



## **TECNOLÓGICO NACIONAL DE MÉXICO INSTITUTO TECNOLÓGICO DE TLAXIACO**

---

### **Seguridad Y Virtualización**

---

#### **Alumnas:**

Julissa Migdalia José Cruz

Luz María Mendoza Cortes

Angeles González Martínez

Elisahadad Mayte León de Jesús

#### **Tema:**

Práctica 2

#### **Docente:**

Ing. Edward Osorio Salinas

#### **Carrera:**

Ingeniera en Sistemas Computacionales

#### **Grupo: 7US**

**Tlaxiaco, Oaxaca.. A 17 de Octubre de 2024.**

*“Educación, Ciencia y Tecnología, Progresos día con día”®*

Boulevard Tecnológico Km. 2.5, Llano Yosovee C.P. 69800. Tlaxiaco. Oax. México.

Tels. Dir. (953) 55 20788, (953) 55 21322, (953) 55 20405 e-mail:

[dir\\_tlaxiaco@tecnm.mx](mailto:dir_tlaxiaco@tecnm.mx) | [www.tlaxiaco.tecnm.mx](http://www.tlaxiaco.tecnm.mx)



## Introducción

En el contexto de la seguridad de la información, dos conceptos fundamentales son la autenticación y la autorización, los cuales juegan un papel crucial en la protección de sistemas y datos.

La autenticación es el proceso de verificar la identidad de un usuario o entidad. A través de diversos mecanismos, como contraseñas, certificados digitales o biometría, se asegura que quien intenta acceder a un sistema sea quien dice ser. Este paso inicial es esencial para evitar accesos no autorizados y suplantaciones de identidad. Por otro lado, la autorización se refiere a la concesión de permisos y privilegios a un usuario ya autenticado. Una vez que el sistema ha confirmado la identidad del usuario, determina qué recursos o acciones tiene permitido realizar. La autorización, por lo tanto, se basa en políticas de acceso que pueden depender del rol del usuario, su nivel de confianza o las necesidades operativas.

## Material

- ✓ Ordenador
- ✓ Xampp
- ✓ Visual studio code

## Objetivo

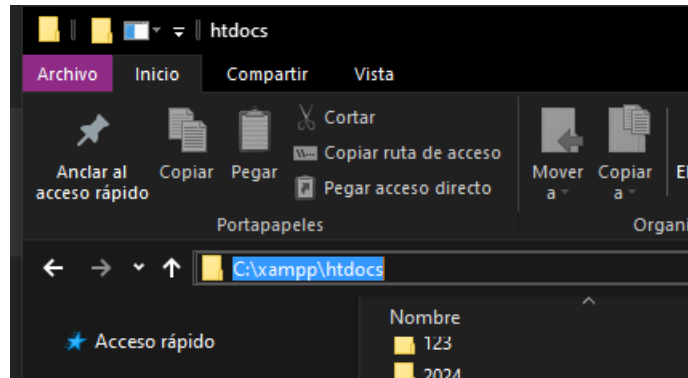
El objetivo de esta práctica es que el alumno conozca y aplique los conceptos de autorización y autenticación en la seguridad de la información

## Instrucciones

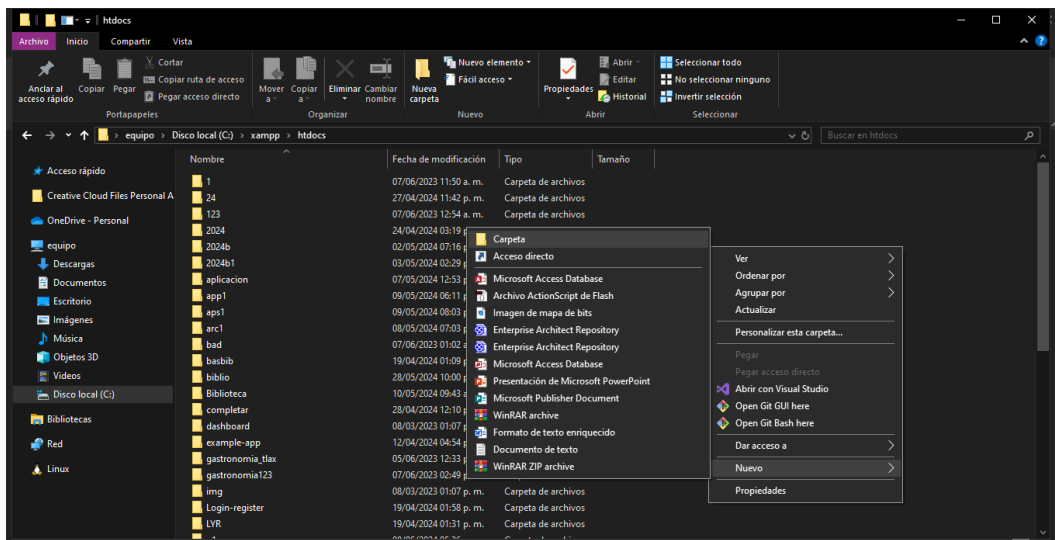
1. Crea una aplicación [web|mobile|escritorio] que permita loguearse con un usuario y contraseña.
  - La aplicación debe tener un formulario de login con los campos de usuario y contraseña.
  - La aplicación debe tener un formulario de registro con los campos de usuario, contraseña y confirmación de contraseña.
  - La aplicación debe decirme si la contraseña es segura o no (extra).
  - La aplicación debe tener una página de inicio que sea accesible para cualquier usuario.
  - La aplicación debe tener una página de perfil que solo sea accesible si el usuario ha iniciado 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.

## DESARROLLO DE LA PRÁCTICA

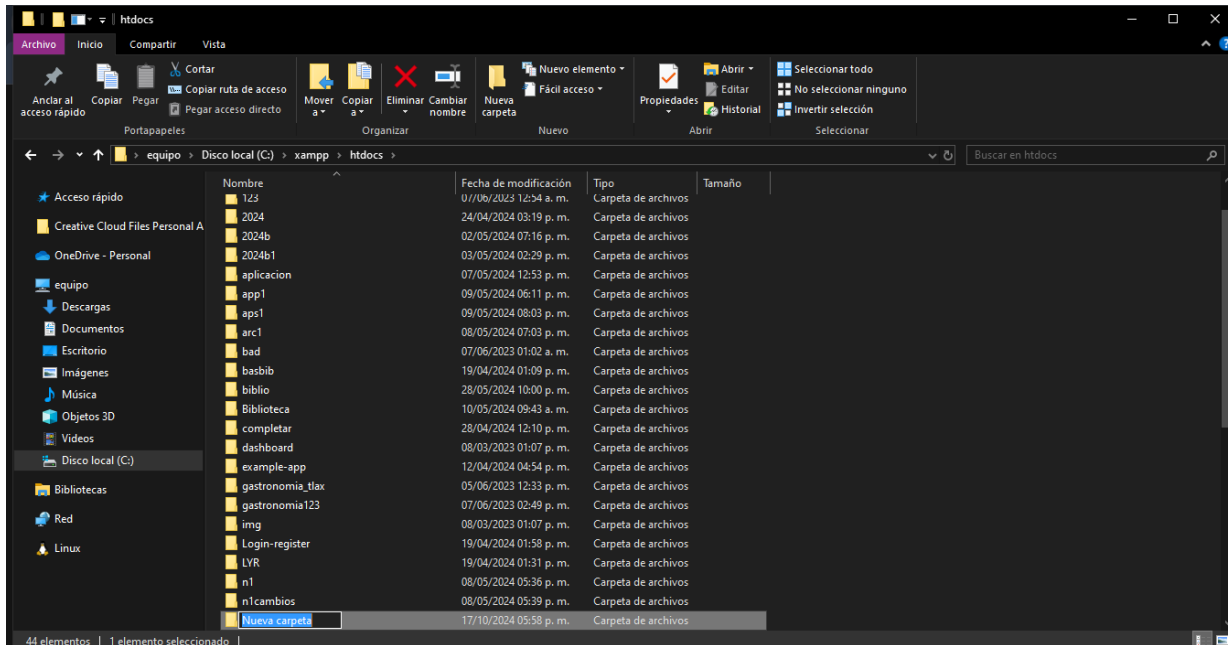
1. Empezaremos por estructurar el almacenamiento de los archivos de la práctica en la siguiente ruta C:/xampp/htdocs



