



TECNOLÓGICO  
NACIONAL DE MÉXICO



EDUCACIÓN  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TLAXIACO

**UNIDAD 1 INTRODUCCIÓN A LA SEGURIDAD - AUTORIZACIÓN Y AUTENTICACIÓN**

Yael de Jesús Santiago Ortiz

Javier Noe Cruz España

Kevin Sánchez Hernández

INGENIERIA EN SISTEMAS COMPUTACIONALES

7MO SEMESTRE 7US



# INDICE

HTML-----	4
CSS-----	6
JS-----	8
Descripción de la interfaz-----	11
LDAP-----	17
RADIUS-----	17
TACACS+ -----	17
Kerberos-----	17
ACL-----	18
RBAC-----	18
ABAC-----	18
PBAC-----	18
CONCLUSION-----	19
BIBLIOGRAFIA-----	20

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.
2. 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.
3. Implementa un mecanismo de autenticación que permita a los usuarios registrarse, iniciar sesión y cerrar sesión.
4. Implementa un mecanismo para cerrar sesión de un usuario si ha pasado un tiempo determinado sin actividad de 5 mins.

# HTML

El HTML define la estructura principal de la aplicación. Incluye lo siguiente:

## 1. Estructura de las páginas:

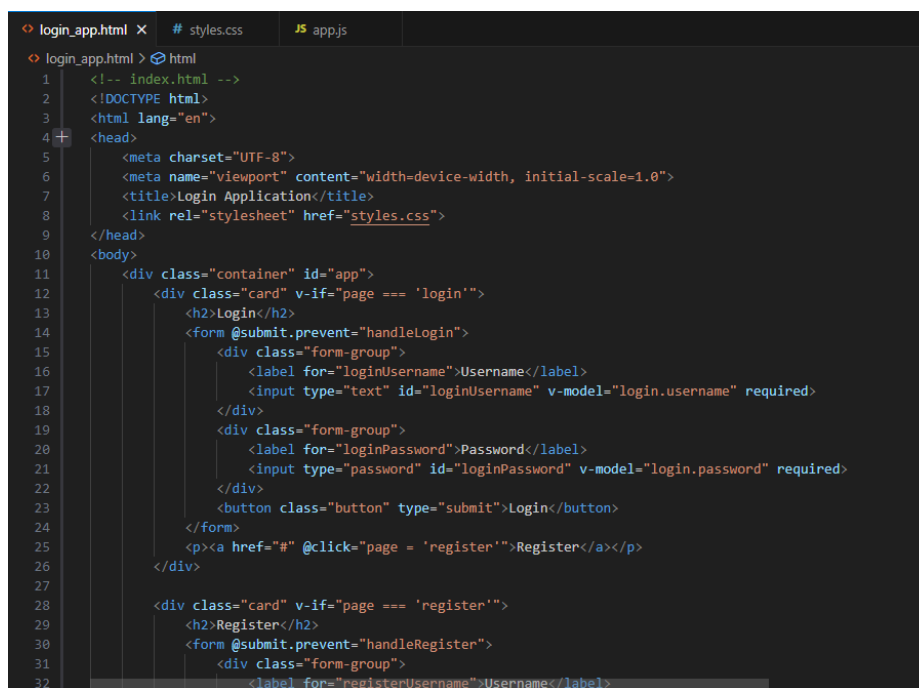
- Formulario de **inicio de sesión** con campos para usuario y contraseña.
- Formulario de **registro** con campos para usuario, contraseña y confirmación de contraseña.
- Páginas de **inicio**, **perfil** y **administración**, que cambian dinámicamente según el estado de la sesión del usuario.

## 2. Enlaces de navegación:

- Permiten cambiar entre las vistas de inicio de sesión y registro.

