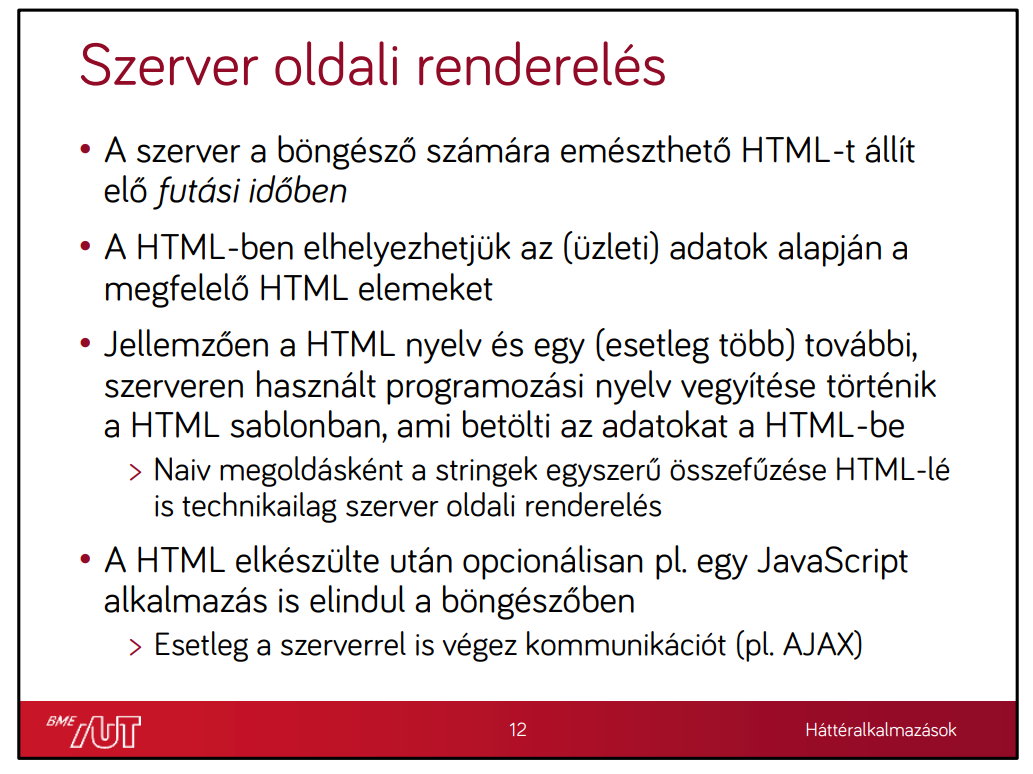
**Milyen módokon szeparálhatjuk a kliens-szerver alkalmazásrétegeket? Mi a szerveroldali renderelés? Milyen előnyei/hátrányai vannak a kliensoldalival szemben? Mi a Razor? Hogyan készül el egy HTML oldal Razor segítségével MVC vagy Razor Pages használatával?**