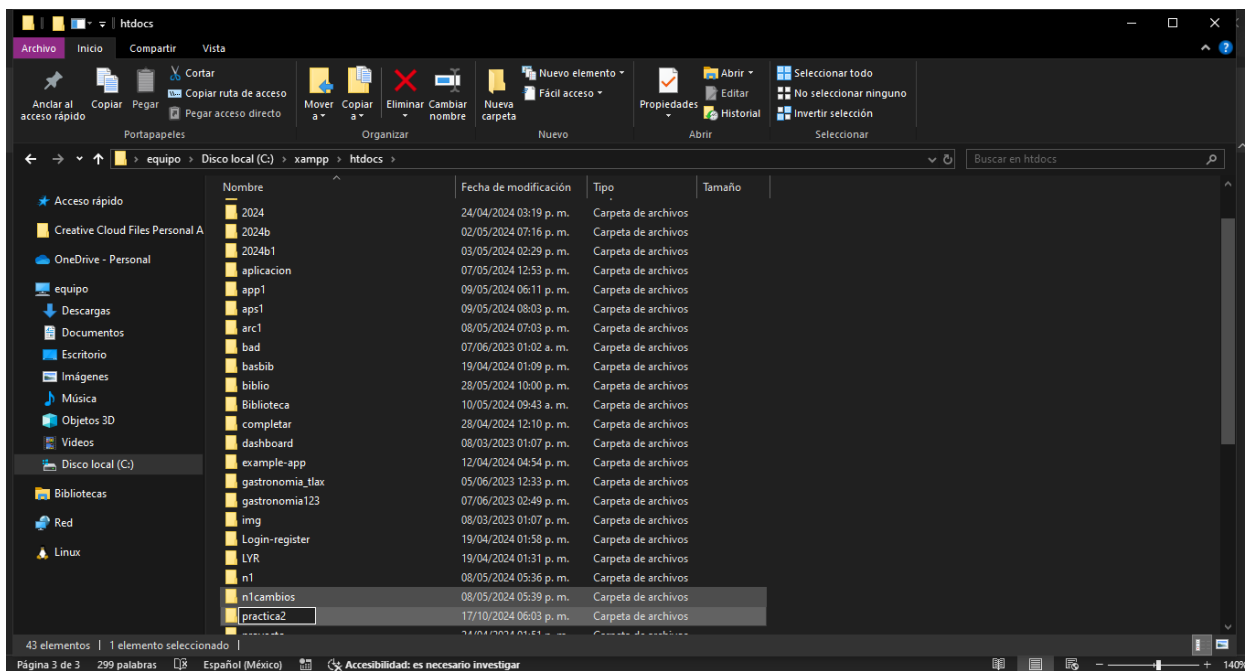
2. El enrutamiento se reestructura seguido por el nombre del proyecto para esto la acción será la siguiente añadiremos una nueva carpeta dentro de la ruta de la siguiente forma estando en la ruta damos clic derecho nos aparece un menú pulsamos en nuevo y seleccionamos carpeta como lo mostramos a continuación:



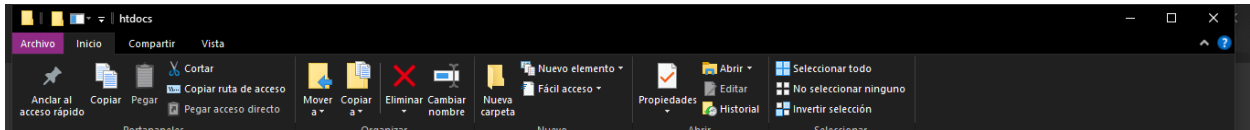
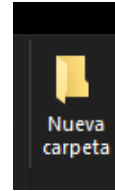
1. Seguido de la acción nos mostrará la nueva carpeta con de manera que aparecerá con la edición para cambiarle el nombre.



2. En nuestro caso el nombre es práctica2, así como se notará en la siguiente imagen.



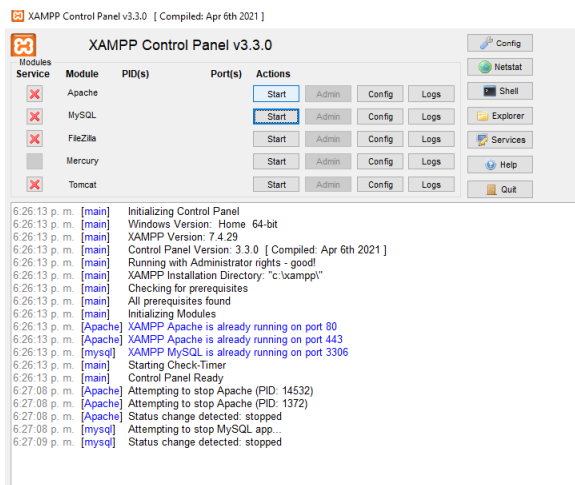
3. Aunque también se puede realizar esta acción de la manera práctica ubicado con la barra de herramientas de arriba con la acción nueva carpeta de modo que se verá de la siguiente manera.



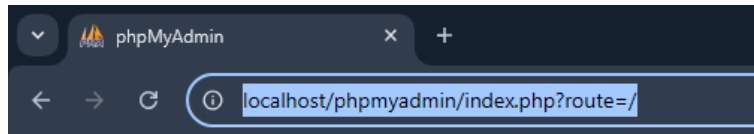
4. Una vez realizado procedemos a crear e insertar la base de datos en nuestro caso con la herramienta de xampp para esto activamos como administrador la aplicación de xampp una vez instalada pulsamos sí.



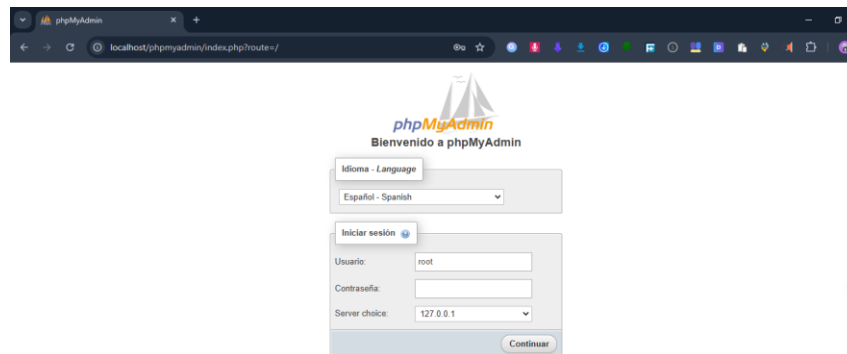
5. Procedemos a activar los puertos y servicios a utilizar esta acción se realiza pulsando en START y seguido de pulsar en admin.



