Angular – HTTP kommunikáció

Observable-alapú aszinkron működés · Error handling · Interceptorok

Frontend ↔ Backend kapcsolat Angularban

- Böngésző → HTTP kérés
- Angular HttpClient
- Backend API (REST)
- JSON adatcsere
- Aszinkron működés

Miért NEM a komponens hívja a backendet?

- **Komponens** feladata az Ul logika
- Ne legyen benne üzleti logika
- Ne legyen benne API hívások

- **Service** feladata: adat és üzleti logika, API hívások
- Újrafelhasználható
- Tesztelhető

HttpClient szerepe

- Angular beépített HTTP kliens
- REST API hívások
- JSON automatikus feldolgozás
- Observable-t ad vissza

Mit jelent az, hogy aszinkron?

- HTTP kérés elindul
- Angular nem vár
- Kód tovább fut
- Válasz később érkezik

Observable alapgondolat

- Időben, egymás után érkező adatok
- Feliratkozás az adatfolyamra
- Több értéket is küldhet
- Aszinkron működés
- Leiratkozás az adatfolyamról

HTTP + Observable együtt

```
this.http.get<User[]>('/api/users')
  .subscribe({
    next: users => { /* adat */ },
    error: err => { /* hiba */ },
    complete: () => { /* lezárás */ }
  });
```

- HTTP kérés csak subscribe-nál indul
- **next** → adat érkezik
- **error** → hiba történt
- **complete** → adatfolyam lezárult
- Observable = **lazy**

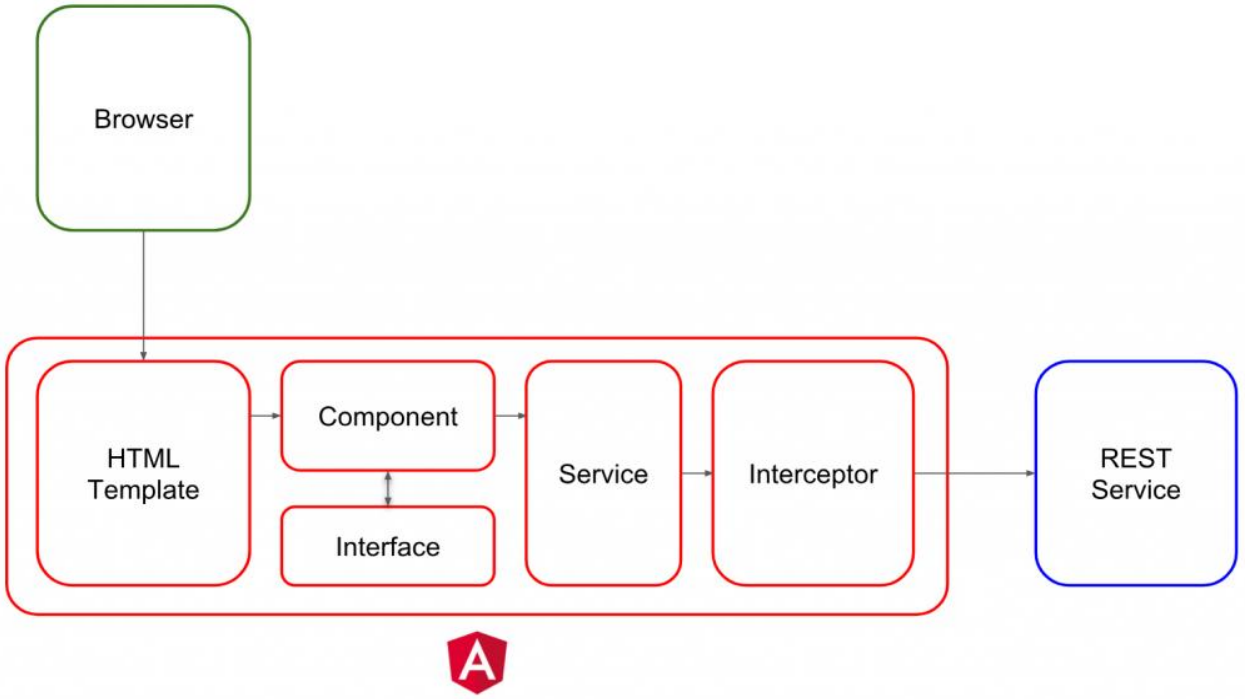
Hibakezelés az Angularban

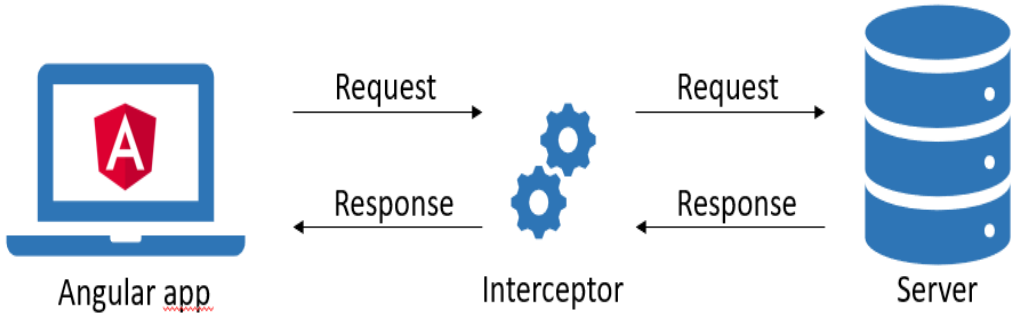
- HTTP hibák kezelése
- Backend error vs network error
- Service-ben kezeljük
- Komponens csak reagál

Interceptor fogalma

- Az interceptor olyan, mint egy **szűrő** a HTTP forgalomban
- HTTP kérés elé/után lép be
- Automatikus header hozzáadás
- Auth token hozzáadása
- Logging
- Globális error kezelés

HTTP kommunikáció teljes képe





- Komponens → Service
- Service → HttpClient
- Interceptor → Backend
- Válasz → Observable → Komponens

Összefoglalás

- **Komponens** = UI kezelés
- **Service** = adat és logika
- **HttpClient** = backend kommunikáció
- **Observable** = aszinkron adatfolyam
- **Interceptor** = globális szabályok