

⚡ PYTHON PIZZA İSTANBUL • 14 KASIM 2026

Bir Dilim FastAPI

10 dakikada, soğumadan.



Erhan BÜTE

erhan.dev



MENÜ

Bugünün menüsü

ANTİPASTİ

FastAPI nedir? 0:50

Margherita: Hello World 0:50

TATLI

Mutfak düzeni: proje yapısı 0:50

Önce sen tat: testler 0:40

Sıcak servis: yayına al 0:30

ANA YEMEK

Dört Peynirli: parametreler 1:00

Ananas Testi: Pydantic 1:00

Menüyü kendisi yazıyor: /docs 0:50

Depends: mutfak yardımcısı 1:00

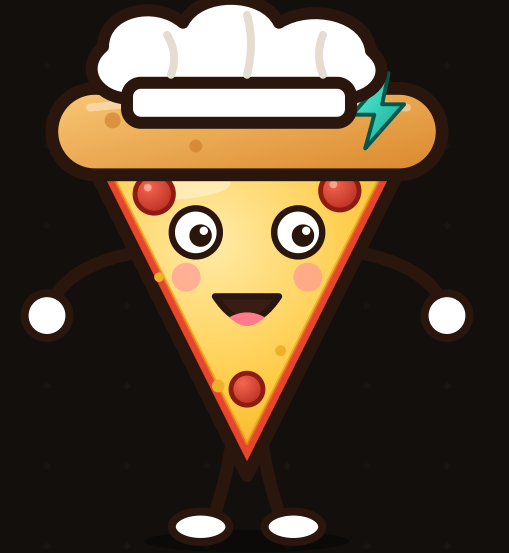
async: bir garson, çok masa 1:10

Hesap: 10 dakika

Bahşış: repoya ★

🚫🍍 Ananas: yok

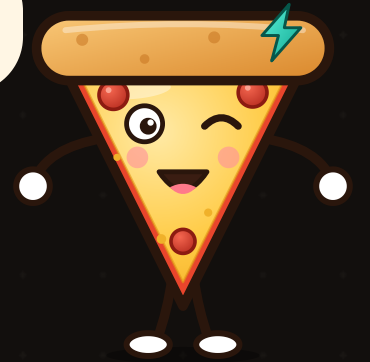
Acıkan var mı?



Tarif: FastAPI nedir?

Modern, hızlı, tip ipuçlarına dayalı bir Python web framework'ü.

Tarif bu kadar.



OpenAPI **ÜSTÜ**
otomatik dokümantasyon

Type hints **PEYNİR**
hepsini birbirine bağlayan

Pydantic **SOS**
veri doğrulama ve serileştirme

Starlette **HAMUR**
ASGI, routing, async

= ⚡ **FastAPI**

Margherita: en sade hali

Üç satır, çalışan bir 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 satır

0 yapılandırma

kendiliğinden yeniden yükleme

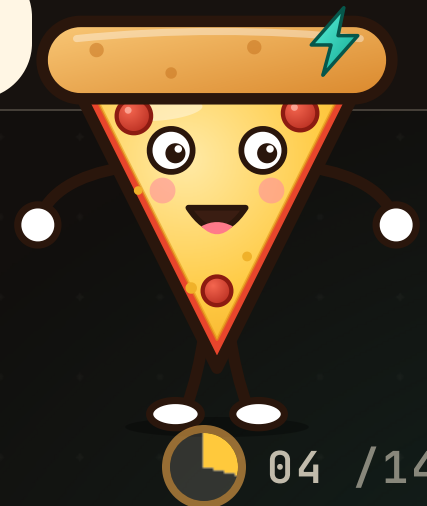
terminal

```
$ 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! 🍕"
}
```

Gerçekten üç satır.



Dört Peynirli

Veri nereden gelir? Tek fonksiyon, dört kaynak.

