ELTE IK, Programtervező informatikus BSc
Webes Alkalmazások Fejlesztése

Rosanics Márton, 2018
marton.rosanics@gmail.com

Feladat: Ételrendelés

1. rész:

Készítsük el egy étel-futár vállalat rendeléseket kezelő rendszerét! A megrendeléseket a vásárlók a webes felületen keresztül adhatják le a cég felé.

- A weblap főoldalán megjelennek a kategóriák (pl. levesek, pizzák, üdítők), valamint a 10 legnépszerűbb (legtöbbet rendelt) étel/ital.
- A kategóriát kiválasztva listázódnak a tételek (név és ár kíséretében), amelyek szűrhetőek név(részlet)re. Ételek esetén leírás is van. Külön meg vannak jelölve a csípős, illetve vegetáriánus ételek.
- Ételek és italok tetszőleges számban helyezhetőek a kosárba egy adott felső korlátig (20.000 Ft), afelett több terméket nem lehet a kosárba helyezni.
- A kosár tartalma bármikor megtekinthető, ekkor látszódnak a felvett tételek, illetve látható az összár. Bármely tétel kivehető a kosárból.
- A rendelést törölhetjük, illetve leadhatjuk. Utóbbi esetben meg kell adnunk a nevünket, címünket, illetve telefonszámunkat, majd elküldhetjük a rendelést.

2. rész:

További feladat egy asztali grafikus felület elkészítése, melyet az alkalmazottak használhatják a rendelések, illetve a weblap tartalmának adminisztrálására.

- Az alkalmazott bejelentkezhet (felhasználónév és jelszó megadásával) a programba, illetve kijelentkezhet.
- Bejelentkezve listázódnak a leadott, illetve teljesített rendelések (leadás időpontja, teljesítés időpontja, név, cím, telefonszám, összeg), egy rendelést kiválasztva pedig listázódnak a tételek. A leadott rendelés teljesítettnek jelölhető, ekkor a rendszer rögzíti a teljesítés időpontját is. A lista szűrhető csak teljesített, illetve csak leadott rendelésekre, továbbá a rendelő nevére, illetve cím(részlet)re.
- Lehetőség van új étel, illetve ital hozzáadására (név, ár, illetve étel esetén leírás, csípős/vegetáriánus tulajdonságok megadásával). Az egyértelműség miatt nem engedélyezett több ugyanolyan nevű étel/ital felvitele.

