Backend REST API fejlesztés

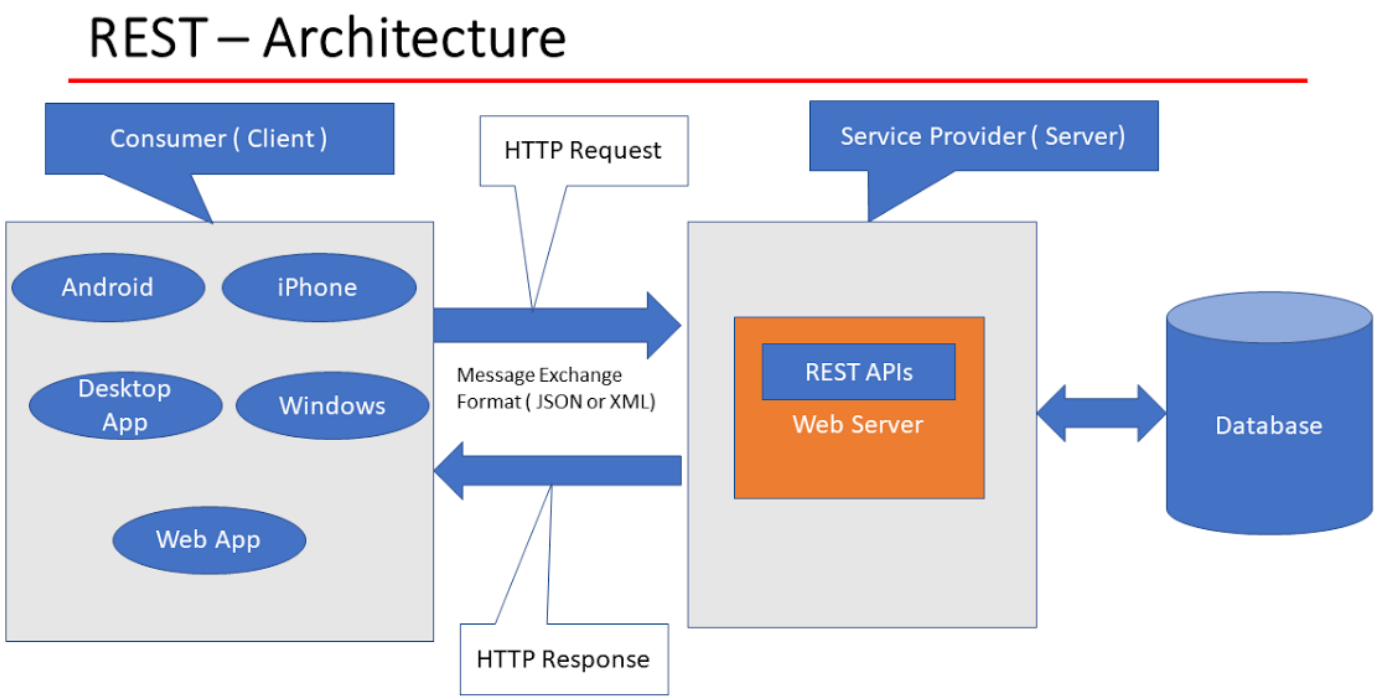
MVC-re épülő backend REST API tervezése és fejlesztése

Bevezetés és célkitűzések

- Mit nevezünk backend REST API-nak?
- Kapcsolat az MVC architektúrával
- Elmélet + gyakorlati szemlélet
- Mit fogunk ma megérteni?

