inicio que sea accesible para cualquier usuario.

Inicio de Sesión

Usuario

Contraseña

Iniciar Sesión

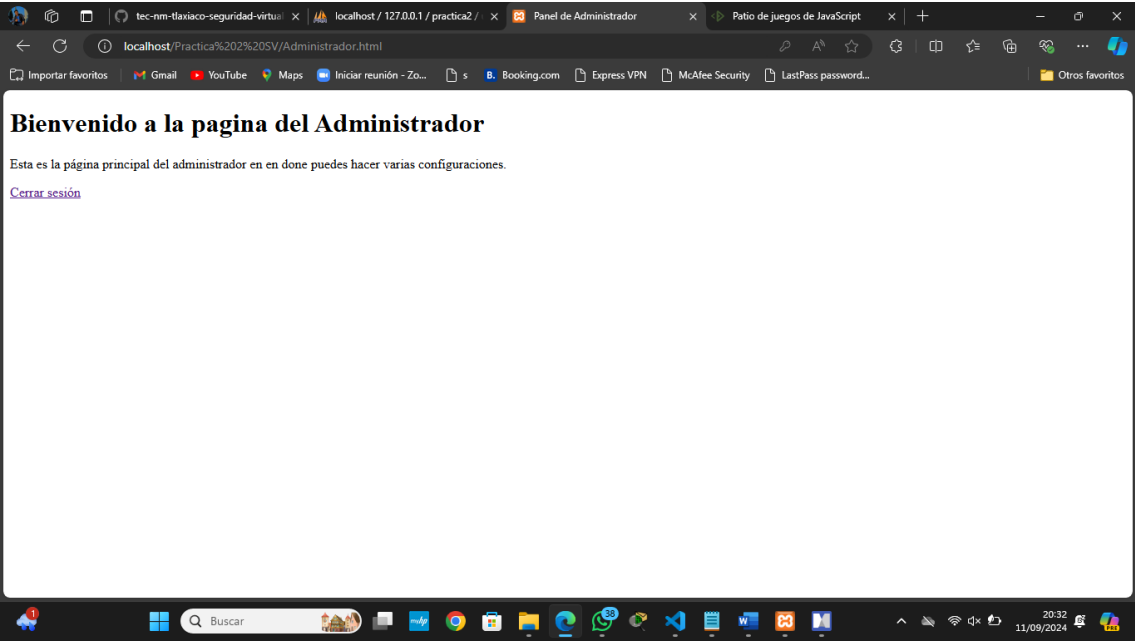
Registrar

La aplicación debe tener una página de perfil que solo sea accesible si el usuario ha iniciado sesión.

Usuario

[Cerrar sesión](#)

La aplicación debe tener una página de administración que solo sea accesible si el usuario ha iniciado sesión y tiene un rol de administrador.



Para serrar sesión en el usuario y administrador.

<b>Usuario</b>	<b>Bienvenido a la pagina del Administrador</b>
Esta es la página principal del administrador en en done puedes hacer varias configuraciones.	Esta es la página principal del administrador en en done puedes hacer varias configuraciones.
<a href="#">Cerrar sesión</a>	<a href="#">Cerrar sesión</a>

## **Conclusión**

Esta aplicación plantea un sistema integral de autenticación y autorización, adecuado para diversas plataformas (web, móvil o escritorio). Se estructura en torno a varios componentes clave:

**Autenticación básica:** Permite a los usuarios registrarse e iniciar sesión con un nombre de usuario y contraseña. El formulario de login permite acceder al sistema, mientras que el formulario de registro asegura que las contraseñas se confirmen correctamente.

**Validación de contraseñas:** Para aumentar la seguridad, se agrega una funcionalidad adicional que evalúa la seguridad de la contraseña, asegurando que cumpla con ciertos criterios.

**Control de acceso basado en roles:** La aplicación cuenta con un sistema de autorización que restringe el acceso a ciertas páginas dependiendo del estado del usuario. Cualquier usuario puede acceder a la página de inicio, pero solo los usuarios autenticados pueden ver su perfil, y solo aquellos con el rol de administrador pueden acceder a la página de administración.

**Gestión de sesiones:** Se incluye un sistema para cerrar la sesión de los usuarios tras 1 minutos de inactividad, mejorando la seguridad y previniendo accesos no autorizados por inactividad prolongada.