

# Consultar y Exportar CSV



Ayúdame a crear un programa en Python para consultar una tabla de mockapi.io en la URL <https://escribetuID.mockapi.io/api/v1/contactos>, que me muestre todos los registros en JSON formateado, en formato DataFrame y los exporte a un csv



## Memoria actualizada

Puedes utilizar la librería `requests` para hacer la consulta a la API, y luego convertir los datos en un DataFrame usando `pandas`. Aquí te dejo un ejemplo del código:

python

```
import requests
import pandas as pd

# URL de la API de MockAPI
url = "https://escribetuID.mockapi.io/api/v1/contactos"

# Realizar la solicitud GET a la API
response = requests.get(url)

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    # Convertir la respuesta JSON a un diccionario de Python
    data = response.json()

    # Mostrar los datos en formato JSON formateado
    print("Datos en JSON formateado:")
    print(pd.json.dumps(data, indent=4))

    # Convertir los datos en un DataFrame de pandas
    df = pd.DataFrame(data)

    # Mostrar el DataFrame
    print("\nDatos en DataFrame:")
    print(df)

    # Exportar el DataFrame a un archivo CSV
    df.to_csv("contactos.csv", index=False)
    print("\nDatos exportados a 'contactos.csv'")
else:
    print("Error al realizar la solicitud:", response.status_code)
```

## Explicación del código:

- Consulta a la API:** Se utiliza la librería `requests` para hacer una solicitud `GET` a la URL de la API.
- JSON Formateado:** La respuesta JSON se formatea y se muestra en la consola.
- Conversión a DataFrame:** Los datos obtenidos de la API se convierten en un DataFrame de `pandas`.
- Exportación a CSV:** El DataFrame se exporta a un archivo CSV llamado `contactos.csv`.

Este script te permitirá consultar los datos de la API, verlos en diferentes formatos y exportarlos fácilmente.



Ayúdame a crear un programa en Python para consultar una tabla de mockapi.io en la URL <https://mockapi.io/projects/66b4e4d59f9169621ea4c11b/api/v1/contactos>, que me muestre todos los registros en JSON formateado, en formato DataFrame y los exporte a un csv



Aquí tienes un ejemplo de cómo puedes crear un programa en Python para consultar la tabla en MockAPI, mostrar los registros en JSON formateado, en formato DataFrame, y exportarlos a un archivo CSV:

python

```
import requests
import pandas as pd
import json

# URL de la API de MockAPI
url = "https://mockapi.io/projects/66b4e4d59f9169621ea4c11b/api/v1/contactos"

# Realizar la solicitud GET a la API
response = requests.get(url)

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    # Convertir la respuesta JSON a un diccionario de Python
    data = response.json()

    # Mostrar los datos en formato JSON formateado
    print("Datos en JSON formateado:")
    print(json.dumps(data, indent=4))

    # Convertir los datos en un DataFrame de pandas
    df = pd.DataFrame(data)

    # Mostrar el DataFrame
    print("\nDatos en DataFrame:")
    print(df)

    # Exportar el DataFrame a un archivo CSV
    df.to_csv("contactos.csv", index=False)
    print("\nDatos exportados a 'contactos.csv'")
else:
    print("Error al realizar la solicitud:", response.status_code)
```

## Explicación del código:

### 1. Librerías Utilizadas:

- `requests`: Para realizar la solicitud HTTP a la API.
- `pandas`: Para manipular los datos en formato tabular.
- `json`: Para formatear la respuesta JSON de manera legible.

### 2. URL de la API:

- La URL de la API proporcionada se utiliza para obtener los datos de la tabla `contactos`.

### 3. Realización de la Solicitud:

- #### 4. Formateo de Datos:

- ## 5. Exportación a CSV:

- Finalmente, el DataFrame se exporta a un archivo CSV llamado ``contactos.csv``.

Este script te permitirá interactuar con la API de MockAPI, visualizar los datos de diferentes maneras y exportarlos para su uso posterior.



como instala la libreria json en python



No necesitas instalar la librería ``json`` en Python, ya que es una librería estándar que viene incluida con Python. Puedes usarla directamente sin necesidad de realizar ninguna instalación adicional. Aquí tienes un ejemplo sencillo de cómo usar la librería ``json``:

python

```
import json

# Convertir un diccionario de Python a un JSON
data = {"nombre": "Juan", "edad": 30, "ciudad": "Madrid"}
json_string = json.dumps(data, indent=4)
print(json_string)

# Convertir un JSON a un diccionario de Python
json_data = '{"nombre": "Juan", "edad": 30, "ciudad": "Madrid"}'
data = json.loads(json_data)
print(data)
```

Así que simplemente asegúrate de tener Python instalado, y ya puedes usar ``json`` sin problemas.



El programa me marca los siguiente errores ""C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\.venv\Scripts\python.exe" "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\getNoSQLAllRecords.py"

Traceback (most recent call last):

File "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\.venv\Lib\site-packages\requests\models.py", line 974, in json

```
return complexjson.loads(self.text, **kwargs)
```

[illegible]

File "C:\Program Files\Python312\Lib\json\\_\_init\_\_.py", line 346, in loads

```
return default_decoder.decode(s)
```

[illegible]

File "C:\Program Files\Python312\Lib\json\decoder.py", line 337, in decode

```
obj, end = self.raw_decode(s, idx=_w(s, 0).end())
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "C:\Program Files\Python312\Lib\json\decoder.py", line 355, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\getNoSQLAllRecords.py",  
line 14, in <module>

```
data = response.json()
```

△△△△△△△△△△△△△△△△

File "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\.venv\Lib\site-packages\requests\models.py", line 978, in json

```
raise RequestsJSONDecodeError(e.msg, e.doc, e.pos)
```

```
requests.exceptions.JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

Process finished with exit code 1

" corrige



El error que estás viendo (`JSONDecodeError: Expecting value: line 1 column 1 (char 0)`) ocurre porque la respuesta de la solicitud HTTP es vacía o no es un JSON válido. Esto puede deberse a varios factores, como un problema con la URL, la conexión, o la configuración de la API.

Para solucionar el problema, puedes seguir estos pasos:

1. **Verifica la URL:** Asegúrate de que la URL es correcta y está activa. Puedes probarla en tu navegador o en una herramienta como Postman.
2. **Verifica la Respuesta:** Antes de intentar convertir la respuesta a JSON, verifica el contenido de la respuesta. Esto te ayudará a entender si la respuesta es vacía o si hay algún otro problema.

