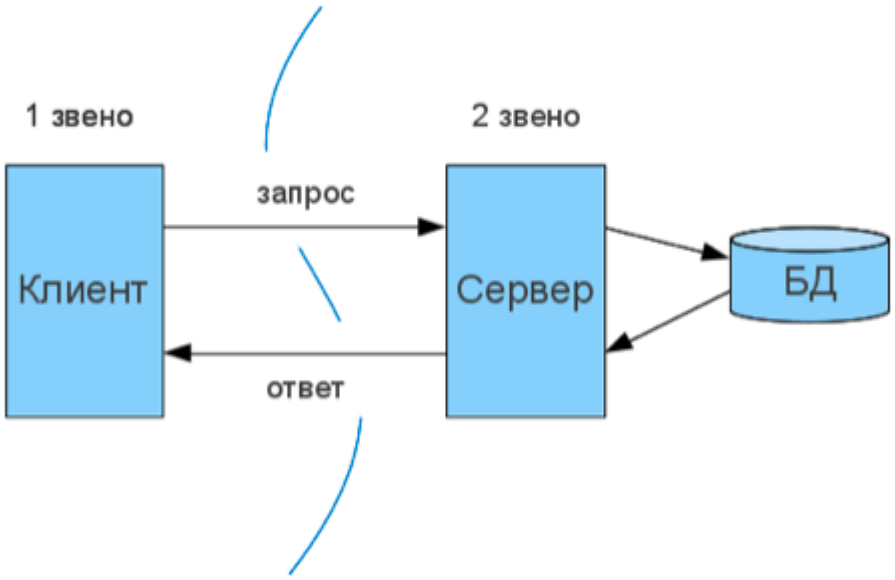
Lesson 31 - Fast Api

Legend

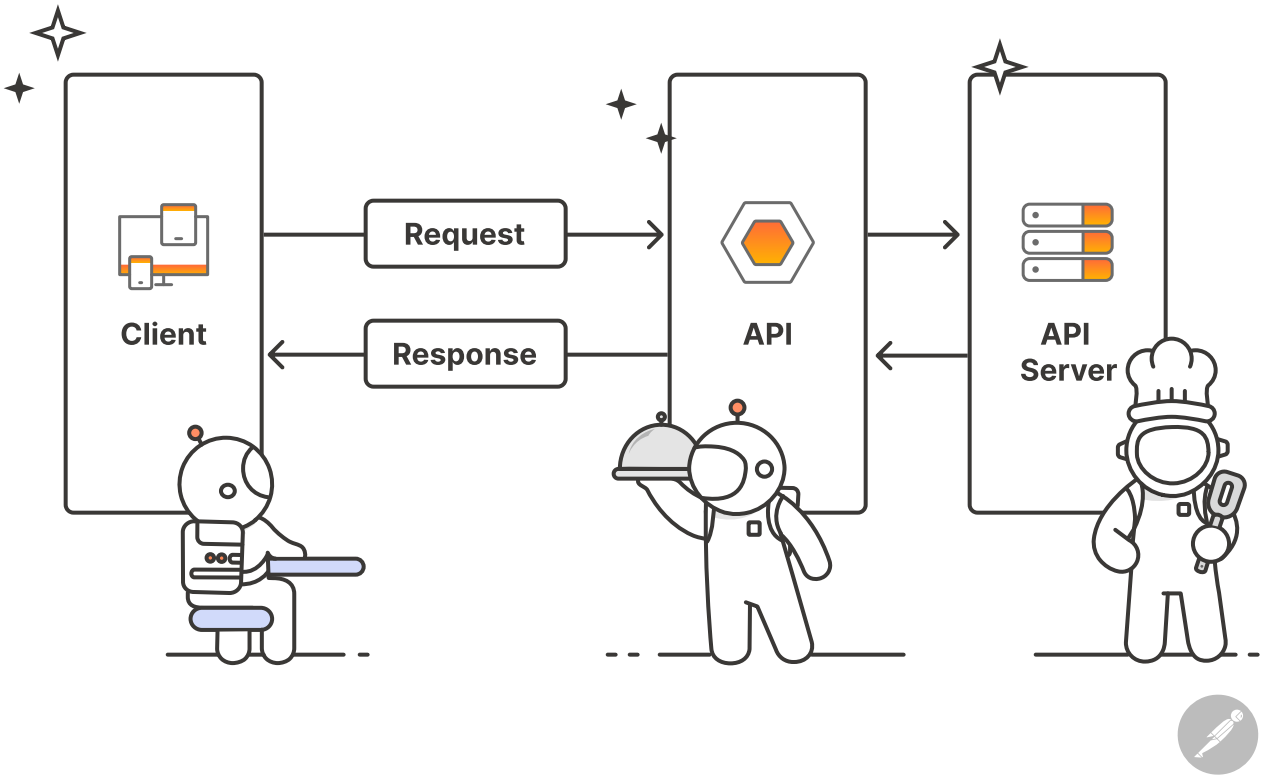
Архитектура веб приложений



API

API (англ. application programming interface — программный интерфейс приложения) — это набор способов и правил, по которым различные программы общаются между собой

и обмениваются данными.



JSON

JSON (англ. JavaScript Object Notation, в английском обычно произносится как /ˈ dʒ eɪ s ən/) — текстовый формат обмена данными, основанный на JavaScript.

```
{
    "type": "message",
    "channel": "C2147483705",
    "user": "U2147483697",
    "text": "Hello world",
    "ts": "1355517523.000005",
    "is_starred": true,
    "pinned_to": ["C024BE7LT", ...],
    "reactions": [
        {
            "name": "astonished",
            "count": 3,
            "users": [ "U1", "U2", "U3" ]
        },
        {
            "name": "facepalm",
            "count": 1034,
            "users": [ "U1", "U2", "U3", "U4", "U5"
]
        }
```


HTTP - Клиенты

- CURL(консольная утилита): curl -X 'GET' 'http://127.0.0.1:8000/users/'
- Веб приложения
- Мобильные приложения
- Postman / Insomnia

FastApi

🚀 Введение в FastAPI

FastAPI — это современный фреймворк для создания API на Python, который отличается высокой производительностью и удобством.

🔷 **Основные преимущества:**