## 3. Vinculación con Vue.js:

- Usa directivas de Vue (v-if, v-model, @submit.prevent, etc.) para manejar interactividad y estado.



```
login_app.html x # styles.css JS app.js
1 <!-- index.html -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Login Application</title>
8   <link rel="stylesheet" href="styles.css">
9 </head>
10 <body>
11   <div class="container" id="app">
12     <div class="card" v-if="page === 'login'">
13       <h2>Login</h2>
14       <form @submit.prevent="handleLogin">
15         <div class="form-group">
16           <label for="loginUsername">Username</label>
17           <input type="text" id="loginUsername" v-model="login.username" required>
18         </div>
19         <div class="form-group">
20           <label for="loginPassword">Password</label>
21           <input type="password" id="loginPassword" v-model="login.password" required>
22         </div>
23         <button class="button" type="submit">Login</button>
24       </form>
25       <p><a href="#" @click="page = 'register'">Register</a></p>
26     </div>
27     <div class="card" v-if="page === 'register'">
28       <h2>Register</h2>
29       <form @submit.prevent="handleRegister">
30         <div class="form-group">
31           <label for="registerUsername">Username</label>
```

```

login_app.html x # styles.css JS app.js
login_app.html > html > body > div#app.container > div.card > form > div.form-group > span.message
29 + <h2>Register</h2>
30 <form @submit.prevent="handleRegister">
31   <div class="form-group">
32     <label for="registerUsername">Username</label>
33     <input type="text" id="registerUsername" v-model="register.username" required>
34   </div>
35   <div class="form-group">
36     <label for="registerPassword">Password</label>
37     <input type="password" id="registerPassword" v-model="register.password" @input="checkPa
38   </div>
39   <div class="form-group">
40     <label for="confirmPassword">Confirm Password</label>
41     <input type="password" id="confirmPassword" v-model="register.confirmPassword" required>
42     <span class="message" v-if="register.password !== register.confirmPassword">Passwords do
43   </div>
44   <div class="form-group">
45     <span class="message" v-if="passwordStrength">Password Strength: {{ passwordStrength }}>
46   </div>
47   <button class="button" type="submit">Register</button>
48 </form>
49 <p><a href="#" @click="page = 'login'">Back to Login</a></p>
50 </div>
51
52 <div class="card" v-if="page === 'home'">
53   <h2>Home Page</h2>
54   <p>Welcome to the application!</p>
55   <button class="button" @click="logout">Logout</button>
56 </div>
57
58 <div class="card" v-if="page === 'profile'">
59   <h2>Profile Page</h2>
60   <p>Hello, {{ user.username }}!</p>

```

```

57
58   <div class="card" v-if="page === 'profile'">
59     <h2>Profile Page</h2>
60     <p>Hello, {{ user.username }}!</p>
61     <button class="button" @click="logout">Logout</button>
62   </div>
63
64   <div class="card" v-if="page === 'admin'">
65     <h2>Admin Page</h2>
66     <p>Admin functionalities here.</p>
67     <button class="button" @click="logout">Logout</button>
68   </div>
69 </div>
70
71 <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
72 <script src="app.js"></script>
73 </body>
74 </html>

```

## CSS

El CSS agrega estilos para mejorar la apariencia y usabilidad:

### 1. Diseño responsivo y centrado:

- Uso de flexbox para centrar los elementos vertical y horizontalmente en la pantalla.

### 2. Estilo limpio y moderno:

- Colores suaves y sombreados para las tarjetas y botones.
- Fuentes sans-serif para un diseño minimalista y profesional.

### 3. Botones interactivos:

- Cambios de color al pasar el cursor sobre los botones (hover).

### 4. Mensajes de validación:

- Estilo específico para mensajes de error como contraseñas no coincidentes o contraseñas débiles.

```
# styles.css > ...
1  +  /* styles.css */
   CodeMate
2  body {
3      font-family: 'Arial', sans-serif;
4      margin: 0;
5      padding: 0;
6      display: flex;
7      justify-content: center;
8      align-items: center;
9      height: 100vh;
10     background: linear-gradient(to right, #4facfe, #00f2fe);
11     color: #333;
12 }
13
   CodeMate
14 .container {
15     width: 100%;
16     max-width: 400px;
17     margin: 0 auto;
18     padding: 20px;
19     box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
20     background-color: #ffffff;
21     border-radius: 8px;
22 }
23
   CodeMate
24 .card {
25     text-align: center;
26     padding: 20px;
27 }
28
   CodeMate
```

```
login_app.html # styles.css x JS app.js
# styles.css > ? .card
28 //
29 CodeMate
30 h2 {
31   margin-bottom: 20px;
32   color: #007bff;
33 }
34 CodeMate
35 .form-group {
36   margin-bottom: 15px;
37   text-align: left;
38 }
39 CodeMate
40 .form-group label {
41   display: block;
42   margin-bottom: 5px;
43   font-weight: bold;
44 }
45 CodeMate
46 .form-group input {
47   width: 100%;
48   padding: 10px;
49   border: 1px solid #ddd;
50   border-radius: 5px;
51   font-size: 16px;
52 }
53 CodeMate
54 .form-group .message {
55   color: red;
```

```
# styles.css > ? a
55 font-size: 0.85em;
56 margin-top: 5px;
57 }
58 CodeMate
59 .button {
60   width: 100%;
61   padding: 10px;
62   background: #007bff;
63   color: #fff;
64   border: none;
65   border-radius: 5px;
66   font-size: 16px;
67   cursor: pointer;
68   transition: background 0.3s ease;
69 }
70 CodeMate
71 .button:hover {
72   background: #0056b3;
73 }
74 CodeMate
75 a {
76   color: #007bff;
77   text-decoration: none;
78 }
79 CodeMate
80 a:hover {
81   text-decoration: underline;
82 }
```