Aquí te muestro cómo puedes modificar el código para manejar mejor los errores y verificar la respuesta:

python

```
import requests
import pandas as pd
import json

# URL de la API de MockAPI
url = "https://mockapi.io/projects/66b4e4d59f9169621ea4c11b/api/v1/contactos"

# Realizar la solicitud GET a la API
response = requests.get(url)

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    # Verificar si la respuesta tiene contenido
    if response.content:
        try:
            # Convertir la respuesta JSON a un diccionario de Python
            data = response.json()
```

```
# Mostrar los datos en formato JSON formateado
print("Datos en JSON formateado:")
print(json.dumps(data, indent=4))

# Convertir los datos en un DataFrame de pandas
df = pd.DataFrame(data)

# Mostrar el DataFrame
print("\nDatos en DataFrame:")
print(df)

# Exportar el DataFrame a un archivo CSV
df.to_csv("contactos.csv", index=False)
print("\nDatos exportados a 'contactos.csv'")
except json.JSONDecodeError as e:
    print(f"Error al decodificar el JSON: {e}")
else:
    print("La respuesta está vacía.")
else:
    print(f"Error al realizar la solicitud: {response.status_code}")
```

## Cambios realizados:

1. **Verificación del contenido de la respuesta:** Antes de intentar convertir la respuesta en JSON, se verifica si la respuesta no está vacía (`response.content`).
2. **Manejo de excepciones:** Se añade un bloque `try-except` alrededor de la conversión JSON para capturar y manejar posibles errores en la decodificación.
3. **Mensajes de error:** Se han añadido mensajes informativos para ayudarte a entender mejor qué está ocurriendo.

Con estos cambios, deberías poder diagnosticar mejor el problema y evitar el error que estabas encontrando. Si la URL no está devolviendo un JSON válido, también podrías investigar si necesitas ajustar la configuración de la API o si hay algún problema temporal con el servicio.



ahora me manda el siguiente error ""C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\venv\Scripts\python.exe" "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\getNoSQLAllRecords.py"  
Error al decodificar el JSON: Expecting value: line 1 column 1 (char 0)"



El error que estás viendo indica que la respuesta de la API está vacía o no contiene un JSON válido. Esto puede deberse a varias razones, como una URL incorrecta, un endpoint que no devuelve datos o un problema con el servidor.

## Soluciones posibles:

1. **Verifica la URL:**
  - Asegúrate de que la URL es correcta y está activa. Intenta acceder a la URL directamente desde un navegador web o usando una herramienta como Postman para verificar si estás obteniendo una respuesta válida.
2. **Imprime el Contenido de la Respuesta:**

- Imprime el contenido de la respuesta antes de intentar decodificarla. Esto te permitirá ver exactamente lo que estás recibiendo desde el servidor.

### 3. Comprueba la Configuración del Proyecto en MockAPI:

- Asegúrate de que el proyecto y el endpoint en MockAPI están configurados correctamente y están activos.

Aquí te muestro cómo puedes modificar el código para imprimir el contenido de la respuesta y diagnosticar el problema:

python

```
import requests
import pandas as pd
import json

# URL de la API de MockAPI
url = "https://mockapi.io/projects/66b4e4d59f9169621ea4c11b/api/v1/contactos"

# Realizar la solicitud GET a la API
response = requests.get(url)

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    # Imprimir el contenido de la respuesta
    print("Contenido de la respuesta:")
    print(response.text) # Esto imprimirá el contenido bruto de la respuesta

    # Verificar si la respuesta tiene contenido
    if response.content:
        try:
            # Convertir la respuesta JSON a un diccionario de Python
            data = response.json()

            # Mostrar los datos en formato JSON formateado
            print("Datos en JSON formateado:")
            print(json.dumps(data, indent=4))

            # Convertir los datos en un DataFrame de pandas
            df = pd.DataFrame(data)

            # Mostrar el DataFrame
            print("\nDatos en DataFrame:")
            print(df)

            # Exportar el DataFrame a un archivo CSV
            df.to_csv("contactos.csv", index=False)
            print("\nDatos exportados a 'contactos.csv'")
        except json.JSONDecodeError as e:
            print(f"Error al decodificar el JSON: {e}")
    else:
        print("La respuesta está vacía.")
else:
    print(f"Error al realizar la solicitud: {response.status_code}")
```

## Explicación:

1. **Imprimir `response.text`**: Esto imprimirá el contenido bruto de la respuesta HTTP tal como se recibió del servidor, antes de intentar convertirlo en JSON. Esto te ayudará a entender si estás recibiendo un error HTML, una respuesta vacía, o algo diferente.
2. **Revisar la Configuración de la API**: Si la respuesta está vacía o es inesperada, verifica la configuración en MockAPI para asegurarte de que estás apuntando al endpoint correcto y que hay datos disponibles.

Si después de hacer esto la respuesta sigue vacía o incorrecta, revisa el endpoint y la configuración del proyecto en MockAPI para asegurarte de que todo está correctamente configurado.



```
ME manda este mensaje ""C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos
NoSQL\venv\Scripts\python.exe" "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos
NoSQL\getNoSQLAllRecords.py"
Contenido de la respuesta:
<!doctype html><html lang="en"><head><meta charset="utf-8"><meta name="viewport"
content="width=device-width,initial-scale=1"><meta name="description" content="MockAPI is a
simple tool that lets you easily mock up APIs, generate custom data, and perform operations on it
using RESTful interface. MockAPI is meant to be used as a prototyping/testing/learning tool.">
<link rel="shortcut icon" href="/favicon.ico"><link rel="shortcut icon" href="/favicon.ico"><link
href="/styles.css" rel="stylesheet"><script src="https://js.stripe.com/v3/"></script>
<title>mockAPI</title><script defer="defer" src="/static/js/main.80f1249e.js"></script><link
href="/static/css/main.9a999eaa.css" rel="stylesheet"></head><link rel="preconnect"
href="https://fonts.googleapis.com"><link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin><link rel="preconnect" href="https://fonts.googleapis.com"><link rel="preconnect"
href="https://fonts.gstatic.com" crossorigin><link href="https://fonts.googleapis.com/css2?
family=JetBrains+Mono:wght@400;500;600;700&display=swap" rel="stylesheet"><link
href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet"><body>
<div id="root"></div><div id="overlay"></div><script async
src="https://www.googletagmanager.com/gtag/js?id=G-2W3SEYRGZ7"></script><script>function
gtag(){dataLayer.push(arguments)}window.dataLayer=window.dataLayer||[],gtag("js",new
Date),gtag("config","G-2W3SEYRGZ7")</script></body></html>
Error al decodificar el JSON: Expecting value: line 1 column 1 (char 0)"
```



El mensaje que recibes es un HTML, no un JSON, lo que indica que la URL que estás utilizando está apuntando a una página web (posiblemente la página principal del sitio) en lugar del endpoint de la API.

