Pablo Sánchez Martínez

Manual de usuario para API de registro de gastos

**Introducción**

La API creada es una aplicación simple para la gestión de gastos. Los usuarios pueden agregar, modificar y eliminar gastos, y ver todos los gastos existentes. La API es compatible con JSON y se puede utilizar con cualquier lenguaje de programación que tenga capacidad para realizar solicitudes HTTP. La API está programada en Python y utiliza el framework Flask para su implementación.

Funciones de la API

La API creada tiene las siguientes funciones:

* Obtener todos los gastos: los usuarios pueden obtener todos los gastos existentes mediante una solicitud GET a la ruta /gastos. La respuesta devuelta es un objeto JSON que contiene los detalles de todos los gastos existentes en la base de datos.
* Agregar un nuevo gasto: los usuarios pueden agregar un nuevo gasto mediante una solicitud POST a la ruta /gastos. La solicitud debe incluir los detalles del nuevo gasto en formato JSON. La respuesta devuelta es un objeto JSON que indica si el gasto se ha agregado correctamente o no.
* Modificar un gasto existente: los usuarios pueden modificar un gasto existente mediante una solicitud PUT a la ruta /gastos/<int:id>. La solicitud debe incluir los detalles del gasto modificado en formato JSON y el ID del gasto que se va a modificar. La respuesta devuelta es un objeto JSON que indica si el gasto se ha modificado correctamente o no.
* Eliminar un gasto existente: los usuarios pueden eliminar un gasto existente mediante una solicitud DELETE a la ruta /gastos/<int:id>. La solicitud debe incluir el ID del gasto que se va a eliminar. La respuesta devuelta es un objeto JSON que indica si el gasto se ha eliminado correctamente o no.

**Instalación y configuración**

Antes de utilizar la API, se deben seguir los siguientes pasos:

1. Instalar Python en el sistema operativo
2. Crear un entorno virtual de Python con el siguiente comando en la terminal: python -m venv env
3. Activar el entorno virtual con el siguiente comando en la terminal: env\Scripts\activate (en Windows) o source env/bin/activate (en Mac/Linux)
4. Instalar las dependencias de la API con el siguiente comando en la terminal: pip install -r requirements.txt
5. Configurar la base de datos SQLite ejecutando el script database.sql
6. Ejecutar el archivo app.py

**Estructura de la API**

La API creada tiene la siguiente estructura:

* app.py: Este archivo es la aplicación principal de Flask que contiene todas las rutas y lógica de la API.
* gastos.db: Este archivo es la base de datos SQLite que almacena todos los datos de gastos.
* index.html: Este archivo es una página HTML simple que muestra cómo utilizar la API.

**Obtener todos los gastos**

Para obtener todos los gastos registrados en la base de datos, se debe realizar una petición GET a la siguiente ruta:

/gastos

Esta petición devolverá un array JSON con todos los gastos registrados en la base de datos. Cada objeto del array representa un gasto y tiene los siguientes atributos:

**id:** identificador único del gasto en la base de datos.

**concepto:** descripción del gasto.

**monto:** cantidad gastada.

**fecha:** fecha en que se realizó el gasto, en formato YYYY-MM-DD.

**Crear un nuevo gasto**

Para crear un nuevo gasto, se debe realizar una petición POST a la siguiente ruta:

/gastos

La petición debe incluir en el cuerpo un objeto JSON con los siguientes atributos:

**concepto:** descripción del gasto.

**monto:** cantidad gastada.

**fecha:** fecha en que se realizó el gasto, en formato YYYY-MM-DD.

Ejemplo:

{

"concepto": "Compra de libros",

"monto": 150.0,

"fecha": "2023-03-11"

}

Si la operación se realiza correctamente, la API devolverá una respuesta en formato JSON con el mensaje **"El gasto ha sido creado correctamente".**

**Modificar un gasto existente**

Para modificar un gasto existente, se debe realizar una petición PUT a la siguiente ruta, indicando el identificador único del gasto a modificar:

/gastos/<id>

En esta ruta, <id> debe ser reemplazado por el identificador único del gasto que se desea modificar.