✅ **Быстродействие** — почти как у Node.js и Golang (использует Starlette + Pydantic)
✅ **Автоматическая документация** — OpenAPI и Swagger
✅ **Асинхронность** — полная поддержка async/await
✅ **Простота кода** — минимальный boilerplate

📌 **Установка и запуск**
Устанавливаем FastAPI и сервер Uvicorn:

```
pip install fastapi uvicorn
```

Создадим файл main.py с простым API:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Привет, FastAPI!"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: str = None):
    return {"item_id": item_id, "query": q}
```

Запускаем сервер командой:

```
uvicorn main:app --reload
```

запуск сервера на определенном порту

```
uvicorn main:app --reload --port 8080
```

🔥 Теперь API доступно по адресу: **http://127.0.0.1:8000/

Документация API автоматически создаётся:
📜 Swagger UI → **http://127.0.0.1:8000/docs**
📜 **ReDoc** → http://127.0.0.1:8000/redoc

📌 Обработка запросов

🔷 Query-параметры

```
@app.get("/users/")
def get_users(name: str = "Гость", age: int = 18):
    return {"name": name, "age": age}
```

Запрос: http://127.0.0.1:8000/users/?name=Иван&age=30
Ответ:

```
{"name": "Иван", "age": 30}
```

🔷 Path-параметры

```
@app.get("/users/{user_id}")
def get_user(user_id: int):
    return {"user_id": user_id}
```

Запрос: http://127.0.0.1:8000/users/42
Ответ:

```
{"user_id": 42}
```

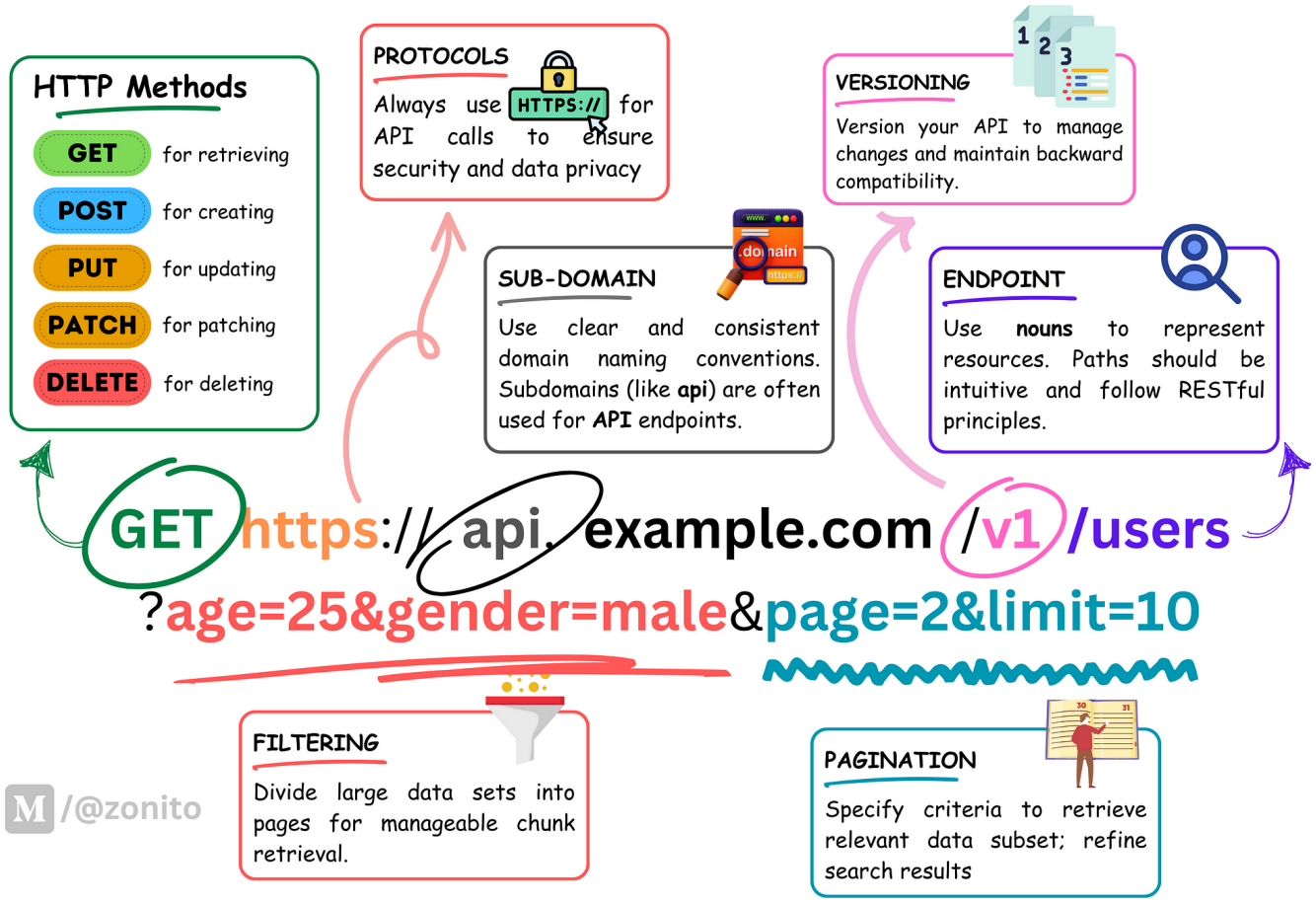
1️⃣ Простой POST-запрос с pydantic

Обычно в POST-методе передаём данные в body, и FastAPI автоматически их валидирует с помощью **Pydantic**.

```
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()

class Item(BaseModel):
    name: str
    price: float
    in_stock: bool = True

@app.post("/items/")
```

```python
def create_item(item: Item):
    return {"message": f"Товар '{item.name}' добавлен!", "data": item}
```

Restfull API



ДЗ-31

Реализовать CRUD для списка книг

- get all
- get by id
- create
- update
- delete

```
{
    "title": "Book1",
    "description": "Very old book",
```

```
    "author": "Vasya Federov"
}
```