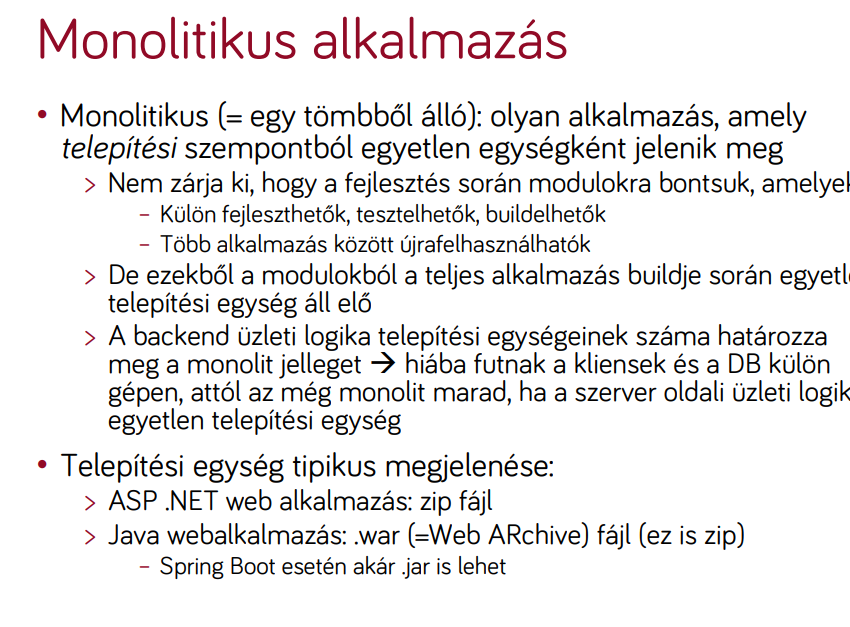
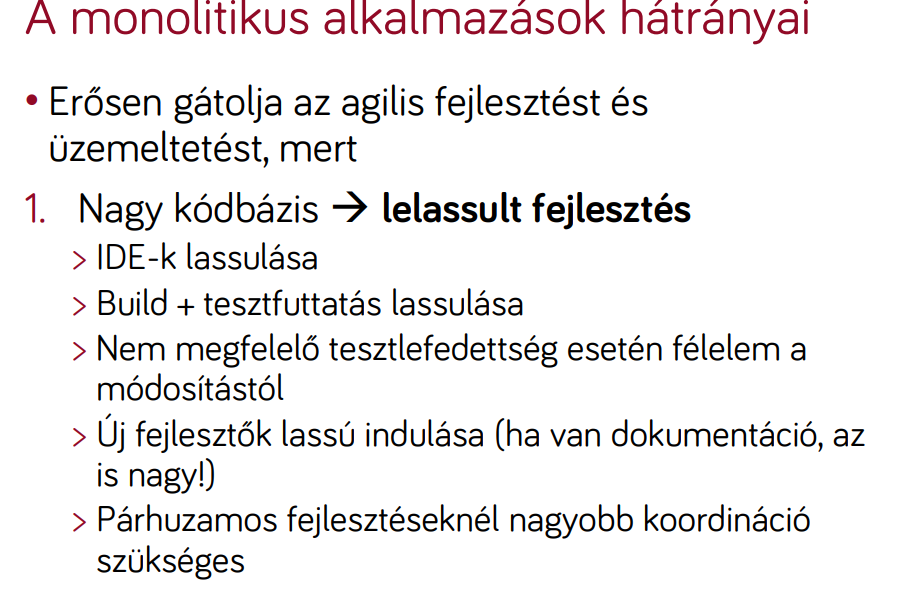
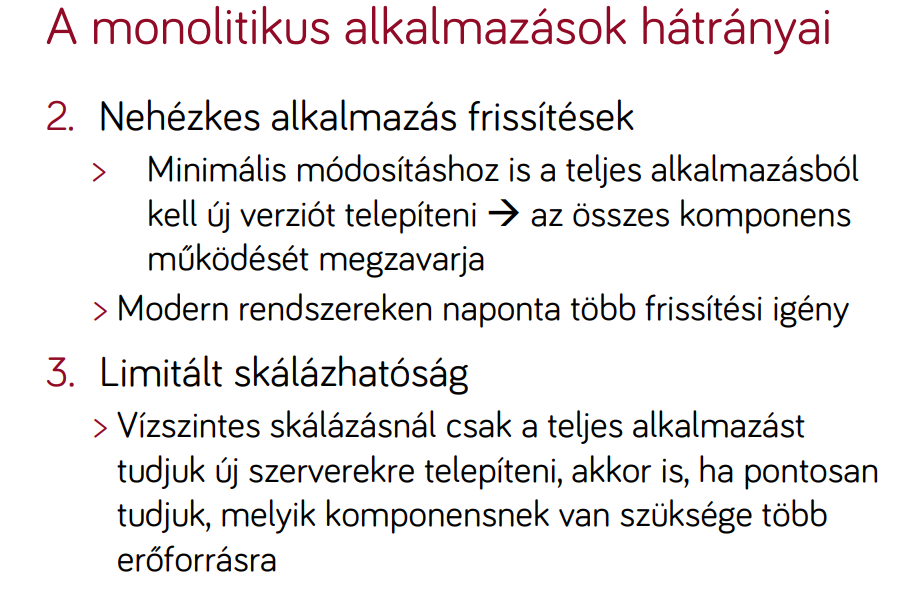
**Milyen előnyökkel és hátrányokkal jár a Mikroszolgáltatás architektúra a monolitikussal szemben? Ismertesd az alábbi Microservice tervezési minták céljait azok előnyeit, hátrányait: Shared database, Database per service, API Composition, CQRS**