La petición debe incluir en el cuerpo un objeto JSON con los siguientes atributos:

**concepto:** descripción actualizada del gasto.

**monto:** cantidad actualizada gastada.

**fecha:** fecha actualizada en que se realizó el gasto, en formato YYYY-MM-DD.

Ejemplo:

{

"concepto": "Compra de libros y revistas",

"monto": 200.0,

"fecha": "2023-03-12"

}

Si la operación se realiza correctamente, la API devolverá una respuesta en formato JSON con el mensaje **"El gasto ha sido modificado correctamente".**

**Eliminar un gasto existente**

Para eliminar un gasto existente, se debe realizar una petición DELETE a la siguiente ruta, indicando el identificador único del gasto a eliminar:

/gastos/<id>

En esta ruta, <id> debe ser reemplazado por el identificador único del gasto que se desea eliminar.

Si la operación se realiza correctamente, la API devolverá una respuesta en formato JSON con el mensaje "El gasto ha sido eliminado correctamente".

**Visualización de gastos**

Para visualizar los gastos registrados en forma de gráfico de pastel, se debe acceder a la siguiente URL: http://localhost:5000/. La API generará el gráfico automáticamente a partir de los datos de la base de datos y lo mostrará en la página HTML.

La API tiene cuatro endpoints para manejar los gastos:

GET /gastos: Obtiene todos los gastos registrados en la base de datos.

POST /gastos: Crea un nuevo gasto en la base de datos.

PUT /gastos/<id>: Actualiza un gasto existente en la base de datos.

DELETE /gastos/<id>: Elimina un gasto existente de la base de datos.

Todos los endpoints responden en formato JSON. Los detalles de cada endpoint se describen a continuación.

**GET /gastos**

Este endpoint devuelve todos los gastos registrados en la base de datos en formato JSON.

Request:

GET /gastos

Response:

[ { "id": 1, "concepto": "Comida", "monto": 50.0, "fecha": "2022-03-10" }, { "id": 2, "concepto": "Transporte", "monto": 10.0, "fecha": "2022-03-11" }]

**POST /gastos**

Este endpoint crea un nuevo gasto en la base de datos. El request debe incluir el concepto, monto y fecha del gasto en formato JSON.

Request:

POST /gastos

Content-Type: application/json

{

"concepto": "Comida",

"monto": 50.0,

"fecha": "2022-03-10"

}

Response:

{

"message": "El gasto ha sido creado correctamente"

}

**PUT /gastos/<id>**

Este endpoint actualiza un gasto existente en la base de datos. El request debe incluir el concepto, monto y fecha del gasto en formato JSON, y el ID del gasto a actualizar se especifica en la URL.

Request:

PUT /gastos/1

Content-Type: application/json

{

"concepto": "Comida rápida",

"monto": 30.0,

"fecha": "2022-03-10"

}

Response:

{

"message": "El gasto ha sido modificado correctamente"

}

**DELETE /gastos/<id>**

Este endpoint elimina un gasto existente de la base de datos. El ID del gasto a eliminar se especifica en la URL.

Request:

DELETE /gastos/1

Response:

{

"message": "El gasto ha sido eliminado correctamente"

}

**Documento de base de datos**

import sqlite3

# Conectar a la base de datos (si no existe, se creará automáticamente)

conexion = sqlite3.connect('gastos.db')

# Crear la tabla gastos

conexion.execute('''CREATE TABLE IF NOT EXISTS gastos

(id INTEGER PRIMARY KEY AUTOINCREMENT,

concepto TEXT,

monto REAL,

fecha TEXT)''')

# Cerrar la conexión a la base de datos

conexion.close()

Conclusiones

La API de registro de gastos es una herramienta útil para llevar un control de los gastos personales y visualizarlos de forma gráfica. Me hubiera gustado añadir un gráfico circular que diferencie los gastos por colores y un botón “Actualizar” que al pulsar actualice el grafico, todo esto con pandas, pero no he sido capaz. La aplicación puede ser mejorada en el futuro para incluir nuevas funcionalidades, como la posibilidad de filtrar los gastos por fecha o categoría.