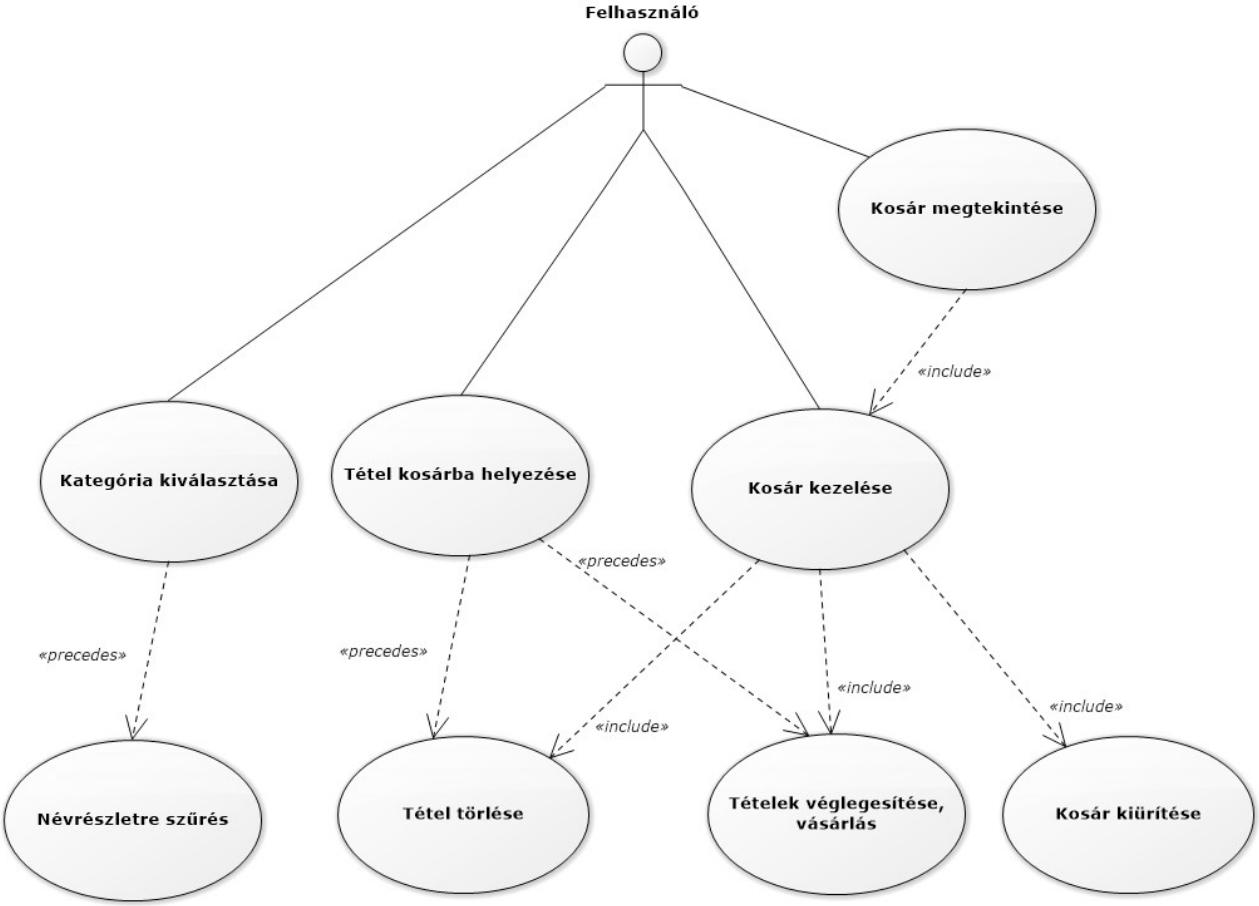
Az adatbázis az alábbi adatokat tárolja:

- kategóriák (név);
- ételek és italok (név, kategória, leírás, ár, csípős-e, vegetáriánus-e);
- munkatársak (teljes név, felhasználónév, jelszó);
- rendelések (név, cím, telefonszám, megrendelt ételek, teljesített-e).

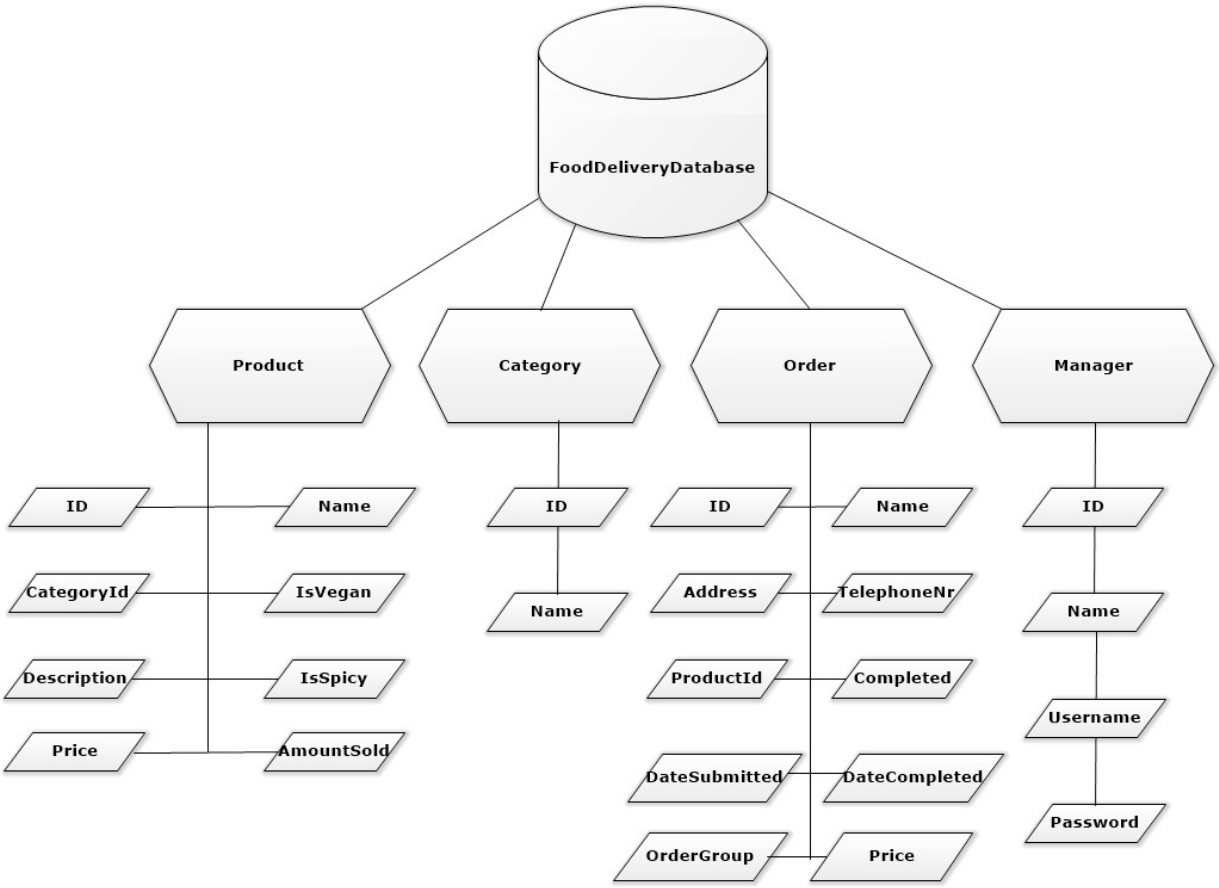
1. rész elemzése:

- A webes felhasználói felületet ASP.NET Core-ban, MVC architektúrával valósítom meg. (A project a *FoodDeliveryService* nevet kapta.)
- Az alkalmazás által használt adatbázishoz Entity Framework segítségével kapcsolódok, az adatbázist kezdetben code-first megközelítéssel hozom létre.
- A fő ablakon a 10 leggyakrabban várásolt étel fog megjelenni, ezek mellett azonban biztosítom kell navigációs lehetőségeket is. Az ablak bal oldalán található menüsorból lehet majd kategóriánként szűrni a különböző ételeket, valamint az ablak tetjén lévő fejlécnél helyezem el a kosarat, melynek tartalma bármikor megtekinthető lesz. A kosárból visszalépéskor a felhasználó arra a lapra fog kerülni ahonnan a kosárra rákattintott, a rendelés leadásánál pedig a főoldalra kerül majd vissza.
- Egy kategóriát kiválasztva listázódnak az adott kategóriához tartozó termékek. Megjelenik egy szövegdoboz is, mellyel az ételek névrészletére lehet szűrni (a kategórián belül). Itt, illetve a főoldalon is lehetőség van a tételeket a kosárba helyezni.
- A kosár tartalmát munkamenet-állapotként kezelem. A vásárlás véglegesítése előtt a felhasználónak meg kell adnia az adatait, ezek helyessége ellenőrizve lesz.

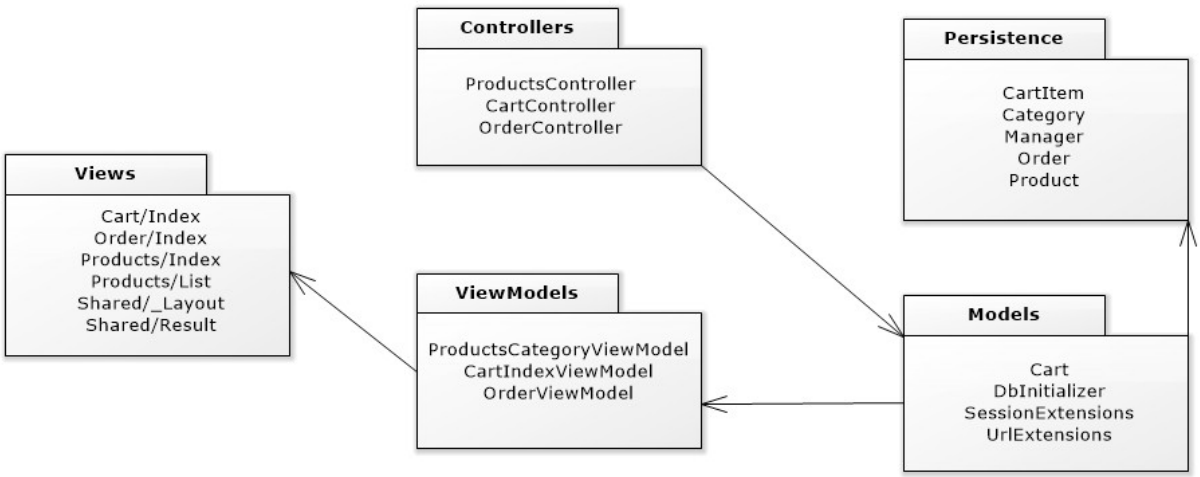
1. rész felhasználói esetei:



