Nagyobb cégek használata:

Java/Kotlin szerverek:

- **Netflix** - teljes backend Java/Kotlin (millió felhasználó)
- **LinkedIn** - Java mikroszolgáltatások
- **Uber** - Java/Kotlin backend szolgáltatások
- **Google** - sok belső szolgáltatás Java-ban
- **Twitter** - Scala (JVM-en fut, mint a Kotlin)

C/C++ használat:

- **Facebook** - kritikus részek C++
- **Google Chrome** - böngésző motor C++
- **Redis/MongoDB** - adatbázis motorok
- **Játékszerverek** - real-time játékok

Tipikus architektúra a gyakorlatban:

E-commerce (pl. Amazon-szerű):

```
Frontend (React/Vue)
    ↓ HTTP/JSON
Backend API (Java/Kotlin Spring Boot)
    ↓ SQL
Adatbázis (PostgreSQL/MySQL)
    ↓
Cache (Redis - C++ alapú)
```

Közösségi média:

```
Mobile App (Swift/Kotlin)
    ↓ REST API
Java/Kotlin mikroszolgáltatások
    ↓
Több adatbázis + Cache
```

Mikor melyiket választják:

Java/Kotlin/Spring Boot:

- **Web API-k** (99% esetben)
- **Mikroszolgáltatások**
- **E-commerce backend**
- **CRM rendszerek**
- **Banki alkalmazások**

C/C++:

- **Adatbázis motorok** (MySQL, PostgreSQL)
- **Cache rendszerek** (Redis, Memcached)
- **Játékszerverek** (real-time)
- **IoT eszközök** (korlátozott erőforrás)
- **Pénzügyi kereskedési rendszerek** (microsecond számít)

Modern trend:

Mikroszolgáltatás architektúra:

```
User Service (Java/Kotlin)
Product Service (Java/Kotlin)
Payment Service (Java/Kotlin)
Notification Service (Node.js/Go)
Analytics Service (Python)
Cache Layer (Redis - C++)
Database (PostgreSQL - C++)
```

API-first fejlesztés:

1. **Backend API** készül először (Java/Kotlin)
2. **Frontend** csatlakozik hozzá (React/Vue/Angular)
3. **Mobile** appok is ugyanazt az API-t használják

Valós példa - Startup indítás:

1. fázis: MVP (Minimum Viable Product)

```
// Spring Boot API - gyors fejlesztés
@RestController
class ProductController {
    @GetMapping("/products")
    fun getProducts() = productService.findAll()
}
```

2. fázis: Növekedés

- Több endpoint hozzáadása
- Adatbázis optimalizáció
- Authentication hozzáadása

3. fázis: Skálázás

- Mikroszolgáltatásokra bontás
- Cache hozzáadása (Redis)
- Load balancer

JSON fájlok használata:

Konfigurációs fájlok:

```
{
  "database": {
    "host": "localhost",
    "port": 5432
  }
}
```

Teszt adatok:

```
{
  "testUsers": [
    {"id": 1, "name": "Test User"}
  ]
}
```

Statikus tartalom:

- **Fordítások** (i18n)
- **Ország/város listák**
- **Konfiguráció**

Összefoglalva: A legtöbb modern webalkalmazás **Java/Kotlin Spring Boot backend-et** használ, ami **JSON API-kon** keresztül kommunikál a frontend-del. A C++ csak speciális esetekben szükséges!