Jak działają aplikacje webowe?

Wiedza o webówce w pigułce

Agenda

Architektura klient-serwer

Co to takiego? Komu to potrzebne? A na co?

Czym jest API?

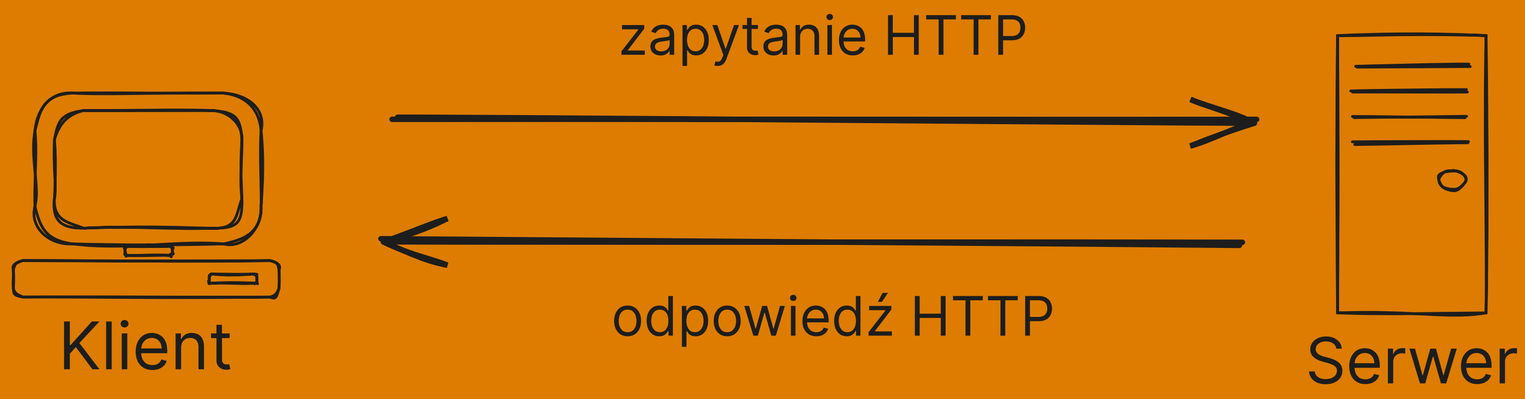
Aaa, wiem, to przez to mogą namierzyć cię hakerzy!

Architektury aplikacji webowych

Nie po to jestem na infie, żeby się teraz w budowlankę pakować...

Klient-serwer

Aplikacje webowe działają w modelu klient-serwer. Klient – czyli przeglądarka – wysyła zapytania, a serwer odpowiada. To podstawa działania wszystkich współczesnych aplikacji webowych



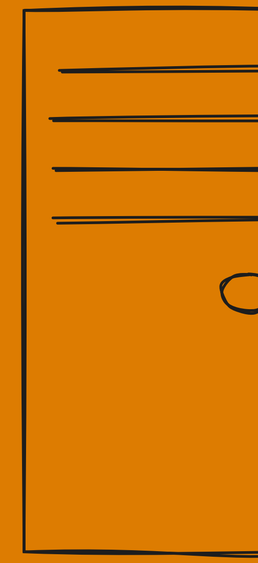
Czym jest zapytanie?

Klient

GET /produkty

Status: 200 OK
Body:

```
[
  { "id": 1, "nazwa": "Draże" },
  { "id": 2, "nazwa": "Korsarze" }
]
```

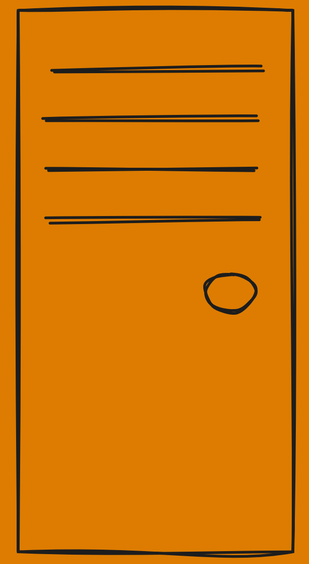


Serwer

API - co to i do czego służy?