## Solución:

### 1. Revisa la URL del Endpoint:

- Asegúrate de que la URL que estás utilizando es la correcta para acceder a la API. El formato correcto para un endpoint de MockAPI debería parecerse a algo como:

```
bash

https://your-mockapi-url.mockapi.io/api/v1/contactos
```

- No debes usar la URL del dashboard del proyecto de MockAPI. La URL de la API debe estar en un formato que finalice con `/api/v1/contactos` u otro endpoint específico.

### 2. Ejemplo de URL Correcta:

- Si tu proyecto está configurado correctamente, la URL del endpoint debería tener un formato similar a:

```
bash
```

```
https://<tu-ID>.mockapi.io/api/v1/contactos
```

- Sustituye `**<tu-ID>**` con el ID correcto de tu proyecto.

### 3. Uso de Postman o Navegador:

- Intenta acceder a la URL directamente desde un navegador o usa una herramienta como Postman para verificar que el endpoint esté funcionando y devolviendo JSON.
- Si todo está en orden, deberías ver una respuesta en formato JSON.

## Código Actualizado (con URL corregida):

```
python
```

```
import requests
import pandas as pd
import json

# Asegúrate de que la URL de la API sea la correcta
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto

# Realizar la solicitud GET a la API
response = requests.get(url)

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    # Imprimir el contenido de la respuesta
    print("Contenido de la respuesta:")
    print(response.text)

    # Verificar si la respuesta tiene contenido
    if response.content:
        try:
            # Convertir la respuesta JSON a un diccionario de Python
            data = response.json()

            # Mostrar los datos en formato JSON formateado
            print("Datos en JSON formateado:")
            print(json.dumps(data, indent=4))

            # Convertir los datos en un DataFrame de pandas
            df = pd.DataFrame(data)

            # Mostrar el DataFrame
            print("\nDatos en DataFrame:")
            print(df)

            # Exportar el DataFrame a un archivo CSV
            df.to_csv("contactos.csv", index=False)
            print("\nDatos exportados a 'contactos.csv'")
        except json.JSONDecodeError as e:
            print(f"Error al decodificar el JSON: {e}")
    else:
        print("La respuesta está vacía.")
else:
    print(f"Error al realizar la solicitud: {response.status_code}")
```

## Paso Adicional:

Si continúas teniendo problemas para encontrar la URL correcta, revisa la configuración de tu proyecto en el dashboard de MockAPI. MockAPI proporciona la URL del endpoint en la sección donde



configuras tu API. Asegúrate de usar la URL proporcionada por MockAPI para tus consultas. Si necesitas más ayuda para encontrar la URL correcta, puedes compartir más detalles sobre cómo estás configurando MockAPI, y te puedo ayudar a identificar el problema.



```
No se ejecuta y me manda este mensaje ""C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos
NoSQL\venv\Scripts\python.exe" "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos
NoSQL\getNoSQLAllRecords.py"
Error al realizar la solicitud: 404

Process finished with exit code 0"
```



El error `404` significa que la URL que estás utilizando no se encuentra, es decir, la dirección que estás intentando alcanzar no existe en el servidor. Esto generalmente ocurre cuando la URL es incorrecta o el endpoint ha sido mal escrito.

## Pasos para Resolver el Problema:

### 1. Verifica la URL del Endpoint:

- Asegúrate de que estás utilizando la URL correcta para acceder a los datos en MockAPI. La URL debe apuntar específicamente al endpoint que contiene los datos que deseas consultar.
- La URL debería verse algo como:

```
bash

https://<tu-ID>.mockapi.io/api/v1/contactos
```

- Sustituye `` con el ID de tu proyecto.

### 2. Confirma la Estructura de la API en MockAPI:

- Ve al dashboard de MockAPI y verifica que el endpoint `/contactos` exista y tenga datos. Si el endpoint no existe o está mal escrito, obtendrás un error 404.

### 3. Prueba la URL en el Navegador o Postman:

- Introduce la URL en un navegador web o en una herramienta como Postman para ver si obtienes una respuesta válida. Si la URL es correcta, deberías ver los datos en formato JSON.

### 4. Asegúrate de que la URL no tenga Errores Tipográficos:

- Verifica que no haya errores tipográficos en la URL que estás utilizando en tu código. Incluso un pequeño error en el nombre del endpoint o en el ID del proyecto puede causar un error 404.

## Código de Ejemplo (Con URL Correcta):

python

```
import requests
import pandas as pd
import json

# URL de la API de MockAPI (verifica que esta sea la correcta)
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

# Realizar la solicitud GET a la API
response = requests.get(url)

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    # Imprimir el contenido de la respuesta
    print("Contenido de la respuesta:")
    print(response.text)

    # Verificar si la respuesta tiene contenido
    if response.content:
        try:
            # Convertir la respuesta JSON a un diccionario de Python
            data = response.json()

            # Mostrar los datos en formato JSON formateado
            print("Datos en JSON formateado:")
            print(json.dumps(data, indent=4))

            # Convertir los datos en un DataFrame de pandas
            df = pd.DataFrame(data)

            # Mostrar el DataFrame
            print("\nDatos en DataFrame:")
            print(df)

            # Exportar el DataFrame a un archivo CSV
            df.to_csv("contactos.csv", index=False)
            print("\nDatos exportados a 'contactos.csv'")
        except json.JSONDecodeError as e:
            print(f"Error al decodificar el JSON: {e}")
    else:
        print("La respuesta está vacía.")
else:
    print(f"Error al realizar la solicitud: {response.status_code}")
```

## Verificación del ID del Proyecto:

Si no estás seguro de cuál es el ID de tu proyecto, puedes obtenerlo desde el dashboard de MockAPI. Es un identificador único que suele estar presente en la URL del proyecto cuando lo estás configurando.