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** yol

/pizzas/3/orders

→ pizza_id = 3

⊕ **Query** sorgu

?extra_cheese=true

→ extra_cheese = True

⊕ **Header** başlık

X-Table-No: 7

→ x_table_no = 7

⊕ **Body** gövde

{"customer": "Ayşe", "quantity": 2}

→ order = Order(...)

Hepsi tip ipucundan.



422 pizza_id: int → /pizzas/abc isteği, fonksiyon çalışmadan reddedilir.

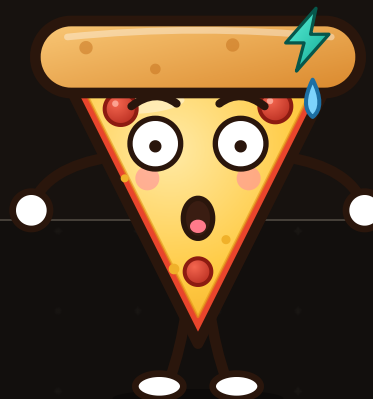
Ananas Testi

Doğrulama kodu yazmadık. Ananas yine de reddedildi.

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)
```



ANANAS MI?!



curl · Swagger UI · Postman...

POST /pizzas

ANANASLI İSTEK

```
{"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 hazırlanıyor

404 Böyle pizza yok

422 Ananas tespit edildi!

418 Çay istedin, ben
çaydanlığım ☕

Menüyü kendisi yazıyor

Tek satır dokümantasyon yazmadan.