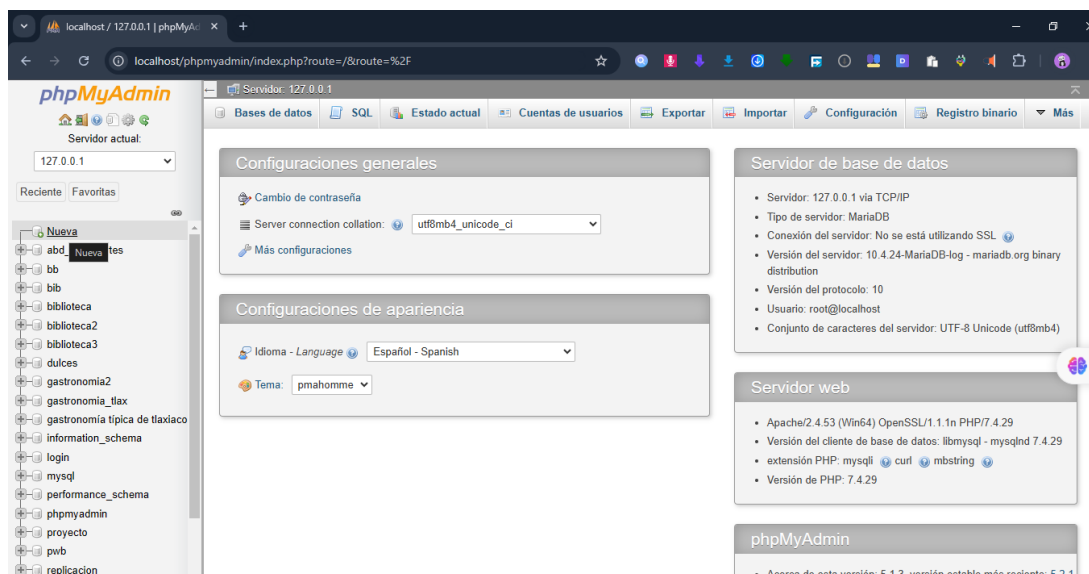
6. Posteriormente nos abrirá en el navegador el xampp con la ruta local como la siguiente .



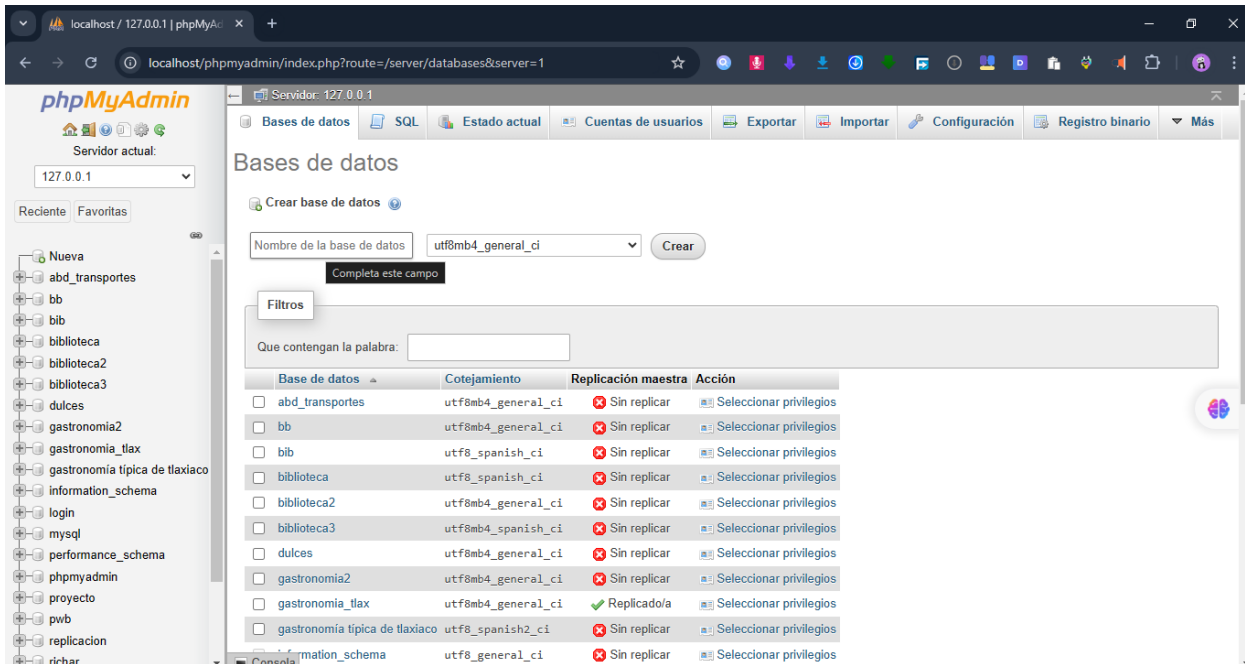
7. En caso de no tener usuario y contraseña o de ser la primera ocasión de usar la aplicación de Xampp la utilidad de root y sin contraseña es por default.



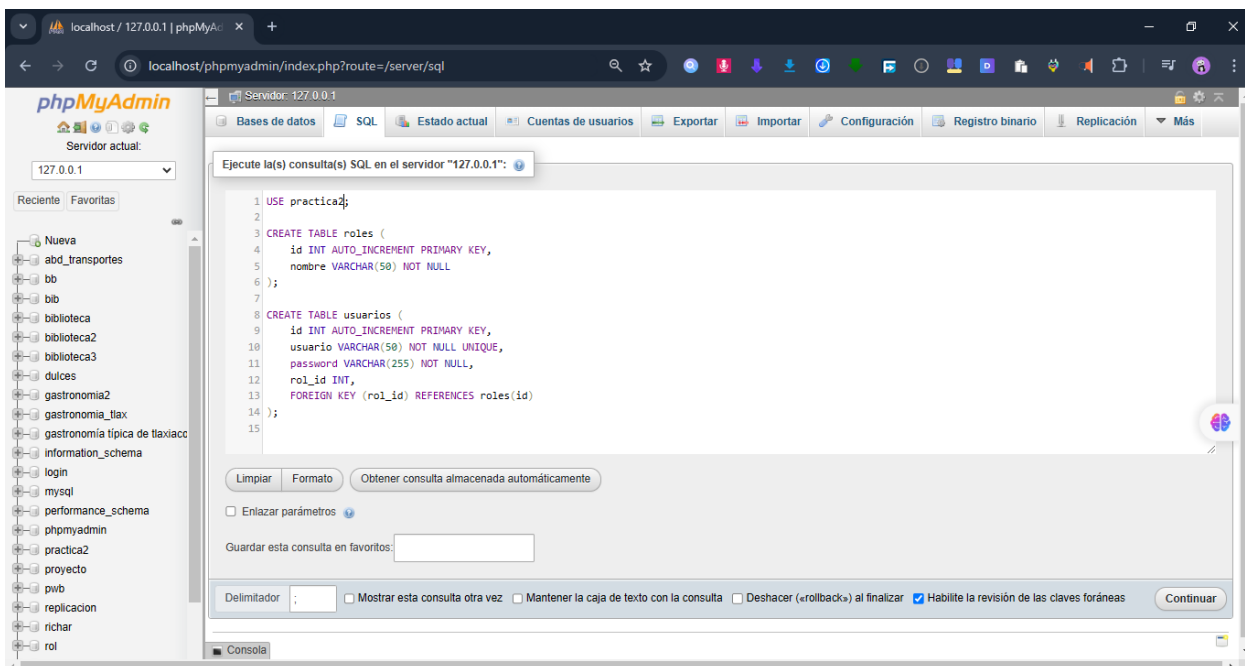
8. Después de haberse logueado Ahora procedemos a entrar la interfaz de la aplicación iremos al menú lateral izquierdo y optaremos por la opción de nueva para la creación de la base de datos.



