Implementar pruebas para tu API utilizando `pytest` es una excelente manera de asegurarte de que tu aplicación funcione correctamente y de que los cambios futuros no rompan la funcionalidad existente. A continuación, te guiaré a través del proceso de configuración y creación de pruebas para tu proyecto "SentinelCore".

### 1. \*\*Instalación de `pytest` y `httpx`\*\*

Primero, asegúrate de que `pytest` y `httpx` estén instalados en tu entorno virtual. `httpx` es una biblioteca que puedes usar para realizar solicitudes HTTP en tus pruebas.

Ejecuta el siguiente comando en tu terminal:

```bash

pip install pytest httpx

```

### 2. \*\*Estructura del Directorio de Pruebas\*\*

Crea un directorio llamado `tests` en la raíz de tu proyecto para almacenar tus pruebas:

```

SentinelCore/

│

├── app/

│ ├── main.py

│ ├── routers/

│ │ ├── \_\_init\_\_.py

│ │ └── mi\_router.py

│ ├── models/

│ ├── services/

│ └── utils/

│

├── tests/

│ └── test\_main.py # Archivo de pruebas

│

├── venv/

└── requirements.txt

```

### 3. \*\*Escribir Pruebas para tu API\*\*

Crea un archivo llamado `test\_main.py` en el directorio `tests` y agrega el siguiente código para probar la ruta raíz y la ruta de ejemplo:

```python

# tests/test\_main.py

import pytest

from fastapi.testclient import TestClient

from app.main import app # Asegúrate de que la ruta sea correcta

client = TestClient(app)

def test\_read\_root():

response = client.get("/")

assert response.status\_code == 200

assert response.json() == {"message": "Bienvenido a SentinelCore"}

def test\_read\_example():

response = client.get("/example")

assert response.status\_code == 200

assert response.json() == {"message": "Este es un ejemplo desde mi\_router"}

```

### 4. \*\*Ejecutar las Pruebas\*\*

Para ejecutar las pruebas, asegúrate de estar en el directorio raíz de tu proyecto (donde se encuentra la carpeta `tests`) y ejecuta el siguiente comando:

```bash

pytest

```

### 5. \*\*Ver Resultados de las Pruebas\*\*

Después de ejecutar `pytest`, deberías ver un resumen de las pruebas que se han ejecutado. Si todas las pruebas pasan, verás un mensaje indicando que todas las pruebas fueron exitosas. Si alguna prueba falla, `pytest` te mostrará información sobre qué prueba falló y por qué.

### 6. \*\*Agregar Más Pruebas\*\*

A medida que desarrolles más funcionalidades en tu API, puedes agregar más pruebas en el directorio `tests`. Asegúrate de probar todas las rutas y servicios importantes para garantizar que tu aplicación funcione como se espera.

### Conclusión

Implementar pruebas con `pytest` es una parte crucial del desarrollo de software, ya que te ayuda a mantener la calidad y la estabilidad de tu aplicación. Siguiendo estos pasos, podrás configurar y ejecutar pruebas para tu API en "SentinelCore". Si tienes más preguntas o necesitas ayuda adicional, no dudes en preguntar. ¡Buena suerte!