Introducción a Fast API

Hilario Tun

FastAPI

Es un micro framework para crear APIs con python 3.7 o superior

- Alto rendimiento
- Fácil de aprender
- Fácil de programar
- Listo para producción

Porque?

Usado por la industria

- Microsoft, Uber, Netflix, etc.
- Buena aceptación en github

Desempeño

- Entre los frameworks con mejor desempeño alcanzado con python

Otras características

Inyección de dependencias

WebSockets

Archivos

Tareas en segundo plano

Integracion con GraphQl o cualquier ORM

Soporte para cualquier editor

¿Cómo funciona?

Base

Pydantic

Validación y administración de preferencias utilizando los type annotations de Python.

Starlette

Framework/Toolkit ligero ideal para construir servicios web asincrónicos en Python.

Uvicorn

Una implementación de servidor web ASGI para Python.

Curva de aprendizaje corta

Sin esfuerzo extra

¿Cómo lo hace?

Basado en estándares

Python

- f-strings
- Type hints

Uso de

1. OpenAPI
2. JSON Schema
3. OAuth2
4. API docs automáticos

Type hints?

Introducido en Python 3.5+

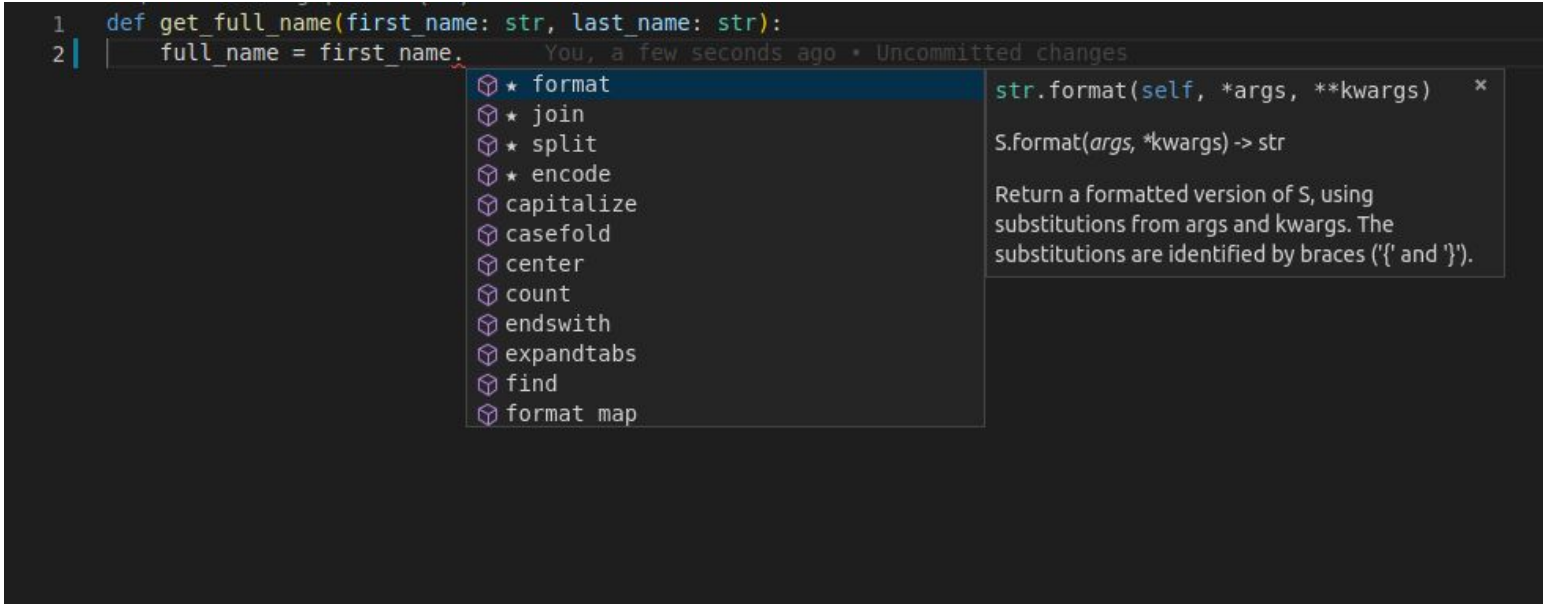
Utilizado por herramientas de terceros

IDE’s

Linters

Type hints?