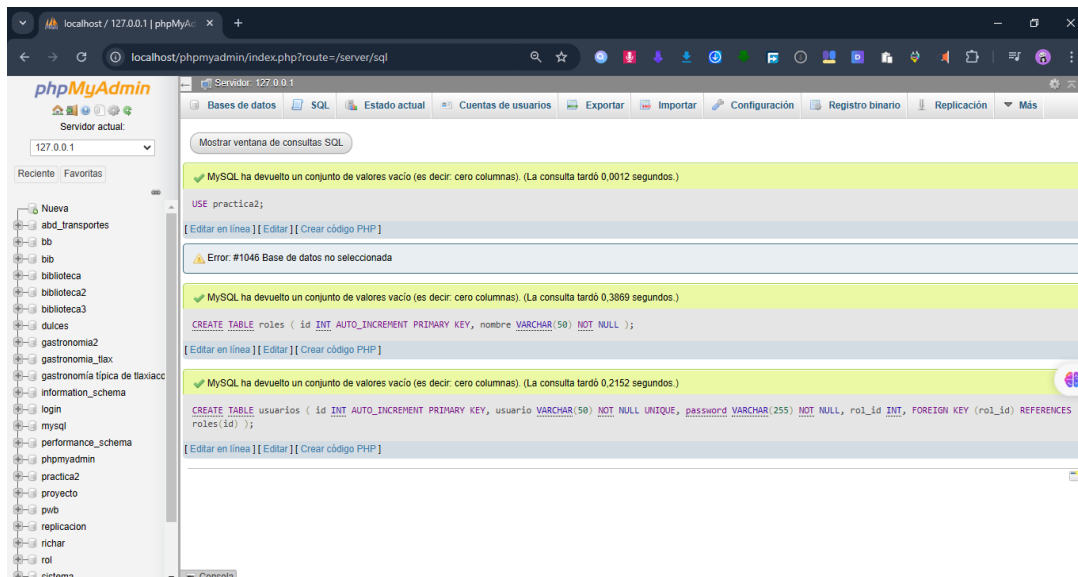
9. Una vez que pulsamos nos recargará la ventana del navegador donde nos aparecerá 2 opciones una es crearla de manera gráfica o por código.



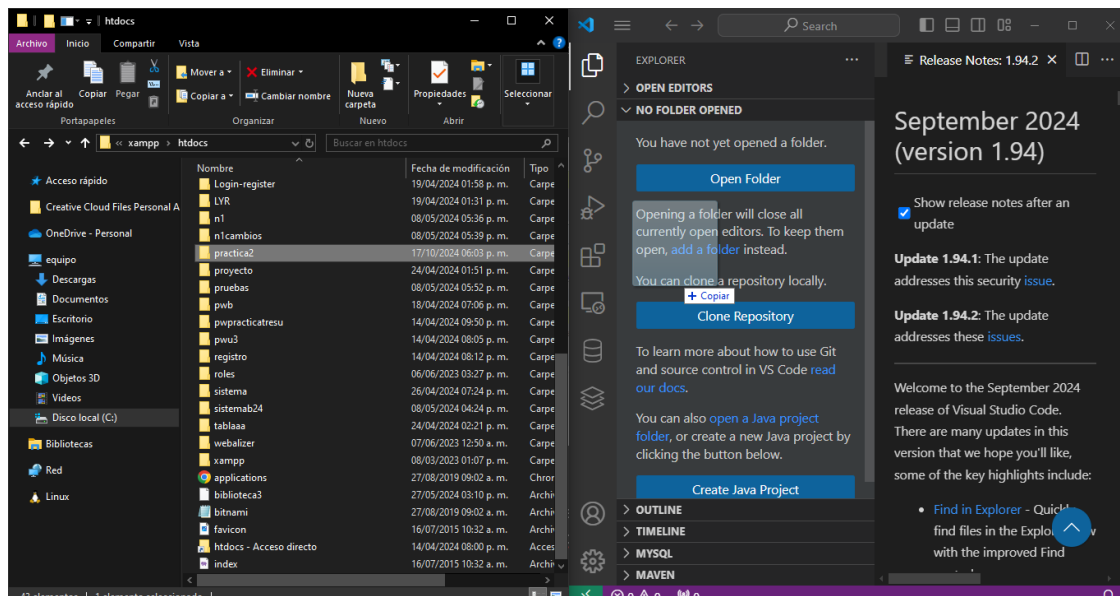
10. En nuestro caso será por código de la siguiente manera pulsaremos en la opción de SQL ubicado en la parte de arriba donde la base de datos será practica2 con dos tablas: usuarios y roles de modo que pulsamos en continuar.



11. Empezará a cargar y este una vez culminada la acción nos mostrará la siguiente.

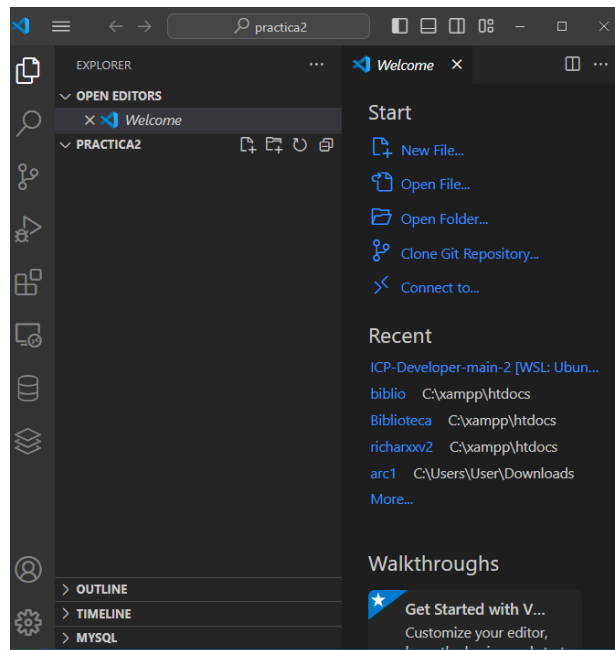


12. Ahora en Visual studio code lo abrimos la carpeta anteriormente creada ya sea que lo abramos desde visual o lo arrastremos a visual studio para que reestructuremos los archivos de almacenamiento.

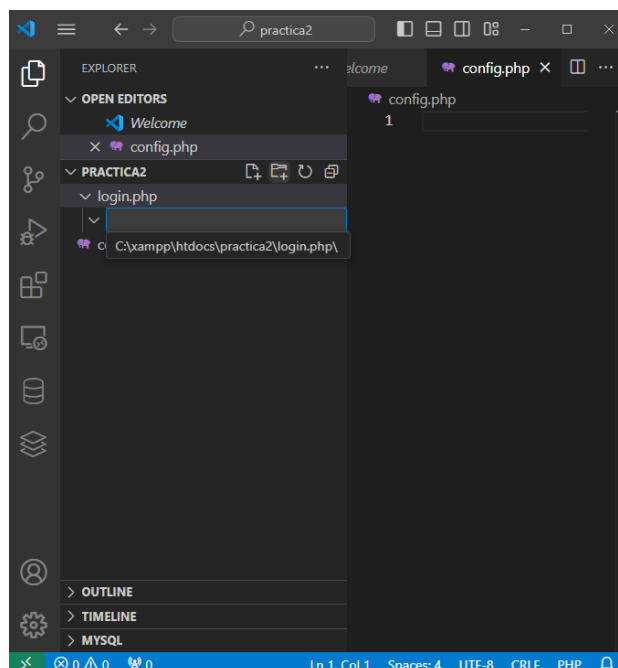




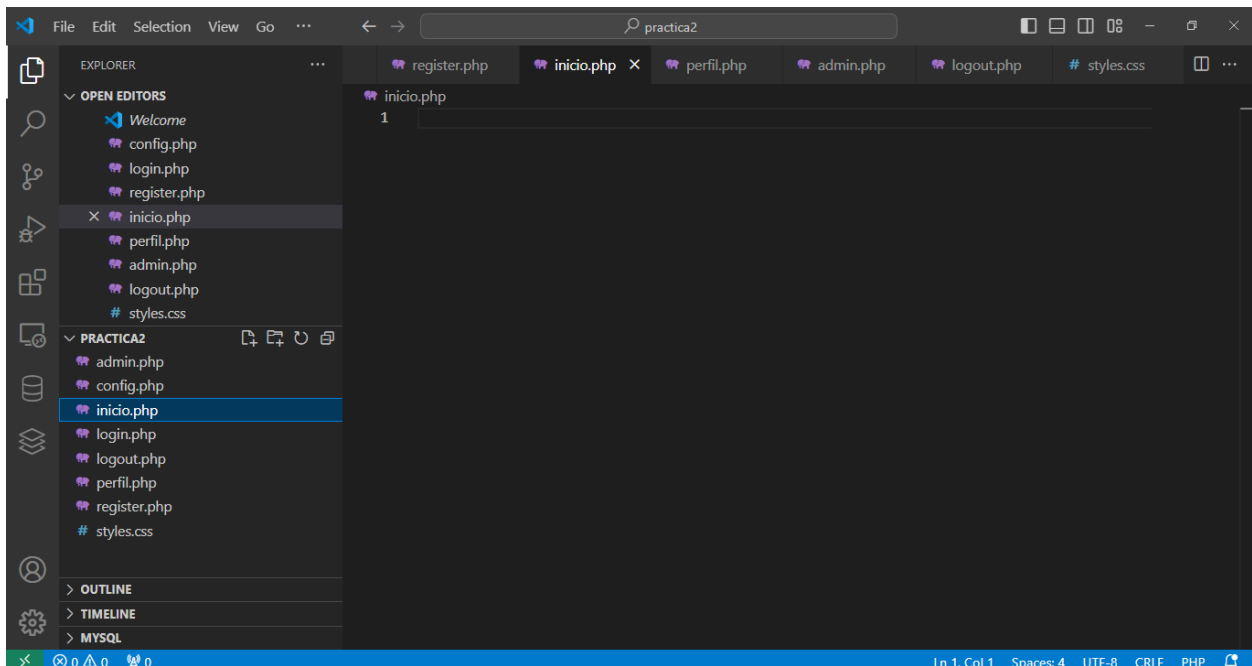
13. Se vera de la siguiente manera para ir insertando los archivos correspondientes.



