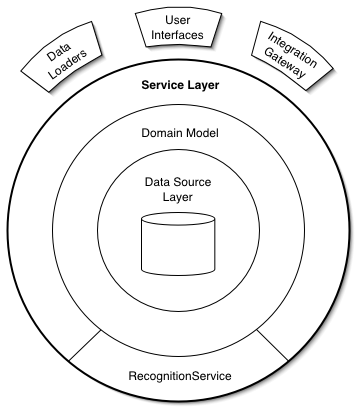
Mikroszervíz architektúra

# Service réteg (domain/infrastructure logika szétválasztása)

* Domain model egységbezárja a Domain Entityket és azok üzleti műveleteit.
* Service-Oriented Architektúra (**Microservices**)
* **Feladata:**
  + Hívások fogadása, továbbítása a Domain Logic felé.
  + Tranzakciókezelés és a lock is.
* Alsóbb rétegekben megjelenik ettől függetlenül az adatbázis szintű tranzakciókezelés is.
* **3 részből áll**

1. Domain/functional model
2. REST endpointok sorozata
3. A domain objektumok tárolására szolgáló eszköz vagy perzisztencia réteg.



## SOA-koncepció

* A SOA amiben nem kód, hanem futó kód alapú elemekből állítható össze az alkalmazás.
  + Emiatt nem kell foglalkozni az adott funkcionalitás futtató környezetével, mert ezt egy megfelelő szolgáltatás végzi el és mi csak azt használjuk.

# Szolgáltatások felépítése és összekapcsolása

## Belső működés

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

## Külső elérés

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

# Üzenetszóró protokollok és szolgáltatások (AMQP/MQTT)**.**

## AMQP – Advanced Message Queueing Protocol

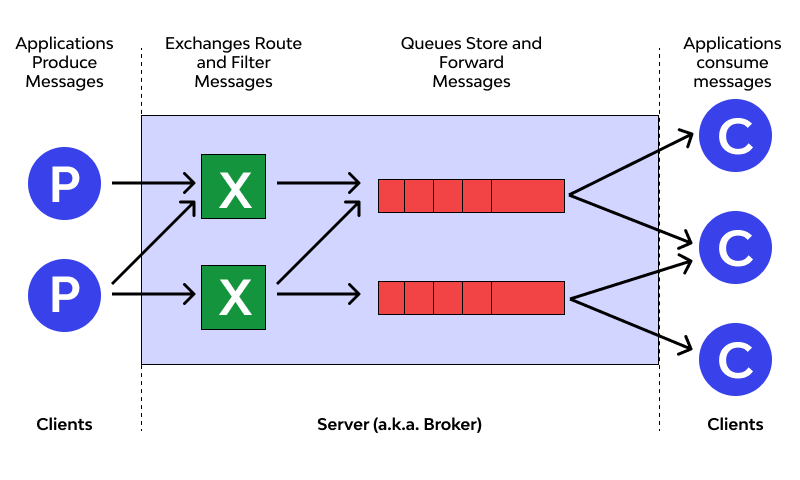
* Általános nyílt protokoll
* Általában PC/WEB
* Alkalmazási rétegen működik kliens és „brókerek” között.
* **A bróker vagy szerver döntő szerpet játszik az AMQP protokoll engedélyezésében.**
  + Felelős a kapcsolatépítésért, ami biztosítja a jobb adatátirányítást és a sorba állítást a kliens oldalon.
* Az üzenetek visszaigazolását a consumer/fogyasztó végzi.

### AMQP komponensek

* **Exchanges:** Gondoskodik az üzenetek lekéréséről és elhelyezése a queue-ban.
  + **Kategóriák:** Fanout, Headers, Topic és Direct
* **Channel:** Az AMQP peerek közötti multiplexelt virtuális kapcsolatra utal, ami egy meglévő kapcsolaton belül épül fel.
* **Message queue:** Azonosított entitás, ami segít összekapcsolni az üzeneteket a forrásaikkal vagy a kiindulási pontjukkal.
* **Binding:** Az üzenetek küldését és kézbesítését kezeli.
* **Virtual Hosztok:** Lehetővé teszi, hogy különböző alkalmazások ugyanazt a broker-t használják anélkül, hogy zavarnák egymást.