Autocompletado para todo

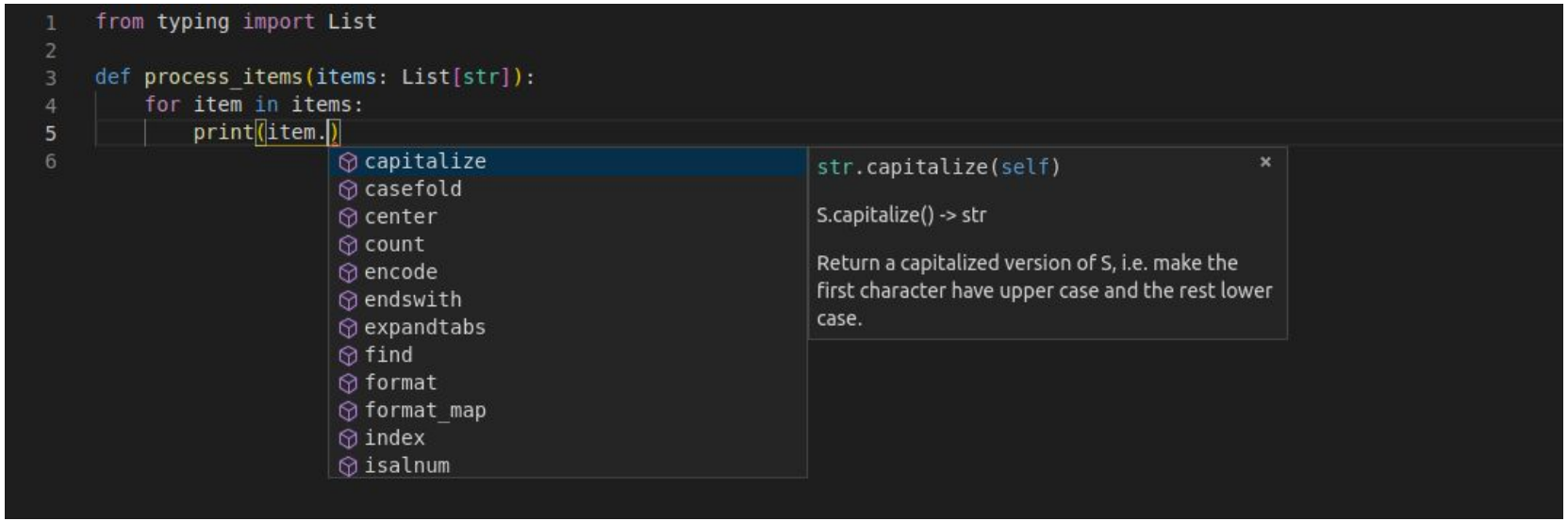


Type hints?

