

API Endpoints and Databases



Cuando se habla de API, ¿a qué se refiere un endpoint?, qué es una base de datos No SQL y qué es un recurso y qué es una API



Aquí tienes las definiciones de los términos que mencionas:

1. Endpoint (en el contexto de una API)

Un **endpoint** es una URL específica en un servidor web donde se puede acceder a un recurso o realizar una acción a través de una API (Application Programming Interface). En términos simples, es el punto de entrada de una API, donde las solicitudes de los clientes (como aplicaciones, sitios web u otros servicios) son enviadas para obtener o modificar datos. Cada endpoint está asociado con una función específica de la API y, a menudo, se define mediante un método HTTP como GET, POST, PUT o DELETE.

2. Base de datos NoSQL

Una **base de datos NoSQL** es un tipo de base de datos que no sigue el modelo tradicional de bases de datos relacionales (SQL). En lugar de almacenar datos en tablas con filas y columnas, las bases de datos NoSQL pueden almacenar datos en una variedad de formatos, como documentos (JSON, BSON), gráficos, pares clave-valor, o tablas de tipo columna ancha. Son especialmente útiles para manejar grandes volúmenes de datos no estructurados o semi-estructurados y son conocidas por su escalabilidad horizontal y flexibilidad. Ejemplos comunes de bases de datos NoSQL incluyen MongoDB, Cassandra, Redis, y CouchDB.

3. Recurso

En el contexto de una API, un **recurso** es una entidad que puede ser identificada y manipulada a través de la API. Los recursos son objetos con los que la API interactúa, como usuarios, artículos, imágenes, etc. En una API REST, los recursos suelen representarse en formatos como JSON o XML y están accesibles mediante endpoints específicos.

4. API (Application Programming Interface)

Una **API** (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí. Las API definen cómo un desarrollador puede interactuar con los servicios, funciones o datos de una aplicación o sistema. Por ejemplo, una

API de redes sociales puede permitir que una aplicación externa publique actualizaciones en el perfil de un usuario o recupere su lista de amigos.

¿Necesitas más detalles sobre alguno de estos conceptos?



que son los metodos web cuando se hace una solicitud o request



Los **métodos web** (también conocidos como **métodos HTTP**) son verbos que indican la acción que se desea realizar sobre un recurso específico cuando se hace una solicitud (request) a un servidor a través de una API o servicio web. Aquí te explico los métodos más comunes:

1. GET

- **Propósito:** Recuperar datos de un recurso específico.
- **Uso común:** Obtener información sin modificarla. Por ejemplo, solicitar un listado de usuarios o los detalles de un artículo específico.
- **Idempotente:** Sí (múltiples solicitudes iguales no cambian el estado del servidor).

2. POST

- **Propósito:** Enviar datos al servidor para crear un nuevo recurso.
- **Uso común:** Crear un nuevo registro en una base de datos, como un nuevo usuario o un nuevo comentario en un blog.
- **Idempotente:** No (múltiples solicitudes iguales pueden crear múltiples recursos).

3. PUT

- **Propósito:** Actualizar o reemplazar un recurso existente con nuevos datos.
- **Uso común:** Modificar la información completa de un recurso específico. Por ejemplo, actualizar todos los detalles de un perfil de usuario.
- **Idempotente:** Sí (múltiples solicitudes iguales tendrán el mismo efecto que una sola).

4. PATCH

- **Propósito:** Actualizar parcialmente un recurso existente.
- **Uso común:** Modificar solo algunos campos de un recurso, como cambiar solo la dirección de correo electrónico de un usuario.
- **Idempotente:** Sí (múltiples solicitudes iguales tendrán el mismo efecto que una sola).

5. DELETE

- **Propósito:** Eliminar un recurso específico del servidor.
- **Uso común:** Borrar un registro, como eliminar una cuenta de usuario o eliminar un comentario.

- **Idempotente:** Sí (eliminar un recurso que ya no existe no cambiará el estado del servidor).

6. OPTIONS

- **Propósito:** Obtener información sobre las opciones de comunicación permitidas para un recurso o servidor.
- **Uso común:** Determinar qué métodos HTTP son soportados por un servidor para un recurso específico.

7. HEAD

- **Propósito:** Similar a GET, pero solo devuelve los encabezados de la respuesta, sin el cuerpo de los datos.
- **Uso común:** Verificar si un recurso existe o para obtener metadatos sobre un recurso sin descargar el contenido completo.
- **Idempotente:** Sí.

