**Actividad:**

**Diseño y desarrollo de servicios web - caso GA7-220501096-AA5-EV01**


**Aprendiz:**

Wilmer Jair Espinosa Silva

CC: 1.095.910.391

Instructor:

ISRAEL ARBONA GUERRERO

Servicio Nacional de aprendizaje-SENA

Curso: TECNOLOGÍA EN ANÁLISIS Y DESARROLLO DE SOFTWARE

Ficha: 2455285

Tomando como referencia lo visto en el componente formativo “Construcción de API” realizar el diseño y la codificación de un servicio web para el siguiente caso:

● Se requiere realizar un servicio web para un registro y un inicio de sesión. El servicio recibirá un usuario y una contraseña, si la autenticación es correcta saldrá un mensaje de autenticación satisfactoria en caso contrario debe devolver error en la autenticación.

RTA:

1. Crear una base de datos para almacenar los usuarios y sus contraseñas.

```
CREATE TABLE users (
    id serial PRIMARY KEY,
    username varchar(255) NOT NULL UNIQUE,
    password varchar(255) NOT NULL
);
```

Este query crea una tabla llamada "users" con 3 columnas:

1. "id": una clave primaria con valores únicos generados automáticamente.
2. "username": una columna de texto (varchar) que almacenará el nombre de usuario, con un tamaño máximo de 255 caracteres y una restricción de unicidad.
3. "password": una columna de texto (varchar) que almacenará la contraseña, con un tamaño máximo de 255 caracteres.

2. Crear un endpoint para el registro de usuarios, donde se reciba una petición POST con los datos del usuario y se almacenen en la base de datos.

   Para crear un endpoint de registro de usuarios en un servicio web que reciba una petición POST con un nombre de usuario y una contraseña, se puede utilizar un framework web como Flask o Django y escribir un controlador o una vista que maneje la petición POST. Aquí hay un ejemplo de código de un endpoint de registro de usuarios en Flask:

```python
from flask import Flask, request
import psycopg2

app = Flask(__name__)

@app.route('/register', methods=['POST'])
def register():
    # Recibir los datos de usuario y contraseña de la petición POST
    username = request.form['username']
    password = request.form['password']

    # Conectarse a la base de datos
    conn = psycopg2.connect(<connection string>)
    cursor = conn.cursor()

    # Ejecutar un query para insertar el nuevo usuario y contraseña en la tabla de usuarios
    cursor.execute("INSERT INTO users (username, password) VALUES (%s, %s)", (username, password))
    conn.commit()

    # Cerrar la conexión a la base de datos
    cursor.close()
    conn.close()

    # Devolver un mensaje de éxito
    return "Usuario registrado exitosamente!", 201
```

   Este ejemplo utiliza la biblioteca psycopg2 para conectarse a la base de datos PostgreSQL y realizar una operación de inserción para registrar un nuevo usuario y contraseña. La respuesta de éxito devuelve un código HTTP 201 (Creado) y un mensaje de éxito.

3. Crear un endpoint para el inicio de sesión, donde se reciba una petición POST con el usuario y la contraseña, se consulte en la base de datos y se devuelva un mensaje de éxito o error en la autenticación.

   Para crear un endpoint de inicio de sesión en un servicio web que reciba una petición POST con un nombre de usuario y una contraseña, se puede utilizar un framework web como Flask o Django y escribir un controlador o una vista que maneje la petición POST. Aquí hay un ejemplo de código de un endpoint de inicio de sesión en Flask:

```python
from flask import Flask, request
import psycopg2

app = Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    # Recibir los datos de usuario y contraseña de la petición POST
    username = request.form['username']
    password = request.form['password']

    # Conectarse a la base de datos
    conn = psycopg2.connect(<connection string>)
    cursor = conn.cursor()

    # Ejecutar un query para consultar el usuario y la contraseña en la tabla de usuarios
    cursor.execute("SELECT password FROM users WHERE username=%s", (username,))
    result = cursor.fetchone()

    # Cerrar la conexión a la base de datos
    cursor.close()
    conn.close()

    # Verificar si la contraseña es correcta
    if result and result[0] == password:
        # Devolver un mensaje de éxito
        return "Autenticación exitosa!", 200
    else:
        # Devolver un mensaje de error en la autenticación
        return "Error en la autenticación", 401

```

   Este ejemplo utiliza la biblioteca psycopg2 para conectarse a la base de datos PostgreSQL y realizar una consulta para verificar si un nombre de usuario y contraseña existen en la tabla de usuarios. La respuesta de éxito devuelve un código HTTP 200 (OK) y un mensaje de éxito, mientras que la respuesta de error

devuelve un código HTTP 401 (No autorizado) y un mensaje de error en la autenticación.

4. Implementar la lógica de autenticación, comparando la información recibida con la almacenada en la base de datos.

```python
from flask import Flask, request
import psycopg2

app = Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    user = request.form['user']
    password = request.form['password']

    conn = psycopg2.connect(
        host="hostname",
        database="dbname",
        user="username",
        password="password"
    )

    cur = conn.cursor()
    cur.execute("SELECT * FROM users WHERE user=%s AND password=%s", (user, password))

    if cur.rowcount == 1:
        return "Autenticación exitosa"
    else:
        return "Error en la autenticación"

if __name__ == '__main__':
    app.run()
```

En este ejemplo, primero importamos los paquetes necesarios: Flask para crear el endpoint y psycopg2 para conectarse a la base de datos de PostgreSQL. Luego creamos un endpoint de tipo POST en la ruta **/login**. En la función de este endpoint, recogemos los valores **user** y **password** de la petición POST. Después, conectamos a la base de datos y ejecutamos una consulta para verificar si el usuario y la contraseña coinciden con los almacenados en la base de datos. Finalmente, si la consulta devuelve un resultado, devolvemos "Autenticación exitosa", de lo contrario devolvemos "Error en la autenticación".

5. Devolver un mensaje JSON en el endpoint de inicio de sesión indicando si la autenticación fue satisfactoria o no.

   Para devolver un mensaje JSON en el endpoint de inicio de sesión en Flask, puedes hacer lo siguiente:

```python
from flask import Flask, request, jsonify
import psycopg2

app = Flask(__name__)

@app.route('/login', methods=['POST'])
def login():
    user = request.form['user']
    password = request.form['password']

    conn = psycopg2.connect(
        host="hostname",
        database="dbname",
        user="username",
        password="password"
    )

    cur = conn.cursor()
    cur.execute("SELECT * FROM users WHERE user=%s AND password=%s", (user, password))

    if cur.rowcount == 1:
        return jsonify({"message": "Autenticación exitosa"})
    else:
        return jsonify({"message": "Error en la autenticación"})

if __name__ == '__main__':
    app.run()

```

   En este ejemplo, utilizamos la función **jsonify** de Flask para crear un objeto JSON con una clave **message** que indica el resultado de la autenticación. De esta manera, puedes devolver un mensaje JSON con la respuesta del servidor.