# Webes alkalmazások fejlesztése

2. feladat - Aukciós

## Feladat

Készítsünk egy aukciókkal foglakozó online rendszert, ahol különböző tárgyakra licitálhatnak a felhasználók.

*1. részfeladat:* a webes felületet a licitálók használhatják a tárgyak megtekintésére, illetve ajánlattételre.

- A főoldalon a legutoljára meghirdetett 20 tárgy listázódik (név, hirdető, jelenlegi licitösszeg), de lehetőségünk van kategóriánként megtekinteni az összes (még aktív) hirdetést. Egy oldalon legfeljebb 20 tárgy látható (a meghirdetés dátuma szerint csökkenő sorrendben), az oldalak között lapozni lehet. A lista szűrhető név(részlet)re. A tárgyat kiválasztva megjelennek a részletes adatok (kép, leírás, lezárás és meghirdetés dátuma, aktuális licit).
- A licitálónak előbb regisztrálnia kell az oldalon (név, telefonszám, e-mail cím, felhasználónév, jelszó, megerősített jelszó), majd ezt követően bejelentkezhet. A bejelentkezett felhasználó kijelentkezhet.
- Bejelentkezést követően érhető el a licitálás minden aktív tárgynál. Licitáláshoz ki kell jelölni a tárgyat és meg kell adni az összeget. Első licit esetén az összegnek a minimális licitnek kell lennie, később pedig mindenképpen nagyobbnak kell lennie a korábbi liciteknél. Egy felhasználó tetszőlegesen sokszor licitálhat egy tárgyra. A licitet visszavonni nem lehet.
- A felhasználó külön listázhatja azokat a tárgyakat, amelyekre legalább egyszer licitált. A listában külön megjelöljük az aktív tárgyakat, valamint azokat, ahol vezeti a licitet.

## Elemzés

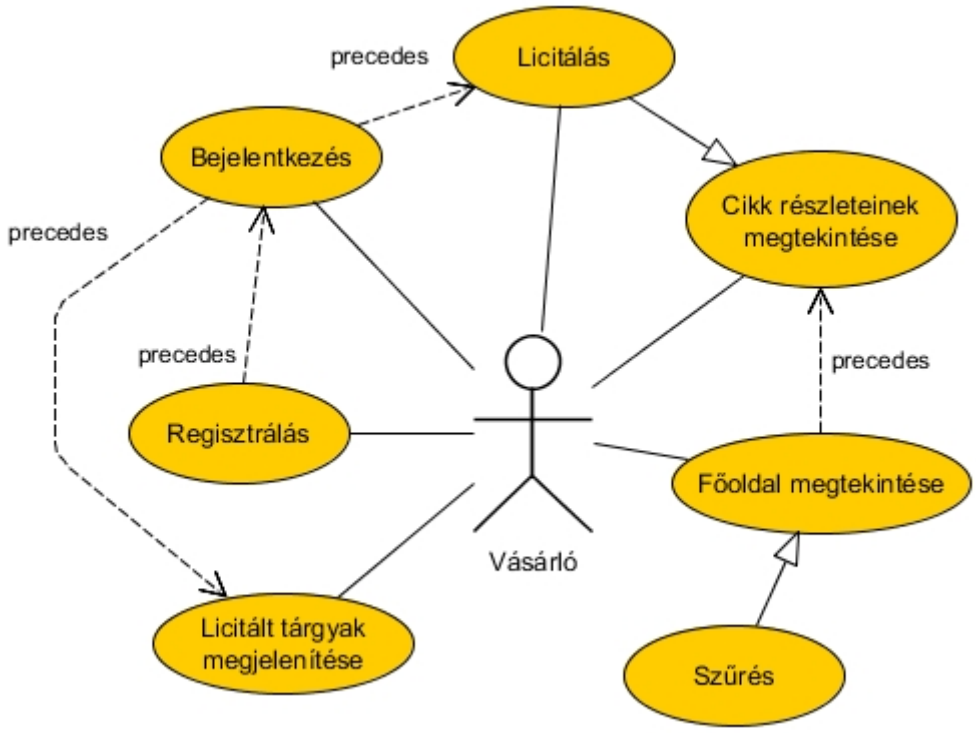
- Az alkalmazást ASP.NET Core 2.1 MVC architektúrával készítjük el
- A perzisztencia réteg Transact-SQL, code-first Entity Framework 4.0 segítségével
- A frontent Razor és ASP.NET Core cshtml

