

Consulta API y Exportación



Ayúdame a crear un programa en Python para consultar una tabla de mockapi.io en la URL <https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos>, que me muestre todos los registros en JSON formateado, en formato Dataframe y los exporte. un archivo csv



Memorias actualizadas

Aquí tienes un ejemplo de cómo podrías crear un programa en Python para consultar la tabla de mockapi.io, mostrar los registros en formato JSON, convertirlos en un DataFrame y exportarlos a un archivo CSV:

pitón

```
import requests
import pandas as pd

# URL de la API
url = "https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"

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

# Verifica si la solicitud fue exitosa
if response.status_code == 200:
    # Obtiene los datos en formato JSON
    data = response.json()

    # Muestra los registros en formato JSON formateado
    print("Registros en formato JSON:")
    print(json.dumps(data, indent=4))

    # Convierte los datos a un DataFrame
    df = pd.DataFrame(data)

    # Muestra el DataFrame
    print("\nDataFrame:")
    print(df)

    # Exporta el DataFrame a un archivo CSV
    df.to_csv("contactos.csv", index=False)
    print("\nArchivo CSV 'contactos.csv' creado con éxito.")
else:
    print(f"Error en la solicitud: {response.status_code}")
```

Pasos del código:

1. **Solicitar datos de la API:** Utiliza `requests.get(url)` para hacer una solicitud GET a la URL proporcionada.
2. **Validar respuesta:** Verifica si la solicitud fue exitosa comprobando el código de estado.
3. **Mostrar datos en formato JSON:** Si la solicitud es exitosa, los datos se formatean y se muestran como JSON utilizando `json.dumps`.
4. **Convertir un DataFrame:** Los datos se convierten en un DataFrame utilizando `pd.DataFrame`.
5. **Exportar a CSV:** El DataFrame se exporta a un archivo CSV utilizando `df.to_csv`.

Notas adicionales:

- **json.dumps** requiere que importe el módulo `json`. Si no está importado, agregue el inicio del script con `import json`.
- Si el DataFrame tiene muchas columnas o filas, considere ajustar cómo se muestra el DataFrame en la consola.

Este script generará un archivo CSV llamado `contactos.csv` en el directorio donde ejecuta el script, con todos los registros de la tabla obtenidos de mockapi.io.



¿Cómo instalar las librerías?



Para instalar las librerías necesarias para ejecutar el script, puedes usar `pip`, que es el administrador de paquetes de Python. Aquí te dejo los comandos para instalar las librerías que necesitas:

1. **solicitudes** : Esta librería se utiliza para hacer solicitudes HTTP.
2. **pandas** : Esta librería se utiliza para manejar y analizar datos, como crear DataFrames y exportar a CSV.

Puedes instalar estas librerías ejecutando los siguientes comandos en tu terminal o en el entorno de Python:

```
intento
```

```
pip install requests pandas
```

Si también necesitas formatear el JSON, asegúrate de que el módulo `json` esté disponible (aunque normalmente viene preinstalado con Python). No necesitas instalar `json` porque es parte de la biblioteca estándar de Python.

Pasos para instalar:

1. Abre tu terminal o el terminal de tu entorno de desarrollo.

2. Escribe o copia los comandos anteriores y presiona ``Enter``.
3. Espere a que termine la instalación.

Una vez instaladas, podrás ejecutar el script sin problemas.



Me marca el siguiente error: "C:\Users\dalij\Documents\Codigos\M3.1 Bases de datos NoSQL\M3.1 Bases de datos NoSQL\venv\Scripts\python.exe" "C:\Users\dalij\Documents\Codigos\M3.1 Bases de datos NoSQL\M3.1 Bases de datos NoSQL\getNoSQLAllRecords.py" Error en la solicitud: 404 Proceso finalizado con código de salida 0



El error "Error en la solicitud: 404" significa que la URL que estás intentando acceder no se encuentra (HTTP 404 es un código de estado que indica "Not Found").

Para solucionar este problema, verifica los siguientes puntos:

1. **Verifica la URL:** Asegúrate de que la URL que estás utilizando es correcta. Intenta abrirla directamente en un navegador para asegurarte de que funciona.
2. **Revisa la API Endpoint:** Es posible que la API o el endpoint haya cambiado o que el recurso no esté disponible temporalmente.
3. **Comprueba la conectividad:** Asegúrate de que tienes acceso a Internet y de que no hay problemas de conectividad que puedan estar causando el error.

Actualiza la URL

Verifica que estás usando exactamente la siguiente URL en tu script:

```
python
```

```
url = "https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"
```

Si confirmas que la URL es correcta y sigue dando el error 404, es posible que la API haya sido eliminada o que el endpoint específico ya no esté disponible.