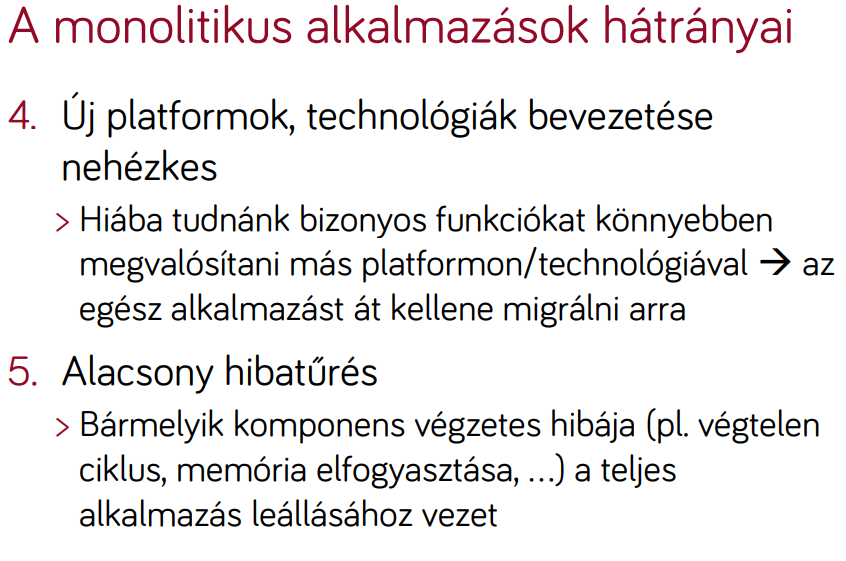
DIA 13

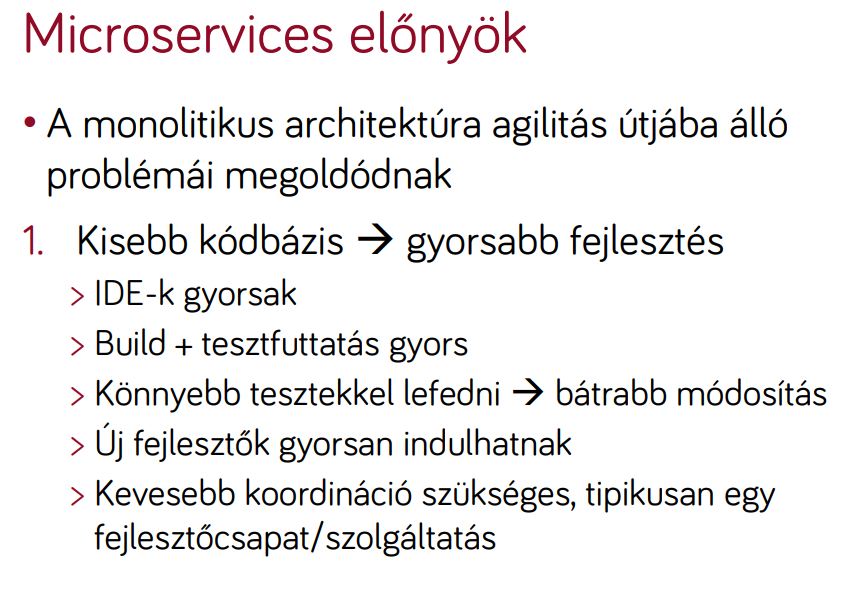
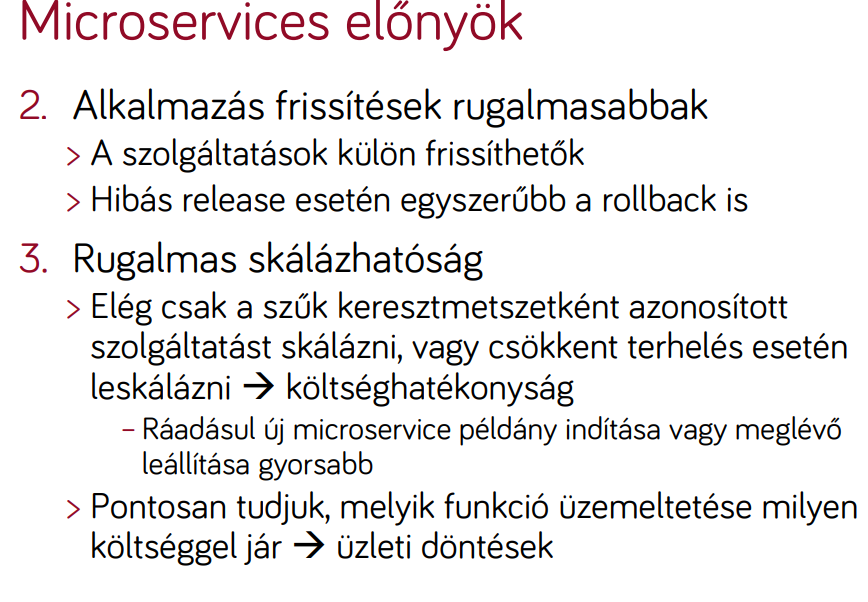
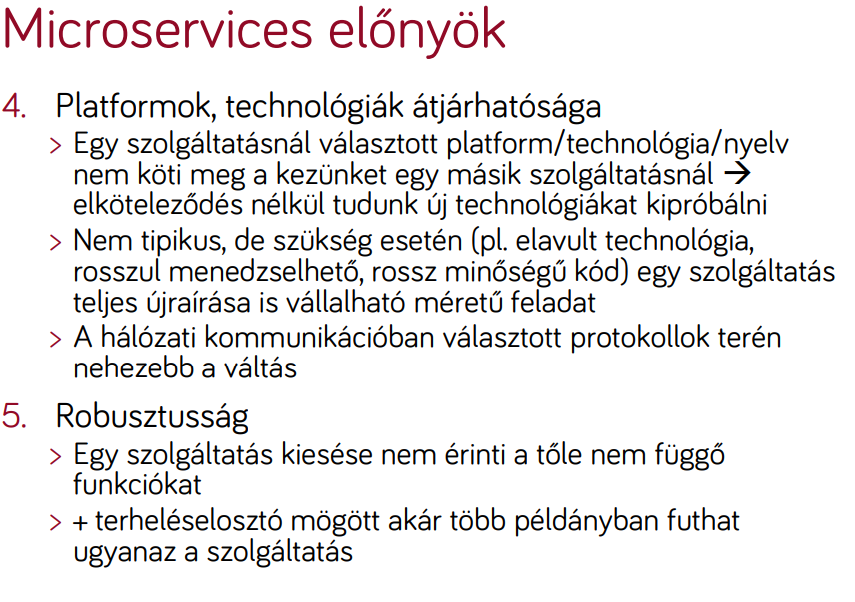
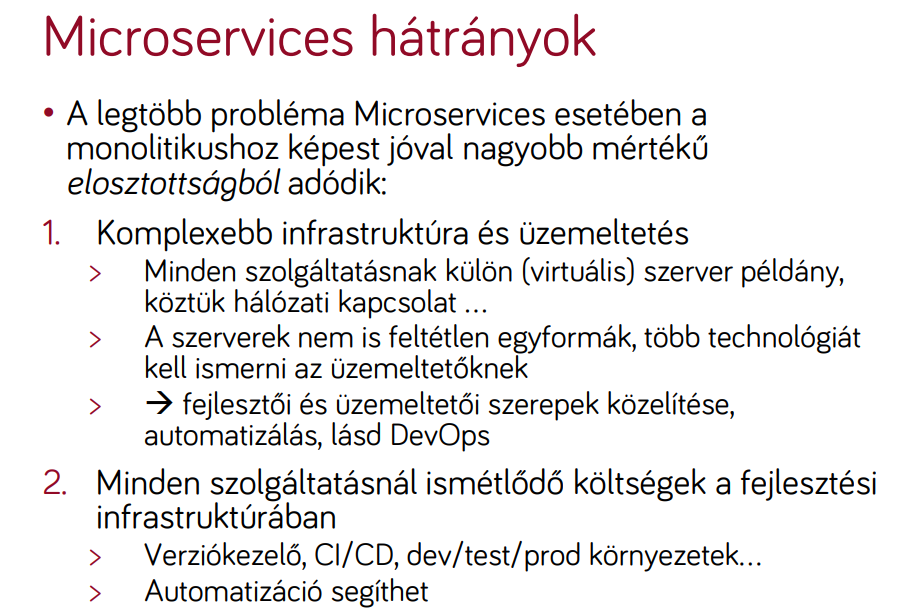
**A Monolitikus alkalmazás egy olyan alkalmazás, amely telepítési szempontból egyetlen egységként jelenik meg. De ez nem jelenti azt, hogy nem bonthatjuk az alkalmazásunkat fejlesztés során modulokra bontsuk. A modulra bontás nem befolyásolja a build-et, mivel a build után ezek az alkalmazások egyetlen telepitési egység áll elő. A monolitikus kialakitást az alkalmazásban az üzleti logika telepitési egységeinek száma határozza meg.**