Si después de verificar todo sigues obteniendo un error 404, podría ser útil revisar la configuración completa del proyecto en MockAPI para asegurarte de que todo está configurado correctamente.



me sigue marcando el error 404 y ya revise todo y esta bien



Si ya revisaste la URL y la configuración del endpoint en MockAPI y aún recibes un error 404, es posible que el problema radique en cómo se está utilizando la URL o en algún detalle específico de la configuración del proyecto.

## Cosas a Verificar:

### 1. Formato de la URL:

- Asegúrate de que la URL esté en el formato correcto. Por ejemplo:

```
bash

https://<tu-ID>.mockapi.io/api/v1/contactos
```

- Reemplaza `**<tu-ID>**` con el ID del proyecto proporcionado por MockAPI. Asegúrate de que la palabra `**contactos**` coincide exactamente con el endpoint que configuraste en MockAPI.

### 2. Endpoint Correcto:

- Revisa en MockAPI que el endpoint que estás intentando alcanzar sea realmente `**/contactos**` y no otro. Asegúrate de que el nombre esté en minúsculas y sin errores tipográficos.

### 3. Accesibilidad del Proyecto:

- Asegúrate de que el proyecto de MockAPI está configurado como "Público" para que sea accesible sin autenticación. Si está configurado como "Privado", necesitarás autenticarte para acceder a los datos.

### 4. Prueba la URL en el Navegador:

- Copia la URL que estás usando en el código y pégala directamente en un navegador web. Si el navegador muestra un error 404, el problema está en la URL o en la configuración del proyecto.

## Ejemplo de Verificación de la URL:

Supongamos que tu ID de proyecto es `**12345**`. La URL debería verse algo así:

```
python

url = "https://12345.mockapi.io/api/v1/contactos"
```

Verifica si esa URL funciona en un navegador. Si sigue mostrando un error 404, entonces es probable que el endpoint `**/contactos**` no esté correctamente configurado o no exista en tu proyecto.

## Herramienta para Diagnosticar el Problema:

Puedes utilizar la herramienta **Postman** para hacer una solicitud GET a la URL de la API y ver exactamente cuál es la respuesta del servidor. Si en Postman también obtienes un 404, es probable que el problema esté en la configuración de la API.

## Ejemplo de Prueba Simple con Python:

Puedes intentar con un código aún más simple para descartar otros factores:

python

```
import requests

url = "https://12345.mockapi.io/api/v1/contactos" # Reemplaza con la URL correcta
response = requests.get(url)

print(f"Estado de la respuesta: {response.status_code}")
print("Contenido de la respuesta:")
print(response.text)
```

## Próximos Pasos:

1. **Confirma la URL de la API en el navegador o Postman.**
2. **Asegúrate de que el endpoint `/contactos` está bien escrito y existe.**
3. **Verifica que el proyecto de MockAPI sea público o que la URL incluya cualquier token de autenticación si es necesario.**

Si nada de esto resuelve el problema, podrías estar lidiando con un error en la plataforma MockAPI o una configuración inusual en tu cuenta que necesitaría revisión detallada.



me manda el siguiente error ""C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\venv\Scripts\python.exe" "C:\Users\leone\PyCharm\Codigos\M3.1 Bases de datos NoSQL\getNoSQLAllRecords.py"  
Estado de la respuesta: 404  
Contenido de la respuesta:  
"Not found"

Process finished with exit code 0"



El error `404` con el mensaje "Not found" indica que la URL a la que estás intentando acceder no existe en el servidor. Dado que ya has revisado la configuración y el formato de la URL, vamos a verificar algunos detalles adicionales:

## Posibles Causas y Soluciones:

1. **Endpoint Incorrecto:**
  - Asegúrate de que el endpoint `/contactos` realmente exista en tu proyecto de MockAPI. Puede que hayas configurado otro nombre de endpoint o la ruta sea diferente. Verifica que el endpoint sea exactamente `/contactos` y que esté bien escrito.
2. **ID de Proyecto Incorrecto:**
  - El ID del proyecto debe ser exacto. Revisa que el ID que estás utilizando en la URL sea el correcto. Puedes encontrar este ID en el dashboard de MockAPI.
3. **Prueba un Endpoint Alternativo:**
  - Si no estás seguro del nombre del endpoint, intenta acceder al endpoint raíz para ver qué recursos están disponibles. Por ejemplo:

```
ruby
```

```
https://<tu-ID>.mockapi.io/api/v1/
```

- Esto debería devolver una lista de los endpoints disponibles. Si esto funciona, podrás ver el endpoint correcto en la respuesta.

#### 4. Prueba con otro proyecto o con un nuevo endpoint:

- Crea un nuevo proyecto en MockAPI con un endpoint simple y prueba acceder a ese endpoint. Esto ayudará a descartar problemas específicos con tu proyecto actual.

## Ejemplo de Código para Probar el Endpoint Raíz:

```
python
```

```
import requests

# Prueba acceder al endpoint raíz para ver qué recursos están disponibles
url = "https://<tu-ID>.mockapi.io/api/v1/" # Reemplaza <tu-ID> con tu ID de proyecto

response = requests.get(url)

print(f"Estado de la respuesta: {response.status_code}")
print("Contenido de la respuesta:")
print(response.text)
```

## Verificación Adicional:

### 1. Revisa el Dashboard de MockAPI:

- Ve al dashboard de MockAPI y confirma que el endpoint `/contactos` realmente existe y que el ID del proyecto es correcto.

### 2. Revisa la URL en el Navegador:

- Pega la URL directamente en un navegador y ve si puedes acceder al endpoint. Si recibes un error 404, el problema está en la URL o en la configuración del proyecto.

### 3. Crea un Nuevo Proyecto o Endpoint:

- Intenta crear un nuevo proyecto en MockAPI y configura un endpoint simple, como `/users`. Luego, prueba acceder a ese endpoint desde tu código.