/docs

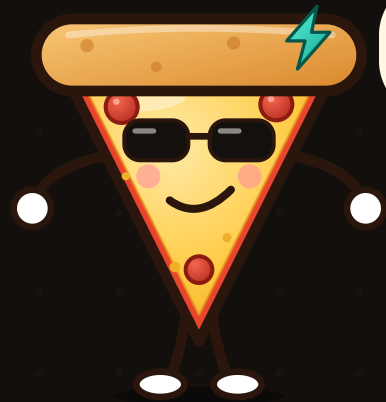
Swagger UI

/redoc

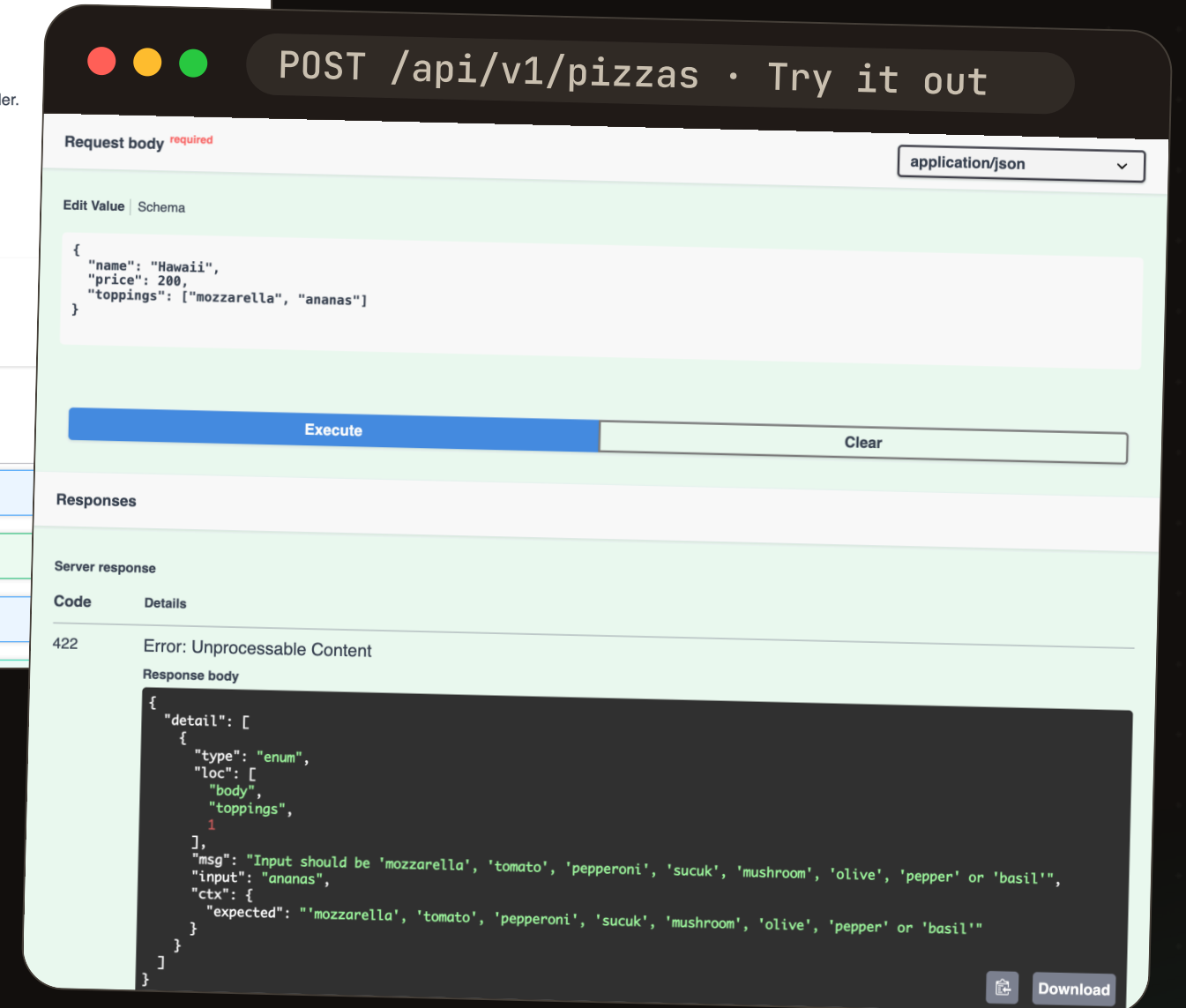
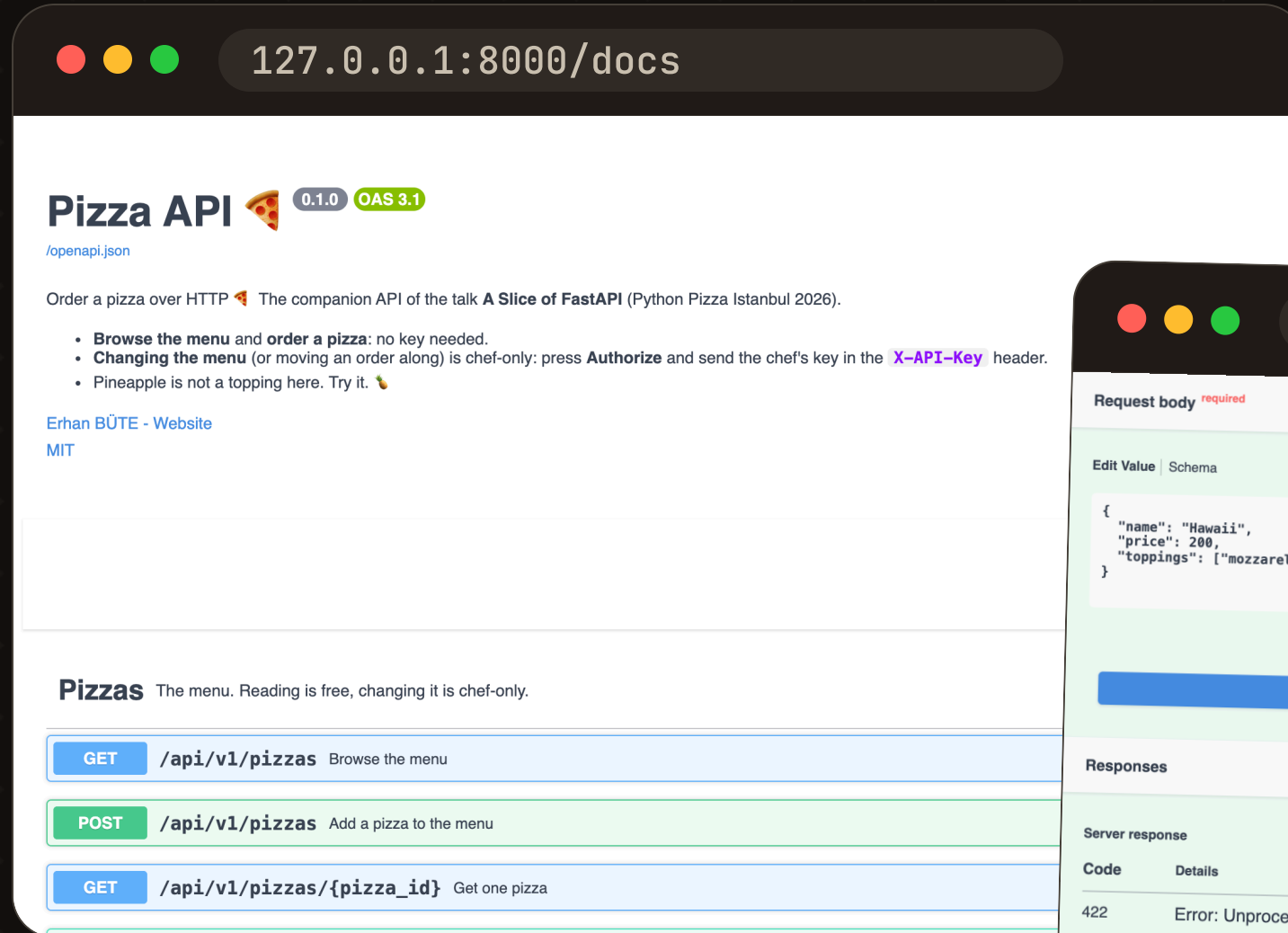
ReDoc

