História, Design e Futuro

Há algum tempo, um usuário **FastAPI** perguntou:

> Qual é a história desse projeto? Parece que surgiu do nada e se tornou incrível em poucas semanas [...]

Aqui está um pouco dessa história.

Alternativas

Eu tenho criado APIs com requisitos complexos por vários anos (Aprendizado de Máquina, sistemas distribuídos, tarefas assíncronas, banco de dados NoSQL etc.), liderando vários times de desenvolvedores.

Como parte disso, eu precisava investigar, testar e usar muitas alternativas.

A história do **FastAPI** é, em grande parte, a história de seus predecessores.

Como dito na seção Alternativas:

FastAPI não existiria se não pelo trabalho anterior de outros.

Há muitas ferramentas criadas antes que ajudaram a inspirar sua criação.

Eu estive evitando a criação de um novo *framework* por vários anos. Primeiro tentei resolver todas as funcionalidades cobertas por **FastAPI** usando muitos *frameworks*, *plug-ins* e ferramentas diferentes.

Mas em algum ponto, não havia outra opção senão criar algo que oferecia todas as funcionalidades, aproveitando as melhores ideias de ferramentas anteriores, e combinando-as da melhor maneira possível, usando funcionalidades da linguagem que nem estavam disponíveis antes (*type hints* do Python 3.6+).

Investigação

Ao usar todas as alternativas anteriores, eu tive a chance de aprender com todas elas, aproveitar ideias e combiná-las da melhor maneira que encontrei para mim e para os times de desenvolvedores com os quais trabalhava.

Por exemplo, estava claro que idealmente ele deveria ser baseado nos *type hints* padrões do Python.

Também, a melhor abordagem era usar padrões já existentes.

Então, antes mesmo de começar a codificar o **FastAPI**, eu investi vários meses estudando as especificações do OpenAPI, JSON Schema, OAuth2 etc. Entendendo suas relações, sobreposições e diferenças.

Design

Eu então dediquei algum tempo projetando a "API" de desenvolvimento que eu queria como usuário (como um desenvolvedor usando o FastAPI).

Eu testei várias ideias nos editores Python mais populares: PyCharm, VS Code, e editores baseados no Jedi.

Pela última Pesquisa do Desenvolvedor Python, isso cobre cerca de 80% dos usuários.

Isso significa que o **FastAPI** foi testado especificamente com os editores usados por 80% dos desenvolvedores Python. Como a maioria dos outros editores tendem a funcionar de forma similar, todos os seus benefícios devem funcionar para virtualmente todos os editores.

Dessa forma eu pude encontrar a melhor maneira de reduzir duplicação de código o máximo possível, ter completação de texto em todos os lugares, conferência de tipos e erros etc.

Tudo de uma forma que oferecesse a melhor experiência de desenvolvimento para todos os desenvolvedores.

Requisitos

Após testar várias alternativas, eu decidi que usaria o **Pydantic** por suas vantagens.

Então eu contribuí com ele, para deixá-lo completamente de acordo com o JSON Schema, para dar suporte a diferentes maneiras de definir declarações de restrições, e melhorar o suporte a editores (conferências de tipos, auto completações) baseado nos testes em vários editores.

Durante o desenvolvimento, eu também contribuí com o **Starlette**, outro requisito chave.

Desenvolvimento

Quando comecei a criar o **FastAPI** de fato, a maior parte das peças já estavam encaixadas, o design estava definido, os requisitos e ferramentas já estavam prontos, e o conhecimento sobre os padrões e especificações estavam claros e frescos.

Futuro

Nesse ponto, já está claro que o **FastAPI** com suas ideias está sendo útil para muitas pessoas.

Ele foi escolhido sobre outras alternativas anteriores por se adequar melhor em muitos casos.

Muitos desenvolvedores e times já dependem do **FastAPI** para seus projetos (incluindo eu e meu time).

Mas ainda há muitas melhorias e funcionalidades a vir.

FastAPI tem um grande futuro à frente.

E sua ajuda é muito bem-vinda.