## JavaScript

El JavaScript maneja toda la lógica de la aplicación utilizando **Vue.js**:

### 1. Gestión de estado:

- Define variables como page, user, y users para controlar el estado de la aplicación.
- page determina qué página mostrar.

### 2. Funcionalidades principales:

- **Registro:** Permite registrar un nuevo usuario, validando que no exista ya y que las contraseñas coincidan.
- **Inicio de sesión:** Valida el nombre de usuario y contraseña, y redirige a la página adecuada según el rol.
- **Cierre de sesión:** Borra los datos del usuario y regresa al formulario de inicio de sesión.

### 3. Persistencia con localStorage:

- Guarda la lista de usuarios para que no se pierda al recargar la página.

### 4. Manejo de seguridad y roles:

- Solo los administradores pueden acceder a la página de administración.
- Usuarios no autenticados no pueden acceder a la página de perfil o administración.

### 5. Control de inactividad:

- Desconecta automáticamente al usuario después de 5 minutos de inactividad.

### 6. Interactividad:

- Valida la fortaleza de la contraseña en tiempo real durante el registro.



```
login_app.html # styles.css JS appjs X
JS appjs > ...
1 new Vue({
2   el: '#app',
3   data: {
4     page: 'login',
5     login: {
6       username: '',
7       password: ''
8     },
9     register: {
10      username: '',
11      password: '',
12      confirmPassword: ''
13    },
14    passwordStrength: '',
15    user: null,
16    users: [], // Store registered users
17    sessionTimeout: null, // Store timeout reference
18    inactivityTime: 300000 // 5 minutes in milliseconds
19  },
20  methods: {
21    handleLogin() {
22      const foundUser = this.users.find(
23        u => u.username === this.login.username && u.password === this.login.password
24      );
25
26      if (foundUser) {
27        this.user = foundUser;
28        this.page = foundUser.role === 'admin' ? 'admin' : 'profile';
29        this.resetTimeout(); // Reset inactivity timer on login
30      } else {
31        alert('Invalid login credentials');
32      }
33    }
34  }
35}
```

```
login_app.html # styles.css JS appjs X
JS appjs > methods > handleLogin
30 } else {
31   alert('Invalid login credentials');
32 }
33 },
34 handleRegister() {
35   if (this.register.password === this.register.confirmPassword) {
36     if (this.users.some(u => u.username === this.register.username)) {
37       alert('Username already exists');
38     } else {
39       const newUser = {
40         username: this.register.username.trim(), // Remove unnecessary spaces
41         password: this.register.password.trim(), // Remove unnecessary spaces
42         role: this.register.username.trim() === 'admin' ? 'admin' : 'user'
43       };
44       this.users.push(newUser);
45       alert('Registration successful! Please login.');
```

```
login_app.html # styles.css JS appjs x
JS appjs > methods > resetTimeout
59 + clearTimeout(this.sessionTimeout); // Clear inactivity timer
60
61 },
62 authorize(route) {
63   if (!this.user) {
64     alert('You must be logged in to access this page.');
```

