

# 0. Topics

Thursday, May 8, 2025

11:03 PM

## **1. What is FastAPI?**

## **2. Key Features**

## **3. Architecture**

## **4. Installation**

## **5. First FastAPI App**

## **6. Comparison**

# 1. What is FastAPI?

Thursday, May 8, 2025 11:08 PM

- FastAPI is a modern, high-performance web framework for building APIs with Python based on standard Python type hints
- It is designed to make it easy and fast to build efficient, production-ready APIs, especially those involving asynchronous I/O operations and data validation
- Requires Python 3.7+



## 2. Key Features

Thursday, May 8, 2025 11:13 PM

### From the original website:

The key features are:

- **Fast:** Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic). [One of the fastest Python frameworks available.](#)
- **Fast to code:** Increase the speed to develop features by about 200% to 300%. \*
- **Fewer bugs:** Reduce about 40% of human (developer) induced errors. \*
- **Intuitive:** Great editor support. [Completion everywhere.](#) Less time debugging.
- **Easy:** Designed to be easy to use and learn. Less time reading docs.
- **Short:** Minimize code duplication. Multiple features from each parameter declaration. Fewer bugs.
- **Robust:** Get production-ready code. With automatic interactive documentation.
- **Standards-based:** Based on (and fully compatible with) the open standards for APIs: [OpenAPI \[↗\]](#) (previously known as Swagger) and [JSON Schema \[↗\]](#).

### 1. High Performance:

- One of the fastest Python web frameworks
- Comparable in speed to Node.js and Go for many API tasks

### 2. Based on Python Type Hints:

- Validate inputs automatically

### 3. Built-in Data Validation:

- Ensures that all input data is structured, typed, and clean

#### 4. Asynchronous Support:

- Designed from the ground up to support async I/O operations

#### 5. Easy Testing & Debugging:

- Works great with pytest and testing tools
- Return types are predictable and readable for debugging

#### 6. Automatic API Documentation:

- Built-in integration with Swagger UI (/docs) and ReDoc (/redoc)
- Helps both backend and frontend teams explore and test endpoints easily

#### 7. Ideal for ML & Data Science Projects:

- Wrapping ML models as APIs (/predict)
- Serving pre/post-processing logic
- Managing model versions and feature validation