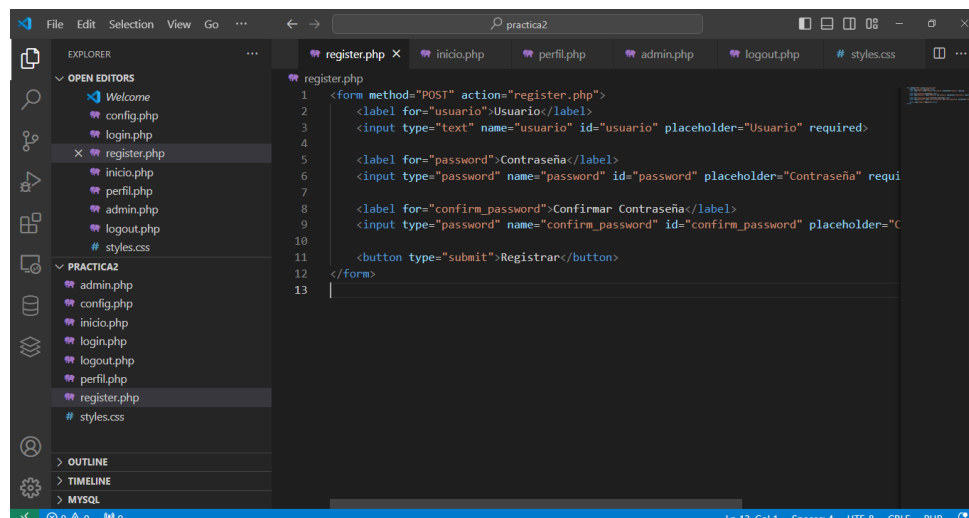
Creamos los siguientes archivos dándole en el icono de crear.



14. Una vez realizado en la inserción de todos los archivos serán los siguientes.



15. En el archivo de register.php el código será lo siguiente.

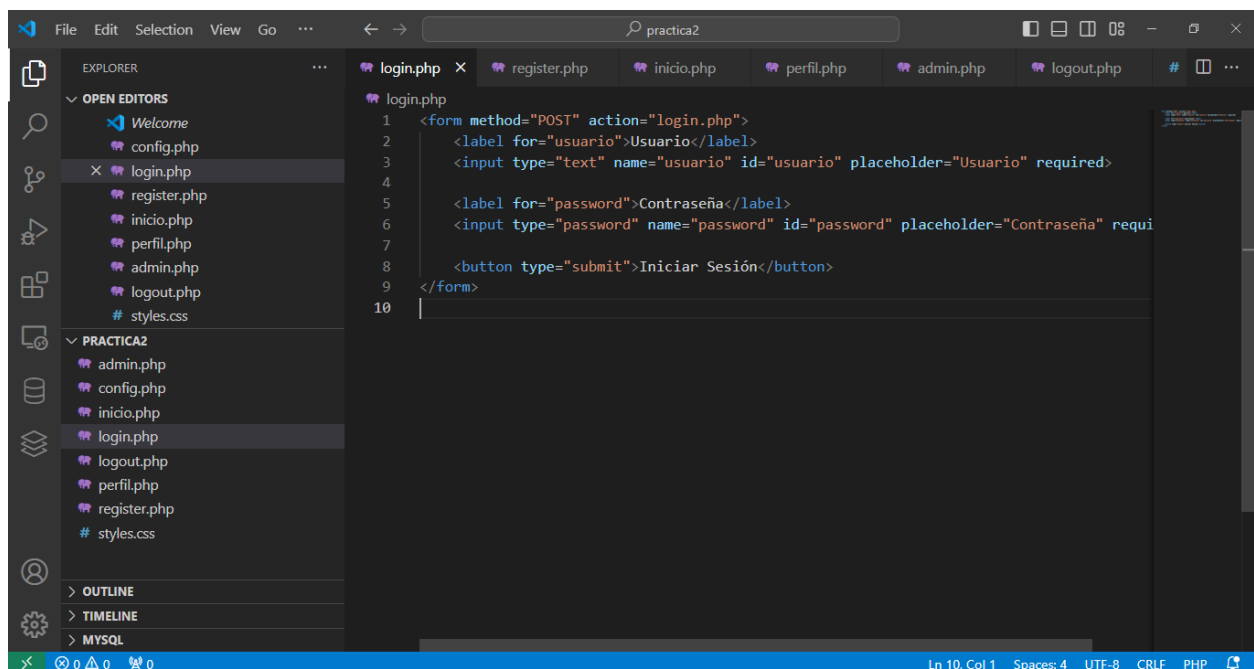


```

1  <?php
2  class UsuariosModel extends Query{
3      private $usuario, $nombre, $clave, $id, $estado;
4      public function __construct()
5      {
6          parent::__construct();
7      }
8      public function getUsuario($usuario, $clave)
9      {
10         $sql = "SELECT * FROM usuarios WHERE usuario = '$usuario' AND clave = '$clave'";
11         $data = $this->select($sql);
12         return $data;
13     }
14     public function getUsuarios()
15     {
16         $sql = "SELECT * FROM usuarios";
17         $data = $this->selectAll($sql);
18         return $data;
19     }
20     public function registrarUsuario($usuario, $nombre, $clave)
21     {
22         $this->usuario = $usuario;
23         $this->nombre = $nombre;
24         $this->clave = $clave;
25         $vericar = "SELECT * FROM usuarios WHERE usuario = '$this->usuario'";
26         $existe = $this->select($vericar);

