

⚡ PYTHON PIZZA ISTANBUL · 14 NOV 2026

A Slice of FastAPI

In 10 minutes, still hot.



Erhan BÜTE
erhan.dev



MENU

Today's menu

STARTERS

What is FastAPI? 0:50

Margherita: Hello World 0:50

DESSERT

Kitchen layout: structure 0:50

Taste first: tests 0:40

Serve it hot: deploy 0:30

MAIN COURSE

Four Cheese: parameters 1:00

The Pineapple Test: Pydantic 1:00

The menu writes itself: /docs 0:50

Depends: your sous-chef 1:00

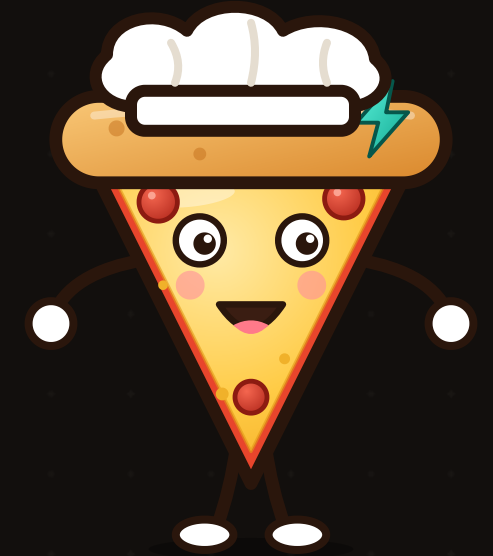
async: one waiter, many tables 1:10

Bill: 10 minutes

Tip: ★ the repo

🚫🍍 Pineapple: none

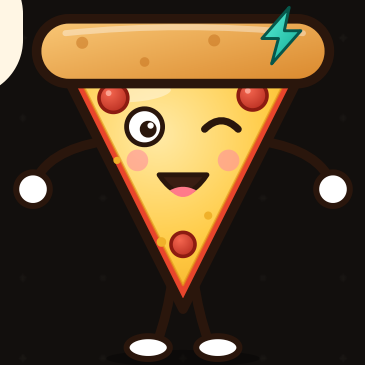
Anyone hungry?



FastAPI's recipe

A modern, fast, type-hint-driven Python web framework.

That's the whole recipe.



OpenAPI **THE TOPPINGS**
automatic documentation

Type hints **THE CHEESE**
binds it all together

Pydantic **THE SAUCE**
validation & serialization

Starlette **THE DOUGH**
ASGI, routing, async

= ⚡ FastAPI

Margherita: the classic

Three lines, one working API.

examples/01_hello.py

main.py

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
async def root() -> dict[str, str]:
    return {"message": "Merhaba Python Pizza! 🍕"}
```

3 lines

0 config

reloads by itself

terminal

```
$ uv run fastapi dev examples/01_hello.py
⚡ Starting FastAPI in development mode
🌐 Server started at http://127.0.0.1:8000
Documentation at http://127.0.0.1:8000/docs
```

127.0.0.1:8000

```
{
  "message": "Merhaba Python Pizza! 🍕"
}
```

Seriously, three lines.



Four Cheese

Where does the data come from? One function, four sources.

examples/02_params.py

```
@app.post("/pizzas/{pizza_id}/orders")
async def order_pizza(
    pizza_id: int, # path
    order: Order, # body
    extra_cheese: bool = False, # query
    x_table_no: Annotated[int | None, Header()] = None, # header
) -> dict[str, object]:
    return {
        "pizza_id": pizza_id,
        "extra_cheese": extra_cheese,
        "table": x_table_no,
        "order": order,
    }
```

⊕ **Path** in the URL

/pizzas/3/orders

→ pizza_id = 3

⊕ **Query** after the ?

?extra_cheese=true

→ extra_cheese = True

⊕ **Header** in the headers

X-Table-No: 7

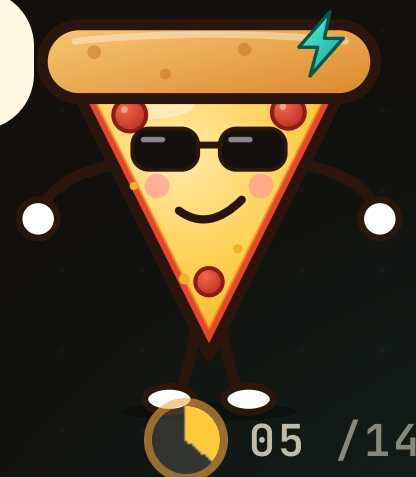
→ x_table_no = 7

⊕ **Body** in the JSON

{"customer": "Ayşe", "quantity": 2}

→ order = Order(...)

All from type hints.



422 pizza_id: int → a request to /pizzas/abc is rejected before your function runs.

The Pineapple Test

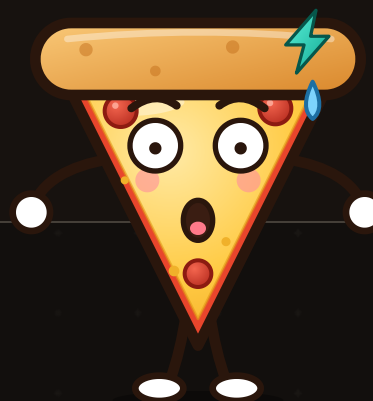
We wrote no validation code. Pineapple was still rejected.



examples/03_models.py