Szerkezet (1. rész):
SQL adatbázisban tárolom az ételekre vonatkozó adatokat. A feladat leírásában felsorolt táblák egy kis kiegészítésre szorulnak; például a Products táblában még az eladott termékek mennyiségét is tároljuk, hiszen erre szükség lesz a főoldalon való sorbarendezéshez. Az adatbázis tervezésekor továbbá figyelembe kellett venni azt is, hogy a feladat 2. részének is ugyanúgy megfeleljen, mint az első részének, hiszen a két részfeladat ugyan azzal az adatbázissal fog dolgozni.



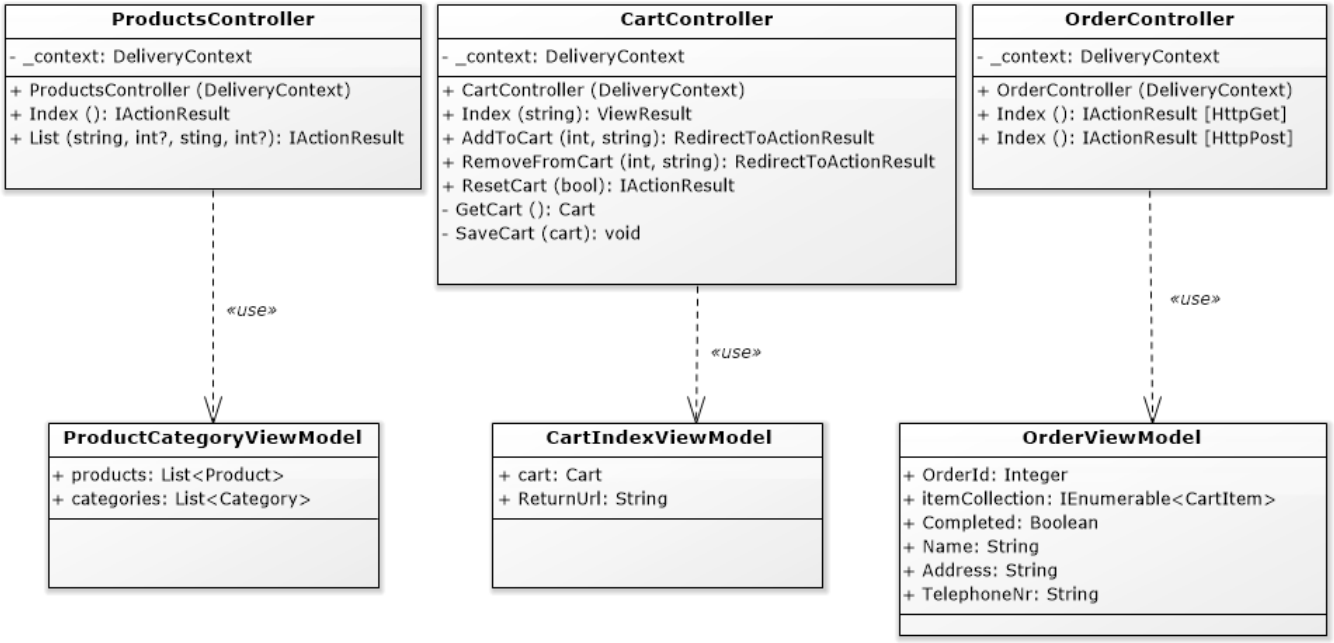
Az MVC architektúrának megfelelően a Conrollerek, a Modelek és a View-k számára külön névtereket hoztam létre az alkalmazáson belül:

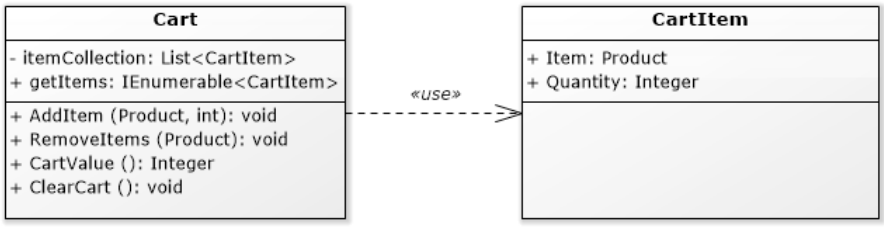


Az alkalmazás elsősorban a Products controllert használja, ennek az Index actionje kezeli a főoldalt, és a List action felelős a szűrésért (kategóriánként és névrészletre egyaránt). Létrehoztam egy ProductsCategoryViewModelt is, mely a controller és a nézet között egy extra rétegként funkcionál.

A CartController felelős a bevásárló kocsi karbantartásáért: segítségével adhatunk hozzá illetve törölhetünk tételeket, nézhetjük meg a kocsi tartalmát vagy éppen üríthetjük ki azt.

Az OrderController a rendelés leadásánál jön képbe, a felhasználó által megadott adatokat kezeli és a rendeléseket menti az adatbázisba.