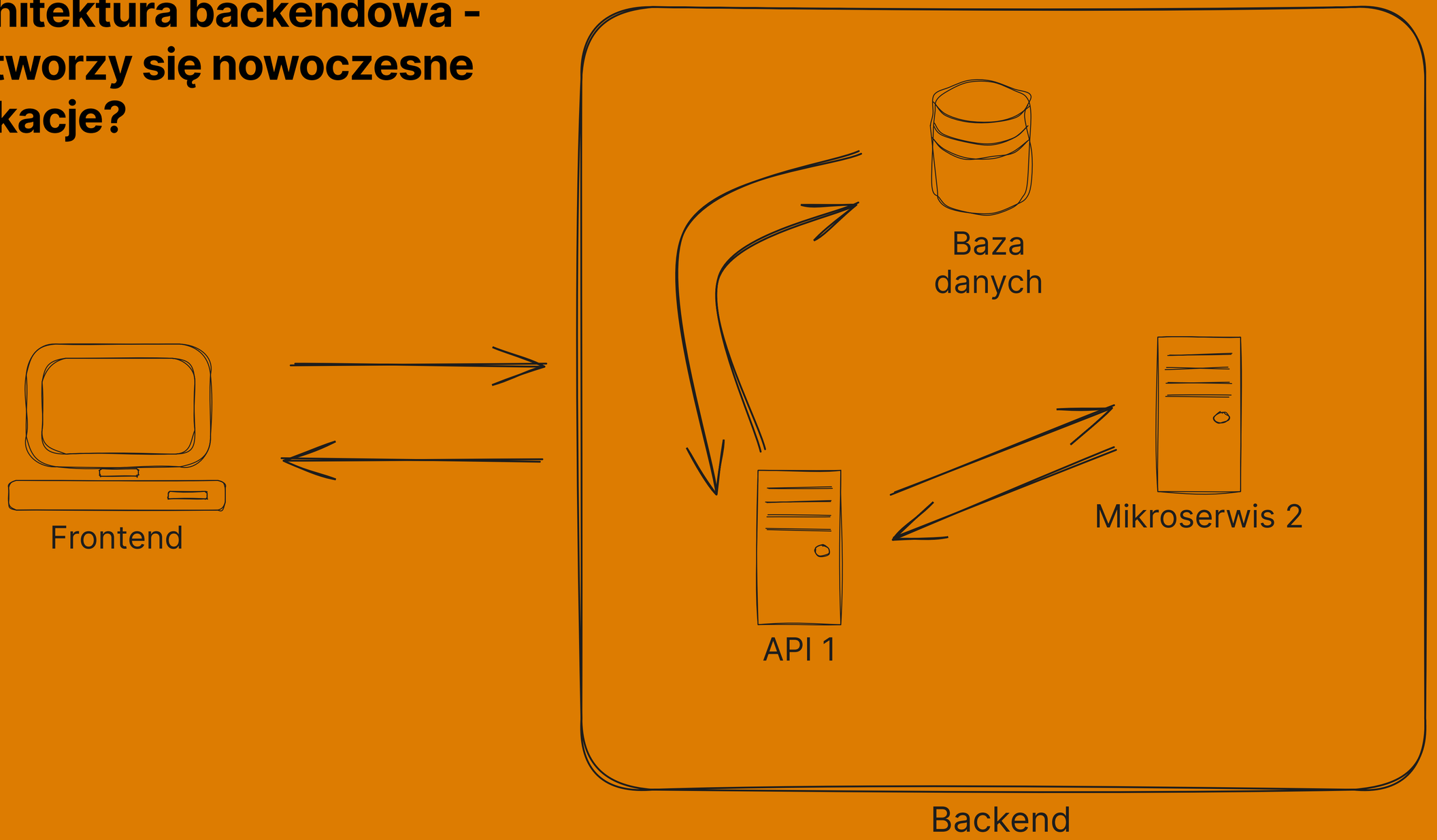
API to zestaw reguł, które opisują, jak możemy rozmawiać z aplikacją. To jak menu w restauracji – pokazuje, co możemy zamówić i czego się spodziewać.

```
GET /produkty

POST /produkty

POST /zamowienia

GET /zamowienia?id=2138
```