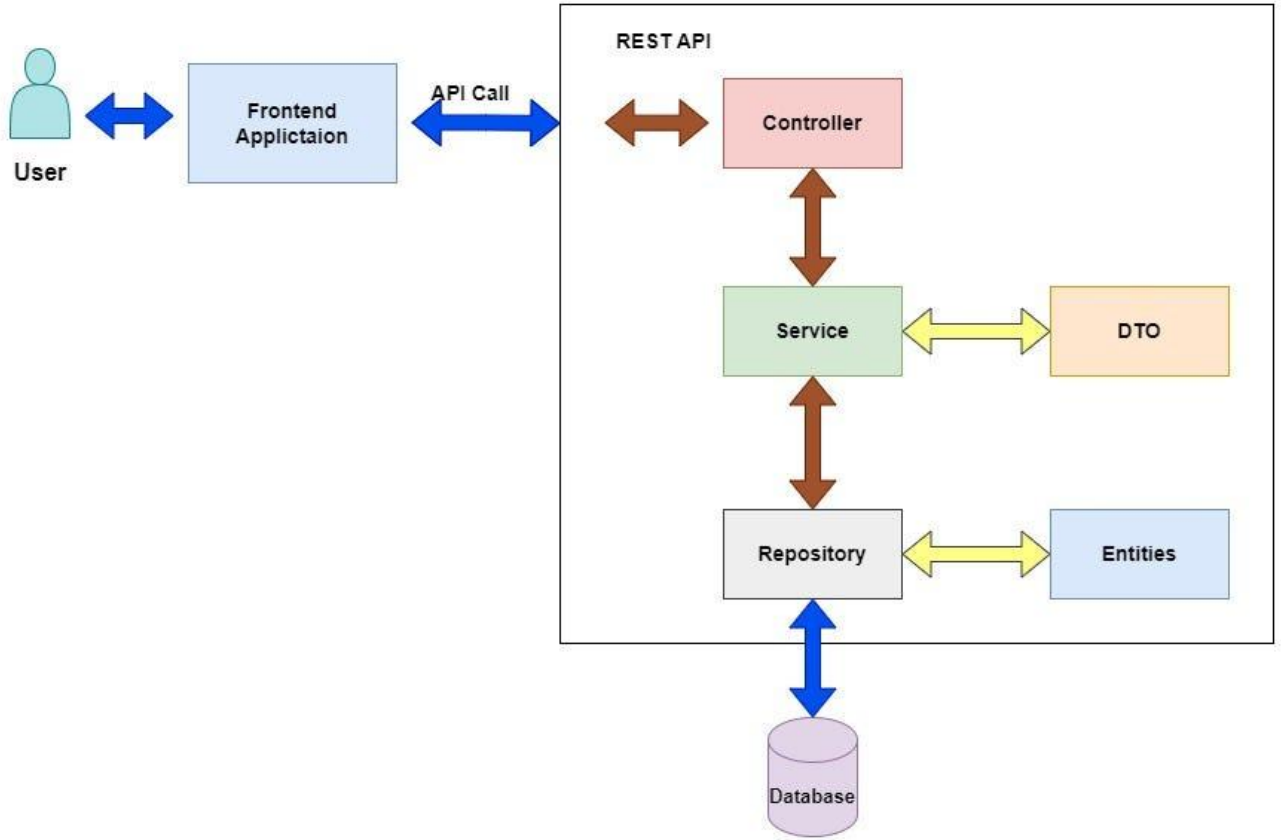
Hol helyezkedik el a Backend REST API?

- Frontend ↔ Backend ↔ Adatbázis
- Backend mint „üzleti logika kapu”
- REST API mint szerződés



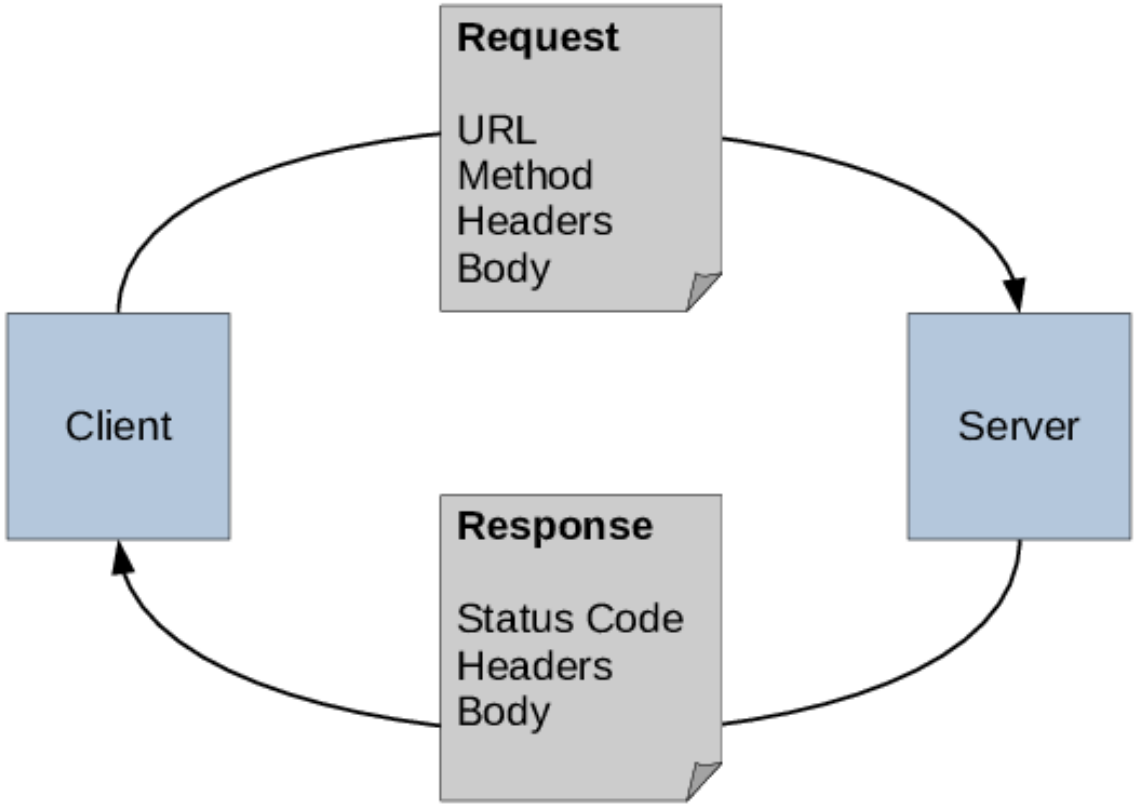
MVC architektúra szerepe Backend oldalon