/openapi.json

makine dostu şema



Ben yazmadım, yemin ederim.



Depends: mutfak yardımcısı

Şef sadece pizza yapar. Malzemeyi mutfak getirir.

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()

yetki kontrolü



get_db()

veritabanı oturumu



route fonksiyonu

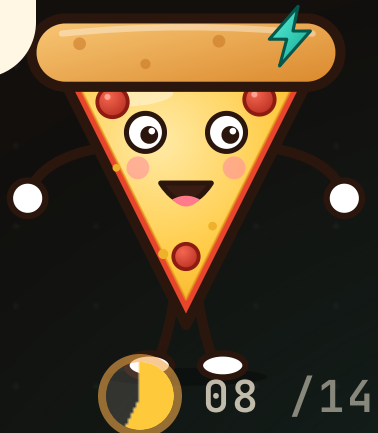
@app.get · @app.delete

app/core/db.py

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

yield = iş bitince temizlik. Bulaşıkları FastAPI yıkar 🧽

Malzemeler hazır, şef!



async def: bir garson, çok masa

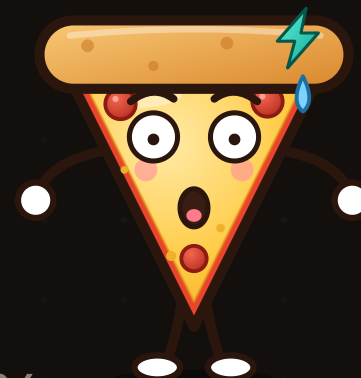
Aynı anda 3 sipariş. Garson kaç tane?

