



Genera un api con fastAPI en python que pueda realizar las operaciones de sumar, restar, multiplicar y dividir con números



Para crear una API con FastAPI en Python que realice operaciones matemáticas básicas, sigue estos pasos:

1. Instala FastAPI y Uvicorn (un servidor web ASGI para Python) utilizando el comando ``pip install fastapi uvicorn``.
2. Crea un archivo llamado ``main.py`` en tu proyecto.
3. Importa las siguientes librerías en tu archivo ``main.py``:

 Copy code

```
from fastapi import FastAPI, HTTPException, Path
```

4. Crea una instancia de ``FastAPI`` en tu archivo ``main.py``:

 Copy code

```
app = FastAPI()
```

5. Crea una ruta para cada operación matemática que desees incluir en tu API. Por ejemplo, para crear una ruta para sumar dos números, puedes utilizar el decorador ``@app.get`` de la siguiente manera:

 Copy code

```
@app.get("/suma/{num1}/{num2}")
async def suma(num1: int, num2: int):
    return {"resultado": num1 + num2}
```

6. Repite el paso 5 para cada operación matemática que desees incluir en tu API.

petición GET a ``http://localhost:8000/suma/5/3``.

Este es un ejemplo básico, pero puedes añadir más funcionalidades y validaciones a tu API según tus necesidades.



Escribe un ejemplo de consumir esta api con retrofit en Java



Para consumir una API creada con FastAPI en Java utilizando Retrofit, sigue estos pasos:

1. Agrega la dependencia de Retrofit en tu archivo ``build.gradle``:

Copy code

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
```

2. Crea una interfaz para definir las operaciones de tu API. Por ejemplo, si tu API tiene una ruta para sumar dos números, puedes crear una interfaz de la siguiente manera:

Copy code

```
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Path;

public interface MathApi {

    @GET("/suma/{num1}/{num2}")
    Call<Result> suma(@Path("num1") int num1, @Path("num2") int
num2);
}
```

3. Crea una clase ``Result`` para manejar la respuesta de tu API:

Copy code

```
public class Result {
    private int resultado;

    public int getResultado() {
        return resultado;
    }

    public void setResultado(int resultado) {
        this.resultado = resultado;
    }
}
```

4. Crea una instancia de Retrofit en tu aplicación:

 Copy code

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://localhost:8000")
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

5. Crea una instancia de la interfaz `MathApi` utilizando la instancia de Retrofit:

 Copy code

```
MathApi mathApi = retrofit.create(MathApi.class);
```

6. Utiliza la instancia de `MathApi` para hacer peticiones a tu API. Por ejemplo, para realizar una suma, puedes utilizar el método `suma` de la siguiente manera:

 Copy code

```
Call<Result> call = mathApi.suma(5, 3);
call.enqueue(new Callback<Result>() {
    @Override
    public void onResponse(Call<Result> call, Response<Result>
```

```
response) {  
    Result result = response.body();  
    int sum = result.getResultado();  
    System.out.println("La suma es: " + sum);  
}  
  
@Override  
public void onFailure(Call<Result> call, Throwable t) {  
    System.out.println("Error: " + t.getMessage());  
}  
});
```

Ten en cuenta que este ejemplo es solo una ilustración básica de como consumir una api con Retrofit, puedes añadir más funcionalidades y validaciones según tus necesidades.