Estos métodos permiten una interacción estándar entre los clientes (como aplicaciones, navegadores web, etc.) y los servidores, facilitando la creación, lectura, actualización y eliminación de recursos a través de la web.

¿Te gustaría profundizar en alguno de estos métodos?



Ayúdame a crear un programa en Python para consultar una tabla de mockapi.io en la URL <https://66b4e4f09f9169621ea4c19d.mockapi.io/api/v1/contactos>, que me muestre todos los registros en JSON formateado, en formato Dataframe y los exporte a un csv



Voy a ayudarte a crear un programa en Python para consultar la tabla de `mockapi.io`, formatear los resultados en JSON y DataFrame, y exportarlos a un archivo CSV.

A continuación, te muestro el código:

python

```
import requests
import pandas as pd

# URL de la API
url = "https://66b4e4f09f9169621ea4c19d.mockapi.io/api/v1/contactos"

# Realizar la solicitud GET para obtener los datos
response = requests.get(url)

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

```
# Mostrar los datos en formato JSON formateado
print("Datos en formato JSON formateado:")
print(pd.json_normalize(data).to_json(orient="records", indent=4))

# Convertir los datos a un DataFrame de Pandas
df = pd.DataFrame(data)

# Mostrar los datos en formato DataFrame
print("\nDatos en formato DataFrame:")
print(df)

# Exportar los datos a un archivo CSV
df.to_csv('contactos.csv', index=False)
print("\nDatos exportados a 'contactos.csv'")
else:
    print(f"Error al obtener los datos. Código de estado: {response.status_code}")
```

Explicación del código:

1. Importación de módulos:

- `requests`: Para realizar la solicitud HTTP a la API.
- `pandas`: Para manejar y transformar los datos en un DataFrame y exportarlos a CSV.

2. Definición de la URL:

- La URL de la API de `mockapi.io` es especificada.

3. Solicitud GET:

- Se realiza una solicitud `GET` a la URL especificada.

4. Verificación del estado de la solicitud:

- Si la solicitud es exitosa (código 200), se procede a procesar los datos.

5. Formateo y presentación de datos:

- Los datos JSON son formateados y mostrados en un formato fácil de leer.
- Los datos se convierten a un `DataFrame` de `pandas` para facilitar su manipulación.

6. Exportación a CSV:

- El `DataFrame` se exporta a un archivo CSV llamado `contactos.csv`.

7. Manejo de errores:

- Si la solicitud falla, se imprime un mensaje con el código de estado de la respuesta.

Requisitos previos:

- Asegúrate de tener instalados los módulos `requests` y `pandas`. Si no los tienes, puedes instalarlos utilizando:

```
bash
```

```
pip install requests pandas
```

Este programa te permitirá consultar la tabla de `mockapi.io`, visualizar los registros en diferentes formatos y exportarlos a un archivo CSV para un análisis adicional.



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



Para mostrar un solo registro de la API en formato plano, puedes modificar el programa para que acepte un ID específico del registro que deseas consultar. Aquí te muestro cómo hacerlo:

python

```
import requests

# URL base de la API
url_base = "https://66b4e4f09f9169621ea4c19d.mockapi.io/api/v1/contactos"

# Función para obtener un registro específico por ID
def obtener_registro_por_id(registro_id):
    # Construir la URL para el registro específico
    url = f"{url_base}/{registro_id}"

    # Realizar la solicitud GET para obtener los datos del registro
    response = requests.get(url)

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

        # Mostrar los datos en formato plano
        print(f"Datos del registro con ID {registro_id}:")
        for key, value in data.items():
            print(f"{key}: {value}")
    else:
        print(f"Error al obtener el registro. Código de estado: {response.status_code}")

# ID del registro que deseas consultar
registro_id = 1 # Cambia este valor por el ID que deseas consultar

# Llamar a la función para mostrar el registro
obtener_registro_por_id(registro_id)
```

Explicación del código:

1. URL base:

- La `url_base` corresponde a la URL principal de la API de `mockapi.io`.

2. Función `obtener_registro_por_id`:

- Esta función recibe como parámetro el `registro_id` que identifica el registro específico que quieres consultar.
- Construye la URL completa para acceder a ese registro añadiendo el ID al final de la URL base.
- Realiza una solicitud `GET` para obtener los datos de ese registro.

- Si la solicitud es exitosa, los datos se muestran en formato plano, donde cada campo se imprime en una línea con su respectivo valor.
- Si la solicitud falla, se muestra un mensaje de error con el código de estado.

3. ID del registro:

- Puedes cambiar el valor de ``registro_id`` por el ID del registro que quieres consultar.

4. Llamada a la función:

- Se llama a la función ``obtener_registro_por_id`` con el ID específico para mostrar los detalles del registro en formato plano.

Ejemplo de salida:

Si tienes un registro con ``ID = 1``, la salida podría verse así:

yaml

