Actividad: Modulo 3 - Lección 1

GitHub: https://github.com/YeshuaCaceresMuniz/COMP2052.WEB-DEV-SERV-SIDE-MICROSER-BKE

app.py

```python
# Importar las bibliotecas necesarias para Flask, autenticación y seguridad
from flask import Flask, render_template, redirect, url_for, request
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
from werkzeug.security import generate_password_hash, check_password_hash

# Inicializar la aplicación Flask
app = Flask(__name__)
# Establecer una clave secreta para proteger las sesiones
app.secret_key = 'mi_clave_secreta'

# Configurar el sistema de autenticación con Flask-Login
login_manager = LoginManager()
login_manager.init_app(app)
# Especificar la vista de login para redireccionar usuarios no autenticados
login_manager.login_view = "login"

# Crear una base de datos simulada de usuarios con roles y contraseñas seguras
usuarios = {
    'Nathan': {'id': '1', 'username': 'Nathan', 'password': generate_password_hash('1234'), 'role': 'admin'},
    'Drake': {'id': '2', 'username': 'Drake', 'password': generate_password_hash('abcd'), 'role': 'user'}
}

# Implementar la clase User que hereda de UserMixin para manejar la autenticación
class User(UserMixin):
    def __init__(self, id, username, password, role):
        self.id = id  # Identificador único del usuario
        self.username = username  # Nombre de usuario para mostrar
        self.password = password  # Contraseña hasheada para seguridad
        self.role = role  # Rol del usuario (admin/user) para control de acceso

    # Implementar método requerido por Flask-Login para obtener el ID
    def get_id(self):
        return self.id

    # Método para verificar si el usuario tiene privilegios de administrador
    def is_admin(self):
        return self.role == 'admin'
```

```python

# Función auxiliar para buscar usuarios por su ID
def get_user_by_id(user_id):
    for user in usuarios.values():
        if user['id'] == user_id:
            return User(user['id'], user['username'], user['password'], user['role'])
    return None

# Función auxiliar para buscar usuarios por su nombre de usuario
def get_user_by_username(username):
    user = usuarios.get(username)
    if user:
        return User(user['id'], user['username'], user['password'], user['role'])
    return None

# Configurar el user_loader requerido por Flask-Login
@login_manager.user_loader
def load_user(user_id):
    return get_user_by_id(user_id)

# Ruta principal de la aplicación
@app.route('/')
def home():
    return render_template('index.html')

# Ruta para manejar el inicio de sesión (GET y POST)
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        # Obtener credenciales del formulario
        username = request.form['username']
        password = request.form['password']

        # Buscar usuario en la base de datos
        user = get_user_by_username(username)
```

```python

        # Verificar credenciales y autenticar al usuario
        if user and check_password_hash(user.password, password):
            login_user(user)
            return redirect(url_for('dashboard'))
        else:
            # Mostrar error si las credenciales son incorrectas
            return render_template('login.html', error='Credenciales inválidas')

    # Mostrar formulario de login para peticiones GET
    return render_template('login.html')

# Ruta protegida para el dashboard (requiere autenticación)
@app.route('/dashboard')
@login_required
def dashboard():
    # Mostrar dashboard diferente según el rol del usuario
    if current_user.is_admin():
        return render_template('admin_dashboard.html', username=current_user.username)
    else:
        return render_template('user_dashboard.html', username=current_user.username)

# Ruta para cerrar sesión (protegida)
@app.route('/logout')
@login_required
def logout():
    logout_user()
    # Redirigir al home después de cerrar sesión
    return redirect(url_for('home'))

# Punto de entrada principal para ejecutar la aplicación
if __name__ == '__main__':
    app.run(debug=True)
```

admin_dashboard.html

templates > admin_dashboard.html > ...

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">

    <!-- Título de la página para el panel de administrador -->
    <title>Admin Dashboard</title>
</head>

<body>
    <!-- Título principal del panel de administración -->
    <h1>Panel de administrador</h1>

    <!-- Mensaje de bienvenida personalizado con el nombre de usuario -->
    <!-- La variable 'username' es proporcionada por Flask/Jinja2 -->
    <p>Bienvenido, {{ username }}.</p>

    <!-- Enlace para cerrar sesión -->
    <!-- url_for() genera dinámicamente la URL para la ruta de logout -->
    <a href="{{ url_for('logout') }}">Cerrar sesión</a>
</body>
</html>
```

index.html

templates > index.html > ...

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <!-- Configuración de codificación de caracteres (UTF-8 para soporte multilenguaje) -->
    <meta charset="UTF-8">

    <!-- Título de la página que se muestra en la pestaña del navegador -->
    <title>Inicio</title>
</head>

<!-- Cuerpo del documento (contenido visible de la página) -->
<body>
    <!-- Título principal de la página de inicio -->
    <h1>Bienvenido a la aplicación</h1>

    <!-- Enlace de navegación a la página de login -->
    <!-- La función url_for() de Flask genera dinámicamente la URL para la ruta 'login' -->
    <a href="{{ url_for('login') }}">Iniciar Sesión</a>
</body>

</html>
```

login.html

templates > login.html > html

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <!-- Título de la página que aparece en la pestaña del navegador -->
    <title>Login</title>
</head>

<body>
    <!-- Título principal del formulario de login -->
    <h1>Iniciar sesión</h1>

    <!-- Bloque condicional para mostrar mensajes de error -->
    <!-- Si existe la variable 'error', se muestra en color rojo -->
    {% if error %}
        <p style="color: red;">{{ error }}</p>
    {% endif %}

    <!-- Formulario de login con método POST -->
    <form method="POST">
        <!-- Campo de entrada para el nombre de usuario -->
        Usuario: <input type="text" name="username" required><br><br>

        <!-- Campo de entrada para la contraseña (oculta) -->
        Contraseña: <input type="password" name="password" required><br><br>

        <!-- Botón para enviar el formulario -->
        <button type="submit">Entrar</button>
    </form>
</body>

</html>
```

user_dashboard.html

templates > user_dashboard.html > html

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">

    <title>User Dashboard</title>
</head>

<body>
    <!-- Encabezado principal de la página -->
    <h1>Panel de usuario</h1>

    <!-- Párrafo que muestra mensaje de bienvenida personalizado con el nombre de usuario -->
    <!-- La sintaxis {{ username }} es de Jinja2 y será reemplazada por el valor real -->
    <p>Bienvenido, {{ username }}.</p>

    <!-- Enlace para cerrar sesión -->
    <!-- url_for('logout') genera dinámicamente la URL para la ruta de logout -->
    <a href="{{ url_for('logout') }}">Cerrar sesión</a>
</body>

</html>
```

Bienvenido a la aplicación

Iniciar Sesión

Iniciar sesión

Usuario: Nathan

Contraseña: ••••

Entrar

Panel de administrador

Bienvenido, Nathan.

Cerrar sesión

Iniciar sesión

Credenciales inválidas

Usuario: Nathan

Contraseña: ••••••••••••

Entrar