### 3. Architecture

Thursday, May 8, 2025 11:28 PM

#### FastAPI is Built on Three Core Libraries:

- Starlette: ✓

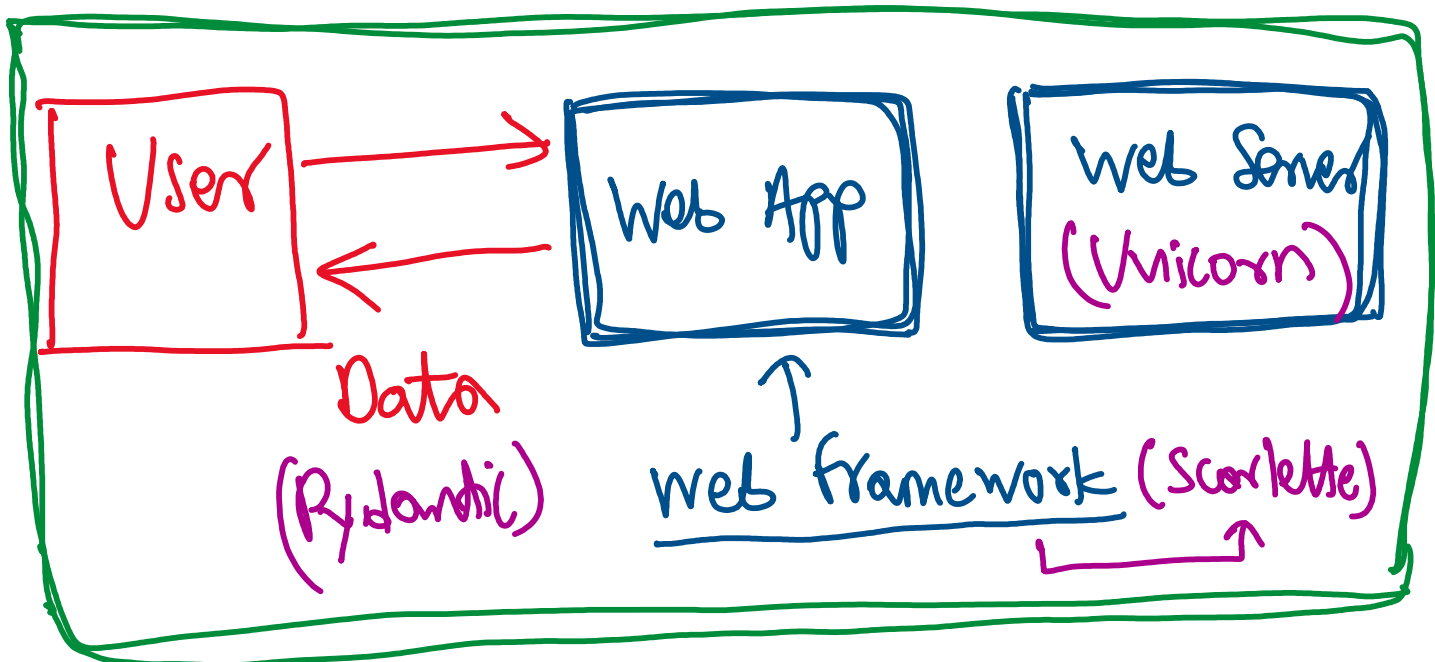
- Lightweight ASGI web framework/toolkit used as the **core web layer** in FastAPI
- Handles the registration and dispatching of HTTP routes (e.g., GET, POST)
- Enables real-time WebSocket communication

- Pydantic: ✓

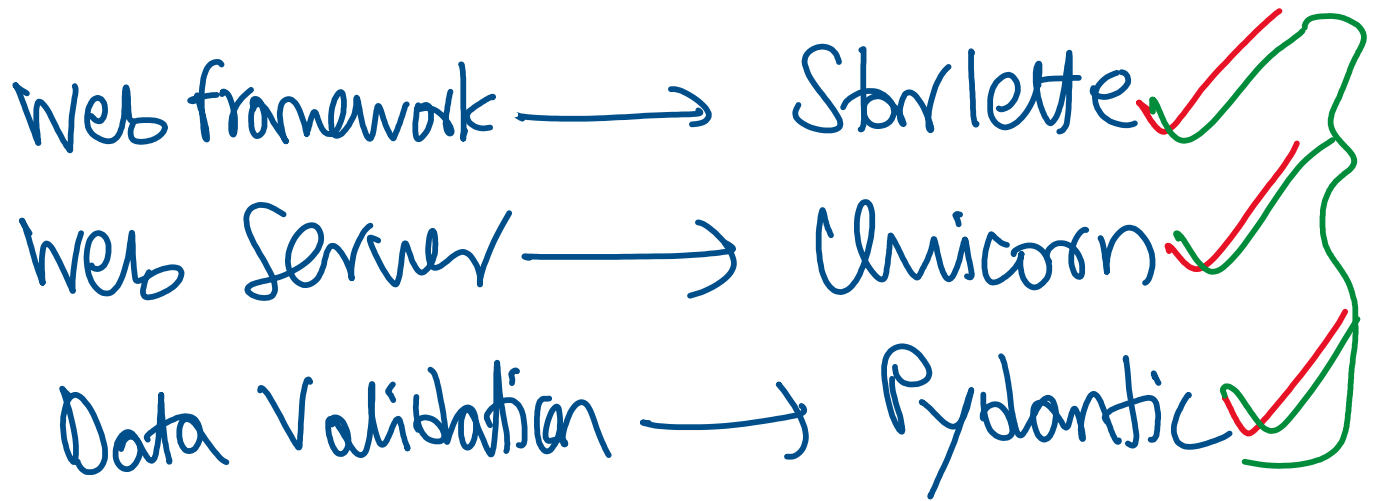
- Data parsing and validation library that leverages Python's type hints to enforce and transform data structures
- Ensures request bodies, query params, headers, and path params match expected types
- Converts data (e.g., string to int, string to datetime) before it reaches your function
- Helps auto-generate JSON Schema for documentation