- **Model** – adat és üzleti logika
- **Controller** – belépési pont
- **View** – JSON adatok



Request–Response életciklus REST API-ban

- HTTP Request
- Routing / Controller
- Feldolgozás
- HTTP Response



Stateless működés jelentése a gyakorlatban

- Nincs szerveroldali session
- Minden request önálló
- A backend nem tárol kliens specifikus állapotot
- Skálázhatóság: több backend példány szolgálhat ki kéréseket

HTTP metódusok szerepe a Backend API-ban

- GET – lekérdezés
- POST – létrehozás
- PUT / PATCH – módosítás
- DELETE – törlés

A metódus **nem csak technikai részlet**, hanem jelentéssel bír. A kliens ebből érti meg a művelet célját.

URI design és erőforrás-orientált gondolkodás

- **Erőforrások** (resources) címzése
- **Főnevek** használata, nem pedig igék
- **Hierarchikus URL**-ek használata

REST-ben nem műveleteket, hanem **erőforrásokat** címzünk.

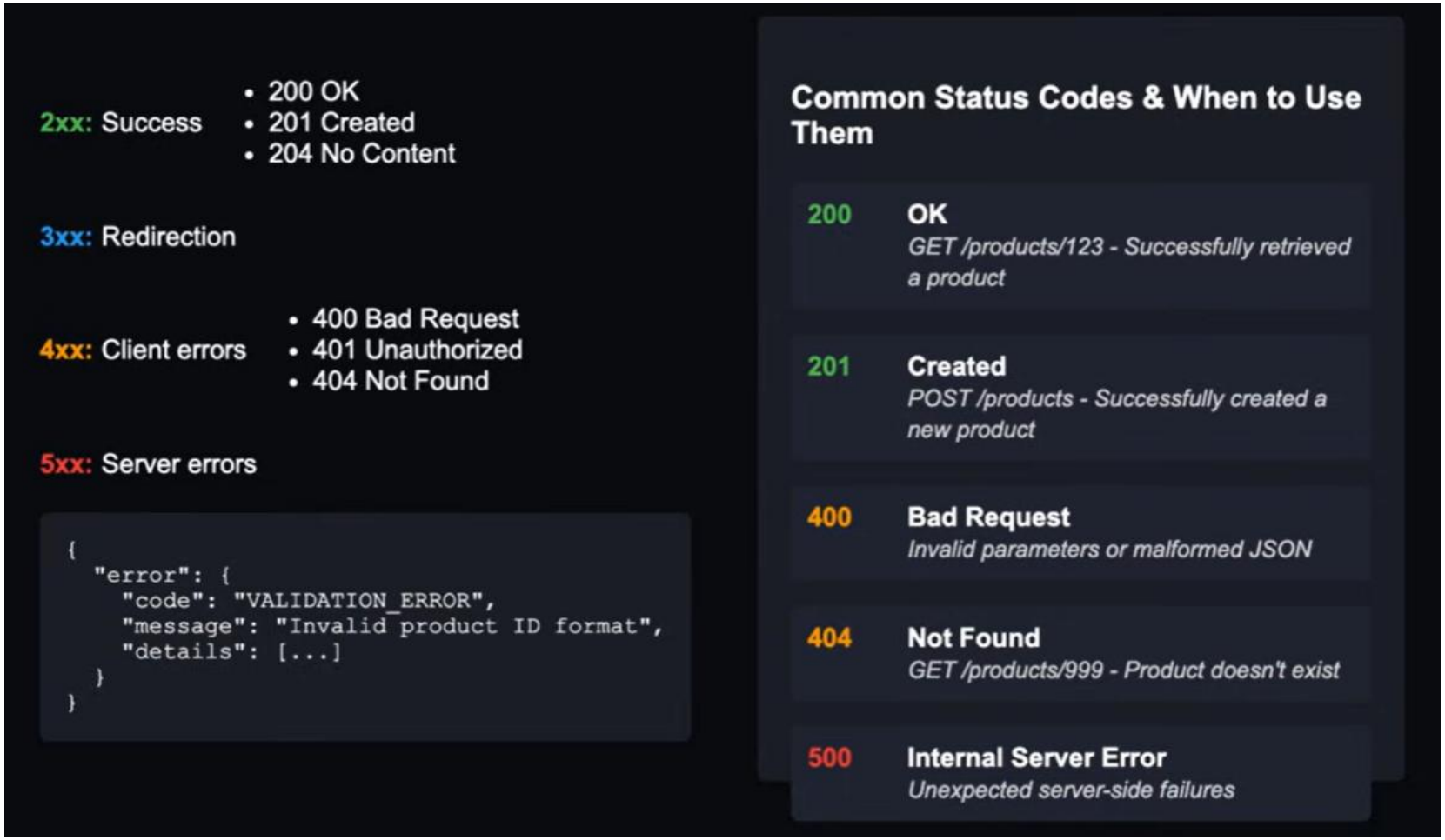
URI design és HTTP metódusok

Method	CRUD	Example
GET	Read	`GET /api/v1/products` Safe, Idempotent
POST	Create	`POST /api/v1/products` Not Safe, Not Idempotent
PUT	Update (Replace)	`PUT /api/v1/products/123` Not Safe, Idempotent
PATCH	Update (Partial)	`PATCH /api/v1/products/123` Not Safe, Not Idempotent
DELETE	Delete	`DELETE /api/v1/products/123` Not Safe, Idempotent

Safe: Does not change server state

Idempotent: Same effect if repeated

HTTP státuszkódok szerepe



JSON mint adatcsere-formátum

- Strukturált felépítésű
- Platform és programozási nyelv független
- Emberileg olvasható, szöveges formátum
- Könnyen feldolgozható

```
{
  "id": 1,
  "name": "John Doe",
  "email": "johndoe@example.com"
}
```

Request adatok kezelése

Egy REST API-ban többféle módon érkezhet adat:

- Path paraméter /api/users/1
- Query paraméter /api/users?userid=1
- Request body

```
{
  "id": 1,
  "name": "John Doe",
  "email": "johndoe@example.com"
}
```

Controller szerepe (Spring Boot példa)

