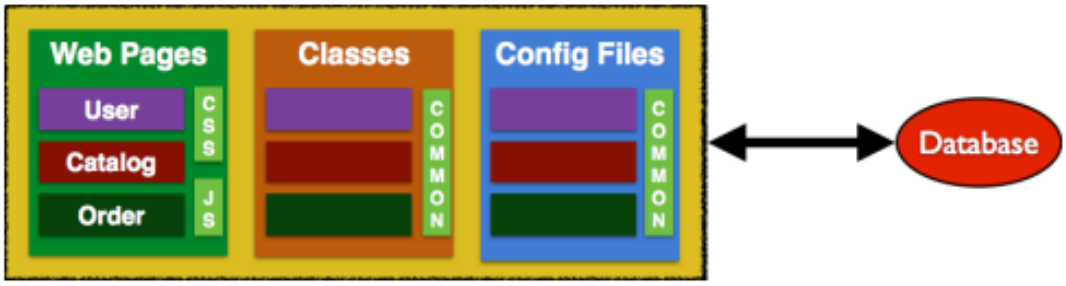
13. Mikroszervíz architektúra

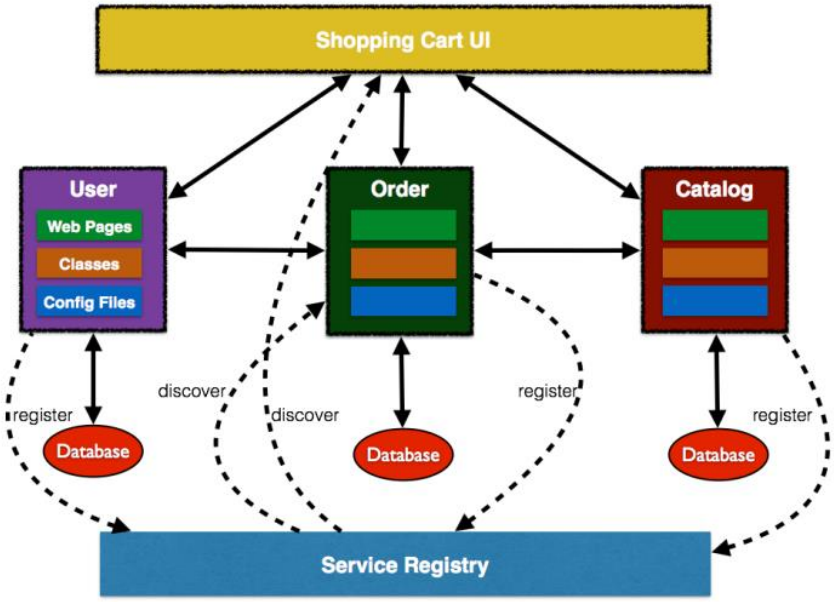
Monolitikus alkalmazás

- Rétegzéssel, egy nagy alkalmazásban, ami például össze van kötve egy adatbázissal.



Mikroszolgáltatások

- Önálló alkalmazások
- **Felbontás alapja a domain model (**Bounded context)
- Önálló fejlesztési ciklus
- Önállóan tesztelhetőek (service stubokkal)
- Önállóan skálázhatóak (valamelyik rétegnek például növeljük a memóriáját, ez a monolitikus alkalmazásokban lehetetlen)
- Más nyelven is írhatóak
- Egymással valamilyen közösen ismert protokollon keresztül kommunikálnak.
- **Például konténerbe zárunk egy-egy mikroszolgáltatást.**



Mikroszolgáltatás felépítése

Belső működés

- Hagyományos rétegezéssel épül fel.
- Saját adatbázissal rendelkezik/rendelkezhet.
- **Kommunikáció**
    - REST API vagy Message bus protokollok

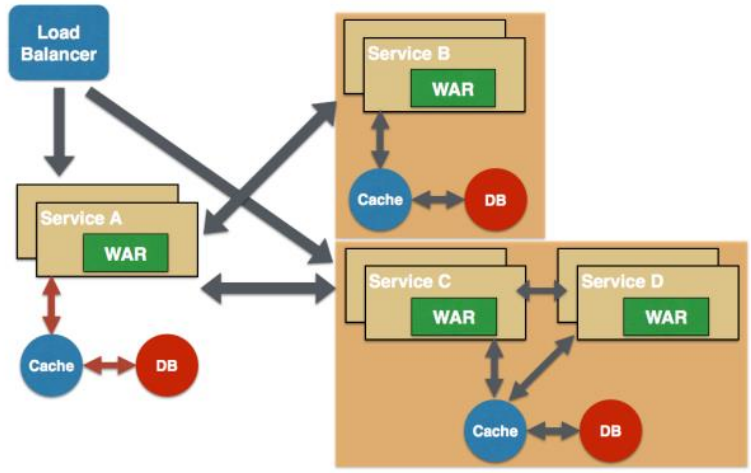
Külső elérés

- Általában REST API-n keresztül.
- UI is egy mikroszolgáltatás, ami megjelenítésért felelős.