```
Datos del registro con ID 1:
id: 1
nombre: Juan Perez
email: juan.perez@example.com
telefono: +1234567890
```

Este programa te permite consultar un registro específico de la API de ``mockapi.io`` y mostrarlo en un formato claro y sencillo.



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



Para agregar un registro a la API de MockAPI en la URL proporcionada, puedes utilizar el método ``POST`` de HTTP. Aquí te muestro cómo hacerlo con un programa en Python:

python

```
import requests

# URL base de la API
url_base = "https://66b4e4f09f9169621ea4c19d.mockapi.io/api/v1/contactos"

# Datos del nuevo registro a agregar
nuevo_registro = {
    "nombre": "Ana Lopez",
    "email": "ana.lopez@example.com",
    "telefono": "+9876543210"
}

# Función para agregar un nuevo registro
def agregar_registro(datos):
    # Realizar la solicitud POST para agregar los datos
```

```
response = requests.post(url_base, json=datos)

# Verificar si la solicitud fue exitosa
if response.status_code == 201:
    # Convertir los datos JSON de la respuesta en un diccionario de Python
    data = response.json()

    print("Registro agregado con éxito. Datos del nuevo registro:")
    for key, value in data.items():
        print(f"{key}: {value}")
else:
    print(f"Error al agregar el registro. Código de estado: {response.status_code}")

# Llamar a la función para agregar el nuevo registro
agregar_registro(nuevo_registro)
```

Explicación del código:

1. URL base:

- La `url_base` es la URL principal de la API donde se almacenan los contactos.

2. Datos del nuevo registro:

- El diccionario `nuevo_registro` contiene los datos que deseas agregar, como `nombre`, `email`, y `telefono`. Puedes personalizar estos campos según lo que permita la API.

3. Función `agregar_registro`:

- Esta función recibe como parámetro el diccionario con los datos del nuevo registro.
- Se realiza una solicitud `POST` a la URL base, enviando los datos en formato JSON.
- Si la solicitud es exitosa (código de estado 201, que indica que un recurso fue creado), se muestran los detalles del nuevo registro.
- Si la solicitud falla, se muestra un mensaje de error con el código de estado.

4. Llamada a la función:

- Se llama a la función `agregar_registro` con los datos del nuevo registro para que este sea añadido a la API.

Ejemplo de salida:

Si el registro es agregado exitosamente, verás algo como:

```
yaml
```

```
Registro agregado con éxito. Datos del nuevo registro:
id: 2
nombre: Ana Lopez
email: ana.lopez@example.com
telefono: +9876543210
```

Requisitos previos:

- Asegúrate de que el formato de los datos coincida con lo esperado por la API de `mockapi.io`. Si el formato no es correcto, la API puede rechazar la solicitud.

Este programa permite agregar un nuevo registro a la tabla de contactos en `mockapi.io`, mostrando luego los detalles del registro agregado.



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



Para modificar un registro existente en la API de MockAPI, puedes utilizar el método `PUT` o `PATCH`. El método `PUT` reemplaza todo el registro, mientras que `PATCH` solo actualiza los campos especificados sin afectar los demás. Aquí te muestro cómo hacerlo usando `PATCH`, que es más flexible si solo deseas modificar algunos campos:

python

```
import requests

# URL base de la API
url_base = "https://66b4e4f09f9169621ea4c19d.mockapi.io/api/v1/contactos"

# ID del registro que deseas modificar
registro_id = 1 # Cambia este valor al ID del registro que deseas actualizar

# Datos que deseas modificar en el registro
datos_modificados = {
    "nombre": "Carlos Gomez",
    "email": "carlos.gomez@example.com"
}

# Función para modificar un registro existente
def modificar_registro(registro_id, datos):
    # Construir la URL para el registro específico
    url = f"{url_base}/{registro_id}"