Si después de hacer todo esto sigues recibiendo un error 404, el problema podría estar relacionado con la configuración del proyecto en MockAPI o podría haber algún error en cómo se está accediendo a la API. Considera contactar con el soporte de MockAPI si el problema persiste.



ya quedo, me faltaba la API



 Memoria actualizada

¡Qué bien que lo resolviste! Ahora que tienes la API correcta, tu programa debería funcionar sin problemas. Si necesitas ayuda con cualquier otra cosa, no dudes en preguntar. ¡Estoy aquí para

ayudarte!



Ahora, ayúdame con el programa para mostrar solamente un determinado registro de la misma URL de MockAPI, muestra todos los campos en formato plano y ya no lo muestres en Dataframe ni lo exportes a CSV



Claro, te puedo ayudar a modificar el programa para que solo muestre un registro específico de la URL de MockAPI en formato plano. Supongamos que quieres mostrar un registro con un ID específico. Aquí tienes un ejemplo de cómo hacerlo:

## Código Modificado para Mostrar un Registro Específico

python

```
import requests
import json

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

# ID del registro que deseas consultar
registro_id = "1" # Reemplaza con el ID del registro que deseas consultar

# Realizar la solicitud GET al endpoint específico del registro
response = requests.get(f"{url}/{registro_id}")

# Verificar si la solicitud fue exitosa
if response.status_code == 200:
    try:
        # Convertir la respuesta JSON a un diccionario de Python
        data = response.json()

        # Mostrar los datos en formato plano
        print("Registro en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")
    except json.JSONDecodeError as e:
        print(f"Error al decodificar el JSON: {e}")
else:
    print(f"Error al realizar la solicitud: {response.status_code}")
```

## Explicación del Código:

### 1. URL del Endpoint:

- Asegúrate de reemplazar `<tu-ID>` con el ID de tu proyecto de MockAPI.

### 2. ID del Registro:

- Cambia el valor de `registro\_id` con el ID del registro que deseas consultar.

### 3. Solicitud al Endpoint Específico:

- La URL se construye para acceder al registro específico usando el ID proporcionado.

### 4. Mostrar el Registro en Formato Plano:

- Una vez que se obtiene la respuesta JSON, se itera sobre los campos del diccionario `data` y se imprime cada campo y su valor en formato plano.

## 5. Manejo de Errores:

- Se maneja la excepción `JSONDecodeError` si la respuesta no es un JSON válido y se muestra un mensaje de error adecuado en caso de problemas con la solicitud.

Este código solo muestra el registro específico y no utiliza `DataFrame` ni exporta datos a CSV. Si tienes más detalles o necesitas ajustar algo, avísame.



mejora el código para que el usuario pueda mostrar el registro que quiera, dando la opción de que ingrese el número de registro, antes debes contar los registros para que se le muestre el rango de id que tienen los registros



Aquí tienes una versión mejorada del código que cuenta los registros disponibles, muestra el rango de IDs y permite al usuario ingresar el ID del registro que desea consultar:

## Código Mejorado:

python

```
import requests
import json

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

def obtener_registros():
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")
        return []

def mostrar_registro(data):
    if data:
        print("Registro en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")

def main():
    # Obtener todos los registros para contar y mostrar el rango de IDs
    registros = obtener_registros()
    if not registros:
        return

    # Mostrar el rango de IDs
    ids = [registro['id'] for registro in registros]
    print(f"Rango de IDs disponibles: {min(ids)} a {max(ids)}")

    # Solicitar al usuario que ingrese el ID del registro que desea mostrar
    registro_id = input("Ingrese el ID del registro que desea mostrar: ")

    # Validar que el ID ingresado esté dentro del rango
    if registro_id not in ids:
        print("ID no válido. Por favor, ingrese un ID dentro del rango mostrado.")
        return
```

```
# Obtener y mostrar el registro específico
response = requests.get(f"{url}/{registro_id}")
if response.status_code == 200:
    try:
        data = response.json()
        mostrar_registro(data)
    except json.JSONDecodeError as e:
        print(f"Error al decodificar el JSON: {e}")
else:
    print(f"Error al realizar la solicitud: {response.status_code}")

if __name__ == "__main__":
    main()
```

## Explicación del Código:

### 1. Función `obtener\_registros()`:

- Realiza una solicitud GET al endpoint de la API para obtener todos los registros.
- Devuelve la lista de registros en formato JSON.

### 2. Función `mostrar\_registro(data)`:

- Muestra el contenido del registro en formato plano.

### 3. Función `main()`:

- Llama a `obtener\_registros()` para obtener todos los registros y contar los disponibles.
- Muestra el rango de IDs disponibles en la API.
- Solicita al usuario que ingrese un ID de registro.
- Verifica si el ID ingresado está dentro del rango de IDs disponibles.
- Realiza una solicitud GET al endpoint específico del registro solicitado y muestra el registro si la solicitud es exitosa.

### 4. Manejo de Errores:

- Si la solicitud para obtener todos los registros falla o no hay registros, se muestra un mensaje de error.
- Se maneja el caso en que el ID ingresado no está en el rango de IDs disponibles.

Este código proporcionará una experiencia de usuario más interactiva y útil, permitiendo a los usuarios seleccionar un registro específico basado en el rango de IDs disponibles. Si necesitas más personalizaciones, avísame.



no me esta dando el rango correcto de los id



Entiendo. El problema puede estar en cómo se están recuperando y procesando los IDs. Vamos a revisar el código para asegurarnos de que estamos obteniendo y mostrando el rango correcto de IDs.

## Código Mejorado para Contar y Mostrar Rango de IDs Correcto

1. Asegúrate de que la API devuelva todos los registros correctamente.
2. Verifica que estés extrayendo correctamente los IDs de los registros.