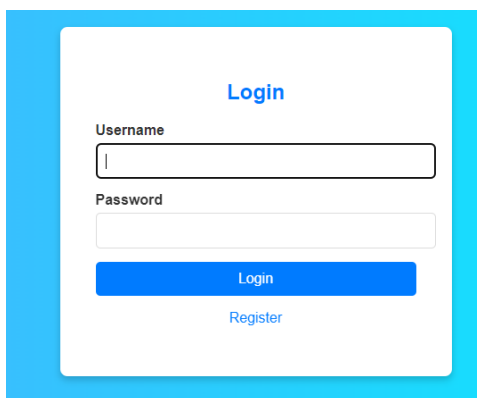
```
65     this.page = 'login';
66     return false;
67   }
68
69   if (route === 'admin' && this.user.role !== 'admin') {
70     alert('You do not have permission to access this page.');
```

```
71     this.page = 'home';
72     return false;
73   }
74
75   return true;
76 },
77 resetTimeout() {
78   clearTimeout(this.sessionTimeout);
79   this.sessionTimeout = setTimeout(() => {
80     alert('You have been logged out due to inactivity.');
```

```
81     this.logout();
82   }, this.inactivityTime);
83 }
84 watch: {
85   page(newPage) {
86     if (newPage === 'profile' || newPage === 'admin') {
87       this.authorize(newPage);
88     }
89   },
90   user(newUser) {
91     if (newUser) {
92       window.addEventListener('mousemove', this.resetTimeout);
93       window.addEventListener('keydown', this.resetTimeout);
94     } else {
95       window.removeEventListener('mousemove', this.resetTimeout);
96       window.removeEventListener('keydown', this.resetTimeout);
97     }
98   }
99 }
100 });
101
```

## Descripción de la interfaz

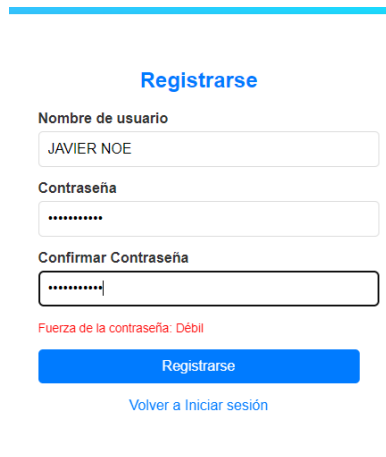
La interfaz muestra un formulario de inicio de sesión limpio y centrado, con un diseño moderno. Tiene dos campos: **Usuario** y **Contraseña**, un botón azul para iniciar sesión y un enlace para registrarse si no tienes cuenta. El fondo tiene un degradado azul, y el formulario está dentro de una tarjeta con bordes redondeados y sombra.

A login form titled "Login" in blue text. It features two input fields: "Username" and "Password". Below the password field is a blue "Login" button and a blue "Register" link. The form is centered on a light blue background with rounded corners and a subtle shadow.

El formulario de registro permite al usuario ingresar su nombre de usuario, una contraseña y confirmarla. Si las contraseñas no coinciden o son débiles, muestra mensajes de advertencia en rojo debajo de los campos correspondientes. Incluye un botón azul para registrarse y un enlace para volver al inicio de sesión. El diseño es limpio, con fondo blanco y bordes redondeados, centrado sobre un fondo degradado azul.

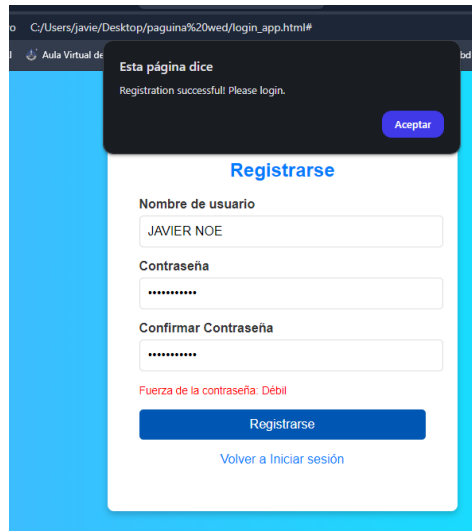
A registration form titled "Registrarse" in blue text. It includes three input fields: "Nombre de usuario" (containing "JAVIER NOE"), "Contraseña" (masked with dots), and "Confirmar Contraseña" (containing "..."). Below the confirmation field, there are two red error messages: "Las contraseñas no coinciden" and "Fuerza de la contraseña: Débil". At the bottom, there is a blue "Registrarse" button and a blue "Volver a Iniciar sesión" link. The form is centered on a light blue background with rounded corners and a subtle shadow.

El formulario de registro permite al usuario introducir un nombre de usuario, una contraseña y confirmarla. En este caso, las contraseñas coinciden, pero el sistema evalúa la fortaleza de la contraseña como "Débil", lo que se indica con un mensaje en rojo debajo del campo de confirmación.