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 ✓
```



Ben tek garsonum 😓

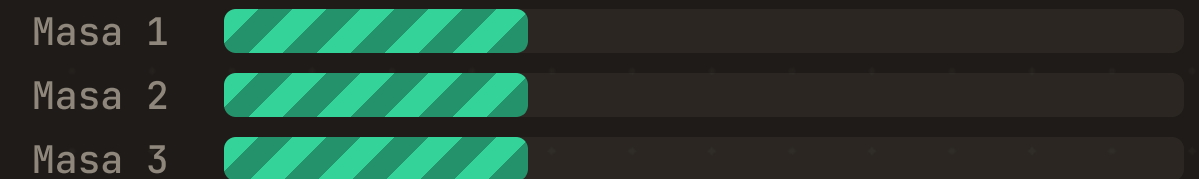
Garson fırının önünde bekliyor

6.05 sn



Fırını beklerken diğer masalara bakıyor

2.05 sn



🕒 Gerçek ölçüm: 3 eşzamanlı istek

await fırını beklerken başka masaya bak

def FastAPI seni thread havuzuna koyar

async def + time.sleep() garson fırının önünde donar

Büyüyen restoran: mutfak düzeni

İstek yukarıdan aşağı akar.

fastapi-pizza/

├─ **app/**

| └─ main.py

| └─ api/

| └─ schemas/

| └─ services/

| └─ crud/

| └─ models/

| └─ core/

├─ examples/

└─ tests/

🚀 **create_app()**

lifespan

🔔 **Garson**

route'lar + bağımlılıklar

📋 **Menü & fiş**

Pydantic modelleri

👨 **Şef**

iş kuralları

🇹🇷 **Kiler**

veritabanı sorguları

📖 **Tarif defteri**

tablolar

🔧 **Ekipman**

ayarlar, DB, hatalar

slayttaki kodlar

önce tat



🌐 **İstek**



route · api/



service · services/

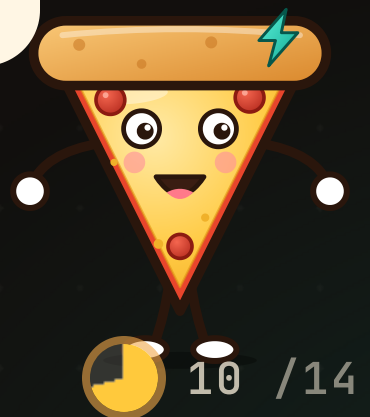


crud · crud/



DB · models/

Her şeyin bir yeri var.



Önce sen tat

TestClient: sunucu açmadan, saniyeler içinde.

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... 422 tam yerinde.

terminal

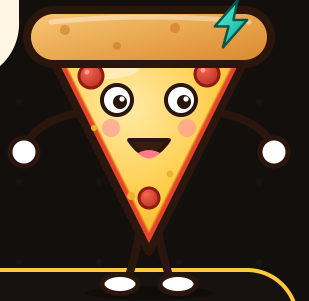
```
$ uv run pytest -v
tests/test_pizzas.py::test_pizza_id_must_be_an_int PASSED [ 65%]
tests/test_pizzas.py::test_pineapple_is_rejected PASSED [ 70%]
===== 44 passed in 2.07s =====
```



Sıcak servis

Yerelden yayına üç komut.

Sıcak sıcak!



\$ fastapi dev

GELİŞTİRME

otomatik yeniden yükleme

\$ fastapi run

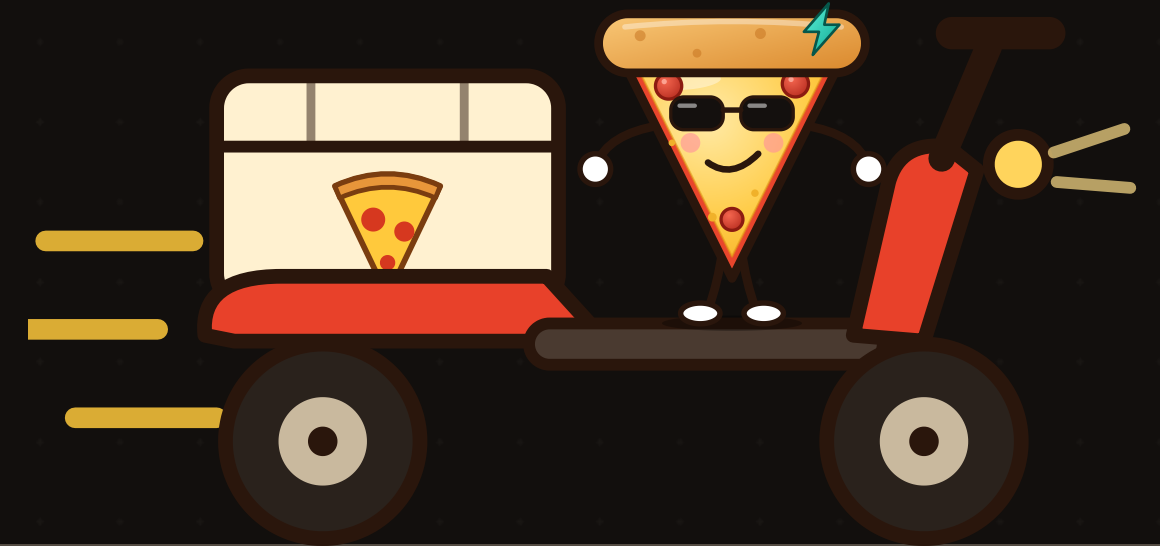
ÜRETİM

tek komut, hazır sunucu

\$ fastapi deploy

FASTAPI CLOUD

tek komutla yayında



Docker ya da kendi sunucunda uvicorn da olur.

Hesap, lütfen!



PIZZA API • FİŞ

14 Kasım 2026 • StartGate, Maslak, İstanbul

Tip ipuçları doğrulama + dokümantasyon
4 x peynir path • query • header • body
Pydantic ananas reddi
/docs menü kendini yazar
Depends malzemeyi mutfak getirir
async bir garson, çok masa
TestClient önce sen tat

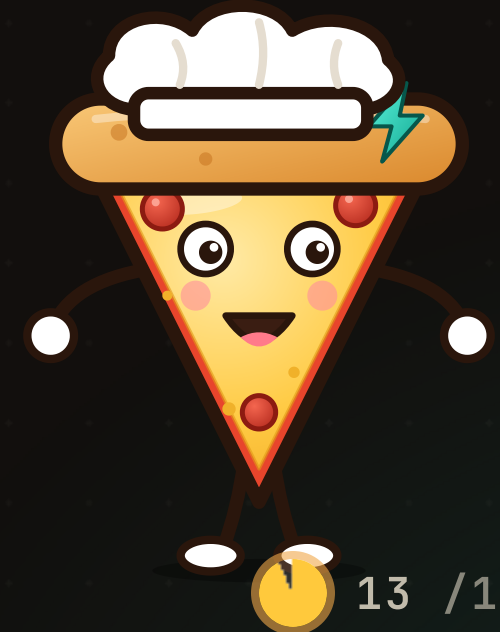
TOPLAM **10 dk**

Bahşiş: repoya ★



AFİYET OLSUN

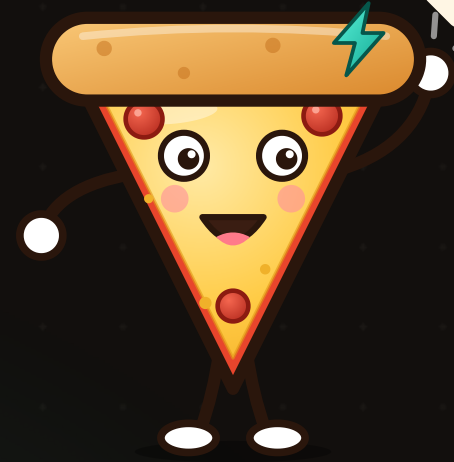
Hesap bende.



Afiyet olsun!

Sorular? (Ananas hariç.)

Teşekkürler! 🙌



erhan.dev



github.com/erhan



python-pizza.erhan.dev



Slaytlar

python-pizza.erhan.dev



yakında

Örnek kod

github.com/erhan/fastapi-pizza