### AMQP Exchange működése

* **Direct exchange:** Összehasonlítja a routing kulcsot a hozzá kötött queue-k routing kulcsaival és az üzenetet bármelyik queue-nak kézbesíti, aminek routing kulcsa megegyezik az üzenet routing kulcsával.
* **Fanout exchange:** Az üzeneteket az összes olyan queue-ra kézbesítik, ami exchange-hez van kötve. A routing kulcsot figyelmen kívül hagyjuk és minden üzenetet az összes kötött queue-ra továbbítunk.
* **Topic exchange:** Az üzenetek egy vagy több, pontokkal elválasztott szóból álló routing kulcs alapján kerülnek a queue-kra. Az exchange a routing kulcsot összehasonlítja a hozzá kötött queue-k routing kulcsaival és a routing kulcsnak való megfeleléshez wildcard-okat használ.



### AMQP API fejlesztés

* Direct üzenetek küldése
* Cache üzenetek sorba állítása a trigger-alapú küldéshez.
* Információkat továbbíthat vagy exchange-eket köthet a kijelölt queue-okhoz.
* Kapcsolatot teremt az exchange-k között a hatékony kommunikáció biztosítása érdekében.
* Automatikus vagy manuális visszaigazolást küldhet.

### Mire használhatjuk az AMQP-t API-kban?

* Pénz feltöltése a digitális pénztárcákhoz
* Hitel-vagy betéti kártyás tranzakció kiskereskedelmi üzletekben
* Kommunikációs rendszerekben

### AMQP példa folyamat

1. Pénz feltöltése digitális pénztárcához, valamilyen szolgáltatás segítségével, mint például Paypal-al.
2. Következő lépésként autentikációs lépés, majd ezután kerül feltöltésre ténylegesen a pénz a tárcához.
3. Előfordulhat, hogy a felhasználó meggondolta magát és nem szeretne feltölteni pénzt, így rendelkeznie kell egy törlési szolgáltatással.
4. Az AMQP üzenetváltás aszinkron módon történik.
   1. Ez azt jelenti, hogy a tranzakciók kritikus fontosságú események és nem lehet bennük inkonzisztencia.
   2. Lehet még egy esemény, ami nyomon követi az üzenetek kimenetelét, hogy a következő eseményt időben lehessen elindítani.

## MQTT – Message Queue Telemetry Transport

* Egy ISO szabvány (ISO/IEC PRF 20922), publish-subscribe alapú üzenetküldő protokoll.
* Alacsony a sávszélesség igénye.
* Szükséges hozzá egy szerver, vagyis broker.

### Broker feladata

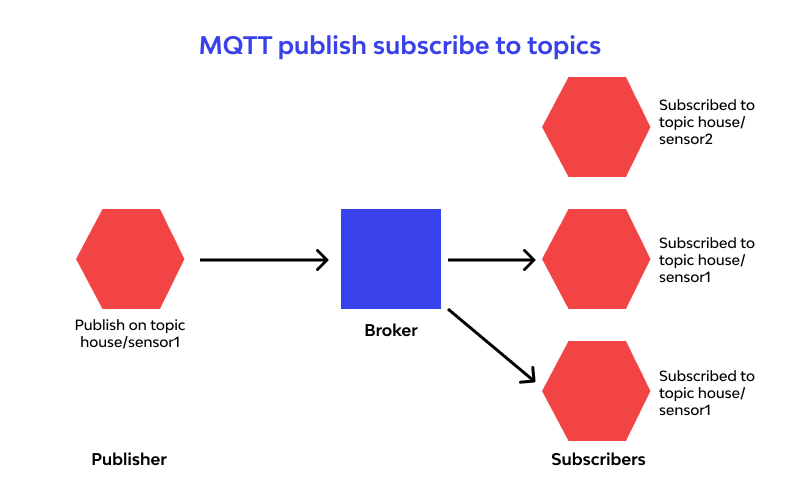
* Broker feladata a beérkező üzenetek továbbküldése a klienseknek, téma alapján.
  + Azaz a kliensek feliratkoznak különböző témákra, majd a broker a témákba érkezett üzeneteket továbbítja a feliratkozott klienseknek.