Revisión de tipos y errores

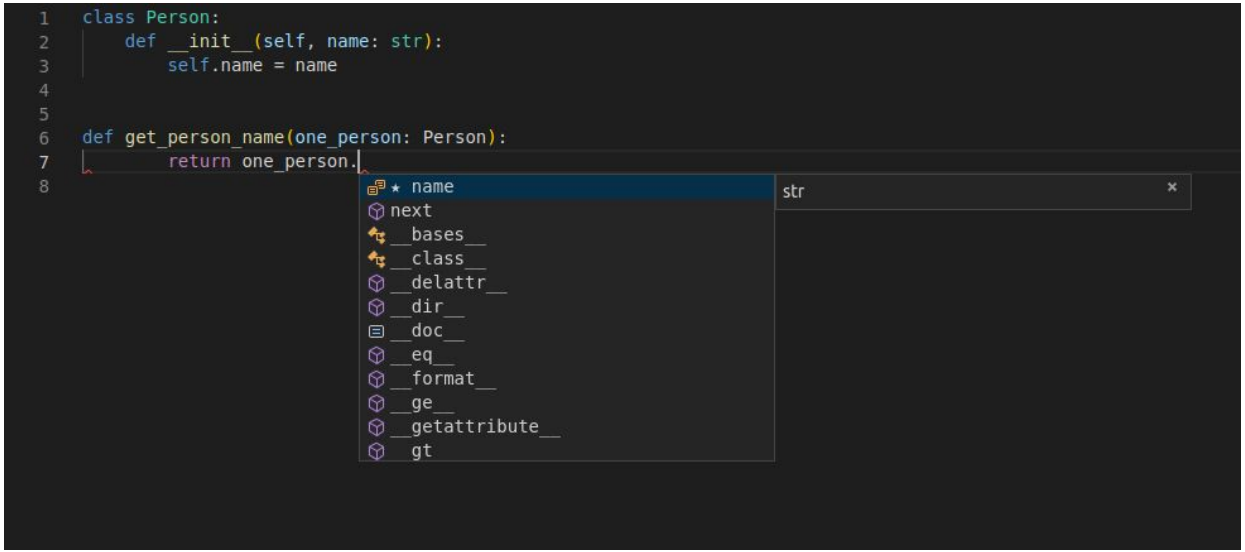
Type hints?

Tipos anidados



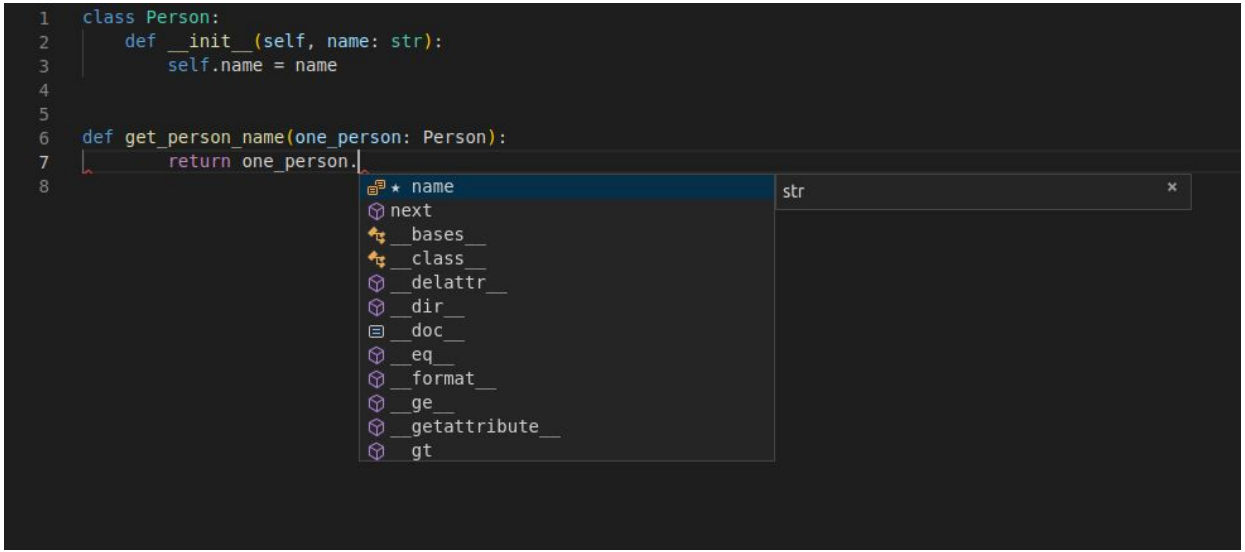
Type hints?

Clases como tipos



Type hints?