## Tervezés

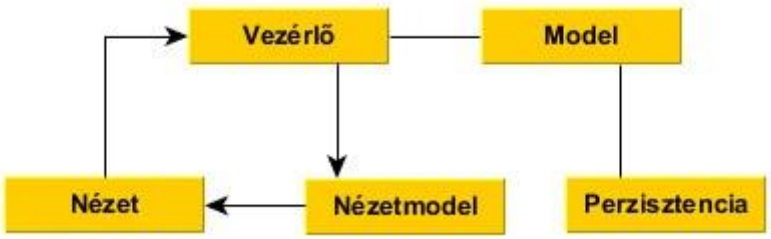
- Két felhasználócsoport van, a vevők és az eladók, de az ASP.Net Core keretrendszer Identity funkciójának kihasználásához őket egy osztály, és az adatbázisban egy tábla reprezentálja. A két csoport elkülönítése a program egy invariánsa.
- A lapozási működést elkülönítjük egy újrafelhasználható osztályba, a *PagingViewModel<T>*-be, amely *T* típusú elemeket aggregál oldalakon egy *IQueryable<T>* lekérdezés alapján.
- A képeket feltöltéskor bájt tömbökként az adatbázisba mentjük, lekéréskor a kép HTML-elem forrás attribútumába töltjük base64-ben elkódolva.

- A tesztelést megkönnyítendő, a szervert ellátjuk egy parancssori paraméterrel, amely törli és a mintaadatokkal újra feltölti az adatbázist. Parancssorból ez a `dotnet run --seed true` parancs kiadásával történik.
- A felhasználókkal és az árucikkekkel kapcsolatos műveleteket kiemeljük saját szolgáltatásba, megvalósításukat a *PortalService* és a *BuyerService* osztályok tartalmazzák, a kontrollerekbe való befecskendezésük az alapértelmezett IoT konténerrel történik.

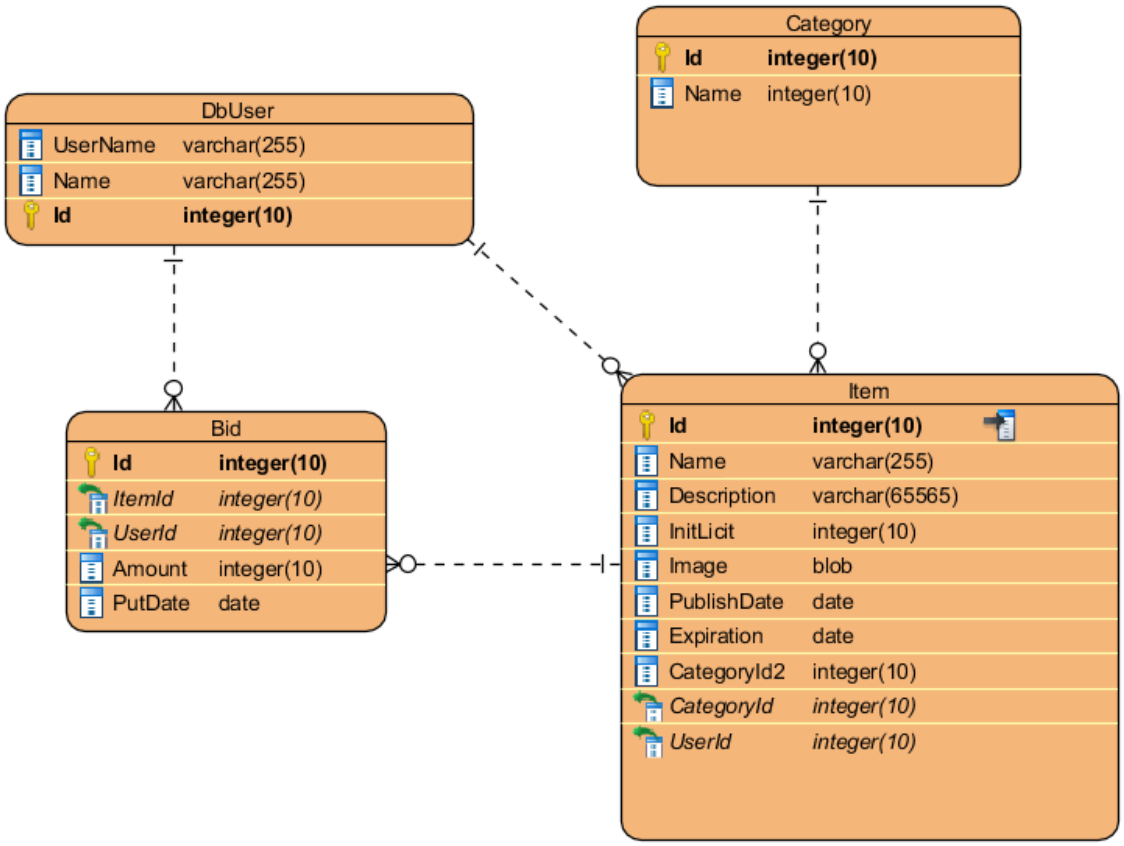
## Használati esetek




## Struktúradiagram



# Entitásdiagram



atadi96@gmail.com
2. feladat

# Osztálydiagram