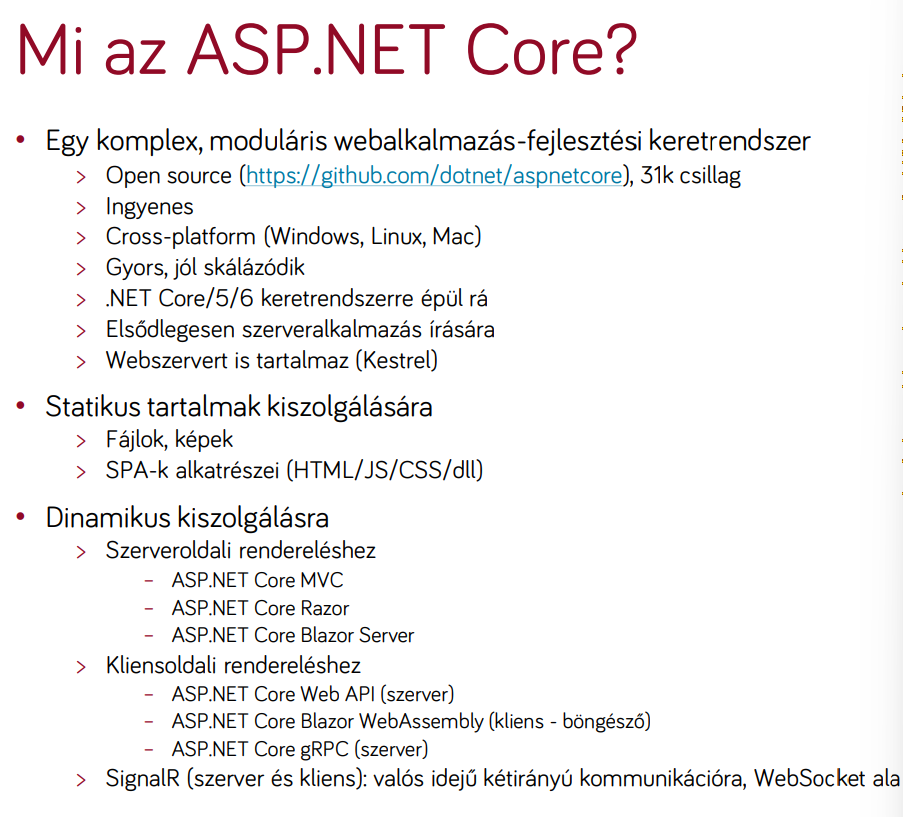
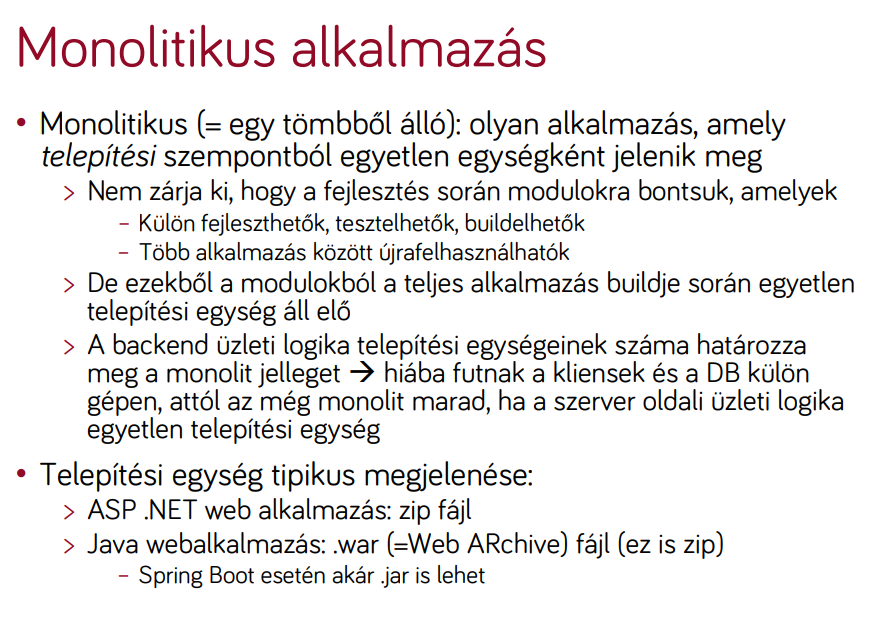
**Mi az ASP.NET Core? Mit jelent, hogy egy alkalmazás monolitikus? Mi a „clean architecture”? Sorolj fel min. 3 általános ASP.NET Core által nyújtott infrastruktúra funkciót! Hogyan indul el egy ASP.NET Core projekt? Milyen lépései vannak, hogy egy Web API-n kiajánljunk egy EF Core által nyújtott adatmodellt (lekérdezésre, módosításra)?**