```

16. Posteriormente el archivo de login.php el código será lo siguiente.



```

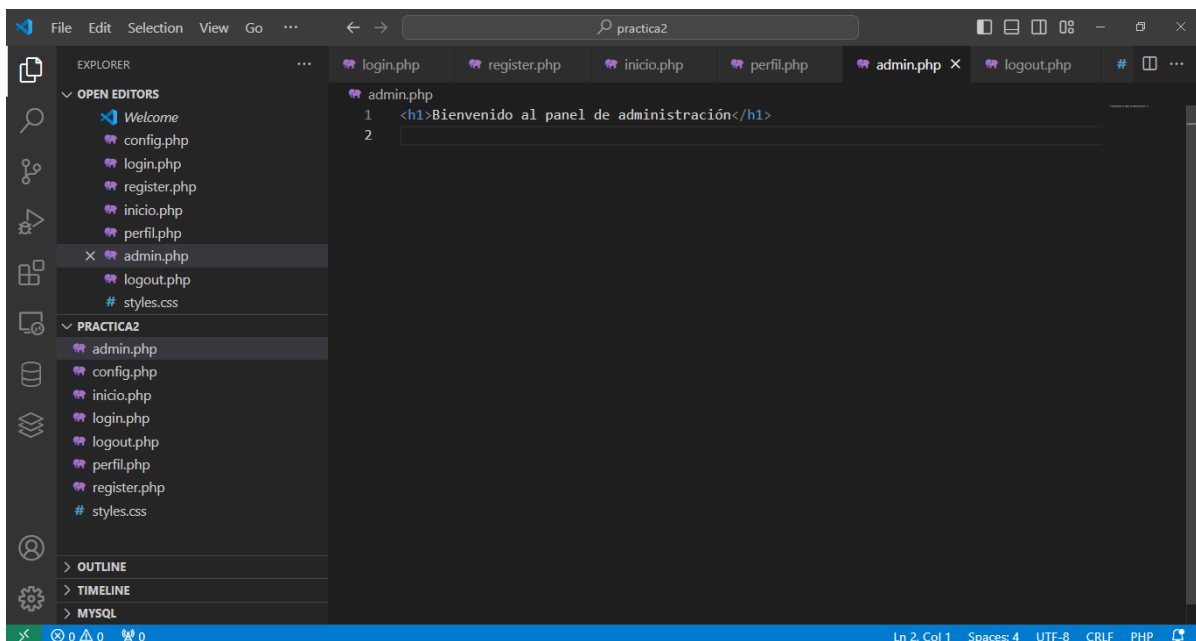
login.php
1  <form method="POST" action="login.php">
2      <label for="usuario">Usuario</label>
3      <input type="text" name="usuario" id="usuario" placeholder="Usuario" required>
4
5      <label for="password">Contraseña</label>
6      <input type="password" name="password" id="password" placeholder="Contraseña" requi
7
8      <button type="submit">Iniciar Sesión</button>
9  </form>
10

```

17. El siguiente archivo de inicio.php el código será lo siguiente.

```
C:\Users\User> Downloads > admin.html > html > head > style > h2 i
2  <html lang="es">
95 <body>
96   <div class="container">
107   <div id="usuarios" class="section">
109     <table>
110       <thead>
111         <tr>
112           <th>ID</th>
113           <th>Nombre</th>
114           <th>Email</th>
115           <th>Acciones</th>
116         </tr>
117       </thead>
118       <tbody>
119         <tr>
120           <td>1</td>
121           <td>Ana López</td>
122           <td>ana@example.com</td>
123           <td>
124             <button class="btn edit"><i class="fas fa-edit"></i></button>
125             <button class="btn delete"><i class="fas fa-trash"></i></button>
126           </td>
127         </tr>
128         <tr>
129           <td>2</td>
130           <td>Carlos Martínez</td>
```

18. Otro del archivo de admin.php el código será lo siguiente.

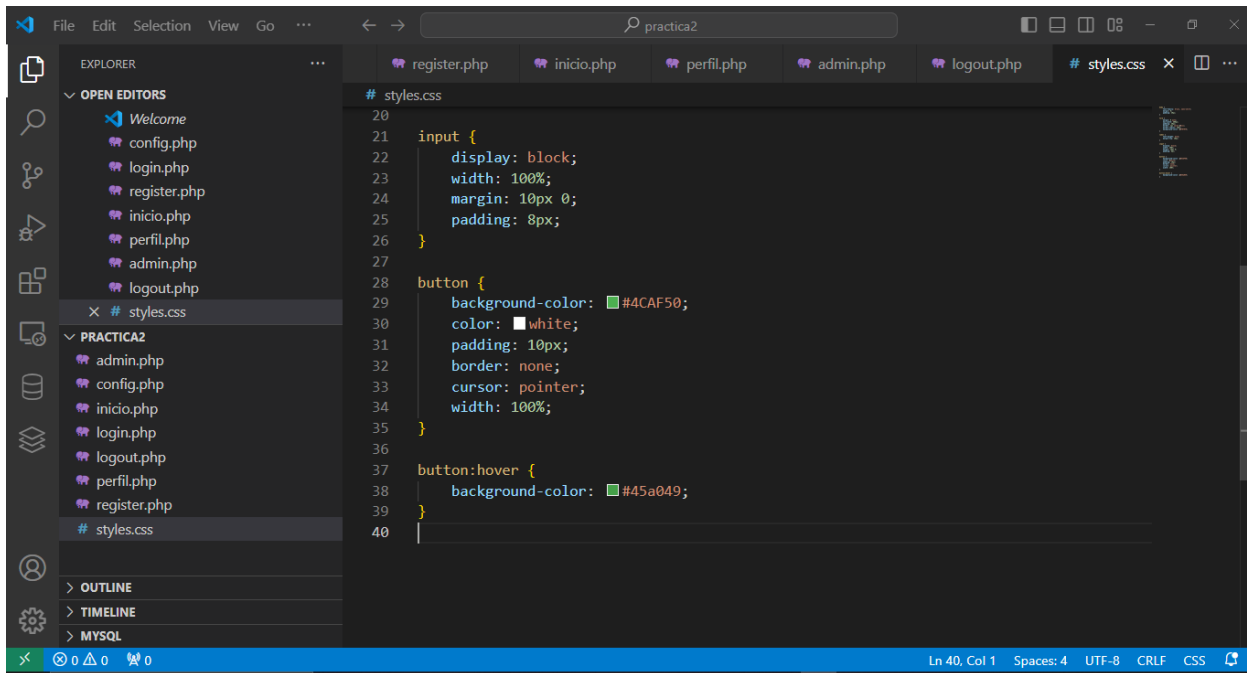


The screenshot shows the Visual Studio Code editor with the 'admin.php' file open. The Explorer sidebar on the left shows a project structure with files like 'login.php', 'register.php', 'inicio.php', 'perfil.php', 'admin.php', 'logout.php', and 'styles.css'. The main editor area shows the content of 'admin.php' with two lines of code:

```
1 <h1>Bienvenido al panel de administración</h1>
2
```

The status bar at the bottom indicates the current position is Line 2, Column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the PHP file type.

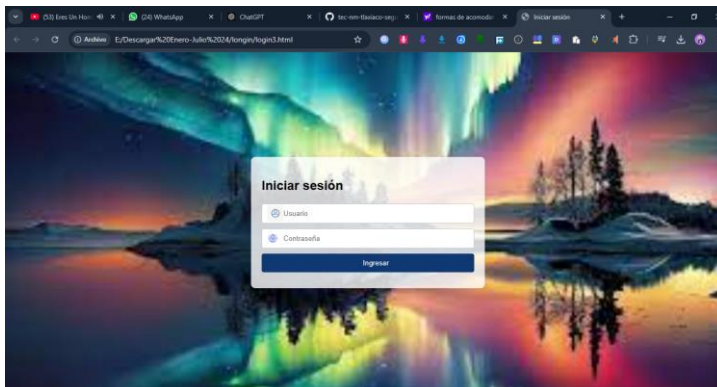
19. Podemos observar los archivos de styles.css el código será lo siguiente.



