歴史、設計、そしてこれから

少し前に、**FastAPI** のユーザーに以下の様に尋ねられました:

> *このプロジェクトの歴史は？何もないところから、数週間ですごいものができているようです。 [...]*

これがその歴史のほんの一部です。

代替手段

数年前から、私は複雑な要件を持つAPI (機械学習、分散システム、非同期ジョブ、NoSQLデータベースなど) を作成しており、いくつかの開発者チームを率いています。

その一環で、多くの方法を調査し、テストし、利用する必要がありました。

FastAPI の歴史は、その前身の歴史が大部分を占めています。

代替ツールから受けたインスピレーションと比較{.internal-link target=_blank}のセクションでこう述べています:

> ***FastAPI**は、代替ツールのこれまでの働きがなければ存在しなかったでしょう。*
>
> *以前に作られた多くのツールが、作成における刺激として役立ってきました。*
>
> *私は数年前から新しいフレームワークの作成を避けてきました。まず、**FastAPI**でカバーされているすべての機能を、さまざまなフレームワーク、プラグイン、ツールを使って解決しようとしました。*
>
> *しかし、その時点では、これらの機能をすべて提供し、以前のツールから優れたアイデアを取り入れ、可能な限り最高の方法でそれらを組み合わせ、それまで利用できなかった言語機能 (Python 3.6以降の型ヒント) を利用したものを作る以外に選択肢はありませんでした。*

調査

すべて既存の代替手段を使うことで、そのすべてを学び、アイデアを得て、自分や一緒に仕事をしてきた開発者のチームにとって最良の方法で組み合わせる機会を得ました。

たとえば、理想的にはPythonの標準的な型ヒントに基づくべきであることが明らかになりました。

また、すでにある規格を利用するのがベストな方法でした。

そこで、**FastAPI**のコードを書き始める前に、OpenAPI、JSON Schema、OAuth2などの仕様を数ヶ月かけて勉強し、それらの関係、重複する箇所、相違点を理解しました。

設計

その後、 (FastAPIを使う開発者として) ユーザーが欲しい「API」の設計に時間を費やしました。

もっとも人気のあるPythonエディターでいくつかのアイデアをテストしました。PyCharm、VS Code、Jediベースのエディターです。

最新の Python開発者調査で、それらのエディターがユーザーの80%をカバーしていました。

これは、**FastAPI**がPython開発者の80%が使用しているエディターで特別にテストされたことを意味します。また、ほとんどの他のエディターも同様に動作する傾向があるため、この恩恵は事実上すべてのエディターでうけられるはずです。

そうすることで、コードの重複を可能な限り減らし、どこでも補完があるようにし、タイプチェックやエラーチェックなどを実現する最善の方法を見つけました。

すべての箇所で、すべての開発者に最高の開発体験を提供しました。

要件

いくつかの代替手法を試したあと、私は**Pydantic**の強みを利用することを決めました。

そして、JSON Schemaに完全に準拠するようにしたり、制約宣言を定義するさまざまな方法をサポートしたり、いくつかのエディターでのテストに基づいてエディターのサポート (型チェック、自動補完) を改善するために貢献しました。

開発中、もう1つの重要な鍵となる**Starlette**、にも貢献しました。

開発

私が**FastAPI**自体の作成を開始した時には、ほとんどの部分がすでに準備されており、設計が定義され、必要な条件とツールの準備ができていました。そして規格や仕様に関する知識が、明確になり、更新されていました。

これから

この時点ですでに、これらのアイデアを持った**FastAPI**が多くの人の役に立っていることは明らかです。

多くのユースケースに適しているため、既存の方法よりも選ばれています。

多くの開発者やチームが、すでに **FastAPI** にプロジェクトを依存しています (私と私のチームも含めて) 。

しかし、まだまだ多くの改善点や機能があります。

FastAPIには大きな未来が待っています。

そして、あなたの助け{.internal-link target=_blank}を大いに歓迎します。