Ejemplo de código actualizado:

```
python
```

```
import requests
import pandas as pd
import json
```

```
# URL de la API
url = "https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"

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

# Verifica si la solicitud fue exitosa
if response.status_code == 200:
    # Obtiene los datos en formato JSON
    data = response.json()

    # Muestra los registros en formato JSON formateado
    print("Registros en formato JSON:")
    print(json.dumps(data, indent=4))

    # Convierte los datos a un DataFrame
    df = pd.DataFrame(data)

    # Muestra el DataFrame
    print("\nDataFrame:")
    print(df)

    # Exporta el DataFrame a un archivo CSV
    df.to_csv("contactos.csv", index=False)
    print("\nArchivo CSV 'contactos.csv' creado con éxito.")
else:
    print(f"Error en la solicitud: {response.status_code}")
```

Si todo está correcto y sigues teniendo problemas, considera crear un nuevo endpoint en mockapi.io o verificar si el endpoint ha sido modificado.



Me marca el siguiente error: "C:\Users\dalij\Documents\Codigos\M3.1 Bases de datos NoSQL\M3.1 Bases de datos NoSQL\venv\Scripts\python.exe"
 "C:\Users\dalij\Documents\Codigos\M3.1 Bases de datos NoSQL\M3.1 Bases de datos NoSQL\getNoSQLAllRecords.py"
 Traceback (most recent call last):
 File "C:\Users\dalij\Documents\Codigos\M3.1 Bases de datos NoSQL\M3.1 Bases de datos NoSQL\getNoSQLAllRecords.py", line 17, in <module>
 print(json.dumps(data, indent=4))
 ^^^^^
 NameError: name 'json' is not definedRegistros en formato JSON:
 . Did you forget to import 'json'?

Process finished with exit code 1



El error `NameError: name 'json' is not defined` significa que el módulo `json` no ha sido importado en tu script. Para solucionar este problema, simplemente necesitas asegurarte de importar el módulo `json` al inicio del archivo.

Aquí te dejo el código corregido:

python

```
import requests
import pandas as pd
import json # Importa el módulo json

# URL de la API
url = "https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"

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

# Verifica si la solicitud fue exitosa
if response.status_code == 200:
    # Obtiene los datos en formato JSON
    data = response.json()

    # Muestra los registros en formato JSON formateado
    print("Registros en formato JSON:")
    print(json.dumps(data, indent=4))

    # Convierte los datos a un DataFrame
    df = pd.DataFrame(data)

    # Muestra el DataFrame
    print("\nDataFrame:")
    print(df)

    # Exporta el DataFrame a un archivo CSV
    df.to_csv("contactos.csv", index=False)
    print("\nArchivo CSV 'contactos.csv' creado con éxito.")
else:
    print(f"Error en la solicitud: {response.status_code}")
```

Pasos:

1. **Importa el módulo `json`:** Agrega `import json` al inicio del script.
2. **Vuelve a ejecutar el script.**

Con esto, deberías poder ejecutar el código sin problemas.



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, aquí tienes un programa en Python que consulta la URL de MockAPI para obtener un registro específico y muestra todos los campos de ese registro en formato plano. En este caso, el registro se selecciona por un identificador único (`id`), que debes especificar en el script.

Aquí está el código modificado para cumplir con estos requisitos:

python

```
import requests
import json

# URL de la API
url = "https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"

# ID del registro que quieres mostrar
registro_id = "1" # Cambia este valor al ID del registro que quieres consultar

# Realiza la solicitud GET a la API para obtener todos los registros
response = requests.get(url)

# Verifica si la solicitud fue exitosa
if response.status_code == 200:
    # Obtiene los datos en formato JSON
    data = response.json()

    # Busca el registro con el ID especificado
    registro = next((item for item in data if item['id'] == registro_id), None)

    if registro:
        # Muestra el registro en formato plano
        print("Registro en formato plano:")
        for campo, valor in registro.items():
            print(f"{campo}: {valor}")
    else:
        print(f"Registro con ID {registro_id} no encontrado.")
else:
    print(f"Error en la solicitud: {response.status_code}")
```

Pasos del código:

1. **Consulta todos los registros:** Se realiza una solicitud GET a la URL para obtener todos los registros.
2. **Busca el registro por ID:** Usa una expresión generadora para buscar el registro con el `id` especificado.
3. **Muestra el registro:** Imprime cada campo y su valor en formato plano.

Notas:

- **ID del Registro:** Asegúrate de cambiar `registro_id` al valor del ID que deseas mostrar.
- **Formato Plano:** El código muestra cada campo del registro en formato clave-valor, sin usar DataFrame ni exportar a CSV.



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



Para agregar un registro en la URL de MockAPI, necesitas hacer una solicitud POST con los datos del nuevo registro. Aquí tienes un ejemplo de cómo hacerlo utilizando el módulo `requests` en Python:

pitón

```
import requests
import json

# URL de la API
url = "https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"

# Datos del nuevo registro
nuevo_registro = {
    "nombre": "Juan Pérez",
    "email": "juan.perez@example.com",
    "telefono": "1234567890",
    "direccion": "Calle Falsa 123",
    "ciudad": "Ciudad Ejemplo"
}

# Realiza la solicitud POST para agregar el nuevo registro
response = requests.post(url, json=nuevo_registro)

# Verifica si la solicitud fue exitosa
if response.status_code == 201:
    # Obtiene los datos del nuevo registro en formato JSON
    data = response.json()

    # Muestra los datos del nuevo registro
    print("Nuevo registro agregado exitosamente:")
    print(json.dumps(data, indent=4))
else:
    print(f"Error en la solicitud: {response.status_code}")
    print(response.text)
```

Pasos del código:

1. **Defina los datos del nuevo registro:** Configure un diccionario con los campos que desea agregar al nuevo registro.
2. **Envía la solicitud POST:** Utiliza `requests.post(url, json=nuevo_registro)` para enviar los datos al servidor.
3. **Verifica la respuesta:** Si la solicitud es exitosa (código de estado 201), muestra los detalles del nuevo registro. Si ocurre un error, se imprime el código de estado y el mensaje de error.