```
class Topping(StrEnum):
    mozzarella = "mozzarella"
    pepperoni = "pepperoni"
    mushroom = "mushroom"
    basil = "basil" # pineapple? not on this menu!
```

```
class PizzaCreate(BaseModel):
    name: str = Field(min_length=2, max_length=40)
    price: int = Field(gt=0)
    toppings: list[Topping] = Field(min_length=1)
```



PINEAPPLE?!

curl · Swagger UI · Postman...

POST /pizzas

PINEAPPLE REQUEST

```
{"name": "Hawaii", "price": 200, "toppings":
["mozzarella", "ananas"]}
```

422 Unprocessable Content

```
{
  "detail": [{
    "loc": ["body", "toppings", 1],
    "msg": "Input should be 'mozzarella', 'pepperoni',
'mushroom' or 'basil'",
    "input": "ananas"
  }]
}
```

201 Pizza is cooking

404 No such pizza

422 Pineapple detected!

418 You wanted tea, I'm a teapot ☕

The menu writes itself

Without writing a single line of docs.

/docs

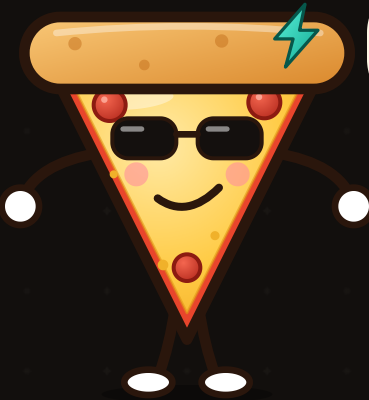
Swagger UI

/redoc

ReDoc

/openapi.json

machine-readable schema



I didn't write it, I swear.

127.0.0.1:8000/docs

Pizza API 🍕 0.1.0 OAS 3.1

[/openapi.json](#)

Order a pizza over HTTP 🍕 The companion API of the talk **A Slice of FastAPI** (Python Pizza Istanbul 2026).

- **Browse the menu and order a pizza:** no key needed.
- **Changing the menu** (or moving an order along) is chef-only: press **Authorize** and send the chef's key in the **X-API-Key** header.
- Pineapple is not a topping here. Try it. 🍍

[Erhan BÜTE - Website](#)
MIT

Pizzas

The menu. Reading is free, changing it is chef-only.

GET /api/v1/pizzas Browse the menu

POST /api/v1/pizzas Add a pizza to the menu

GET /api/v1/pizzas/{pizza_id} Get one pizza

POST /api/v1/pizzas · Try it out

Request body required

application/json

Edit Value | Schema

```
{
  "name": "Hawaii",
  "price": 200,
  "toppings": ["mozzarella", "ananas"]
}
```

Execute

Clear

Responses

Server response

Code	Details
422	Error: Unprocessable Content

Response body

```
{
  "detail": [
    {
      "type": "enum",
      "loc": [
        "body",
        "toppings",
        1
      ],
      "msg": "Input should be 'mozzarella', 'tomato', 'pepperoni', 'sucuk', 'mushroom', 'olive', 'pepper' or 'basil'",
      "input": "ananas",
      "ctx": {
        "expected": "'mozzarella', 'tomato', 'pepperoni', 'sucuk', 'mushroom', 'olive', 'pepper' or 'basil'"
      }
    }
  ]
}
```

Download

🍕 Python Pizza Istanbul 2026

07 / 14

Depends: your sous-chef

The chef just makes pizza. The kitchen brings the ingredients.

examples/04_depends.py

```
def pagination(limit: int = 10, offset: int = 0) -> dict[str, int]:  
    return {"limit": limit, "offset": offset}
```