- Uvicorn:

- A lightning-fast ASGI server implementation based on uvloop and httptools
- Launches FastAPI apps and handles incoming client requests
- Communicates with FastAPI via the ASGI specification
- Supports async I/O, which makes it fast and scalable



Web framework → Starlette ✓  
Web Server → Unicorn ✓  
Data Validation → Pydantic ✓

A handwritten list of three items, each consisting of a category followed by an arrow pointing to a specific tool. The categories are 'Web framework', 'Web Server', and 'Data Validation'. The tools are 'Starlette', 'Unicorn', and 'Pydantic' respectively. Each tool name is followed by a red checkmark. A large green curly bracket on the right side groups all three items together.

## 4. Installation

Friday, May 9, 2025 2:06 AM

### 1. Create/Activate a virtual environment:

### 2. Install FastAPI and Uvicorn:

- `pip install fastapi uvicorn`

## 5. First FastAPI App

Friday, May 9, 2025 2:09 AM

### 1. Write the Python script

### 2. Run the start-up command:

- **uvicorn main:app --reload**
- **Telling uvicorn:** Run the **app** object defined in the **main.py** file, and keep watching for any file changes so you can auto-reload the server

### Understanding the Start-up Command:

PART	MEANING
<b>uvicorn</b>	The command-line tool to start the Uvicorn ASGI server. It must be installed (pip install uvicorn).
<b>main</b>	Refers to the Python file named main.py (without the .py extension).
<b>app</b>	Refers to the FastAPI application instance in the main.py file. This should be defined as: app = FastAPI().
<b>--reload</b>	Enables auto-reload: the server will automatically restart when you make changes to the code. Useful in development (not recommended in production).

- **Server starts at:** <http://127.0.0.1:8000>
- **You can visit:**
  - <http://127.0.0.1:8000> → Your API endpoint
  - <http://127.0.0.1:8000/docs> → Swagger UI
  - <http://127.0.0.1:8000/redoc> → ReDoc docs



## 6. Comparison

Friday, May 9, 2025 2:24 AM

FEATURE	FastAPI	Flask	Django	Falcon
Release Year	2018	2010	2005	2013
Asynchronous Support	Native (async/await)	Limited (via extensions)	Partial (since Django 3.1)	Full
Performance	Very High (Starlette + async)	Moderate	Moderate	Very High
Type Hinting / Validation	Built-in with Pydantic	Manual	Manual or via DRF	Manual
API Documentation	Auto-generated (Swagger, ReDoc)	No (3rd-party plugins)	Yes (only with DRF)	No
ORM	None built-in (can use SQLAlchemy, Tortoise, etc.)	None built-in	Django ORM	None
Admin Interface	No	No	Built-in	No
Learning Curve	Easy (if familiar with type hints)	Very Easy	Steep (due to monolithic structure)	Medium
Best Use Cases	Modern APIs, ML apps, async microservices	Simple apps, prototyping	Large web apps, admin panels	Ultra-fast microservices, low-latency APIs
Community & Ecosystem	Growing fast	Mature, large	Very mature	Niche, smaller
Built-in Features	Light but powerful	Lightweight	Full-stack batteries-included	Very minimal
Project Structure	Flexible (modular)	Very flexible	Strict	Very flexible
Deployment Ready	(ASGI-ready)	(WSGI)	(WSGI & partial ASGI)	(ASGI/WSGI)