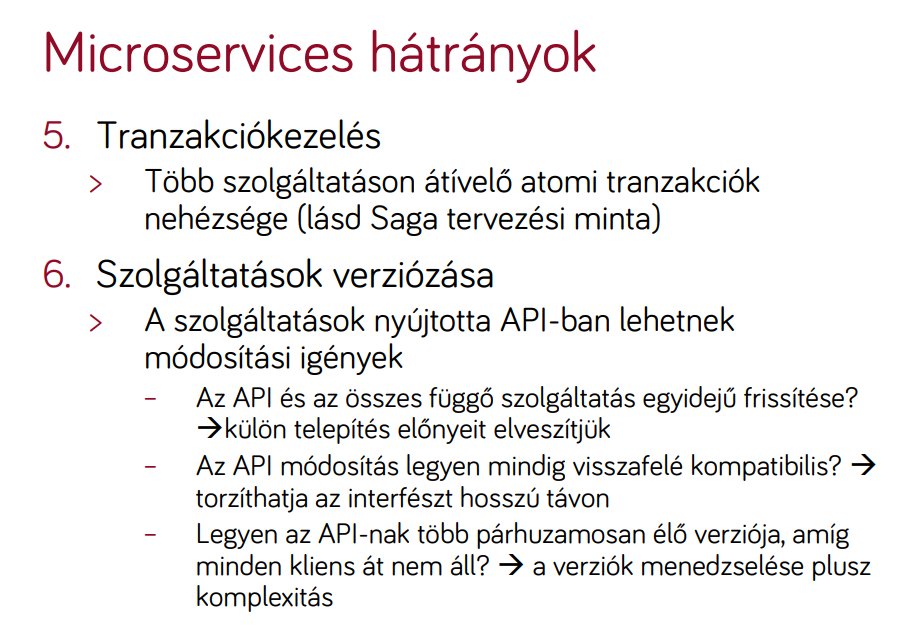
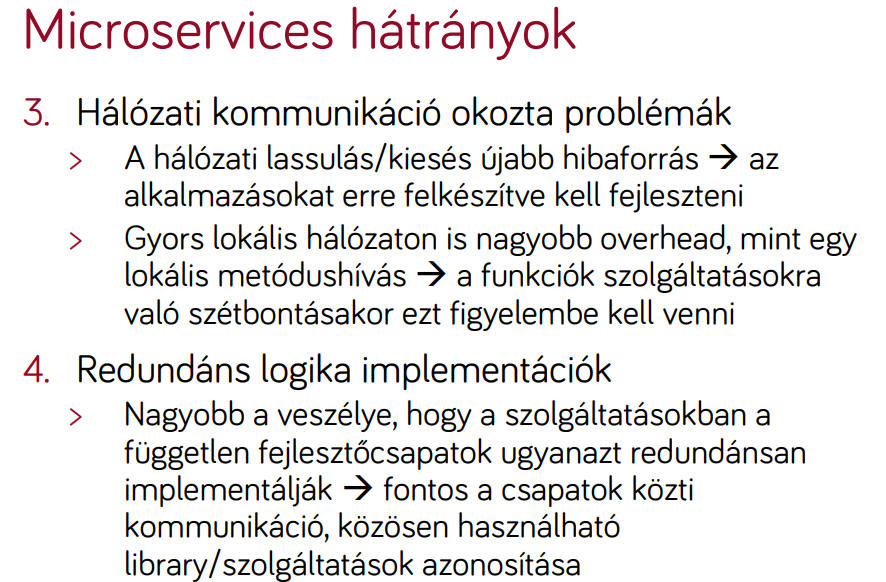
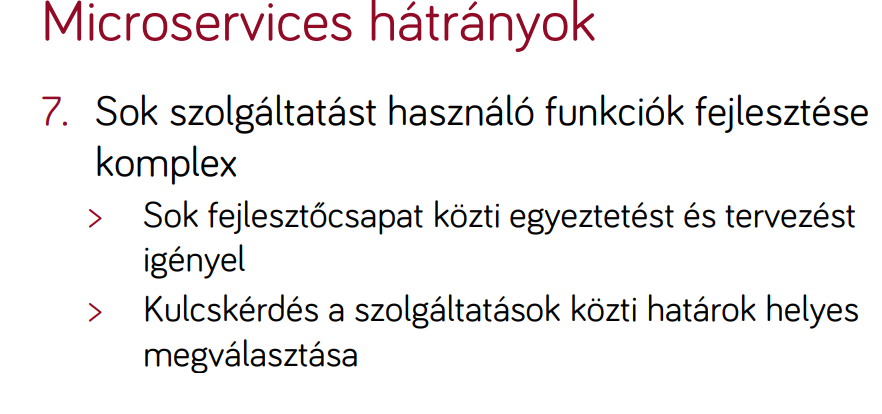
Ezzel szemben a Mikroszolgáltatás architektúrában több üzletti logikai együtt müködése a cél, ami általában HTTP-n keresztül történik.