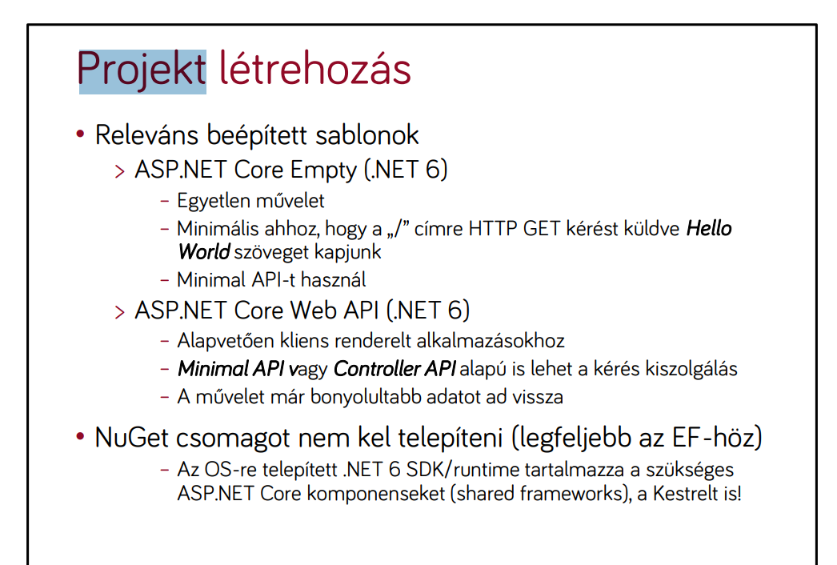
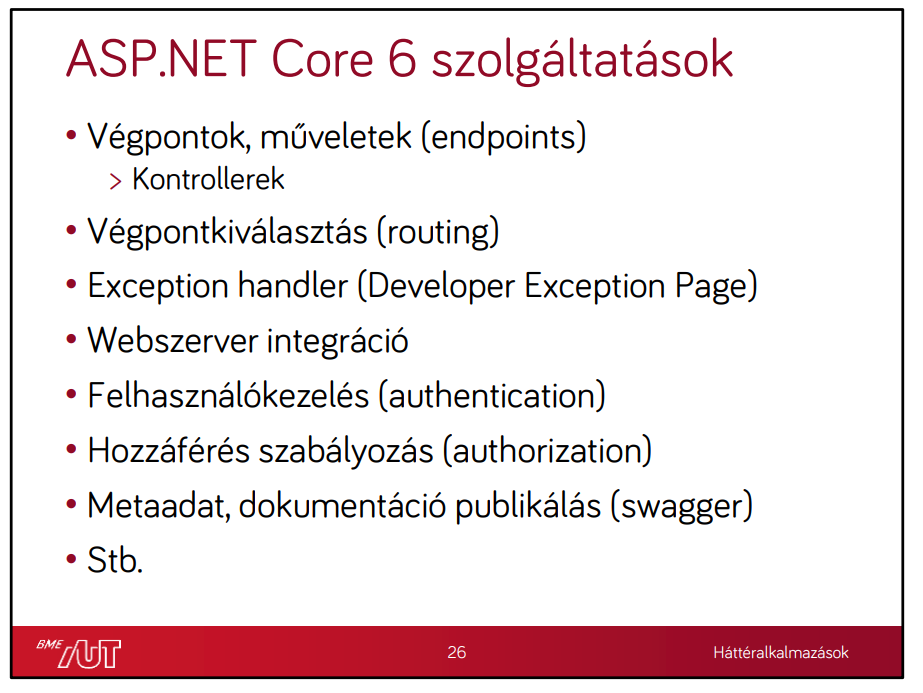
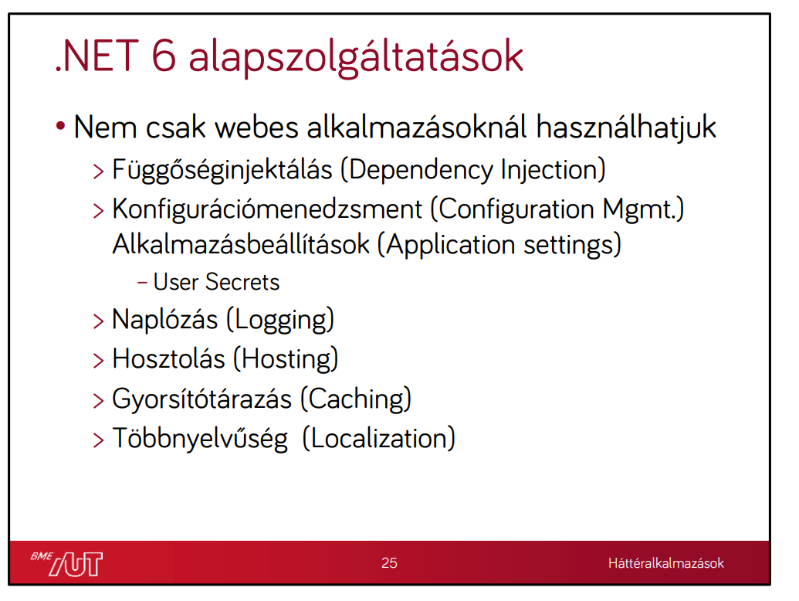
DIA 8 oldal 13

