History, Design and Future

Some time ago, a **FastAPI** user asked:

> What's the history of this project? It seems to have come from nowhere to awesome in a few weeks [...]

Here's a little bit of that history.

Alternatives

I have been creating APIs with complex requirements for several years (Machine Learning, distributed systems, asynchronous jobs, NoSQL databases, etc), leading several teams of developers.

As part of that, I needed to investigate, test and use many alternatives.

The history of **FastAPI** is in great part the history of its predecessors.

As said in the section Alternatives:

FastAPI wouldn't exist if not for the previous work of others.

There have been many tools created before that have helped inspire its creation.

I have been avoiding the creation of a new framework for several years. First I tried to solve all the features covered by **FastAPI** using many different frameworks, plug-ins, and tools.

But at some point, there was no other option than creating something that provided all these features, taking the best ideas from previous tools, and combining them in the best way possible, using language features that weren't even available before (Python 3.6+ type hints).

Investigation

By using all the previous alternatives I had the chance to learn from all of them, take ideas, and combine them in the best way I could find for myself and the teams of developers I have worked with.

For example, it was clear that ideally it should be based on standard Python type hints.

Also, the best approach was to use already existing standards.

So, before even starting to code **FastAPI**, I spent several months studying the specs for OpenAPI, JSON Schema, OAuth2, etc. Understanding their relationship, overlap, and differences.

Design

Then I spent some time designing the developer "API" I wanted to have as a user (as a developer using FastAPI).

I tested several ideas in the most popular Python editors: PyCharm, VS Code, Jedi based editors.

By the last Python Developer Survey, that covers about 80% of the users.

It means that **FastAPI** was specifically tested with the editors used by 80% of the Python developers. And as most of the other editors tend to work similarly, all its benefits should work for virtually all editors.

That way I could find the best ways to reduce code duplication as much as possible, to have completion everywhere, type and error checks, etc.

All in a way that provided the best development experience for all the developers.

Requirements

After testing several alternatives, I decided that I was going to use **Pydantic** for its advantages.

Then I contributed to it, to make it fully compliant with JSON Schema, to support different ways to define constraint declarations, and to improve editor support (type checks, autocompletion) based on the tests in several editors.

During the development, I also contributed to **Starlette**, the other key requirement.

Development

By the time I started creating **FastAPI** itself, most of the pieces were already in place, the design was defined, the requirements and tools were ready, and the knowledge about the standards and specifications was clear and fresh.

Future

By this point, it's already clear that **FastAPI** with its ideas is being useful for many people.

It is being chosen over previous alternatives for suiting many use cases better.

Many developers and teams already depend on **FastAPI** for their projects (including me and my team).

But still, there are many improvements and features to come.

FastAPI has a great future ahead.

And your help is greatly appreciated.