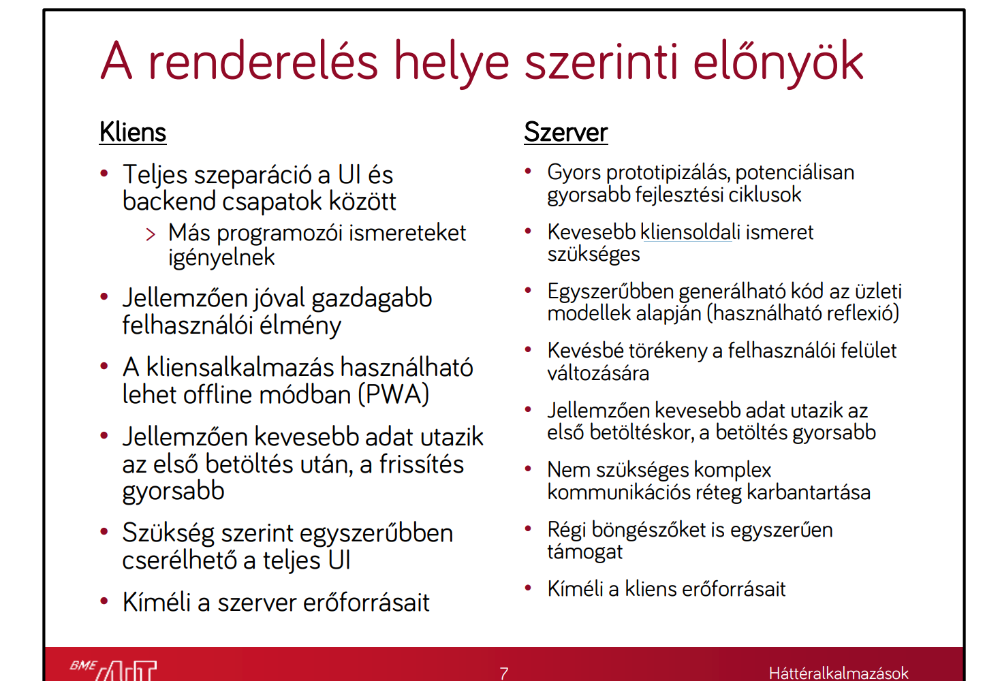
1. **Háromrétegű (Three-tier) architektúra**: Ebben a megközelítésben az alkalmazás három rétegre oszlik:
   * **Prezentációs réteg**: Az a rész, amely az ügyféllel közvetlenül kommunikál, például egy böngészőben futó webalkalmazás.
   * **Üzleti logika réteg**: Ahol az üzleti szabályok és az alkalmazás logikája működik.
   * **Adatréteg**: Az adatok tárolásának és kezelésének helye, például egy adatbázis.
2. ***MV* Minták** (MVC, MVVM, MVP, stb.)\*\*:
   * **MVC (Model-View-Controller)**: A modell tárolja az adatokat, a nézet felelős a megjelenítésért, a vezérlő pedig kezeli az interakciókat.
   * **MVVM (Model-View-ViewModel)**: A nézetmodell kapcsolatot teremt a modell és a nézet között, lehetővé téve az adatkötést és a nézet aktualizálását.
   * **MVP (Model-View-Presenter)**: A presenter összeköti a modellt és a nézetet, hasonlóan az MVVM-hez, de nagyobb hangsúlyt helyez a nézetfüggetlenségre.
3. **Mikroszolgáltatások**: A nagyobb alkalmazás felosztása kisebb, önálló szolgáltatásokra, amelyek egymással kommunikálnak, gyakran HTTP vagy üzenetközpontú közvetítés révén. Ez lehetővé teszi a szolgáltatások független skálázását és karbantartását.
4. **Backend-for-Frontend (BFF)**: Az elkülönítés olyan módja, amelyben az alkalmazás különböző típusú klienseket szolgál ki (például web- és mobilalkalmazásokat) különböző backend rétegeken keresztül, hogy jobban illeszkedjen a kliensspecifikus igényekhez.
5. **Service-Oriented Architecture (SOA)**: Az alkalmazás különböző részeit elkülönítik külön szolgáltatásokra, amelyek lazán kapcsolódnak egymáshoz. Ez lehetővé teszi a komponensek cseréjét vagy független fejlesztését.
6. **Domain-Driven Design (DDD)**: Az alkalmazást a tartományokon vagy üzleti területeken alapuló koncepciók alapján szervezik, így a különböző tartományok elkülönülnek egymástól, de mégis kommunikálhatnak.

DIA 9

**A szerver oldali renderelés annyit jelent, hogy a szerver a bőngésző számára érthető HTML-t állit elő futtatási időben.**

**Ebben persze ebben elhelyezhetünk üzleti logika alapját amit HTML és programozási nyelv vegyitésével tehetünk meg.**

DIA 8

**A szerver oldali rendelésnek fő előnye a kliens oldalival szemben, hogy sokkal gyorsabb fejlesztést érhetünk el. Mivel ehhez sokkal kevesebb kliens oldali ismeret szükséges. Ezenkivül ebben az esetben sokkal kevesebb adat utazik a betöltéskor igy ez egy gyorsabb müködést eredményez. Azonban ez egy jóval szegényebb felhasználói élményt nyújt a felhasználó számára mint a kliens oldali renderelés. Illetve ebben az esetben a szerver oldal dolgozik majdnem midnenen igy a kliens oldali reőforrás igény csökken, viszont a szerver oldali annyival nő.**

**A Razor egy szintaxis és templating engine, amelyet az ASP.NET Core (és korábban az ASP.NET MVC) keretrendszerben használnak a dinamikus weboldalak készítéséhez. A Razor szintaxis lehetővé teszi a C# és HTML kódok keverését egyetlen fájlban, így a fejlesztők könnyen készíthetnek dinamikus, adatvezérelt weboldalakat.**

