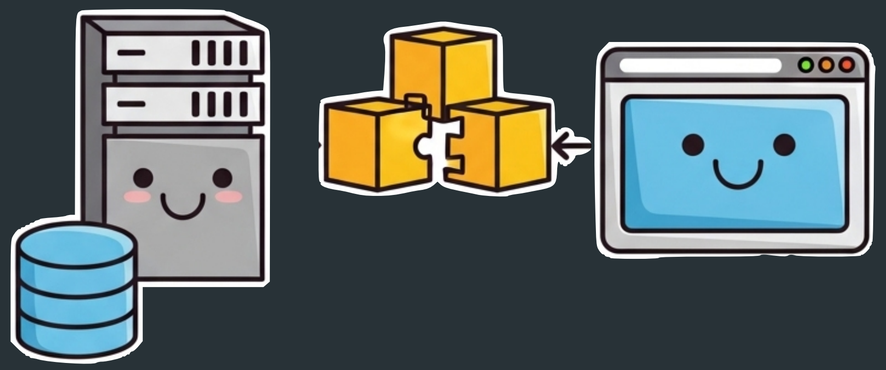
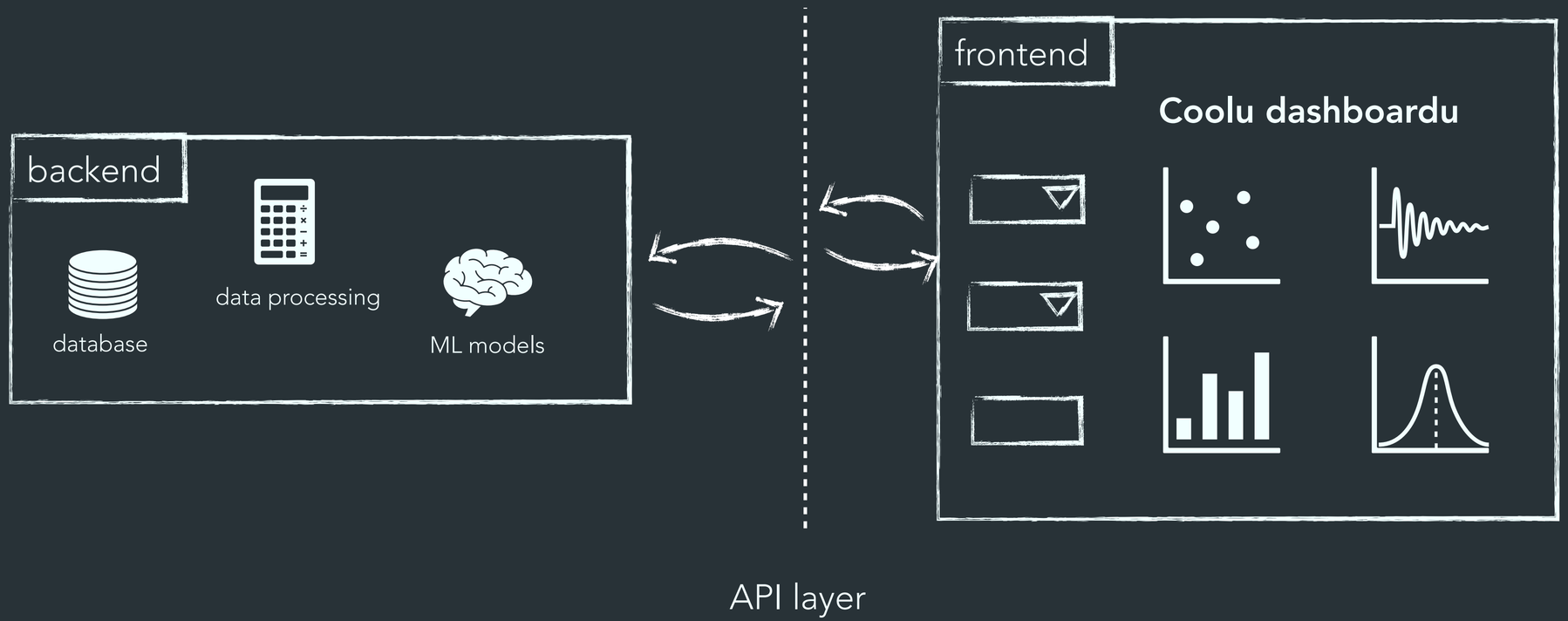


kokchun giang

fastapi - a python library to create APIs really simple and fast

web **API** is a layer or an interface to facilitate communication between frontend and backend



CRUD is a simple pattern for building data APIs

POST	**C**reate	adds new data
GET	**R**ead	get the data
PUT	**U**pdate	change an existing data
DELETE	**D**elete	remove data

a look into **FastAPI**

creates an API application

api.py

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/books")
async def read_books():
    return books
```

data returned at this endpoint

creating an endpoint /books

in terminal run this command

```
uvicorn api:app --reload
```

navigate to http://127.0.0.1:8000 in your browser

http://127.0.0.1:8000/books to see the data

create data with **post request**

makes this endpoint
a post request

```python
@app.post("/books/create_book")
async def create_book(book_request: Book):
    new_book = Book.model_validate(book_request)
    books.append(new_book)

    return new_book
```

a pydantic model

good to return the
object that is posted

request body is a json object