Serwer

Architektura backendowa - jak tworzy się nowoczesne aplikacje?



API Design patterns

Czyli jak tworzyć backend tak, żeby miał ręce i nogi, a conajmniej - endpointy

Kilka polularnych wzorców projektowych

CSR

Controller - Service - Repository
Popularny w Spring Boot. Dane i logika oddzielone w warstwach Service i Repository.

REST

Styl, w którym komunikacja oparta jest na zasobach (np. `/produkty`), z metodami HTTP (GET, POST, itd)

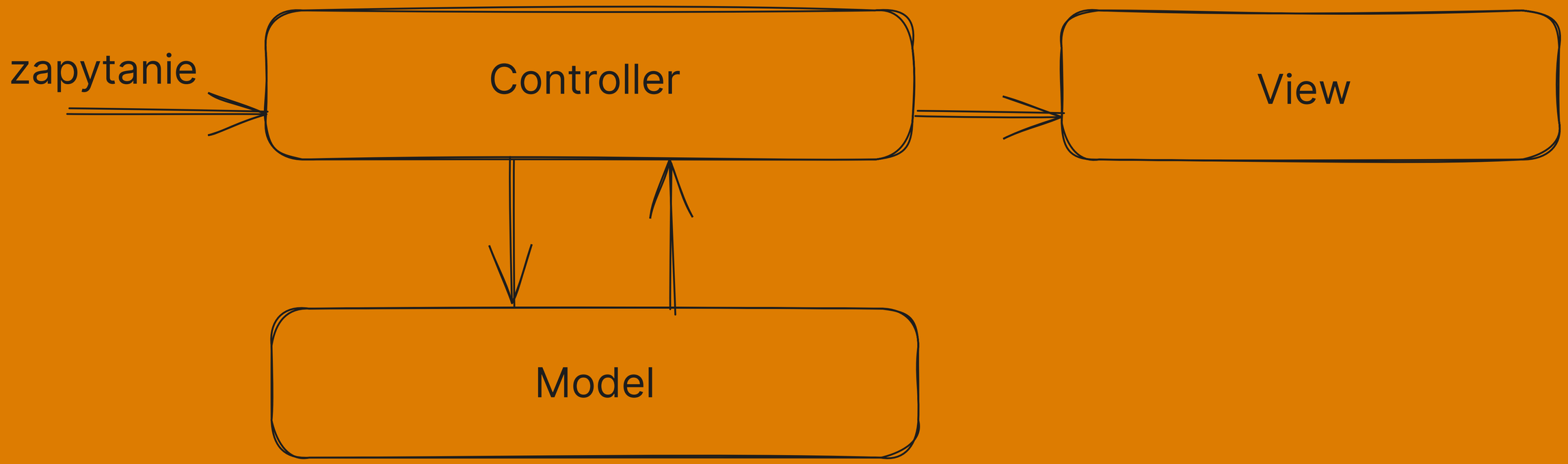
MVC

Model - View - Controller
Rozdziela widok, logikę i dane – klasyka w aplikacjach webowych.

