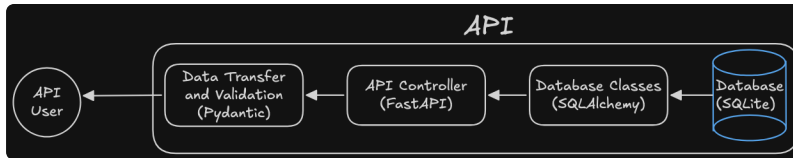


## Chapter 04 - Developing the FastAPI Code

- Endpoint Description + HTTP Verb + URL
- `v0` at the start, so consumer know that product is not that reliable, there can be some potential breaking changes



- We implemented Database + Database Classes in [Chapter 03 - Creating Your Database](#)
- We Implement the API Controller + Data Transfer in [Chapter 04 - Developing the FastAPI Code](#)
- Tools
  - [FastAPI](#): Web framework to build the API
    - FastAPI CLI: Command-line interface for FastAPI
      - `main.py`: FastAPI file that defines routes and controls API
      - `test_main.py`: The pytest file for the FastAPI program
  - [HTTPX](#): HTTP client for Python
  - [Pydantic](#): Validation Library
    - `schemas.py`: Define the Pydantic classes that validates data sent to the API
  - [Uvicorn](#): Web server to run the API

### FastAPI

FastAPI is a Python Web Framework that is designed for building APIs.

#### Web Framework

Web Framework is a set of libraries that simplify common tasks for web applications.

Other Web Frameworks:

- Express
- Flask
- Django
- Ruby on Rails

## Advantages of FastAPI

- Fast in both application performance and developer productivity by:
  - simplifying different tasks related to API building and publishing, such as:
    - It handles HTTP traffic, requests/responses, other "plumbing" jobs with a few line of code
    - It automatically generates an OpenAPI specification file for your API <sup>which</sup> → useful for integrating with other application
    - It includes interactive documentation for your API
    - It supports API versioning, security, and etc.
- FastAPI CLI is a separate python library that is used to run FastAPI from the command line
- Current version of FastAPI is 0.x which means can have breaking changes

## HTTPX

- HTTPX is a python HTTP client.
- It support asynchronous calls <sup>means</sup> → some tasks to finish while others process ≠ requests library <sup>which is</sup> → synchronous calls = wait until they receive a response before continuing
- HTTPX is used by pytest to test FastAPI programs
- You can create your Python SDK with it too

## Pydantic


- Pydantic is a data validation library
- key part in the APIs <sup>why?</sup> → Because APIs are used to communicate between systems, so a critical piece of their functionality is the validation of inputs and outputs → Programmers and Data Scientists spend a lot of time to check the data types and validate values that go into and out of the API endpoints
- Pydantic is fast because:
  - It is written in Rust programming language, so the pydantic validation code runs much faster.
  - No need to write Python validation code by hand.
- FastAPI uses Pydantic to generate JSON Schema representations from Python code.
  - JSON Schema is a standard that ensures consistency in JSON data structure.
  - This feature enables FastAPI to automatically generate the OpenAPI specification.


## Uvicorn

- Uvicorn is a web server
  - handle various administrative tasks related to handling requests and responses
- It is based on ASGI specification
  - Support both synchronous processes & asynchronous processes

## Creating Pydantic Schemas

- `schema.py` file
- Pydantic classes define the structure of the data that the consumer will receive in their API responses
- Uses a [software design pattern](#) called [Data Transfer Objects - DTO](#):
  - You define a format for transferring data between a producer and consumer, without the consumer needing to know the backend format.
  - Enable Complete flexibility.
- We convert Database objects by use of ORM tool such as SQLAlchemy, which converted them to Python Classes which we can interact with them, but our consumer will receive them in an HTTP request as a JSON object.  $\xrightarrow{\text{Problem}}$  How to convert this Python Classes to JSON objects? Pydantic → What is this process called? Serialization process: Converting the Python Objects into JSON for the API response.
- With Pydantic you don't need to manage the serialization process in your python code  $\xrightarrow{\text{result into}}$  simplifies your program + Pydantic is written in rust which means it is faster than python
- De-serialization task: Python also defines the response format in the `openapi.json` file, which is a standard contract that uses OpenAPI and JSON Schema

 **Pydantic take the data from SQLAlchemy, Convert it to JSON and deliver it to the API user**

 **Both in SQLAlchemy and Pydantic documentation they refer to their classes as models! This will be confusing for Data Scientist which think of model as machine learning model! Keep this in mind!**

`schema.py` : in this file we form the responses to the API endpoints.

- The primary schemas are directly returned to the endpoints and the secondary schemas are returned as an attribute of the primary schemas.
- `BaseModel` class from Pydantic provides a lot of built-in capabilities, including:
  - Validating the data types
  - Converting the Python object to JSON (Serializing)
  - Raising intelligent errors
  - Connecting automatically to the SQLAlchemy models

⚡ **Pydantic data types of individual class elements are assigned with a colon, and not an equals sign which is what SQLAlchemy uses.**

- Breaking data in two (multiple) schemas allows you to share a limited version of the data in some situation and a full version in others
- Why it is like this 'Teams → Player → Performance'? to reduce the amount of data transmitted in the API call
  - Team is the Primary Schema and Player is a Secondary Schema, which add additional data to team data

## Creating Your FastAPI Controller

- `main.py` : tie together all the files that you have created till now!
- Primary class in FastAPI is `FastAPI` class!
  - This class by default includes functionality to handle much of the work that an API needs to perform, without requiring you to specify every detail.
  - You just need to create `FastAPI` instance

🔗 **When running API through CLI with Uvicorn, you will reference `main:app`, referring to the app object in `main.py`**

- A decorator is a statement that is added above a function definition, to give special attributes to it
  - In this case decorator defines that these functions definitions will be a FastAPI request handlers.
- Endpoints = Routes
  - Endpoints/Routes are defined with the decorators above each function
  - All of our Endpoint at this state use the `HTTP GET` verb, which is defined by `@app.get()`

- the first parameter of the `@app.get()` is the relative URL
- `response_model`, inform FastAPI which type of data return from this endpoint
  - This information will be included in the OpenAPI specification that FastAPI automatically create.
  - Help the consumer to be more confident about the data being valid according to this definition.
- You have some named parameters in your functions such as `limit`, `skip` and etc. which FastAPI will automatically include these parameters as query parameters in the API definition.
  - Query parameters are included in the URL path with question mark (?) in front and an ampersand (&) between
    - Exp:  
URL: {base URL}/v0/players/?first\_name=Bryce&last\_name=Young
- Matching between FastAPI definition and `crud.py` definition:
  - if the match between data types in `get_player()` from `crud.py` and `read_players()` decorator definition which is `list[schemas.Player]` happens then the FastAPI uses Pydantic to serialize the Python objects into a text JSON string and sends the response to the consumer.
- Path Parameter <sup>exp.</sup> = `{player_id}`:
  - Which is an API request parameter that is included in the URL path instead of being separated like Query parameter
- `HTTPExceptions`: standard method that web applications use to communicate status.

🔗 It is a good RESTful API design to use the standard HTTP status codes to communicate with customers.

## Test API

- `test_main.py`
- `TestClient` is a special class that allow the FastAPI program to be tested without running it on a web server

## Lunch The API

```
fsatapi run main.py
```

## Additional Resources

- [Uvicorn Documentation](#)
- [Pydantic Documentation](#)
- [GitHub - Kludex/fastapi-tips: FastAPI Tips by The FastAPI Expert!](#)
- [FastAPI O'reilly Book](#)
- [Reference - FastAPI](#)
- [FastAPI](#)