Clases como tipos



Pydantic

```python
class Person:
    def __init__(self, name: str):
        self.name = name


def get_person_name(one_person: Person):
    return one_person.name
```

1. Validará los valores
2. Los convertirá al tipo apropiado (si ese es el caso)
3. Te dará un objeto con todos los datos

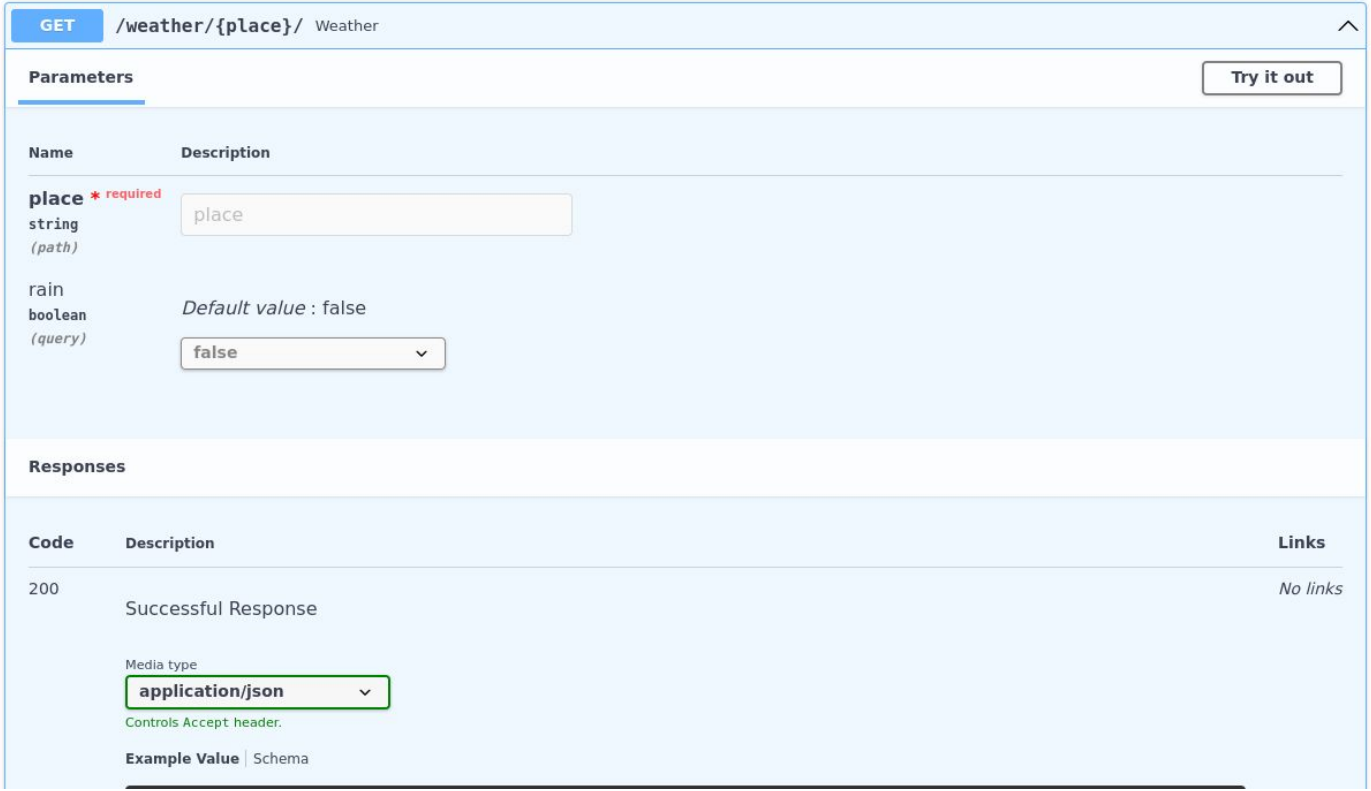
Aplicación básica

```python
from fastapi import Depends, FastAPI,

app = FastAPI()

@app.get("/weather/{place}/", tags=["Basic usage"])
def weather(place: str, rain: bool = False):
    if rain:
        return {"Hello": f"Hello {place}, today is a rainy day"}
    return {"Hello": f"Hello {place}, today is a sunny day"}
```