```
Pagination = Annotated[dict[str, int], Depends(pagination)]
```

```
@app.get("/menu")
```

```
async def menu(page: Pagination): ...
```

```
def require_chef(x_api_key: Annotated[str | None, Header()] = None):  
    if x_api_key != "chef-secret":  
        raise HTTPException(401, "Only the chef can do that 🧑🍳")
```

```
@app.delete("/menu/{name}", dependencies=[Depends(require_chef)])
```

```
async def remove(name: str): ...
```



pagination()

limit & offset



require_chef()

authorization



get_db()

database session



route function

@app.get · @app.delete

app/core/db.py

```
async def get_db() -> AsyncIterator[AsyncSession]:  
    async with SessionLocal() as session:  
        yield session
```

yield = cleanup when done. FastAPI does the dishes 🍳

Ingredients ready, chef!



async def: one waiter, many tables

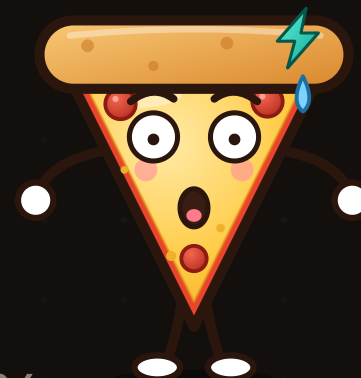
Three orders at once. How many waiters do you need?

examples/05_async.py

```
@app.get("/bake/async")
async def bake_async():
    await asyncio.sleep(BAKE_SECONDS) # waiter serves other tables ✓

@app.get("/bake/blocking")
async def bake_blocking():
    time.sleep(BAKE_SECONDS) # waiter freezes at the oven ✗