Notas:

- **Campo `id`:** En MockAPI, el campo `id` generalmente se genera automáticamente, por lo que no es necesario incluirlo en el diccionario `nuevo_registro`.
- **Datos del Registro:** Ajusta los campos en `nuevo_registro` según la estructura esperada por la API.



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



Para modificar un registro en la URL de MockAPI, necesitas hacer una solicitud
`PUT` o `PATCH` específico `PUT`, What

pag

```
import requests

import json

# URL base de la API
url =
url

u
"https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"

# ID del registro que quieres modificar
registro_id =
registro_id

registro_
regis

r
"1" # Cambia este valor al ID del registro que deseas modificar

# Datos que deseas actualizar en el registro
actualizacion_registro = {

actualizacion_registro = {

actualizacion_registro = {
actualizacion_registr
actualizacion_re
actualizaci
actuali

act
"nombre": "Juan Pérez Modificado",

"email": "juan.perez.modificado@example.com",

"telefono": "0987654321",

"direccion": "Avenida Siempre Viva 742",
"ciudad": "Ciudad Actualizada"
}
```



```

}
# Construye la URL completa para el registro específico
url_completa =
url_completa = f

url_completa

url
f"{url}/{registro_id}"

# Realiza la solicitud PUT para modificar el registro
response = requests.put(url_completa, json=actualizacion_registro)

response = requests.put(url_completa, json=actualizacion
response = requests.put(url_completa, json=actual
response = requests.put(url_completa, json
response = requests.put(url_completa
response = requests.put(url
response =

response
# Verifica si la solicitud fue exitosa
if response.status_code == 200:

# Obtiene los datos del registro actualizado en formato JSON
data = response.json()

data = response.json()

data =

data

# Muestra los datos del registro actualizado
print("Registro modificado exitosamente:")

print(json.dumps(data, indent=4))
else:

print(f"Error en la solicitud: {response.status_code}")
print(response.text)

```

Pasos

1. **Defina la URL del registro específico:** Con el ID del
2. **Configura los datos de actualización:** Crea
3. **Envíe la solicitud PUT:** Utilice `requests.put(url_completa, json=actualizacion_registro)` pag
4. **Verifica la respuesta:** S

No un

- **Campo `ID`:** Como `ID` (es `registro_id`)
- **Método `PUT` vs `PATCH`:** ``PUT` reemplazo `PATCH` solo `requests.patch` en lu `requests.put` si solo que

Si prefieres modificar solo algunos campos, el código sería similar pero usando `PATCH`:

pitón

```
response = requests.patch(url_completa, json=actualizacion_registro)
```

Este enf



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



Para eliminar un registro en la URL de MockAPI, puedes usar una solicitud `DELETE` específica

pitón

```
import requests

# URL base de la API
url =
url
"https://66b4e5159f9169621ea4c277.mockapi.io/api/v1/contactos"

# ID del registro que quieres eliminar
registro_id = "1" # Cambia este valor al ID del registro que deseas eliminar

# Construye la URL completa para el registro específico
url_completa =
url_co
f"{url}/{registro_id}"

# Realiza la solicitud DELETE para eliminar el registro
response = requests.delete(url_completa)

response = requests.dele

respo
# Verifica si la solicitud fue exitosa
if response.status_code == 200:
    print(f"Registro con ID {registro_id} eliminado exitosamente.")
else:
    prin
    print(f"Error en la solicitud: {response.status_code}")
```

```
prin
print(response.text)
``
```

Pasos del código:

1. **Define la URL del registro específico:** Combina la URL base con el `ID` del registro que deseas eliminar.
2. **Envía la solicitud `DELETE`:** Utiliza `requests.delete(url_completa)` para eliminar el registro.
3. **Verifica la respuesta:** Si la solicitud es exitosa (código de estado 200), se confirma que el registro ha sido eliminado.

Notas:

- **Campo `ID`:** Asegúrate de que el `ID` (`registro_id`) que proporcionas corresponda al registro que deseas eliminar.
- **Respuesta de la API:** Si el registro se elimina correctamente, recibirás un mensaje de éxito. Si ocurre un error (por ejemplo, si el `ID` no existe), se imprimirá el código de estado y un mensaje de error.

Este código elimina el registro específico de la API MockAPI basado en el `ID` proporcionado.

¿Esta respuesta fue mejor o peor?

MejorPeorIgual

ChatGPT puede cometer errores. Comprueba la información importante.