Documentación Automática



Documentación Personalizada

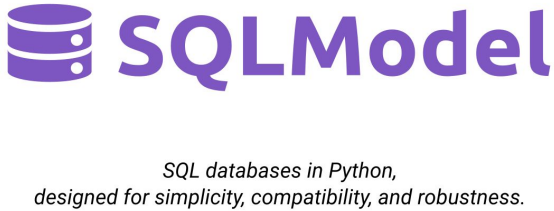
```python
@app.get("/custom", tags=["Docs"])
def custom_docs(days: int = Query(default=1, gt=0, le=10, description="Number of days")):
    """
    ### Return the following info:

    - **message**: a message to say hello world

    ### This allow md
    - *Line 1*: Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    - **Line 2**: sed do eiusmod tempor incididunt ut labore et dolore
    - *Line 3*: magna aliqua. Ut enim ad minim veniam.
    """
    return {"message": "Hello World"}
```

Bases de datos

Compatibilidad con múltiples ORMs



Async

Se pueden declarar funciones asíncronas

```
@app.get("/waiting/", tags=['Async'])
async def get_resource():
    async with httpx.AsyncClient() as client:
        response = await client.get(f"https://httpstat.us/201?sleep=5000" ,
timeout=None)
        print("Done")
        return {'status_code': response.status_code}
```

Async

Se pueden declarar funciones asíncronas

```python
@app.get("/sleep", tags=['Async'])
async def sleep_response(ms: int):
    urls = [
        f"https://httpstat.us/200?sleep={ms}",
        f"https://httpstat.us/200?sleep={ms}",
        f"https://httpstat.us/200?sleep={ms}"
    ]
    tasks = [make_request(url) for url in urls]
    responses = await asyncio.gather(*tasks)
    for response in responses:
        if response.status_code != 200:
            raise HTTPException(status_code=response.status_code, detail=response.text)
    return {'status_code': response.status_code}
```

WebSockets

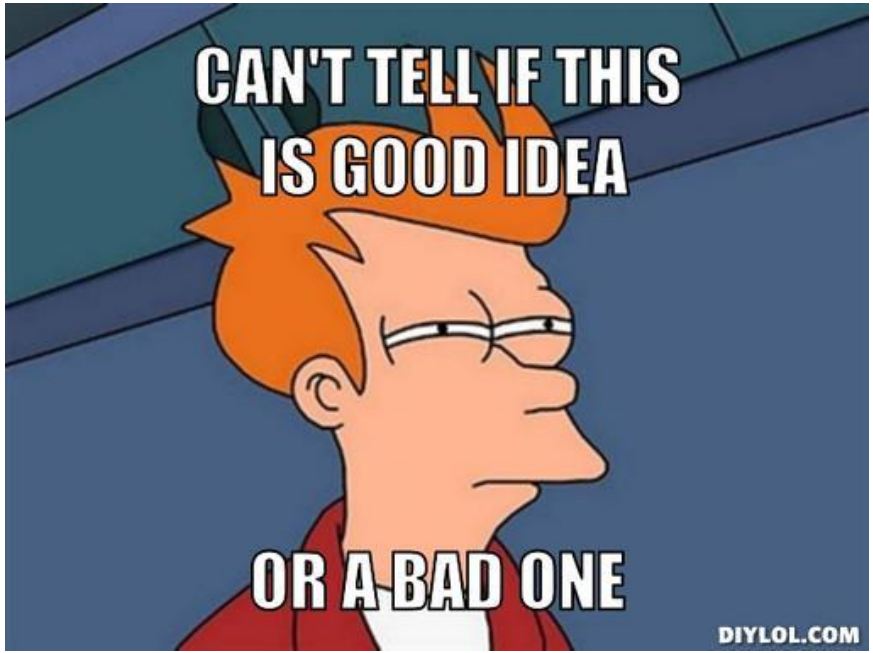
Con soporte para websockets

```python
@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    while True:
        data = await websocket.receive_text()
        await websocket.send_text(f"Message text was: {data}")
```

Tareas en segundo plano

Se ejecutan después de la respuesta.

```python
@app.get("/background/{email}")
async def background(email: str, background_tasks: BackgroundTasks):
    background_tasks.add_task(write_notification, email, message="some notification")
    return {"message": "Notification sent in the background" }
```



¿Cuándo no es buena idea?

- App estable escrita en otro framework
- No necesita nuevas características o no hace uso de ellas
- No es una API

Gracias