### MQTT protokoll jellemzői

* **Az MQTT-t tartják a legmegfelelőbbnek az IoT számára, mivel rendelkezik bizonyos tulajdonságokkal:**
  + Könnyű használni és azonnali használatra elkészített broker-t és klienst kínál.
  + Csökkenti az alkalmazásfejlesztési időt.
  + Megbízható kapcsolatot kínál, mert az MQTT csökkenti a csatlakozási problémát egy QoS funkcióval, ami sorba állítja az üzeneteket és elmenti őket az MQTT broker-nél és megváratja őket, amíg a célzott eszköz készen áll az elfogadásra.
    - Ez csökkenti az üzenetek rossz elhelyezésének esélyét, így az üzenet biztosan eljut a célállomásra.
  + Skálázható üzenetek

### MQTT működése

* Az MQTT modellben a kommunikáció közvetve, a PUSH/SUBSCRIBE topológián keresztül történik.
* A kliens, például az MQTT Explorer, csatlakozik a kapcsolódó publisher-hez és továbbítja az üzenetet.
* Ezt a megosztott adatot leválasztják a kliensről és továbbítják a következő szakaszba.
* A folyamat során a broker megkapja a szétválasztott adatot és továbbítja a subscriber-eknek.
* Ha a broker-subscriber kapcsolat megszakad, az üzenet a broker-nél elmentésre kerül és a kapcsolat helyreállásakor újra továbbításra kerül.
* A publisher-eknél, ha a broker kapcsolata értesítés nélkül megszakad, a broker saját maga tárolja az üzenetet a kapcsolódó subscriber-eknek.
* Mivel az MQTT eseményvezérelt, nem támogatja a folyamatos adatátvitelt és kontroll alatt tartja azt.
  + Az adatok csak akkor kerülnek továbbításra, amikor szükség van rájuk.



### MQTT használati esetek

* Remote monitoring alkalmazások fejlesztése.
* Olyan célokra használják, mint a veszélyekre való figyelmeztetés, tűzérzékelők, lopásérzékelés vagy egy cél követése.
* Valós idejű kommunikációs alkalmazások fejlesztésére is tökéletes, ilyen például a Facebook Messenger is.
  + Lightweight/”könnyű”, nem fogyasztja annyira a telefon akkumulátorját és gyorsan kézbesíti az üzeneteket.

### MQTT session/munkamenet szakaszai

* **Connection:** 
  + Ha egy MQTT kliens-broker TCP/IP kapcsolatot létesít, akkor indul el ez a session.
  + A feladatot egy szabványos vagy egyéni port használatával hajtják végre.
  + Legfontosabb, hogy biztosítani kell, hogy a TCP/IP-n ne fusson régi session.
    - Ha ez megtörténik, akkor a kapcsolat megszakad.
* **Authentication:**
  + A kapcsolat befejezése előtt a kliens ellenőrzi a szerver tanúsítványának hitelességét és jóváhagyja azt.
    - Ehhez a kliens megadja az SSL/TLS tanúsítvány adatait a broker-nek, aki ellenőrzi a kiszolgálói tanúsítvány adatait.
  + Ha az SSL/TLS nem kínálja fel a szerver tanúsítványt, akkor a felhasználók ellenőrzése vagy a hitelesítés a felhasználói hitelesítő adatokon keresztül történik, amiket plain-text formában küldenek el.
    - Ha az open-source broker-ek anonim klienseket fogadnak el, akkor a felhasználónév és jelszó szakaszban nem kínálnak beviteli adatokat.
* **Communication:**
  + Miután a hitelesítés befejeződött, az MQTT session eléri a kommunikációs szakaszt, amiben a kliensek engedélyezik a subscribe-ot és a pingelést, valamint az üzenetek/műveletek közzétetelét.
  + Az MQTT legfeljebb 256 MB méretű üzenetadatokat képes továbbítani.
* **Termination:**
  + Akkor történik meg a befejezés, ha bárki, subscriber vagy publisher véget akar vetni folyamatban lévő MQTT session-nek.
  + DISCONNECT üzenetet küld, amit a broker feldolgoz.