Aquí tienes un código revisado que realiza estos pasos:



python

```
import requests
import json

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

def obtener_registros():
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")
        return []

def mostrar_registro(data):
    if data:
        print("Registro en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")

def main():
    # Obtener todos los registros para contar y mostrar el rango de IDs
    registros = obtener_registros()
    if not registros:
        return

    # Extraer y mostrar el rango de IDs
    ids = [int(registro['id']) for registro in registros if 'id' in registro]
    if ids:
        print(f"Rango de IDs disponibles: {min(ids)} a {max(ids)}")
    else:
        print("No se encontraron IDs en los registros.")
        return

    # Solicitar al usuario que ingrese el ID del registro que desea mostrar
    try:
        registro_id = int(input("Ingrese el ID del registro que desea mostrar: "))
    except ValueError:
        print("ID no válido. Debe ser un número entero.")
        return

    # Validar que el ID ingresado esté dentro del rango
    if registro_id not in ids:
        print("ID no válido. Por favor, ingrese un ID dentro del rango mostrado.")
        return

    # Obtener y mostrar el registro específico
    response = requests.get(f"{url}/{registro_id}")
    if response.status_code == 200:
        try:
            data = response.json()
            mostrar_registro(data)
        except json.JSONDecodeError as e:
            print(f"Error al decodificar el JSON: {e}")
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")

if __name__ == "__main__":
    main()
```

## Cambios Realizados:

### 1. Conversión a Entero:

- Convierte los IDs a enteros para manejar correctamente los valores numéricos y comparar adecuadamente.

## 2. Chequeo de Presencia del ID:

- Asegura que el campo ``id`` esté presente en los registros antes de intentar extraerlo.

## 3. Manejo de Errores de Entrada:

- Agrega manejo de errores para asegurar que el ID ingresado sea un número entero.

## 4. Validación de ID:

- Valida que el ID ingresado esté en la lista de IDs disponibles antes de hacer la solicitud para ese registro específico.

## Verifica:

- **Datos de la API:** Asegúrate de que la API devuelve los datos en el formato esperado y que los IDs son numéricos.
- **Navegador:** Puedes usar un navegador o herramientas como Postman para verificar que los datos de la API sean correctos y que los IDs estén presentes.

Si sigues teniendo problemas, asegúrate de que los datos devueltos por la API estén correctamente formateados y que no haya problemas con la forma en que los registros se están estructurando o serializando.



Ahora dame el programa para agregar un registro en la misma URL de MockAPI, pero que el id se automático, es decir el siguiente del último y se muestre



Claro, para agregar un nuevo registro en MockAPI y asegurarte de que el ID se asigne automáticamente como el siguiente del último, puedes seguir estos pasos:

## 1. Obtener el Último ID:

- Realiza una solicitud para obtener todos los registros y determina el ID máximo actual.

## 2. Agregar un Nuevo Registro:

- Realiza una solicitud POST a la API para agregar un nuevo registro con el ID incrementado automáticamente.

## 3. Mostrar el Nuevo Registro:

- Verifica que el nuevo registro se haya añadido correctamente y muestra sus detalles.

## Código para Agregar un Nuevo Registro

Aquí tienes un código que realiza estas tareas:

python

```
import requests
import json

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

def obtener_registros():
    response = requests.get(url)
    if response.status_code == 200:
```

```

        return response.json()
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")
        return []

def agregar_registro(nuevo_registro):
    response = requests.post(url, json=nuevo_registro)
    if response.status_code == 201:
        return response.json()
    else:
        print(f"Error al agregar el registro: {response.status_code}")
        return None

def mostrar_registro(data):
    if data:
        print("Nuevo registro agregado en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")

def main():
    # Obtener todos los registros para encontrar el ID máximo actual
    registros = obtener_registros()
    if not registros:
        max_id = 0
    else:
        ids = [int(registro['id']) for registro in registros if 'id' in registro]
        max_id = max(ids)

    # Crear un nuevo registro con el ID siguiente
    nuevo_id = max_id + 1
    nuevo_registro = {
        "id": nuevo_id,
        "nombre": input("Ingrese el nombre: "),
        "email": input("Ingrese el email: "),
        # Agrega aquí los campos adicionales que esperas en el registro
    }

    # Agregar el nuevo registro
    registro_agregado = agregar_registro(nuevo_registro)
    if registro_agregado:
        # Mostrar el nuevo registro agregado
        mostrar_registro(registro_agregado)

if __name__ == "__main__":
    main()

```

## Explicación del Código:

### 1. Función `obtener\_registros()`:

- Realiza una solicitud GET para obtener todos los registros y devolver la lista en formato JSON.

### 2. Función `agregar\_registro(nuevo\_registro)`:

- Realiza una solicitud POST para agregar un nuevo registro con los datos proporcionados. Retorna el registro agregado si la solicitud es exitosa.

### 3. Función `mostrar\_registro(data)`:

- Muestra el contenido del nuevo registro en formato plano.

### 4. Función `main()`:

- Obtiene todos los registros y encuentra el ID máximo actual.
- Incrementa el ID máximo para crear un nuevo registro.
- Solicita al usuario que ingrese los detalles del nuevo registro.
- Agrega el nuevo registro a la API y muestra los detalles del registro agregado.

## Notas:

- **Campos del Registro:** Asegúrate de incluir todos los campos necesarios en el nuevo registro (`"nombre"`, `"email"`, etc.). Ajusta el diccionario `nuevo\_registro` según la estructura de tus datos.
- **Manejo de Errores:** El código maneja errores básicos en la solicitud GET y POST, pero puedes añadir más validaciones según tus necesidades.

Este código debería ayudarte a agregar un nuevo registro a MockAPI y mostrar los detalles del nuevo registro agregado. Si necesitas más ajustes o tienes alguna otra pregunta, avísame.



mejora el código para que se muestre el ID del registro que se va a agregar



Para mejorar el código y asegurarte de que se muestre el ID del registro que se va a agregar antes de realizar la solicitud POST, puedes seguir estos pasos:

1. **Calcula el nuevo ID antes de agregar el registro.**
2. **Muestra el ID y los detalles del nuevo registro antes de enviarlo.**

