A szoftverfejlesztésben a réteg egy olyan fogalom, mely egy bizonyos absztrakció szinthez kötötten definiálja az adott szoftver egy részét. Több ismert különböző rétegzési módszer, minta ismert mindegyik célja az, újrahasznosítható függőség alapú fejlesztés folyjék, a kód karbantarthatósága érdekében.

A klasszikus három rétegű megközelítés során két ilyen rétegzési módot tekintettünk meg a tanulmányaink során.

Az egyik ilyen megközelítés az MVVM (Model View ViewModel) megközelítés.

Ez tipikusan grafikai interfésszel rendelkező szoftverek fejlesztés során használt, például WPF-ben, UWP-ben.

A felhasználó a View réteggel lép kapcsolatba, ennek pedig a vezérlőjén (pl Gombon) keresztül, jut el az adat a ViewModel réteghez végül pedig a Model réteghez. Fontos, hogy a View és ViewModel között adatkötés van. (Ez lehet egy vagy kétoldalú adatkötés)

Az MVVM pattern esetében a Model általánosságban „üzleti logikai modelt” jelent.

A Model a ViewModelt valamint a ViewModel a Viewt eseményeken keresztül változtatja.

Tehát a legfőbb cél az, hogy a View (UI felület) és a Model (logika) úgy tudjon kommunikálni egymással, hogy ne függjenek egymástól, ezért a kettő közé bekerül a View Model réteg, ami ezt a kommunikációt lehetővé teszi.

A View tud a ViewModelről a ViewModel tud a Modelről de ez visszafelé nem igaz a Model nem tud a ViewModelről, a View Model nem tud a Viewról, tehát ez egy egyirányú függőség.

Egyszerű MVVM példa:

A View-n szerepel egy szöveg (label) valamint egy gomb. Mind a kettő adatkötéssel kötve lett a ViewModel rétegben. A gomb lenyomását követően a ViewModel a Modelhez fordul. A model végrehajta az adott logikai műveletet, ezt visszaadja a ViewModelnek aki ennek megfelelően frissíti a szöveges mező tartalmát. A szöveges mező a kötésnek köszönhetően pedig automatikusan frissül.

Ez megengedi a különböző rétegek cseréjét, újrahasznosíthatóságát, valamint segíti a karbantarthatóságot. (Pl. int érték kötésénél a ViewModel-t nem érdekli, ha cseréljük az input mezőt counterről sliderre, mind a kettőt meg fogja enni a bindolást követően)

A másik megtekintett az MVC. (Model View Controller)

Tipikusan webes alkalmazásoknál használt rétegzés. (Pl. ASP.NET MVC, SPRING, Symfony stb.)

MVC esetében a felhasználó a Controller réteggel lép kapcsolatba, tipikusan webes kérések formájában. Itt a Controller rétegben vagy mögötte történik az üzleti logikai módosítás. A Model réteg egyfajta adathalmazként szolgál, ezen adathalmaz felhasználásával alakul ki a View. A View a megjelenített UI felület, a controller hívás egy viewval vagy view kombinációkkal tér vissza ez alakítja ki végül azt, amit a böngésző megjeleníti a felhasználó számára.

ASP.NET MVC esetében az alkalmazás Razor pages-ben definiálja a View-t. Ez lehetőséget ad objektumokat átadni és felhasználni a viewnak. Fontos azonban, hogy itt nem szabad üzleti logikát írni, mivel ez rétegsértésnek minősül. Itt tipikusan megjelenítéssel kapcsolatos kód kerül implementálásra. Lehetőség van loop-ok, elágazások stb. használatára. (pl foreach egy lista megjelenítésre, elágazás eldöntés alapú megjelenítésekre)

A REST hívás során is ez a rétegzési logika van jelen, viszont abban az esetben általánosságban JSON és/vagy XML üzenetekben szokás válaszokat küldeni.