@app.get("/bake/threadpool")
def bake_threadpool():
    time.sleep(BAKE_SECONDS) # `def` runs in a thread pool ✓
```



I'm the only waiter 🥲

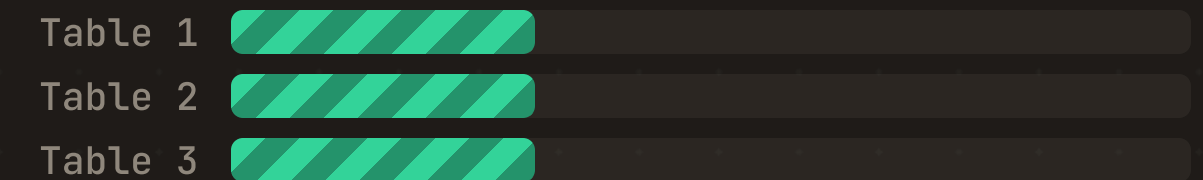
The waiter stands at the oven

6.05 s



The waiter serves other tables while the oven works

2.05 s



🕒 Measured for real: 3 concurrent requests

await serve other tables while the oven works

def FastAPI runs you in a thread pool

async def + time.sleep() the waiter freezes at the oven

A growing restaurant: kitchen layout

Requests flow from top to bottom.

fastapi-pizza/

├─ app/

│ ├── main.py

│ ├── api/

│ ├── schemas/

│ ├── services/

│ ├── crud/

│ ├── models/

│ └─ core/

└─ examples/

└─ tests/

 create_app()

 Waiter

 Menu & receipt

 Chef

 Pantry

 Recipe book

 Equipment

lifespan

routes + dependencies

Pydantic models

business rules

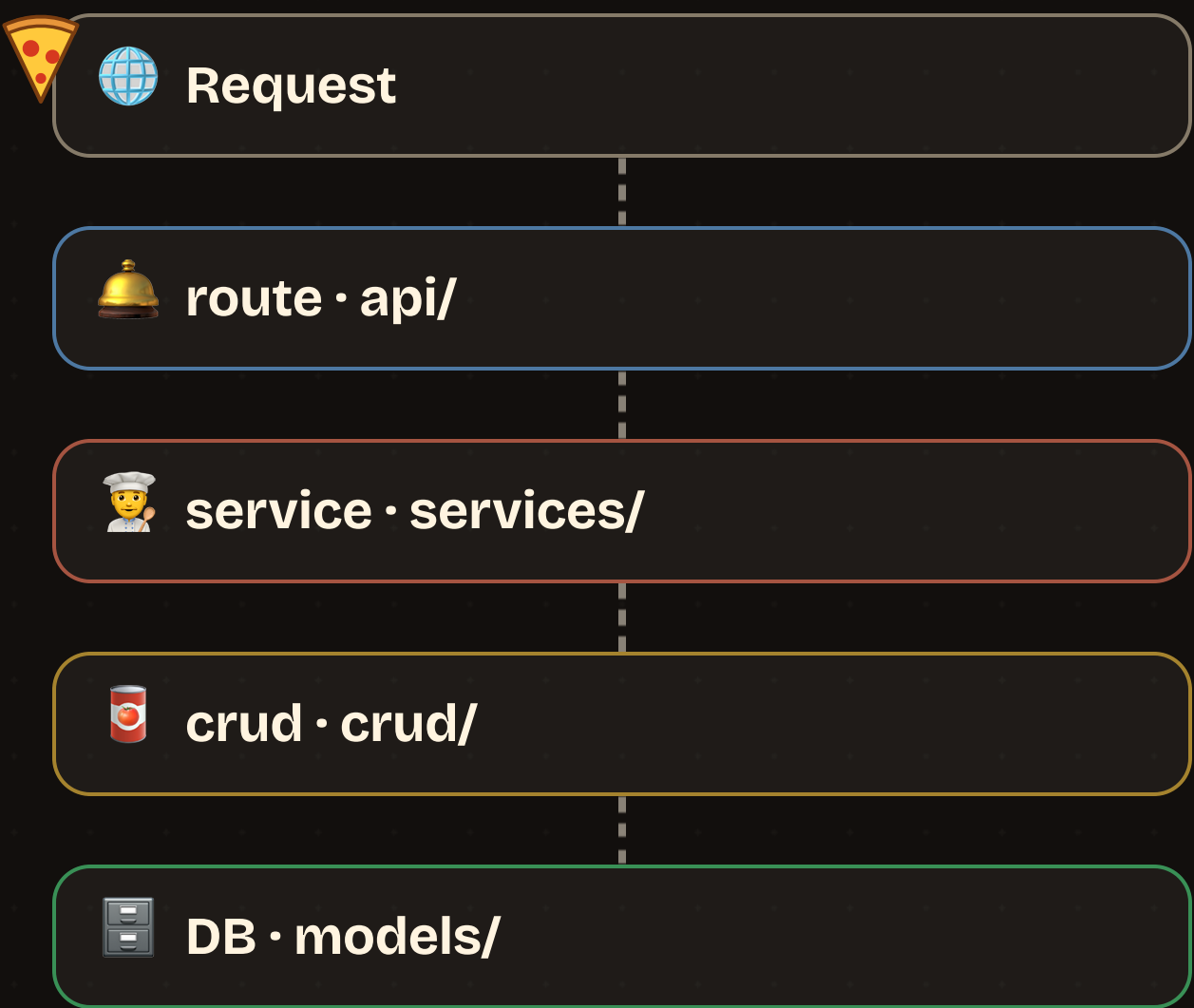
database queries

tables

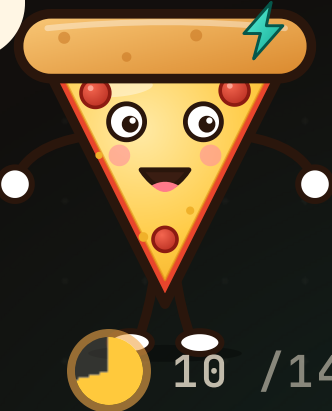
settings, DB, errors

the code on these slides

taste first



A place for everything.



Taste it first

TestClient: no server needed, done in seconds.

tests/test_pizzas.py

pytest