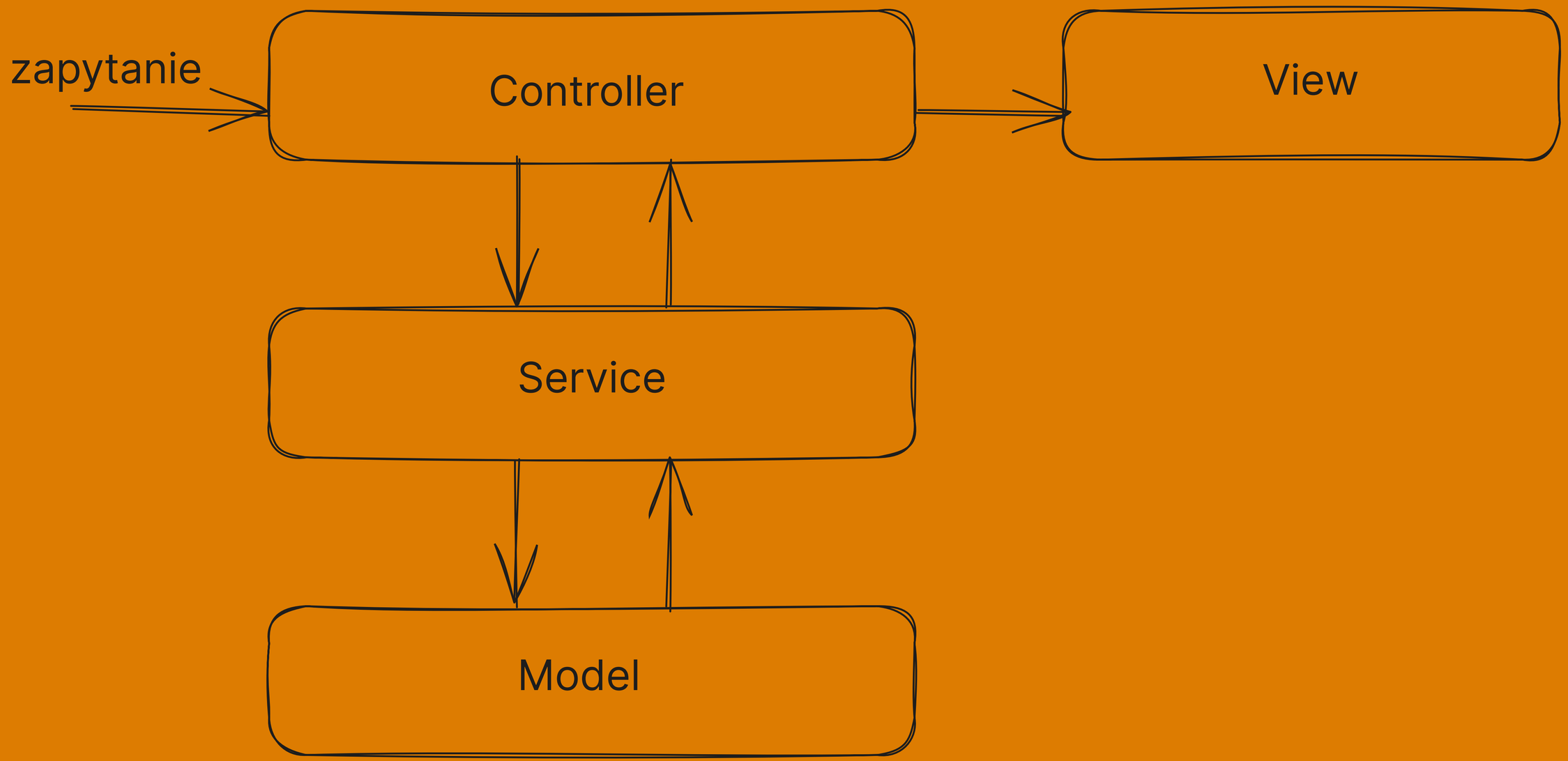
MVCS

Model - View - Controller - Service
Dodaje warstwę z logiką biznesową. Controller staje się "cieńszy".

Flow w wzorcu MVC



Flow w wzorcu MVCS



Dzięki za uwagę!

Czas na klepanie kodu!