Felhasználó lekérése ID alapján

GET api/users/{id}

- Belépési pont
- Request paraméterek kezelése
- Model réteg szolgáltatási hívása
- Response visszaadása

```java
@RestController
@RequestMapping("/api/v1/users")
public class UserController {

    @GetMapping("/{id}")
    public ResponseEntity<UserDto> getUser(@PathVariable Long id) {
        UserDto user = userService.findById(id);
        return ResponseEntity.ok(user);
    }
}
```

POST: User létrehozása (Spring Boot)

Új felhasználó létrehozása JSON request body alapján

POST /api/users

```java
@RestController
@RequestMapping("/api/v1/users")
public class UserController {

    @PostMapping
    public ResponseEntity<UserDto> createUser(@RequestBody CreateUserDto dto) {
        UserDto createdUser = userService.create(dto);
        return ResponseEntity.status(HttpStatus.CREATED).body(createdUser);
    }
}
```

PUT: User módosítása (Spring Boot)

Meglévő felhasználó teljes módosítása

PUT /api/users/{id}

```
@PutMapping("/{id}")
public ResponseEntity<UserDto> updateUser(
        @PathVariable Long id,
        @RequestBody UpdateUserDto dto
) {
    UserDto updatedUser = userService.update(id, dto);
    return ResponseEntity.ok(updatedUser);
}
```

DELETE: User törlése (Spring Boot)

Felhasználó törlése azonosító alapján

DELETE /api/users/{id}

```
@DeleteMapping("/{id}")
public ResponseEntity<Void> deleteUser(@PathVariable Long id) {
    userService.delete(id);
    return ResponseEntity.noContent().build();
}
```

REST végpont FastAPI-val (Python)

Python FastAPI – egyszerű REST API példa

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/users/{user_id}")
def get_user(user_id: int):
    return {
        "id": user_id,
        "name": "John Doe"
    }
```

Összefoglalás

- Backend REST API = koncepciók + technológia
- MVC tovább él a backendben
- Következő lépés: gyakorlati API építés