**A Mikorszolgáltatás architectúra ebből adodó elönye, hogy több kisebb kódbázis jön létre aminek a fejlesztése és karbantartása gyorsabb. Ezzel együtt egyes frissitések emiatt a kialakitás miatt sokkal rugalmasabban telepithetőek és készithetőek el. A kialakitása miatt a mükődésben sokkal rugalmasabb skálázhatóságot érhetünk el, mivel egyes üzleti logikát képesek vagyunk skálázni.**

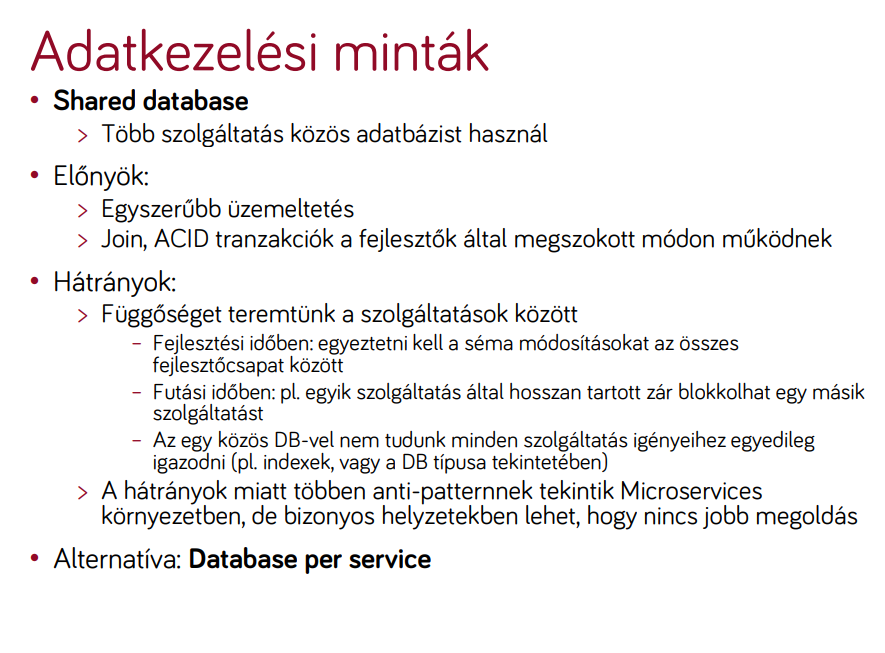
**Ezenkivül az ilyen rendszerek egyes részei nem lesznek olyan robosztusak mivel ezeket megprobáluk különálló logikákra bontani. Viszont ennek a hátránya, hogy ezek között a logikák között nehezebb a kommunikáció megoldása mint a monolitikus alkalmazásokban.**

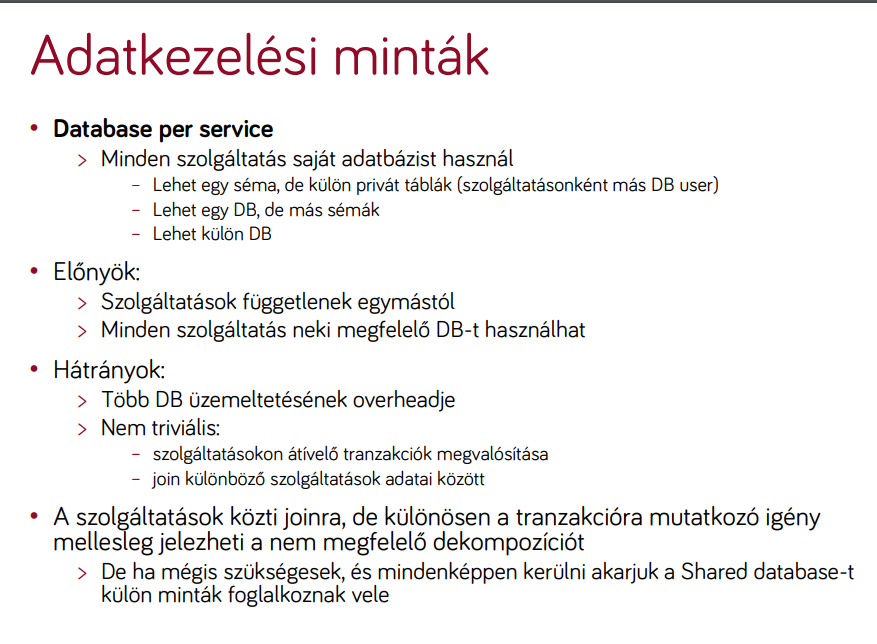
**Ezenkivül az ilyen alkalmazásokban amiatt, hogy szét bontunk miatt egy nehezebben üzemeltethető alkalmazást kapunk, ahol minden modulban ismétlődik a fejlesztési költsége például a CI/CD** **rendszereknek. Mert azok csak egyes modulokra használhatóak.**