Aquí está el código mejorado:

python

```
import requests
import json

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

def obtener_registros():
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")
        return []

def agregar_registro(nuevo_registro):
    response = requests.post(url, json=nuevo_registro)
    if response.status_code == 201:
        return response.json()
    else:
        print(f"Error al agregar el registro: {response.status_code}")
        return None

def mostrar_registro(data):
    if data:
        print("Registro agregado en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")

def main():
    # Obtener todos los registros para encontrar el ID máximo actual
    registros = obtener_registros()
    if not registros:
        max_id = 0
    else:
        ids = [int(registro['id']) for registro in registros if 'id' in registro]
        max_id = max(ids)
```

```
# Crear un nuevo registro con el ID siguiente
nuevo_id = max_id + 1
nuevo_registro = {
    "id": nuevo_id,
    "nombre": input("Ingrese el nombre: "),
    "email": input("Ingrese el email: "),
    # Agrega aquí los campos adicionales que esperas en el registro
}

# Mostrar el ID y los detalles del nuevo registro
print(f"\nSe va a agregar el siguiente registro con ID: {nuevo_id}")
mostrar_registro(nuevo_registro)

# Confirmar antes de agregar el registro
confirmar = input("\n¿Desea agregar este registro? (sí/no): ").strip().lower()
if confirmar in ['sí', 'si', 'yes', 'y']:
    # Agregar el nuevo registro
    registro_agregado = agregar_registro(nuevo_registro)
    if registro_agregado:
        # Mostrar el nuevo registro agregado
        mostrar_registro(registro_agregado)
else:
    print("Registro no agregado.")

if __name__ == "__main__":
    main()
```

## Explicación del Código Mejorado:

### 1. Mostrar ID del Registro:

- Antes de realizar la solicitud POST, el código muestra el ID y los detalles del nuevo registro.

### 2. Confirmación del Usuario:

- Se solicita al usuario que confirme si desea agregar el registro con el ID mostrado. Esto evita la adición accidental de registros.

### 3. Mostrar Registro:

- La función `mostrar\_registro()` se utiliza para mostrar tanto el registro a agregar como el registro agregado una vez que se realiza la solicitud POST.

### 4. Manejo de Errores y Confirmaciones:

- Se manejan los errores de solicitud y se confirma con el usuario antes de realizar la adición del registro.

Este código proporciona una experiencia más interactiva y permite al usuario verificar el ID y los detalles del registro antes de confirmarlo. Si necesitas más personalización o tienes más preguntas, no dudes en preguntar.



mejora el código para que se muestre primero el id del nuevo registro que se va a capturar



### Memoria actualizada

Claro, para mejorar el código y asegurarte de que el ID del nuevo registro se muestre primero, antes de que el usuario ingrese los datos, puedes seguir este enfoque:

### 1. Calcula y muestra el ID del nuevo registro.

## 2. Solicita al usuario que ingrese los detalles del registro después de mostrar el ID.

Aquí está el código actualizado:

python

```
import requests
import json

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

def obtener_registros():
    response = requests.get(url)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")
        return []

def agregar_registro(nuevo_registro):
    response = requests.post(url, json=nuevo_registro)
    if response.status_code == 201:
        return response.json()
    else:
        print(f"Error al agregar el registro: {response.status_code}")
        return None

def mostrar_registro(data):
    if data:
        print("Registro en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")

def main():
    # Obtener todos los registros para encontrar el ID máximo actual
    registros = obtener_registros()
    if not registros:
        max_id = 0
    else:
        ids = [int(registro['id']) for registro in registros if 'id' in registro]
        max_id = max(ids)

    # Crear un nuevo ID para el registro
    nuevo_id = max_id + 1

    # Mostrar el ID del nuevo registro
    print(f"El ID del nuevo registro será: {nuevo_id}")

    # Solicitar al usuario que ingrese los detalles del nuevo registro
    nuevo_registro = {
        "id": nuevo_id,
        "nombre": input("Ingrese el nombre: "),
        "email": input("Ingrese el email: "),
        # Agrega aquí los campos adicionales que esperas en el registro
    }

    # Mostrar los detalles del nuevo registro antes de agregarlo
    print("\nDetalles del nuevo registro:")
    mostrar_registro(nuevo_registro)

    # Confirmar antes de agregar el registro
    confirmar = input("\n¿Desea agregar este registro? (sí/no): ").strip().lower()
    if confirmar in ['sí', 'si', 'yes', 'y']:
        # Agregar el nuevo registro
        registro_agregado = agregar_registro(nuevo_registro)
        if registro_agregado:
            # Mostrar el registro agregado
            mostrar_registro(registro_agregado)
        else:
            print("Registro no agregado.")
    else:
        print("Registro no agregado.")
```

```
if __name__ == "__main__":
    main()
```

## Explicación del Código Mejorado:

### 1. Mostrar ID del Nuevo Registro:

- El código calcula el nuevo ID (`nuevo\_id`) y lo muestra al usuario antes de solicitar cualquier detalle del registro.

### 2. Ingreso de Datos del Usuario:

- Después de mostrar el ID, solicita al usuario que ingrese los detalles del nuevo registro.

### 3. Mostrar Detalles del Nuevo Registro:

- Muestra los detalles del nuevo registro antes de realizar la confirmación final y agregar el registro a la API.

### 4. Confirmación del Usuario:

- Se solicita al usuario que confirme si desea agregar el registro con el ID mostrado. Esto ayuda a evitar errores antes de hacer la solicitud POST.

Este enfoque garantiza que el ID del nuevo registro se muestre al usuario antes de que se ingresen los datos, brindando una vista clara del ID que se asignará. Si necesitas más ajustes o tienes más preguntas, no dudes en preguntar.



Ahora dame el programa para modificar un registro en la misma URL de MockAPI



Claro, para modificar un registro en la misma URL de MockAPI, necesitarás:

### 1. Obtener el registro existente:

- Realiza una solicitud GET para obtener el registro que deseas modificar.

### 2. Actualizar los detalles del registro:

- Permite al usuario ingresar los nuevos valores para los campos que se deben actualizar.

### 3. Enviar los datos actualizados:

- Realiza una solicitud PUT para actualizar el registro en la API.

### 4. Mostrar el registro actualizado:

- Muestra los detalles del registro modificado para confirmar que la actualización fue exitosa.