**Az ASP.NET Core egy complex moduláris webalkalmazás-fejelsztési keretrendszer. Ami open source és ingyenes. Jelenleg elérhető rengeteg platformra mint például a Linux, Mac és Windows. Az ASP.NET Core ezen kivül dinamikus kiszolgálásra is képes attól függően, hogy kliens vagy szerver oldali a renderelés.**

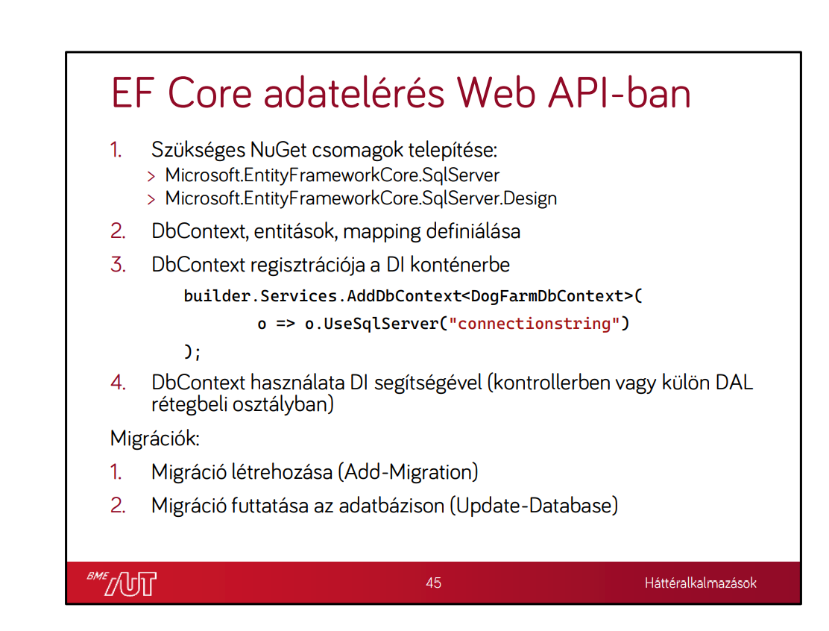
DIA 13 oldal 4

**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.**

**A clean architecture egy olyan elv ami a SOLID elvre épül és lényegében arról szól, hogy az architekruális tervezés egyik fő feladata a komponensek közötti határok meghuzása és komponensekre bontása.**

**A ASP.NET Core az alap szolgáltatásokon kivül nyújt plussz szolgáltatásokat. Ilyenek például a Végpontkiválasztás, Webszerver integráció, Felhasználókezelés**

**Elöszőr létrehozzuk a projektet ahol használhatunk sablonokat ha szeretnénk de ezeket mi magunk is kidolgozhatjuk. Ezután a projektet beállitjuk és konfiguráljuk a Program vagy Startup.cs -ben. Ezután futtatjuk az alkalmazásunkat esetleg más NuGet csomagokat telepitünk. Ezután teszteljük és végül hibát keressünk az alkalmazásban.**

**Elöszőr létrehozunk egy projektet az előbb felvetett módon. Ezután az adatmodellünket határozzuk meg. Ezzel együtt magát a DBContext-et is konfiguráljuk az adatmodellünk miatt. Ha ezzel megvoltunk akkor Startup.cs -ben beállitjuk a DBContext-et az adatbázis elérés miatt.**

**Ha ezzel megvoltunk készitünk egy controller-t a amiben meghatározzuk, hogy milyen cimen szeretnénk kiszolgálni majd a kéréseket.**

**A controller-ben ha nem bontjuk jobban szét modulokra az alkalmazásunkat akkor már magából a controllerből kiszolgálhatunk. Itt a meghatározott modón lekérjük az adatot az adatbázisból a context segitségével. Majd ezt szerkeszthetjük törölhetjük vagy vissza adhatjuk. Ha szerkesztünk bármit akkor azt a context-ben visszakell iktatni majd azt mentenünk kell és cask utána less iktatva az adatbázisban.**