Tervezési minták

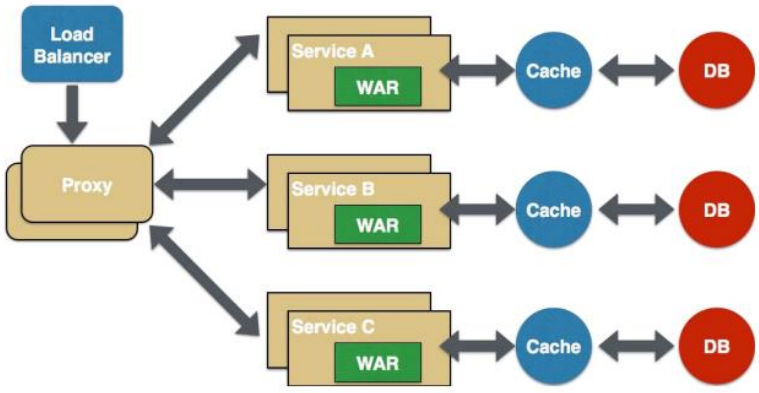
Nem strukturált

- Nem használunk semmilyen tervezési mintát.
- Oda-vissza, kölcsönösen kommunikálnak egymással.
    - Minden mikroszolgáltatás egymással össze van kötve.



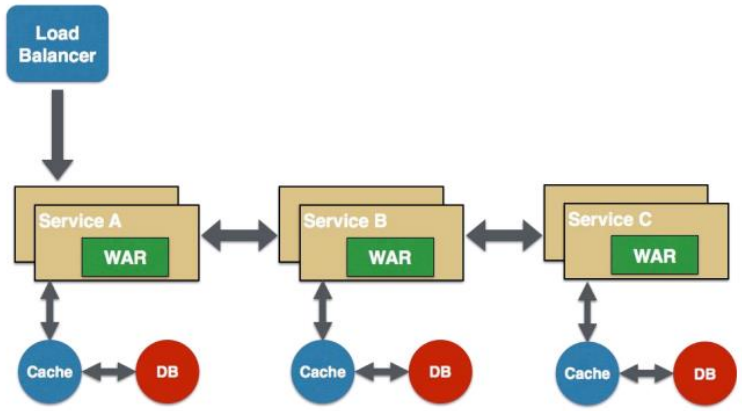
Proxy/Facade

- Fogjuk a mikroszolgáltatásokat és azoknak a különböző szolgáltatás készletét azt elrejtjük egy, összegezzük egy Proxy/Facade-ba.



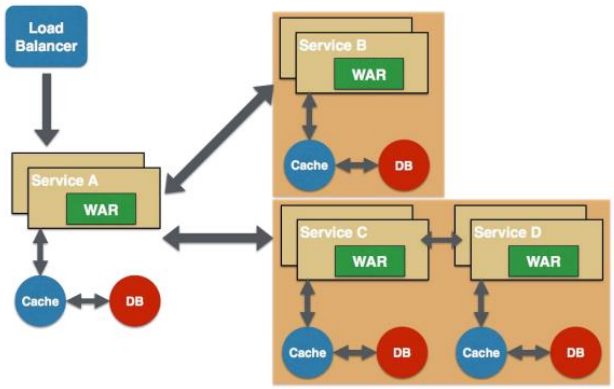
Chain of Responsibility

- Kommunikálunk egy mikroszolgáltatással, küldünk neki egy adatot és visszakapunk egy eredményt, amit továbbküldünk a következő mikroszolgáltatásnak és így tovább.



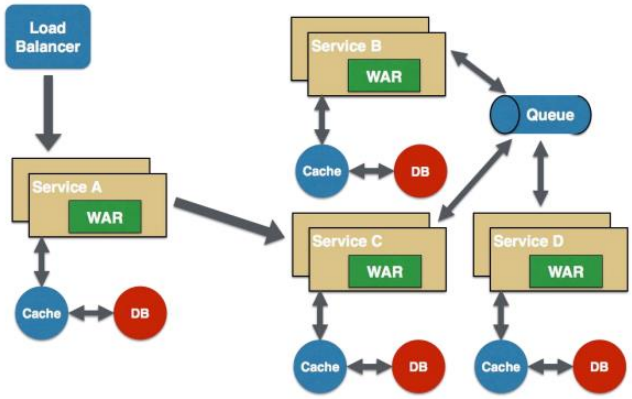
Composite

- Fa struktúrát alkotunk az adatokból, tehát a mikroszolgáltatások egymásból leszármaznak.



Mediator

- Elterjedt, mert van benne egy Messenger, amin keresztül kommunikálnak egymással a mikroszolgáltatások.
    - Tehát így nem feltétlenül kell tudniuk egymásról a mikroszolgáltatásoknak.
    - **Igény** alapján **feliratkozik** egy mikroszolgáltatásra, kap egy eredményt, visszateszi a Queue-ba és a másik mikroszolgáltatás onnan kiveszi.



Mediator implementálása

- **AMQP – Advanced Message Queueing Protocol**
  - Általános nyílt protokoll
  - Tipikusan PC/WEB-re használhatjuk
- **MQTT – Message Queue Telemetry Transport**
  - ISO szabvány
  - Publish – Subscribe üzenetküldésre tökéletes
  - Kis overhead
  - Tipikusan mobil/IoT (szenzorok)
  - Brokerek
    - Mosquitto: 30e üzenet / sec
    - Moquette: 30-100e üzenet / sec
    - HiveMQ: 800e üzenet / sec
    - Redis: 1m üzenet / sec