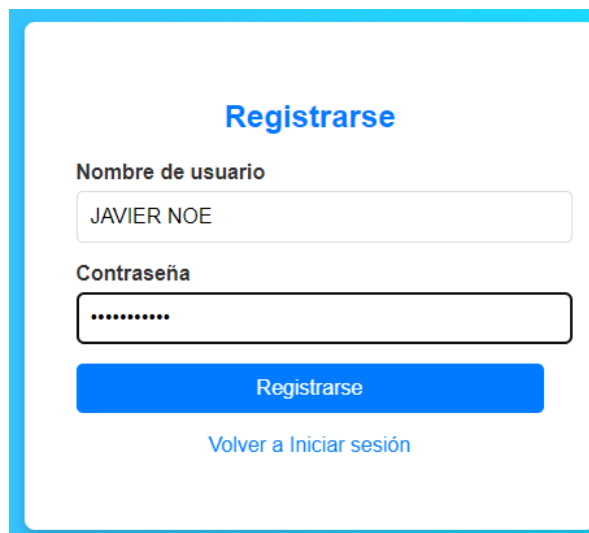
The image shows a registration form titled "Registrarse" in blue. It contains three input fields: "Nombre de usuario" with the text "JAVIER NOE", "Contraseña" with masked characters "\*\*\*\*\*", and "Confirmar Contraseña" also with "\*\*\*\*\*". Below the confirmation field, a red message reads "Fuerza de la contraseña: Débil". At the bottom, there is a blue "Registrarse" button and a blue link "Volver a Iniciar sesión".

En la siguiente imagen se presenta que la creación de la cuenta fue éxitos y así podemos iniciar sesión.



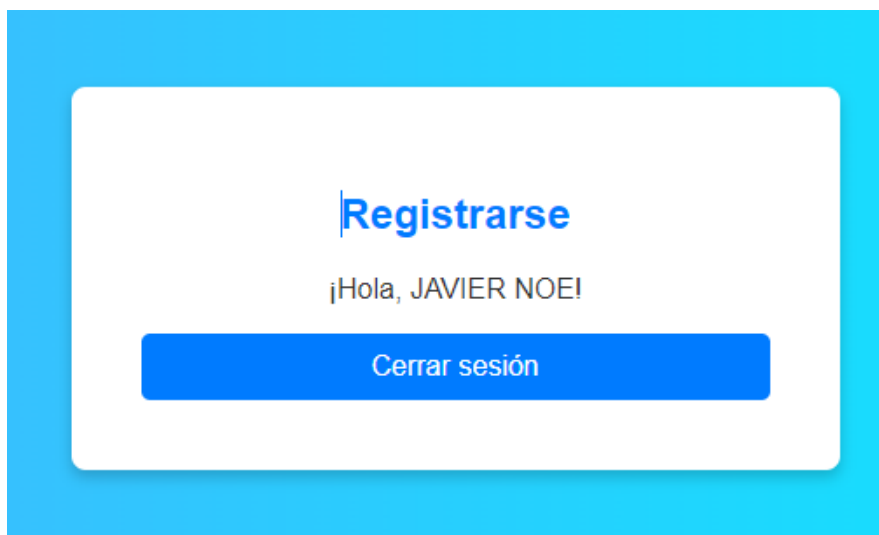
This image shows the same registration form as before, but with a dark grey overlay at the top. The overlay contains the text "Esta página dice" and "Registration successful! Please login.", along with a purple "Aceptar" button. The form fields and the "Registrarse" button remain visible below the overlay.

Aquí nos redirige al login para poder iniciar sesión ya que el registro fue un éxito.



A registration form titled "Registrarse" in blue. It contains two input fields: "Nombre de usuario" with the text "JAVIER NOE" and "Contraseña" with masked characters ".....". Below the fields is a blue "Registrarse" button and a blue link "Volver a Iniciar sesión".

Como se muestra en la imagen el inicio de sesión fue exitoso.



A login success screen with a blue background. It features the title "Registrarse" in blue, followed by the greeting "¡Hola, JAVIER NOE!". At the bottom is a blue "Cerrar sesión" button.

Aquí regresamos al inicio de sesión para verificar otros datos.