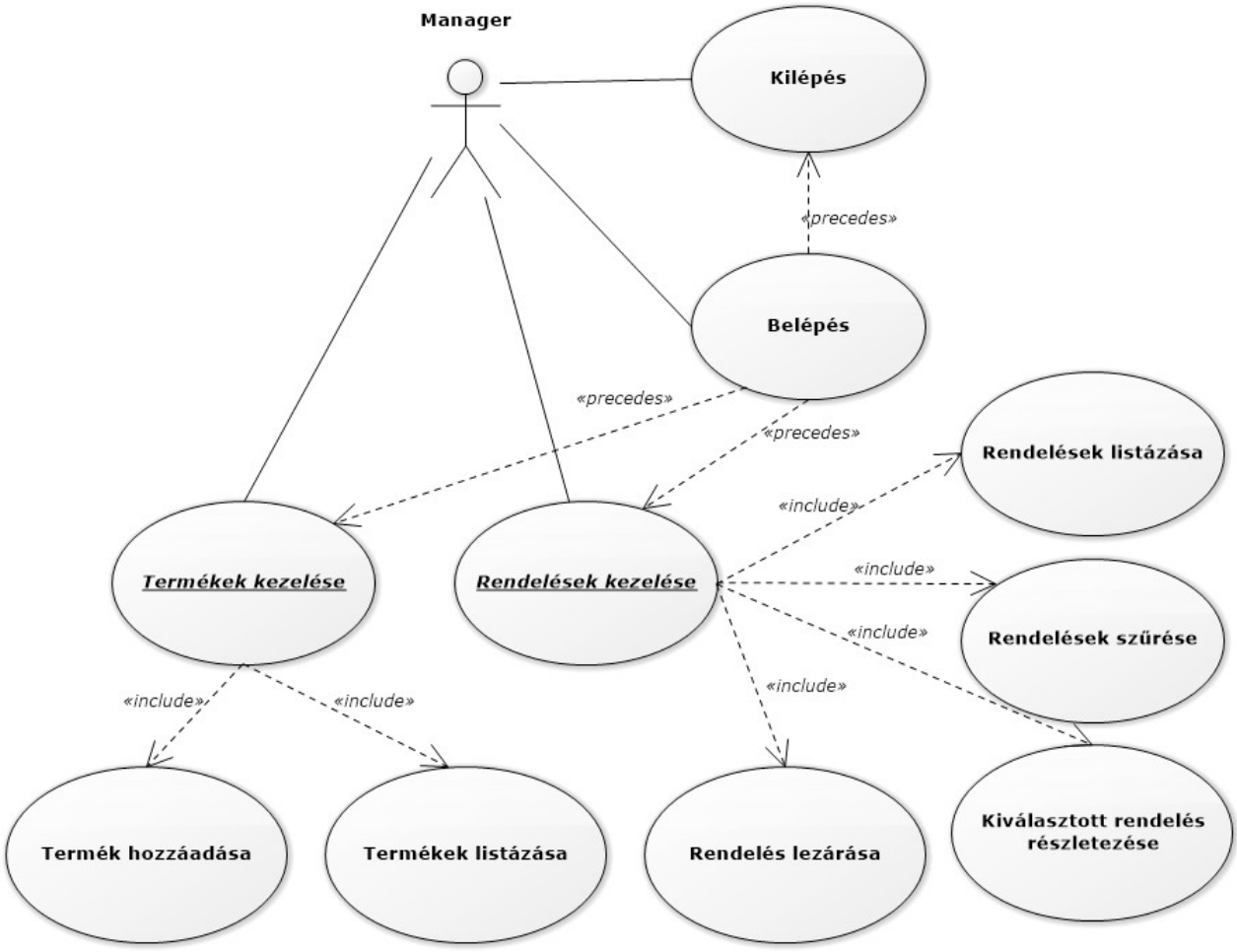




2. rész elemzése

- A feladat megvalósítását 3 projecten keresztül történik: az első (*FoodDeliveryAPI*) egy ASP.NET Core WebAPI, mely segítségével létrehozzuk a webszolgáltatást. A második (*FoodDeliveryAdmin*) egy WPF grafikus felületű asztali alkalmazás, mely MVVM+P (*Model, View, ViewModel + Persistence*) architechtúrában készül. A harmadik (*FoodDeliveryData*) pedig egy project mely a Data Transfer Object osztályokat tartalmazza, melyek segítségével a kommunikáció a különböző rétegek között lebonyolítható.
- Az adatbázishoz a webszolgáltatás Entity Framework segítségével kapcsolódik.
- Az alkalmazás indulásakor felugró bejelentkező-ablak után jelenik meg a főablak, amin elhelyezett nyomógombokon, illetve a menüsoron keresztül érhetőek el a különböző funkcionalitások.
- Új termék hozzáadása esetén új ablak ugrik fel, ahol meg lehet adni az új termék adatait, illetve meg lehet tekinteni az eddigi termékek listáját.
- A főablak bezárásakor lesz vége a munkamenetnek.

2. rész felhasználói esetei:

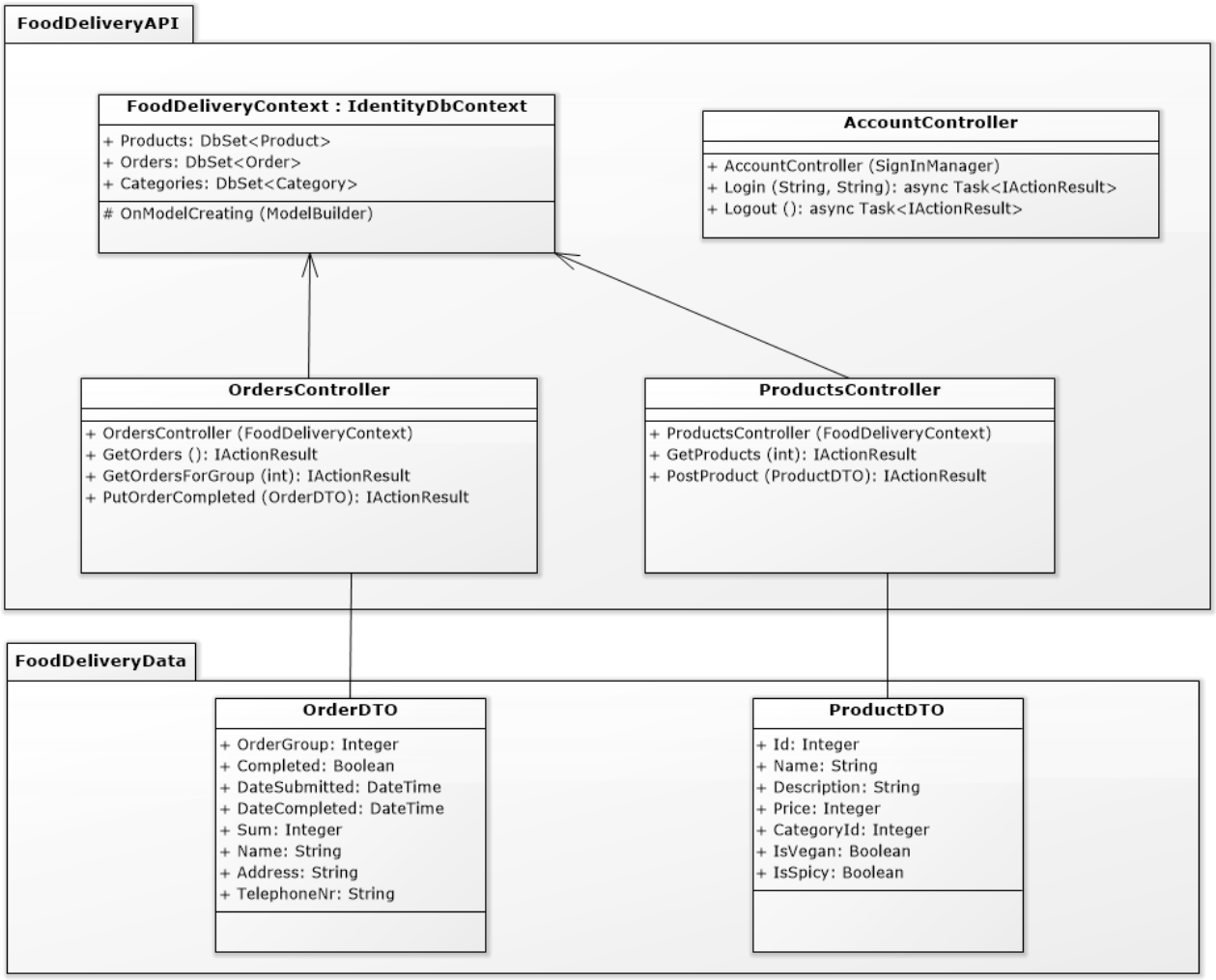


Szerkezet (2. rész):
SQL adatbázisban tároljuk az ételekre vonatkozó adatokat; az adatbázis felépítését az 1. részben taglaltam. Az elsődleges szempont az volt, hogy mindkét rész (*customer* és *manager*) akár párhuzamosan, egymástól függetlenül is működhessen.

Az adatbázissal a FoodDeliveryAPI tartja a kapcsolatot, míg a WPF alkalmazás és az API közötti adatátvitel az OrderDTO és a ProductDTO osztályok segítségével történik.