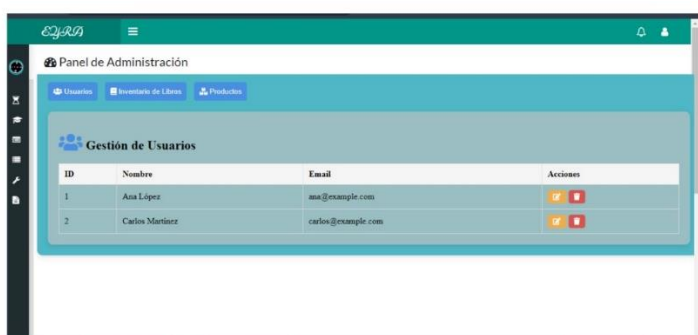
```
# styles.css
20
21 input {
22     display: block;
23     width: 100%;
24     margin: 10px 0;
25     padding: 8px;
26 }
27
28 button {
29     background-color: #4CAF50;
30     color: white;
31     padding: 10px;
32     border: none;
33     cursor: pointer;
34     width: 100%;
35 }
36
37 button:hover {
38     background-color: #45a049;
39 }
40
```

Cada pantalla se vera de la siguiente manera, de acuerdo a las peticiones

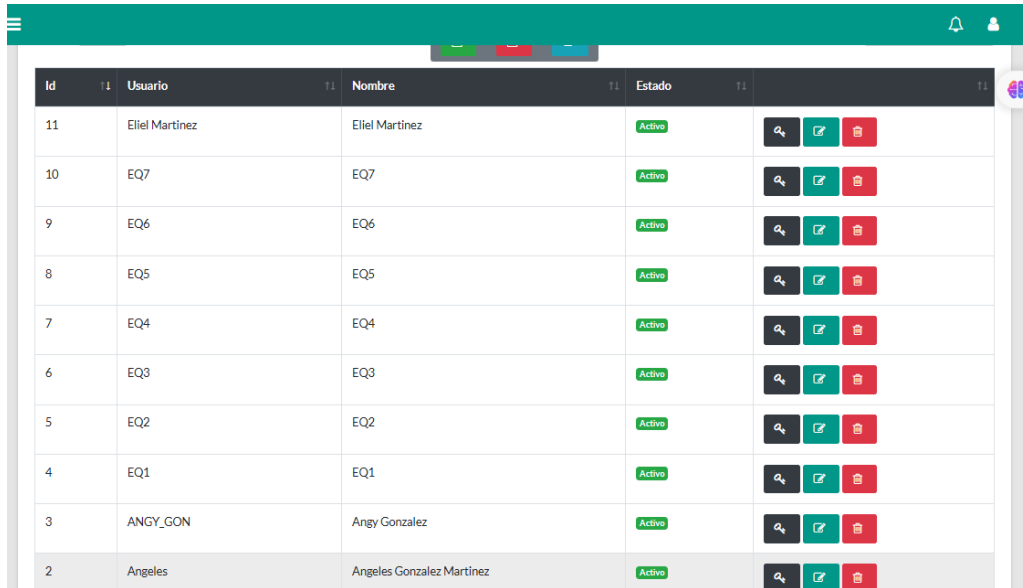
## Login

































## Panel de administración

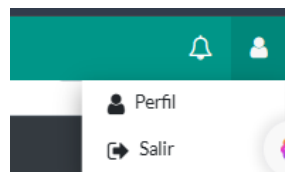


- Implementa un mecanismo de autorización que permita o deniegue el acceso a ciertas rutas de la aplicación, en este caso la página de perfil y la página de administración si en dado caso el usuario no ha iniciado sesión o no tiene el rol de administrador.



Id	Usuario	Nombre	Estado	
11	Eliel Martinez	Eliel Martinez	Activo	  
10	EQ7	EQ7	Activo	  
9	EQ6	EQ6	Activo	  
8	EQ5	EQ5	Activo	  
7	EQ4	EQ4	Activo	  
6	EQ3	EQ3	Activo	  
5	EQ2	EQ2	Activo	  
4	EQ1	EQ1	Activo	  
3	ANGY_GON	Angy Gonzalez	Activo	  
2	Angeles	Angeles Gonzalez Martinez	Activo	  

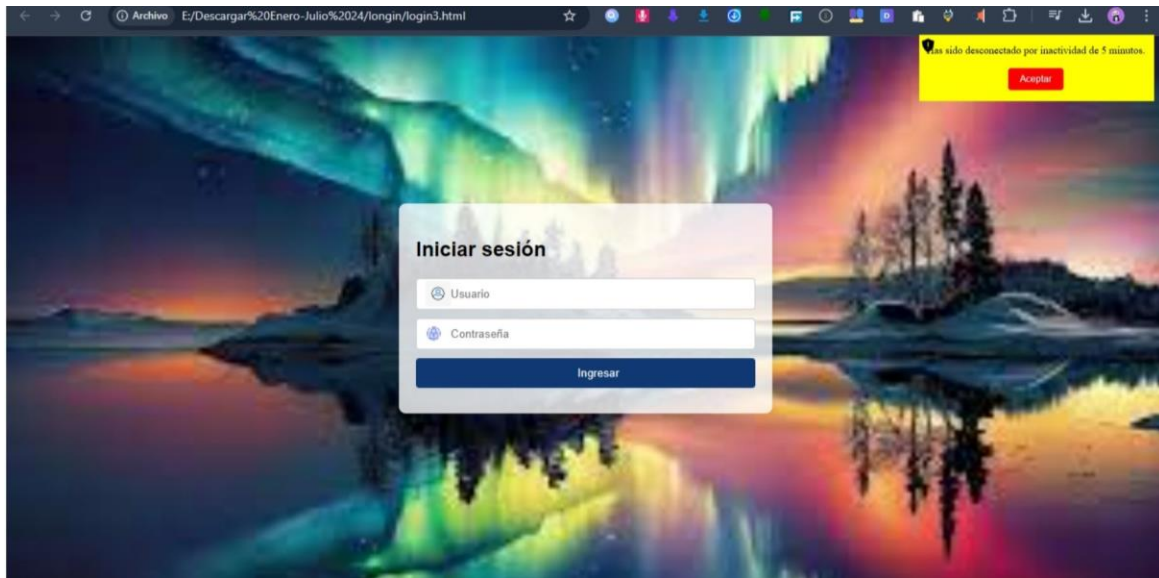
- Implementa un mecanismo de autenticación que permita a los usuarios registrarse, iniciar sesión y cerrar sesión.



- Implementa un mecanismo para cerrar sesión de un usuario si ha pasado un tiempo determinado sin actividad de 5 mins.

Añadimos la funcion en este fragmento de código para que cierre y reinicie

```
08
09     function logOut() {
10         alert("Has sido desconectado por inactividad.");
11         window.location.href = "login.html"; // Cambia esto a tu URL de inicio de sesión
12     }
13
14     // Reiniciar el temporizador en eventos de interacción
15     window.onload = resetTimer;
16     document.onmousemove = resetTimer; // Reinicia al mover el ratón
17     document.onkeypress = resetTimer; // Reinicia al presionar una tecla
18     document.ontouchstart = resetTimer; // Reinicia en dispositivos táctiles
19
```



5.- Investiga y describe los siguientes servicios de autenticación

### **LDAP (Lightweight Directory Access Protocol)**

El LDAP es un protocolo utilizado para acceder y mantener servicios de directorio distribuidos a través de una red, como los que contienen datos sobre usuarios, equipos, permisos y configuraciones. Funciona como un mecanismo de autenticación, permitiendo a los usuarios acceder a diversos recursos en una red. Aunque no está diseñado específicamente como un protocolo de autenticación, suele utilizarse junto con otros sistemas para verificar identidades.

Características:

- Protocolo ligero y optimizado para leer, buscar y autenticar contra bases de datos de directorios.
- Usa una estructura jerárquica similar a los árboles, con entradas que pueden ser usuarios, grupos, dispositivos, etc.
- Utilizado comúnmente en aplicaciones como Microsoft Active Directory, OpenLDAP y Novell eDirectory.
- Funciona sobre el protocolo TCP/IP, generalmente en los puertos 389 (no seguro) y 636 (seguro con SSL/TLS).
- Permite tanto autenticación simple como autenticación basada en mecanismos más robustos como SASL (Simple Authentication and Security Layer).

Casos de uso: Gestión de usuarios y autenticación centralizada en entornos de red empresariales.

## **RADIUS (Remote Authentication Dial-In User Service)**

RADIUS es un protocolo de red que proporciona autenticación, autorización y contabilidad (AAA) para la administración centralizada de acceso a redes, como VPNs, Wi-Fi, servidores de acceso remoto, etc. Es muy utilizado en ISP y entornos empresariales que requieren control sobre el acceso a la red.