**Ezenkivül az ilyen alkalmazásokban nehezebb a tranzakció kezelés, egyes szolgáltatások verziójának számon tartása, illetve kód szinten figyelni arra, hogy a kód ne legyen redundás és egyes logikákat különböző egyságekben ne valósitsunk meg külön.**

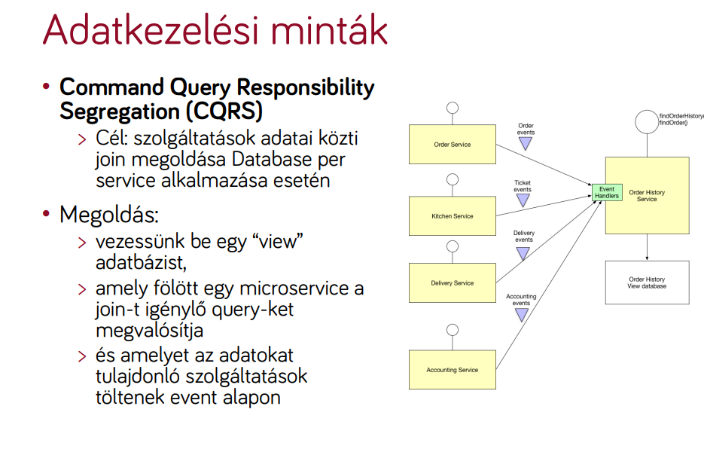


**Ezekre a hátrányokra ma már rengeteg dekompoziciós minta van amik azért léteznek, hogy pontositsák, hogy ezeket az alkalmazásokat hogyan bontsuk szét ahhoz, hogy ezeket a hátrányokat redukáljuk.**

**A Shared database -ben több szolgáltatás egy azon adatbázist használ a tranzakciók könnyitése végett. Ennek a mintának az elönye, hogy egyszerübb less az üzemeltetés és könnyebbek a lekérdezések mivel az ACID tranzakciók használhatóak a megszokott modón. Viszont ennek a mintának a hátránya, hogy függősségeket teremtünk egyes szolgáltatások között és ennek hála fejlesztési időben állandó séma egyeztetés kell, illetve futtatási időben előfordulhat, hogy egy szolgáltatás blokkol egy másik szolgáltatást. Emiatt ezt a mintát főként anti pattern-ként szokták emlegetni.**

**A Databas per service mintában minden szolgáltatásnak külön adatbzisa van amiben dolgozik. Ezzel a szolgáltatások teljesen függetlenek egymástól és nem blokkolják egymás mivel mindegyik a neki megfelelő DB-t használja. Ennek viszont a hátránya, hogy több adatbázist kell egyszerre karbantartani. Illetve megvalósitási szinten nem egyértelmü a tranzakciók megvalósitása és az összetett adatok lekérése. Ez alatt lehet érteni azt ha JOIN által több tábla egyben van.**

**Az API Composition mintában létrehozunk egy szolgáltatást ami azért felelős, hogy több szolgáltatás által uralt adatott összegyüjtse és azt egyben oda adja. Ez megoldás arra, hogy a JOIN által egyesitett adatokat, hogyan kapjuk meg. Ennek pontosan az a célja, hogy több szolgáltatás közti join-ra megoldást adjon a Database per service mintában.**

**A Command Query Responsibility Segregation azaz CQRS mintának az a célja, hogy szolgáltatások adatai közti join-t megoldja. Ez szintúgy egy jó megoldás lehet a Database per service minta használata mellett. Ennek megvalósitásának céljából készitünk egy view adatbázist amely a microservice által igényelt join query-ket megvalósitja. Ezeket majd az adatokat tulajdonló szolgáltatások töltenek event alapon.**