```python
#Prueba Tecnica Axel Mateo Martínez Rovira

#Importamos las librerias necesarias para nuestro codigo
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy


#Generamos la aplicacion con el FrameWork Flask
app = Flask(__name__)


#Hacemos una direccion para la base de datos
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///usuarios.db'

db = SQLAlchemy(app)


#Creamos el modelo de la base de datos
class UsuariosBD(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.String(120), nullable=False)

    #Este es una funcion que nos permite de forma dinamica acceder a los valores
    def constructor(self):
        return{
            'id': self.id,
            'username': self.username,
            'password': self.password
        }

#Con este codigo, creamos las tablas, en esta caso la tabla en la BD
```

```python
with app.app_context():
    db.create_all()


#Rutas de nuestra api


# Lista de todos los usuarios
@app.route('/usuarios', methods=['GET'])
def get_usuarios():
    usuarios = UsuariosBD.query.all()
    return jsonify({'respuesta': [usuario.constructor() for usuario in usuarios]})


# Lectura de Usuario
@app.route('/usuario/<int:idUsuario>', methods=['GET'])
def get_usuario(idUsuario):
    usuario = UsuariosBD.query.get(idUsuario)
    if not usuario:
        return jsonify({'respuesta': 'Usuario no Encontrado'}), 404
    return jsonify(usuario.constructor())



# Creacion de Usuario
@app.route('/usuario', methods=['POST'])
def create_usuario():
    data = request.get_json()
    usuario = UsuariosBD(username=data['username'], password=data['password'])
    db.session.add(usuario)
    db.session.commit()
    return jsonify({'respuesta':'Se creo el usuario con exito', 'usuario':
usuario.constructor()}),201
```

```python
# Actualizar un Usuario
@app.route('/usuario/<int:idUsuario>', methods=['PUT','PATCH'])
def update_usuario(idUsuario):
    usuario = UsuariosBD.query.get_or_404(idUsuario)
    data = request.get_json()
    if 'username' in data:
        usuario.username = data['username']
    if 'password' in data:
        usuario.password = data['password']
    db.session.commit()
    return jsonify({'respuesta':'Se Actualizo el usuario con exito', 'usuario':
usuario.constructor()}),200



# Eliminar un Usuario
@app.route('/usuario/<int:idUsuario>', methods=['DELETE'])
def delete_usuario(idUsuario):
    usuario = UsuariosBD.query.get(idUsuario)
    if not usuario:
        return jsonify({'respuesta': 'Usuario no Encontrado'}), 404
    db.session.delete(usuario)
    db.session.commit()
    return jsonify({'respuesta': 'Usuario Eliminado'})


#Es una validacion, para que el archivo pueda ser probado y ejecitado
if __name__ == '__main__':
    app.run(debug=True)
```

Especificaciones de las funciones

 # Lista de todos los usuarios

@app.route('/usuarios', methods=['GET']) ← Declaramos la ruta en la que se desarollara nuestra función, y declaramos el método por el cual será activado

def get_usuarios(): ← Declaración de función

usuarios = UsuariosBD.query.all() ←Dado a que es una Base de Datos, en una variable, buscamos con un query TODOS los elementos

return jsonify({'respuesta': [usuario.constructor() for usuario in usuarios]}) ← Retornamos como un diccionario, cada uno de los valores dentro de nustros registros.

 # Lectura de Usuario

@app.route('/usuario/<int:idUsuario>', methods=['GET'])← Nuevamente declaramos la ruta, y el método a usar, pero en esta ocasión, generamos la petición de una variable

def get_usuario(idUsuario): ←Declaración de función, pidiendo una variable

usuario = UsuariosBD.query.get(idUsuario) ←Buscamos en la Base de Datos un registro que tenga el Id solicitado

if not usuario:

return jsonify({'respuesta': 'Usuario no Encontrado'}), 404←Hacemos un IF, donde si no existe el registro, apoya con un mensaje de respuesta

return jsonify(usuario.constructor()) ← Se imprime el usuario encontrado

 # Creacion de Usuario

@app.route('/usuario', methods=['POST'])

def create_usuario():

data = request.get_json() ←Generamos el registro de los datos proporcionados por medio de json

usuario = UsuariosBD(username=data['username'], password=data['password'])← Agregamos la construcción de un nuevo objeto

Axel Mateo Martínez Rovira – 12/03/2024

db.session.add(usuario) ←Agregamos a la base de datos, el nuevo objeto, convirtiéndolo en un registro

db.session.commit() ← Ejecutamos los cambios generados

return jsonify({'respuesta':'Se creo el usuario con exito', 'usuario': usuario.constructor()}),201 ←Se crea una respuesta de creación

Actualizar un Usuario

```
@app.route('/usuario/<int:idUsuario>', methods=['PUT','PATCH'])
def update_usuario(idUsuario):
    usuario = UsuariosBD.query.get_or_404(idUsuario)
    data = request.get_json()
    if 'username' in data:
        usuario.username = data['username'] ←Sobrescribe los datos nuevos
    if 'password' in data:
        usuario.password = data['password'] ←Sobrescribe los datos nuevos
    db.session.commit()
    return jsonify({'respuesta':'Se Actualizo el usuario con exito', 'usuario': usuario.constructor()}),200 ←Se crea una respuesta de satisfacción
```

Eliminar un Usuario

```
@app.route('/usuario/<int:idUsuario>', methods=['DELETE'])
def delete_usuario(idUsuario):
    usuario = UsuariosBD.query.get(idUsuario)
    if not usuario:
        return jsonify({'respuesta': 'Usuario no Encontrado'}), 404 ←Se crea una respuesta de no encontrado
    db.session.delete(usuario) ←Se elimina el usuario dado
    db.session.commit()
    return jsonify({'respuesta': 'Usuario Eliminado'}) ←Se crea una respuesta de satisfacción
```