Características:

- Proporciona autenticación centralizada para servicios de red.
- Divide las funciones de autenticación, autorización y contabilidad en un solo servicio.
- Soporta varios métodos de autenticación, como contraseñas o certificados.
- Utiliza el protocolo UDP (puertos 1812 y 1813).
- Diseñado para autenticar de manera eficiente a usuarios que intentan acceder remotamente a una red o recurso.
- Funciona bien en entornos distribuidos, donde el cliente envía una solicitud al servidor RADIUS, que valida las credenciales contra una base de datos.

Casos de uso: Acceso remoto a VPNs, autenticación para redes Wi-Fi empresariales, autenticación de servicios de red como routers o switches.

## **TACACS+ (Terminal Access Controller Access Control System Plus)**

TACACS+ es un protocolo similar a RADIUS, pero enfocado en proporcionar más control granular sobre el acceso y la autorización de dispositivos y recursos en una red. Fue desarrollado por Cisco y es ampliamente utilizado en entornos de red para la autenticación de administradores y la gestión de equipos de red como routers y switches.

Características:

- Proporciona AAA (autenticación, autorización y contabilidad), pero separa las tres funciones, a diferencia de RADIUS.
- Funciona sobre TCP (puerto 49), lo que lo hace más confiable y asegura la entrega de paquetes.
- Cifra todo el contenido del mensaje, no solo la contraseña (a diferencia de RADIUS, que solo cifra las credenciales).
- Mejora el control sobre los comandos ejecutados en dispositivos de red.
- Ofrece un nivel de autorización más detallado, permitiendo control sobre qué comandos pueden ejecutar ciertos usuarios.



Casos de uso: Control de acceso de administradores a dispositivos de red, como routers y switches, en grandes infraestructuras empresariales.

### **Kerberos**

Kerberos es un protocolo de autenticación basado en un sistema de "tickets", diseñado para proporcionar autenticación fuerte en redes de computadora. Utiliza criptografía de clave simétrica para autenticar a usuarios y dispositivos en una red. Fue desarrollado por el MIT y es utilizado comúnmente en entornos como Microsoft Active Directory.

Características:

- Protocolo seguro basado en el uso de criptografía simétrica.
- Opera bajo el concepto de "tickets" otorgados por un KDC (Key Distribution Center), que el cliente usa para acceder a servicios dentro de la red.
- Evita el envío de contraseñas sin cifrar a través de la red.
- Utiliza el puerto 88 (UDP o TCP).
- Es capaz de autenticar tanto usuarios como dispositivos dentro de un dominio o entorno distribuido.
- Asegura que las contraseñas no se transmitan por la red, solo se utilizan tokens cifrados.

Casos de uso: Autenticación en redes empresariales que requieren alta seguridad, principalmente en sistemas operativos como Windows (con Active Directory), Linux y Unix.

6.- Investiga y describe los siguientes servicios de autorización

### **ACL (Access Control List)**

ACL (Lista de Control de Acceso) es una técnica de control de acceso que define qué usuarios o sistemas tienen permiso para acceder a un recurso específico, como un archivo, directorio, servicio o dispositivo. Una ACL es básicamente una lista que contiene reglas de permisos asociadas a un recurso, indicando qué operaciones (lectura, escritura, ejecución, etc.) pueden realizarse y por quién.

Características:

- Reglas de acceso sencillas: Cada entrada en la lista especifica un sujeto (usuario o grupo) y el permiso asociado (lectura, escritura, ejecución, etc.).
- Estática: Las reglas no cambian en función de atributos dinámicos o contexto (como la hora o ubicación).

- Directamente asignada al recurso: Cada recurso (archivo, carpeta, etc.) tiene su propia ACL.

Tipos comunes de ACL:

- ACL estándar: Define permisos a nivel de IP o dirección de red.
- ACL extendida: Define permisos en función de servicios, tipos de tráfico y otros criterios.

Casos de uso: Control de acceso a archivos y directorios en sistemas operativos como Linux o Windows, restricciones de tráfico en dispositivos de red (firewalls, routers).

### **RBAC (Role-Based Access Control)**

RBAC (Control de Acceso Basado en Roles) es un modelo de autorización que asigna permisos a los usuarios en función de los "roles" que ocupan dentro de una organización. Los usuarios no reciben permisos directos, sino que obtienen acceso en función de los roles a los que están asociados. Un rol representa un conjunto de permisos que están agrupados según las responsabilidades comunes.

Características:

- Asignación basada en roles: Los usuarios se agrupan en roles (por ejemplo, administrador, editor, lector), y cada rol tiene permisos predefinidos.
- Jerarquía de roles: Los roles pueden organizarse en una estructura jerárquica, donde los roles de mayor nivel heredan permisos de los roles de nivel inferior.
- Simplicidad en la gestión: Facilita la administración de permisos, ya que no se asignan a usuarios individuales, sino a roles, lo que mejora la escalabilidad y seguridad.
- Menor flexibilidad: No permite condiciones contextuales, como basar el acceso en atributos dinámicos del usuario o del entorno.

Casos de uso: Gestión de acceso en sistemas empresariales grandes, como bases de datos, sistemas operativos, aplicaciones empresariales (ERP, CRM).

### **ABAC (Attribute-Based Access Control)**

ABAC (Control de Acceso Basado en Atributos) es un modelo de control de acceso más dinámico y flexible, donde los permisos se otorgan en función de múltiples atributos. Los atributos pueden estar asociados con el sujeto (usuario), el objeto (recurso), la acción o el entorno. Esto permite tomar decisiones de autorización más sofisticadas en función de reglas basadas en diversas características.

#### Características:

- Acceso flexible y contextual: Las decisiones de autorización dependen de atributos del usuario (por ejemplo, departamento, cargo), del recurso (tipo de archivo, clasificación de seguridad), y del entorno (hora, ubicación geográfica).
- Atributos múltiples: Se pueden utilizar múltiples atributos simultáneamente para determinar el acceso.
- Escalabilidad: Se puede adaptar a situaciones más complejas y entornos más dinámicos.
- Política basada en reglas: Las políticas de acceso están escritas en forma de reglas lógicas, que especifican condiciones bajo las cuales se concede el acceso.

Casos de uso: Autorización en sistemas con necesidades complejas o donde los factores contextuales son relevantes, como sistemas de gobierno, aplicaciones en la nube, entornos con usuarios móviles.

#### **PBAC (Policy-Based Access Control)**

PBAC (Control de Acceso Basado en Políticas), también conocido como Control de Acceso Basado en Políticas de Seguridad, es un modelo que toma decisiones de autorización basándose en políticas predefinidas. Las políticas son un conjunto de reglas que especifican qué acciones están permitidas o denegadas para ciertos sujetos en determinadas condiciones.

#### Características:

- Políticas centralizadas: Las decisiones de acceso se basan en un conjunto de políticas que definen condiciones, restricciones y permisos.
- Control granular: Permite un control detallado de acceso basado en criterios variados (roles, atributos, contexto, entre otros).
- Integración con otros modelos: Puede implementar reglas similares a las de RBAC, ABAC, y otros modelos, lo que lo convierte en un enfoque híbrido y flexible.
- Adaptable: Las políticas pueden ser modificadas para adaptarse a nuevos requisitos de seguridad sin tener que cambiar directamente los permisos individuales.

Casos de uso: Grandes infraestructuras de TI donde se requiere un control centralizado y flexible sobre políticas de acceso, como aplicaciones en la nube y sistemas de seguridad de datos.

## **Conclusiones**

En conclusión, tanto la autenticación como la autorización son procesos esenciales para garantizar la seguridad en cualquier sistema de información. La autenticación asegura que los usuarios sean quienes dicen ser, mientras que la autorización delimita los permisos y accesos adecuados a los recursos, basándose en las políticas de seguridad y los roles definidos. El correcto diseño e implementación de mecanismos de autenticación y autorización no solo fortalece la protección frente a accesos no deseados, sino que también ayuda a mantener la integridad y confidencialidad de los datos. La combinación de estos dos procesos, junto con una gestión de identidades y accesos efectiva, es fundamental para prevenir vulnerabilidades y proteger a las organizaciones de amenazas cada vez más sofisticadas.