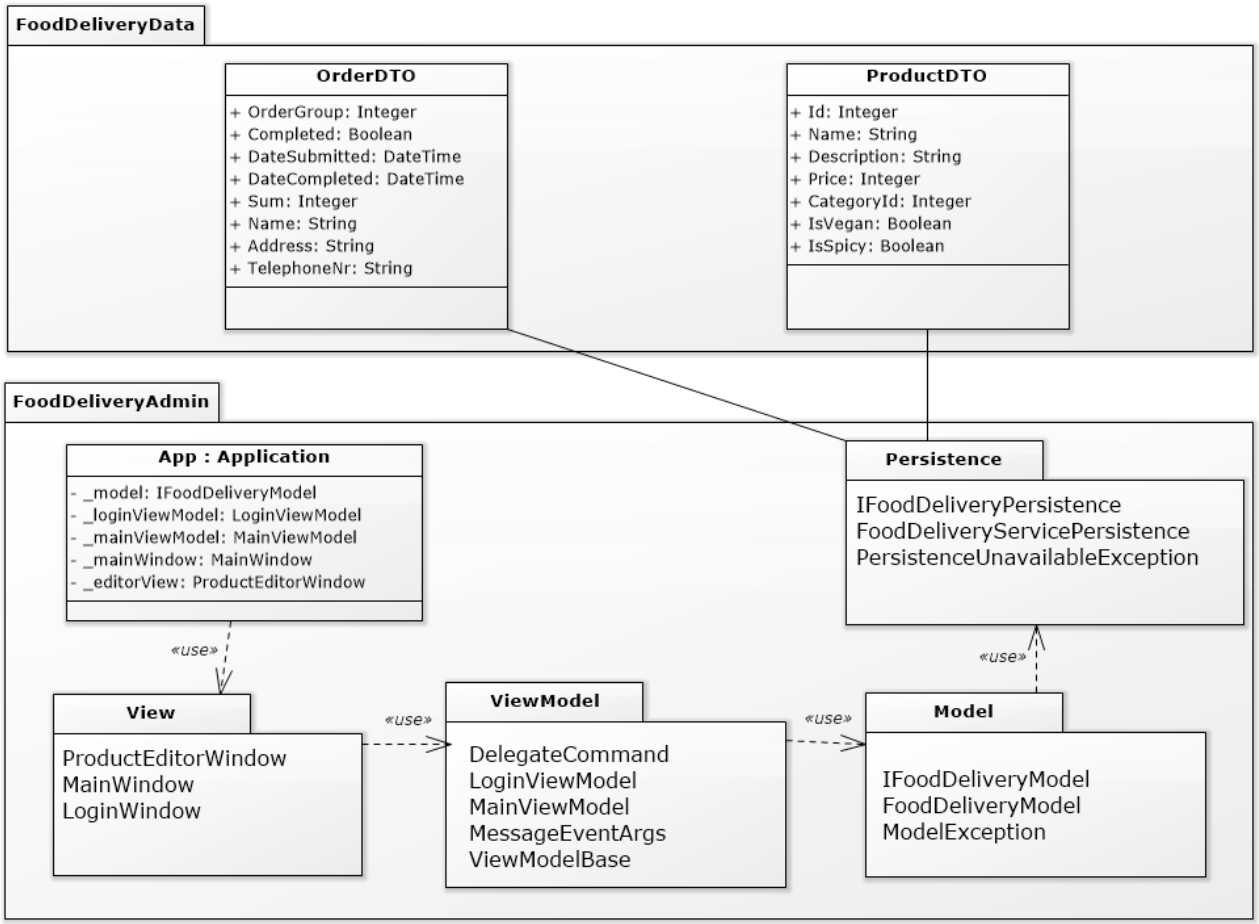
WebApi felépítése:

- A megvalósítás MVC (*Model, View, Controller*) architechtúrában történik.
- A Controller osztályok szolgáltatják azt az interfészt, melyen keresztül az adatbázisbeli elemeket le tudjuk kérdezni, illetve módosítani tudjuk azokat. (OrdersController, ProductsController, AccountController)
- Az adatbázis kontextust a FoodDeliveryContext határozza meg, az adatbázis létrehozását (ha az még nem létezik) a DbInitializer osztály végzi.



WPF alkalmazás felépítése:

- A megvalósítás MVVM architechtúrában történik.
- A View névtérben található összes nézet osztálya XAML-ben készültek, a nézetmodell megfelelő tagjára data binding (delegátumok) segítségével kapcsolódnak.
- A ViewModel névtérbeli osztályok a ViewModelBase ősosztály leszármazottjai, ez biztosítja azt, hogy a nézet jelezni tudja a változtatásokat. Minden nézethez tartozik azonban egy saját nézetmodell is, mely az adatokat továbbítja a modell felé.
- A modell osztályok dolgozzák fel az adatokat és továbbítják azokat a perzisztencia felé ha kell. A viewmodel a modeltől tudja lekérni a számára szükséges információkat.
- A perzisztencia az IFoodDeliveryPersistence-ből származtatott FoodDeliveryServicePersistence osztály segítségével tartja a kapcsolatot az API-val.
- Az adatforgalom az API és a WPF alkalmazás között az OrderDTO és ProductDTO osztályok segítségével jön létre.



Tesztelés:

A webes szolgáltatás XUnit egységtesztek segítségével a helyes működés érdekében tesztelve lett. Az API Controller osztályainak összes metódusa átment a teszteseteken.