```
def test_pineapple_is_rejected(client: TestClient, chef: dict[str, str]) -> None:
    body = {"name": "Hawaii", "price": 200, "toppings": ["mozzarella", "ananas"]}
    response = client.post(URL, json=body, headers=chef)
    assert response.status_code == 422
    error = response.json()["detail"][0]
    assert error["loc"] == ["body", "toppings", 1]
    assert error["input"] == "ananas"
    assert client.get(URL).json() == [] # nothing was created
```

Mmm... that 422 is spot on.

terminal

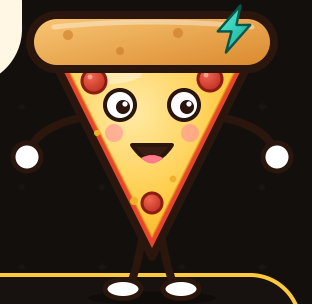
```
$ uv run pytest -v
tests/test_pizzas.py::test_pizza_id_must_be_an_int PASSED [ 65%]
tests/test_pizzas.py::test_pineapple_is_rejected PASSED [ 70%]
===== 44 passed in 2.07s =====
```



Serve it hot

From local to live in three commands.

Piping hot!



\$ fastapi dev

DEVELOPMENT

auto-reload

\$ fastapi run

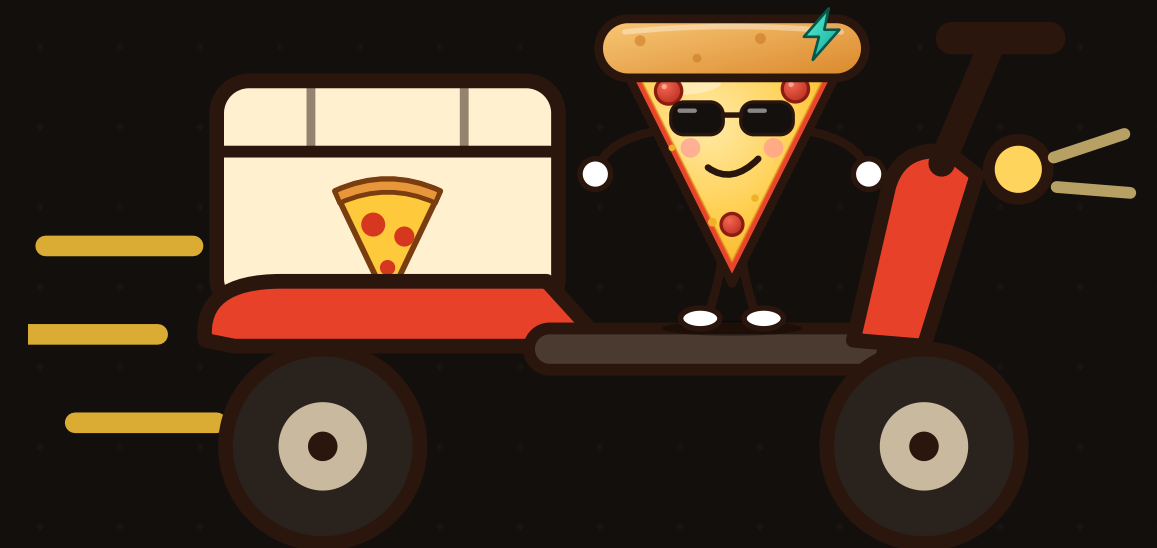
PRODUCTION

one command, ready server

\$ fastapi deploy

FASTAPI CLOUD

live in one command



Docker or plain uvicorn on your own server work too.

The check, please!



PIZZA API • RECEIPT

14 November 2026 • StartGate, Maslak, Istanbul

Type hints validation + docs
4 × cheese path • query • header • body
Pydantic pineapple rejection
/docs the menu writes itself
Depends the kitchen brings ingredients
async one waiter, many tables
TestClient taste first

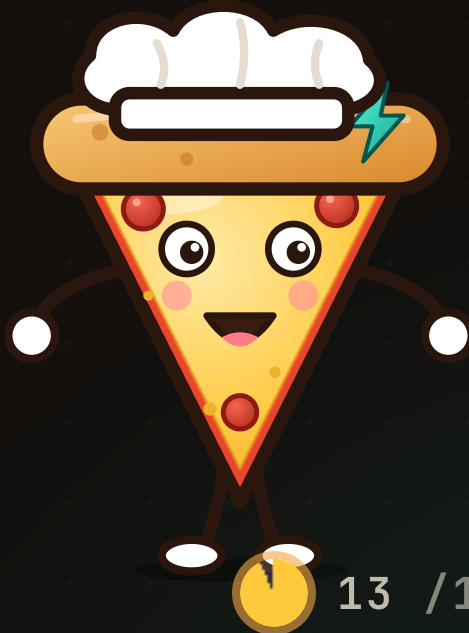
TOTAL **10 min**

Tip: ★ the repo



AFİYET OLSUN

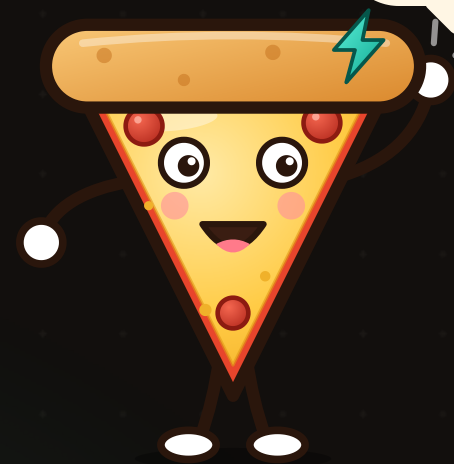
It's on me.



Afiyet olsun!

Questions? (No pineapple.)

Thank you! 🙌



 erhan.dev

 github.com/erhan

 python-pizza.erhan.dev



Slides

python-pizza.erhan.dev/en



coming soon

Example code

github.com/erhan/fastapi-pizza