    # Realizar la solicitud PATCH para actualizar los datos
    response = requests.patch(url, json=datos)

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

        print("Registro modificado con éxito. Datos actualizados:")
        for key, value in data.items():
            print(f"{key}: {value}")
    else:
        print(f"Error al modificar el registro. Código de estado: {response.status_code}")

# Llamar a la función para modificar el registro
modificar_registro(registro_id, datos_modificados)
```

Explicación del código:

1. URL base:

- ``url_base`` es la URL principal de la API donde se almacenan los contactos.

2. ID del registro:

- ``registro_id`` corresponde al ID del registro que deseas modificar. Cambia este valor según el registro específico que quieras actualizar.

3. Datos modificados:

- ``datos_modificados`` es un diccionario que contiene los campos y sus nuevos valores. Solo se actualizarán los campos especificados.

4. Función ``modificar_registro``:

- Esta función recibe el ``registro_id`` y el diccionario con los ``datos`` a modificar.
- Se construye la URL específica del registro añadiendo el ID al final de la ``url_base``.
- Se realiza una solicitud ``PATCH`` a esa URL, enviando los nuevos datos en formato JSON.
- Si la solicitud es exitosa (código de estado 200, que indica que el recurso fue actualizado), se muestran los detalles del registro modificado.
- Si la solicitud falla, se muestra un mensaje de error con el código de estado.

5. Llamada a la función:

- Se llama a la función ``modificar_registro`` con el ID del registro que deseas modificar y los nuevos datos.

Ejemplo de salida:

Si el registro se modifica exitosamente, verás algo como:

```
yaml
```

```
Registro modificado con éxito. Datos actualizados:
id: 1
nombre: Carlos Gomez
email: carlos.gomez@example.com
telefono: +1234567890
```

Requisitos previos:

- Asegúrate de que el registro con el ``ID`` especificado exista en la API. Si no existe, la API puede devolver un error 404 (no encontrado).
- El formato de los datos debe coincidir con lo que espera la API.

Este programa te permite modificar un registro existente en la tabla de contactos de ``mockapi.io``, actualizando solo los campos especificados y dejando los demás intactos.



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



Para eliminar un registro existente en la API de MockAPI, puedes utilizar el método `DELETE`. Aquí te muestro cómo hacerlo con un programa en Python:

python

```
import requests

# URL base de la API
url_base = "https://66b4e4f09f9169621ea4c19d.mockapi.io/api/v1/contactos"

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

# Función para eliminar un registro existente
def eliminar_registro(registro_id):
    # Construir la URL para el registro específico
    url = f"{url_base}/{registro_id}"

    # Realizar la solicitud DELETE para eliminar el registro
    response = requests.delete(url)

    # Verificar si la solicitud fue exitosa
    if response.status_code == 200:
        print(f"Registro con ID {registro_id} eliminado con éxito.")
    else:
        print(f"Error al eliminar el registro. Código de estado: {response.status_code}")

# Llamar a la función para eliminar el registro
eliminar_registro(registro_id)
```

Explicación del código:

1. URL base:

- `url_base` es la URL principal de la API donde se almacenan los contactos.

2. ID del registro:

- `registro_id` corresponde al ID del registro que deseas eliminar. Cambia este valor según el registro específico que quieras borrar.

3. Función `eliminar_registro`:

- Esta función recibe el `registro_id` que identifica el registro que deseas eliminar.
- Se construye la URL específica del registro añadiendo el ID al final de la `url_base`.
- Se realiza una solicitud `DELETE` a esa URL para eliminar el registro.
- Si la solicitud es exitosa (código de estado 200, que indica que el recurso fue eliminado), se muestra un mensaje confirmando la eliminación.
- Si la solicitud falla, se muestra un mensaje de error con el código de estado.

4. Llamada a la función:

- Se llama a la función `eliminar_registro` con el ID del registro que deseas eliminar.

Ejemplo de salida:

Si el registro se elimina exitosamente, verás algo como:

Registro con ID 1 eliminado con éxito.

Requisitos previos:

- Asegúrate de que el registro con el `ID` especificado exista en la API. Si no existe, la API puede devolver un error 404 (no encontrado).

Este programa te permite eliminar un registro específico en la tabla de contactos de `mockapi.io`. Es útil para mantener la base de datos limpia eliminando los registros que ya no son necesarios.

ChatGPT puede cometer errores. Considera verificar la información importante.