**Registrarse**

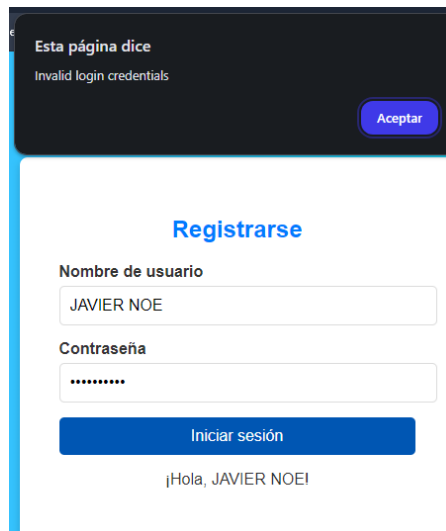
Nombre de usuario

Contraseña

Iniciar sesión

¡Hola, JAVIER NOE!

En esta parte se verifica el inicio de sesión para verificar que si ingresamos el usuario o la contraseña incorrecta no nos dejara ingresar al inicio de sesión nos mostrara un mensaje que los datos son incorrectos.



Esta página dice  
Invalid login credentials

Aceptar

**Registrarse**

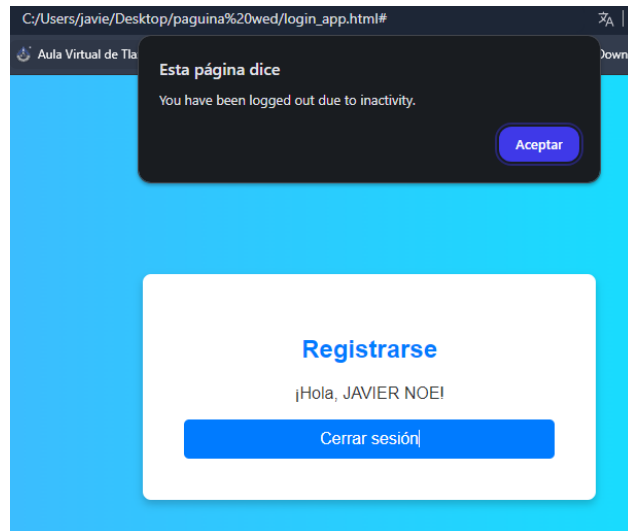
Nombre de usuario

Contraseña

Iniciar sesión

¡Hola, JAVIER NOE!

Aquí se nos muestra el siguiente mensaje que la sesión caduco por inactividad y si se actualiza la pagina se serrara en automático.



Nuevamente nos regresamos a login si queremos ingresar sin agregar el usuario y contraseña no nos dejara ingresar nos muestra el siguiente mensaje como se muestra en la imagen.

**Registrarse**

Nombre de usuario

Contraseña

 Completa este campo

**Iniciar sesión**

¡Hola, JAVIER NOE!

## Servicios de Autenticación

### 1. LDAP (Lightweight Directory Access Protocol):

- Es un protocolo estándar para acceder y gestionar directorios distribuidos en red.
- Usado comúnmente para autenticar usuarios en sistemas grandes, como aplicaciones empresariales o redes corporativas.
- Ejemplo: Microsoft Active Directory utiliza LDAP para consultar información del directorio.
- **Ventajas:** Escalabilidad, capacidad de centralizar la administración de usuarios.
- **Desventajas:** Seguridad limitada en su versión estándar (puede ser mitigada con LDAPS, una versión segura).

### 2. RADIUS (Remote Authentication Dial-In User Service):

- Protocolo para autenticación, autorización y contabilidad (AAA).
- Usado principalmente en redes para autenticar accesos remotos, como conexiones VPN o Wi-Fi.



- **Ventajas:** Soporta múltiples usuarios y dispositivos, ampliamente adoptado en redes empresariales.
- **Desventajas:** No cifra completamente los datos, solo las credenciales.

### 3. TACACS+ (Terminal Access Controller Access-Control System Plus):

- Protocolo AAA, similar a RADIUS, pero con más granularidad en la autorización.
- Diseñado para redes más críticas, como las de proveedores de servicios o grandes infraestructuras.
- **Ventajas:** Seguridad mejorada, separa funciones de autenticación, autorización y contabilidad.
- **Desventajas:** Más complejo de implementar en comparación con RADIUS.

### 4. Kerberos:

- Protocolo de autenticación basado en tickets.
- Utiliza cifrado simétrico para autenticar usuarios y servicios en una red de manera segura.
- Usado ampliamente en entornos Windows y Unix.
- **Ventajas:** Alta seguridad, evita el envío de contraseñas en texto plano.
- **Desventajas:** Puede ser complejo de configurar y mantener.