```json
{
  "id": 11,
  "title": "Learn with AIgineer",
  "author": "Kokchun Giang",
  "year": 2025
}
```

update data with **put request**

makes this endpoint
a put request

```python
@app.put("/books/update_book")
async def update_book(updated_book: Book):
    for i, book in enumerate(books):
        if book.id == updated_book.id:
            books[i] = updated_book
    return updated_book
```

good to return the
object that is put

request body is a json object

```json
{
  "id": 11,
  "title": "Learn with AIgineer",
  "author": "Kokchun Giang",
  "year": 2025
}
```

delete request to remove data

```python
@app.delete("/books/delete_book/{id}")
async def delete_book(id: int):
    for i, book in enumerate(books):
        if book.id == id:
            del books[i]
            break
```

path parameter -
value in endpoint used
in the function

delete book if id in
path parameter
matches book id

example endpoint
/books/delete_book/5

deletes book
with id 5

query parameter for key-value pairs in the url

```python
@app.get("/books/")
async def filter_books(
    start_year: int = Query(
        1950,
        gt=1500,
        lt=CURRENT_YEAR + 1,
        description="Filters books that are newer than this year"
    ),
    author: str = Query(None, description="Authors firstname and lastname "),
):
    filtered_books = [book for book in books if start_year < book.year]

    if author:
        filtered_books = [
            book
            for book in filtered_books
            if author.casefold() == book.author.casefold()
        ]

    return filtered_books
```

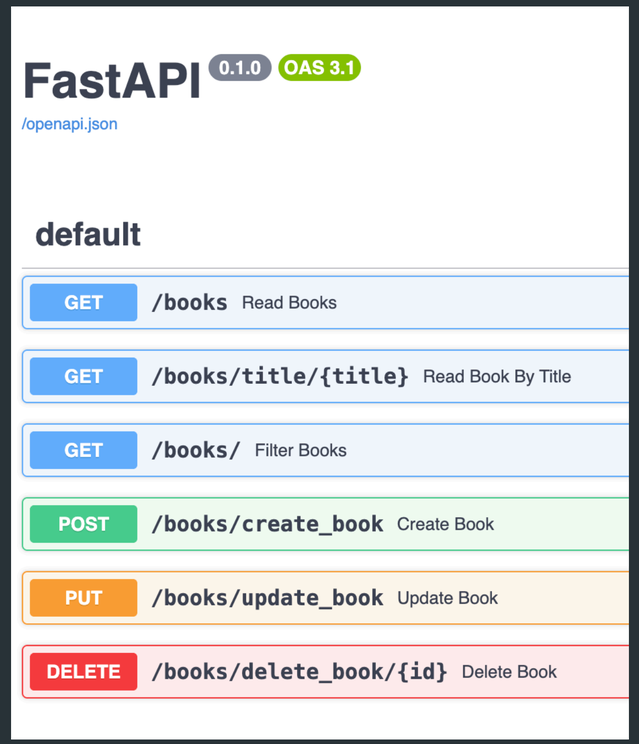
query parameter with validation and description

filters data to be returned based on query parameters

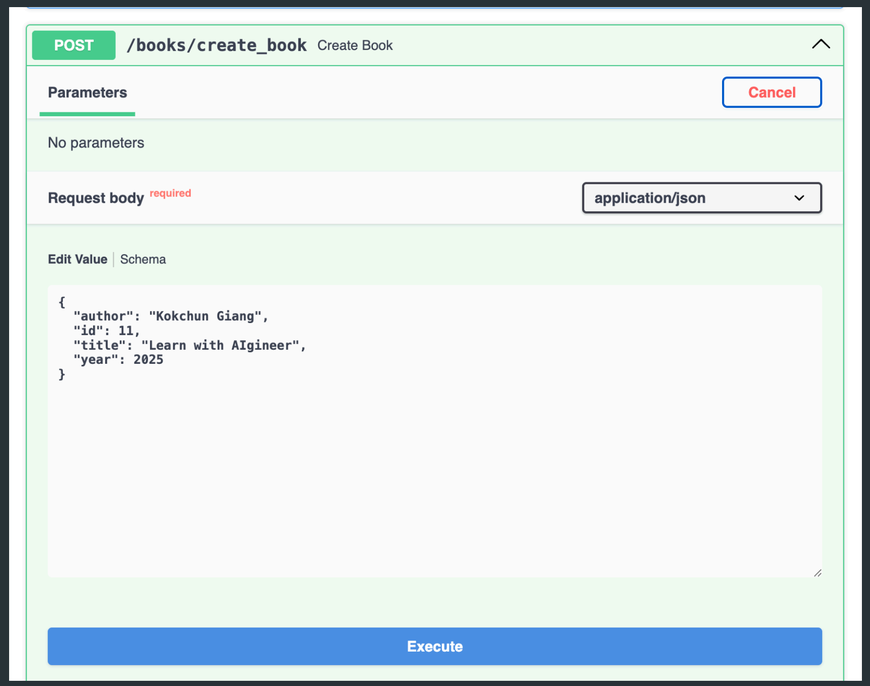
http://127.0.0.1:8000/books/?start_year=1999&author=kokchun%20giang

swagger ui to interact with your API

http://127.0.0.1:8000/docs



all endpoints



try out an endpoint