Aquí tienes el código para modificar un registro:

python

```
import requests
import json

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

def obtener_registro(registro_id):
    response = requests.get(f"{url}/{registro_id}")
    if response.status_code == 200:
```

```

        return response.json()
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")
        return None

def actualizar_registro(registro_id, datos_actualizados):
    response = requests.put(f"{url}/{registro_id}", json=datos_actualizados)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error al actualizar el registro: {response.status_code}")
        return None

def mostrar_registro(data):
    if data:
        print("Registro en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")

def main():
    # Solicitar al usuario que ingrese el ID del registro que desea modificar
    registro_id = input("Ingrese el ID del registro que desea modificar: ")

    # Obtener el registro existente
    registro_existente = obtener_registro(registro_id)
    if not registro_existente:
        return

    # Mostrar los detalles del registro existente
    print("Detalles del registro existente:")
    mostrar_registro(registro_existente)

    # Solicitar al usuario que ingrese los nuevos valores
    datos_actualizados = {
        "nombre": input(f"Ingrese el nuevo nombre (actual: {registro_existente['nombre']}): ") or registro_existente['nombre'],
        "email": input(f"Ingrese el nuevo email (actual: {registro_existente['email']}): ") or registro_existente['email'],
        # Agrega aquí los campos adicionales que esperas en el registro
    }

    # Confirmar antes de actualizar el registro
    print("\nDetalles actualizados del registro:")
    mostrar_registro(datos_actualizados)

    confirmar = input("\n¿Desea actualizar este registro? (sí/no): ").strip().lower()
    if confirmar in ['sí', 'si', 'yes', 'y']:
        # Actualizar el registro
        registro_actualizado = actualizar_registro(registro_id, datos_actualizados)
        if registro_actualizado:
            # Mostrar el registro actualizado
            print("\nRegistro actualizado:")
            mostrar_registro(registro_actualizado)
        else:
            print("Actualización no realizada.")
    else:
        print("Actualización no realizada.")

if __name__ == "__main__":
    main()

```

## Explicación del Código:

### 1. Función `obtener\_registro(registro\_id)`:

- Realiza una solicitud GET para obtener los detalles del registro con el ID especificado.

### 2. Función `actualizar\_registro(registro\_id, datos\_actualizados)`:

- Realiza una solicitud PUT para actualizar el registro con el ID especificado con los nuevos datos proporcionados.

### 3. Función `mostrar\_registro(data)`:

- Muestra los detalles del registro en formato plano.



#### 4. Función `main()`:

- Solicita al usuario que ingrese el ID del registro que desea modificar.
- Obtiene y muestra el registro existente.
- Solicita al usuario que ingrese nuevos valores para los campos del registro, usando los valores actuales como valores predeterminados.
- Muestra los detalles actualizados y solicita confirmación antes de realizar la solicitud PUT para actualizar el registro.
- Muestra el registro actualizado si la actualización fue exitosa.

#### Notas:

- **Campos del Registro:** Ajusta el diccionario `datos\_actualizados` para incluir todos los campos necesarios para tu registro.
- **Validaciones Adicionales:** Puedes agregar más validaciones según los requisitos específicos de tu API y datos.

Este código te permitirá modificar un registro existente en MockAPI, proporcionando una forma interactiva para ingresar y confirmar las actualizaciones. Si tienes más preguntas o necesitas ajustes adicionales, no dudes en preguntar.



Ahora dame el programa para eliminar un registro en la misma URL de MockAPI



Para eliminar un registro en la URL de MockAPI, sigue estos pasos:

1. **Obtener el registro existente (opcional, para confirmar que existe).**
  - Realiza una solicitud GET para verificar que el registro que deseas eliminar existe.
2. **Eliminar el registro:**
  - Realiza una solicitud DELETE a la API para eliminar el registro.
3. **Confirmar que el registro ha sido eliminado.**
  - Verifica si la eliminación fue exitosa y muestra un mensaje al usuario.

Aquí tienes el código para eliminar un registro:

python

```
import requests

# URL de la API de MockAPI
url = "https://<tu-ID>.mockapi.io/api/v1/contactos" # Reemplaza <tu-ID> con el ID
correcto de tu proyecto

def obtener_registro(registro_id):
    response = requests.get(f"{url}/{registro_id}")
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error al realizar la solicitud: {response.status_code}")
        return None

def eliminar_registro(registro_id):
    response = requests.delete(f"{url}/{registro_id}")
```

```

if response.status_code == 200:
    return True
else:
    print(f"Error al eliminar el registro: {response.status_code}")
    return False

def mostrar_registro(data):
    if data:
        print("Registro en formato plano:")
        for key, value in data.items():
            print(f"{key}: {value}")

def main():
    # Solicitar al usuario que ingrese el ID del registro que desea eliminar
    registro_id = input("Ingrese el ID del registro que desea eliminar: ")

    # Obtener el registro existente
    registro_existente = obtener_registro(registro_id)
    if not registro_existente:
        print("El registro no existe o no se pudo recuperar.")
        return

    # Mostrar los detalles del registro existente
    print("Detalles del registro a eliminar:")
    mostrar_registro(registro_existente)

    # Confirmar antes de eliminar el registro
    confirmar = input("\n¿Desea eliminar este registro? (sí/no): ").strip().lower()
    if confirmar in ['sí', 'si', 'yes', 'y']:
        # Eliminar el registro
        exito = eliminar_registro(registro_id)
        if exito:
            print("Registro eliminado exitosamente.")
        else:
            print("No se pudo eliminar el registro.")
    else:
        print("Eliminación no realizada.")

if __name__ == "__main__":
    main()

```

## Explicación del Código:

### 1. Función `obtener\_registro(registro\_id)`:

- Realiza una solicitud GET para obtener los detalles del registro con el ID especificado. Esto es opcional pero útil para confirmar que el registro existe antes de intentar eliminarlo.

### 2. Función `eliminar\_registro(registro\_id)`:

- Realiza una solicitud DELETE para eliminar el registro con el ID especificado. Retorna `True` si la eliminación fue exitosa, y `False` en caso contrario.

### 3. Función `mostrar\_registro(data)`:

- Muestra los detalles del registro en formato plano.

### 4. Función `main()`:

- Solicita al usuario que ingrese el ID del registro que desea eliminar.
- Obtiene y muestra el registro existente para confirmación.
- Solicita confirmación al usuario antes de realizar la eliminación.
- Muestra un mensaje indicando si la eliminación fue exitosa o no.

## Notas:

- **Validaciones:** El código maneja errores básicos y confirma con el usuario antes de proceder con la eliminación.

- **Opcional:** Obtener el registro antes de eliminar es opcional, pero es útil para confirmar que el registro existe y para mostrar sus detalles.

Este código te permitirá eliminar un registro en MockAPI de manera interactiva, asegurando que el usuario confirme la acción antes de proceder. Si tienes más preguntas o necesitas ajustes adicionales, no dudes en preguntar.

ChatGPT puede cometer errores. Comprueba la información importante.