## Servicios de Autorización

### 1. ACL (Access Control List):

- Define permisos sobre recursos específicos, como archivos o redes.
- Cada recurso tiene una lista que especifica qué usuarios o grupos tienen acceso y con qué privilegios.
- **Ventajas:** Fácil de entender e implementar.
- **Desventajas:** Escalabilidad limitada en sistemas con muchos recursos.

### 2. RBAC (Role-Based Access Control):

- Controla el acceso basado en roles asignados a los usuarios.
- Cada rol tiene permisos predefinidos que determinan lo que un usuario puede hacer.
- **Ventajas:** Simplifica la gestión en entornos grandes, coherente con políticas organizacionales.
- **Desventajas:** Rigidez si los roles no están bien diseñados.

### 3. ABAC (Attribute-Based Access Control):

- Controla el acceso basado en atributos del usuario, del recurso o del entorno.
- Ejemplo: Permitir acceso solo durante horas laborales o desde ubicaciones específicas.
- **Ventajas:** Flexibilidad y capacidad de adaptación.
- **Desventajas:** Mayor complejidad en la implementación y administración.

### 4. PBAC (Policy-Based Access Control):

- Similar a ABAC, pero utiliza políticas definidas por reglas específicas.
- Estas políticas pueden ser más detalladas y adaptarse a situaciones específicas.
- **Ventajas:** Alta personalización y control.
- **Desventajas:** Complejo de implementar en grandes entornos.

## Conclusión

Este proyecto demostró cómo implementar un sistema básico de autenticación y autorización para una aplicación web. Aunque funcional, carece de robustez para entornos reales debido a la ausencia de seguridad avanzada (como cifrado y verificación en servidores). Para mejorar:

1. **Autenticación:** Se podría integrar con servicios como LDAP o Kerberos para gestionar usuarios centralizados y mejorar la seguridad.
2. **Autorización:** Migrar a un enfoque RBAC o ABAC para un control más granular de los permisos.
3. **Escalabilidad y Persistencia:** Utilizar una base de datos para almacenar usuarios y roles de forma segura.
4. **Seguridad:** Incorporar protocolos como HTTPS y almacenamiento cifrado para proteger los datos.

## Bibliografía

### 1. LDAP:

- a. Howes, T., Smith, M., & Good, G. (2003). *Understanding and Deploying LDAP Directory Services*. Addison-Wesley Professional.
- b. OpenLDAP Project. (2024). <https://www.openldap.org>.

### 2. RADIUS:

- a. Rigney, C., et al. (2000). *Remote Authentication Dial In User Service (RADIUS)*. RFC 2865.  
<https://tools.ietf.org/html/rfc2865>.
- b. Juniper Networks: RADIUS Overview.  
<https://www.juniper.net>.

### 3. TACACS+:

- a. Cisco Systems, Inc. (2024). *Understanding TACACS+*.  
<https://www.cisco.com>.

### 4. Kerberos:

- a. Neuman, C., et al. (2005). *The Kerberos Network Authentication Service (V5)*. RFC 4120.  
<https://tools.ietf.org/html/rfc4120>.
- b. Kerberos Consortium. (2024). <https://web.mit.edu/kerberos>.

### 1. ACL:

- Stallings, W. (2017). *Network Security Essentials: Applications and Standards*. Pearson.
- Linux ACL Documentation. <https://linux.die.net/man/5/acl>.

### 2. RBAC:

- Sandhu, R., et al. (1996). *Role-Based Access Control Models*. IEEE Computer.  
<https://csrc.nist.gov>.

### 3. ABAC:

- Hu, V., Ferraiolo, D., et al. (2014). *Guide to Attribute Based Access Control (ABAC)*. NIST Special Publication 800-162.  
<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf>.

### 4. PBAC:

- **Oracle. (2024). *Policy-Based Access Control in Oracle Cloud*.**  
<https://www.oracle.com>.
  - **IBM Security. *Policy-Based Access Control Overview*.**  
<https://www.ibm.com>.
5. • Bishop, M. (2018). *Computer Security: Art and Science*. Addison-Wesley.
  6. • Stallings, W. (2020). *Cryptography and Network Security: Principles and Practice*. Pearson.
  7. • Schneier, B. (2015). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley.