Webes alkalmazások fejlesztése 2. beadandó

Készítő neve: Kovács Levente
Neptun kód: BK29F1

A feladat leírása:

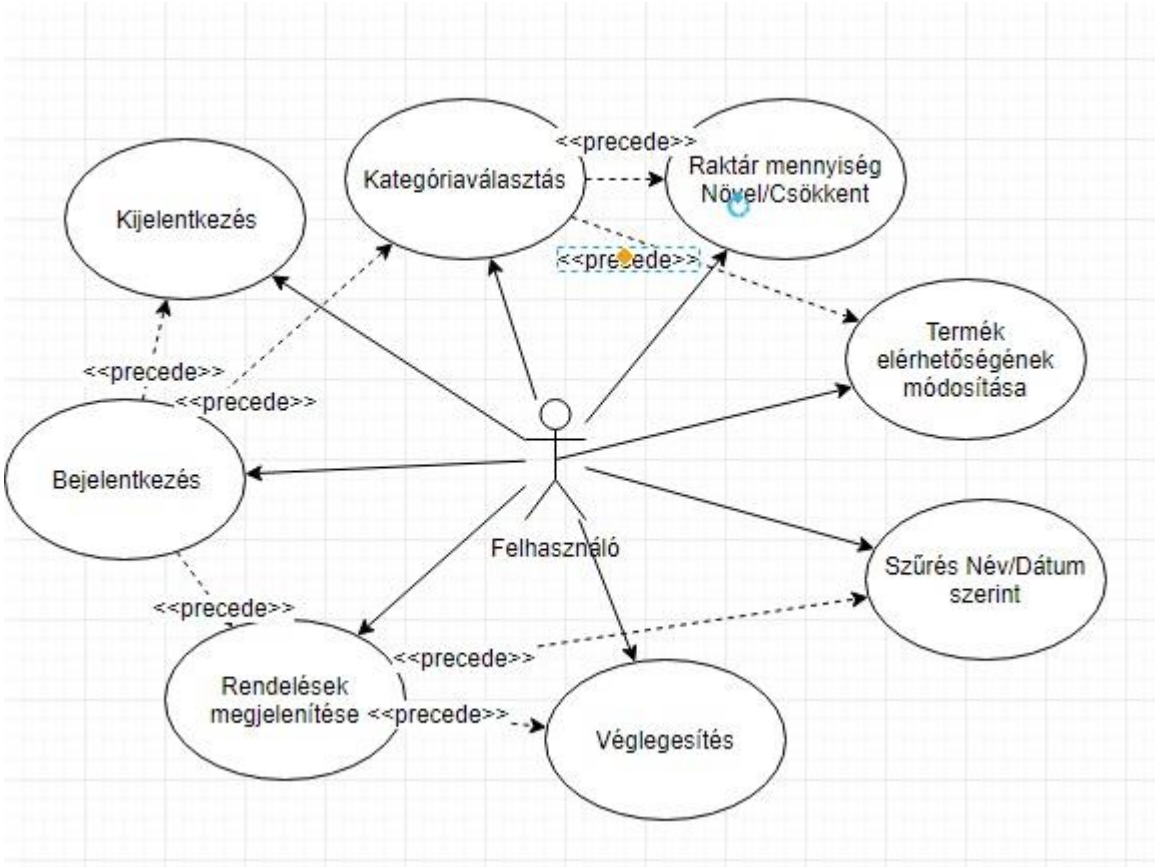
Készítsük el egy elektronikus termékekkel foglalkozó cég online rendszert, amelyben a vevők megrendelhetik termékeiket.

2. részfeladat: az alkalmazottak az asztali grafikus felületen keresztül adminisztrálhatják a rendeléseket, illetve raktárkészleteket.
• Az alkalmazott bejelentkezhet (felhasználónév és jelszó megadásával) a programba, illetve kijelentkezhet.
• Bejelentkezve az alkalmazás listázza kategóriánként a termékeket (gyártó, típus, leírás, nettó/bruttó ár, illetve raktárkészlet), és lehetőség van a raktárkészlet növelésére az egyes termékeknél.
• Amennyiben a termék hosszabb távon nem hozzáférhető, lehet inaktív állapotba helyezni. Ez azt jelenti, hogy a webes felületen csak addig jelenik Webes alkalmazások fejlesztése 2019/2020 őszi félév 7 meg, és addig rendelhető, amíg van belőle raktárkészlet. Továbbá az asztali grafikus felületen sem lehet a raktárkészletét növelni. Inaktív terméket természetesen lehet újra aktiválni.
• Az alkalmazás listázza a rendeléseket (dátum, név, cím, telefonszám, e-mail cím, termékek listája). A teljesítést a munkatárs kijelöléssel tudja kezdeményezni, amelyre a rendszer megerősítést kér, és automatikusan módosítja a raktárkészletet. A rendelések szűrhetőek a megrendelő név(részlet), dátum, valamint teljesítettség állapota szerint.

Az adatbázis az alábbi adatokat tárolja:
• kategóriák (név);
• termékek (gyártó, modellszám, leírás, kategória, nettó ár, raktárkészlet, elérhető-e);
• munkatársak (teljes név, felhasználónév, jelszó);
• rendelések (név, cím, telefonszám, e-mail cím, termékek listája, teljesítettség)

A feladat elemzése:

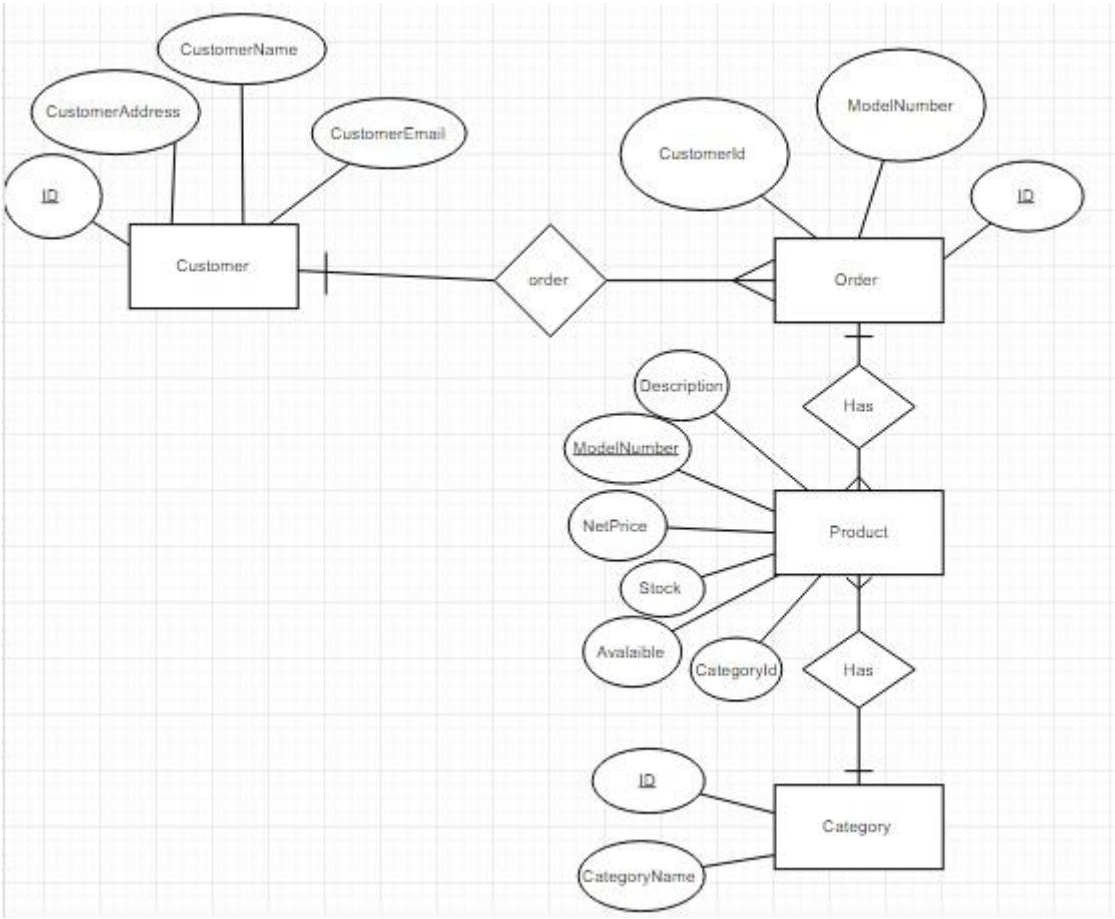
- A felhasználónak az alkalmazás indítását követően be kell jelentkeznie, hogy használhassa a többi funkciót.
- Bejelentkezést követően megjelennek a kiválasztható kategóriák
- Egy kategóriára kattintva megjelennek a hozzá tartozó termékek a megfelelő információkkal
- A felhasználónak lehetősége van letiltani egy terméket, növelni illetve csökkenteni a raktárkészletét egyesével.
- A felhasználónak lehetősége van a rendelések megtekintésére a „Rendelések” gombra kattintással. Ezt követően kiválaszthatja ,hogy a teljesített, vagy a nem teljesített rendelések jelenjenek meg a képernyőn. Ezt kiválasztva megjelennek a rendelések a megfelelő információkkal. További beállításként kereshet a rendelések között, név illetve dátum szerint is, valamint a „véglegesít” gombra kattintva véglegesítheti is a rendelést, amit megelőz egy megerősítő rendszerüzenet.



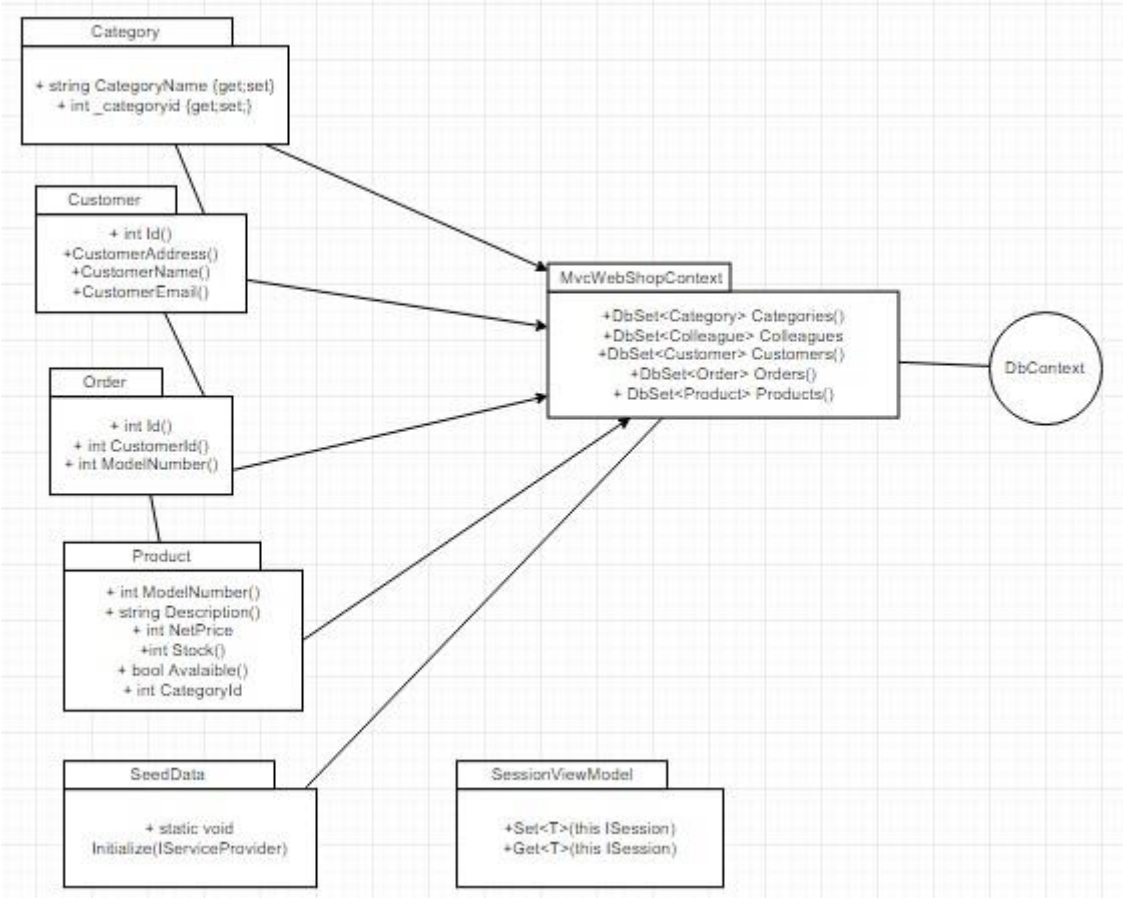
1. ábra: felhasználói esetek

Az feladat megvalósítása:

- **Perzisztencia:**
  A perzisztenciát az Entity Framework segítségével valósítjuk meg. Az adatbázis négy táblát tartalmaz, név szerint: Categories, Products, Orders, Customers, . A négy táblát az MvcWebShopContext nevű modell fogja össze. A táblák össze vannak kapcsolva. Minden Product-beli termék csak egy Category-ba tartozhat, de egy kategória több termékhez is tartozhat. A Customers tábla tartalmazza a rendelésekhez tartozó felhasználókat, az Orders tábla pedig a rendelésekhez tartozó további információkat.



2. ábra: az adatbázis egyedkapcsolat diagrammja



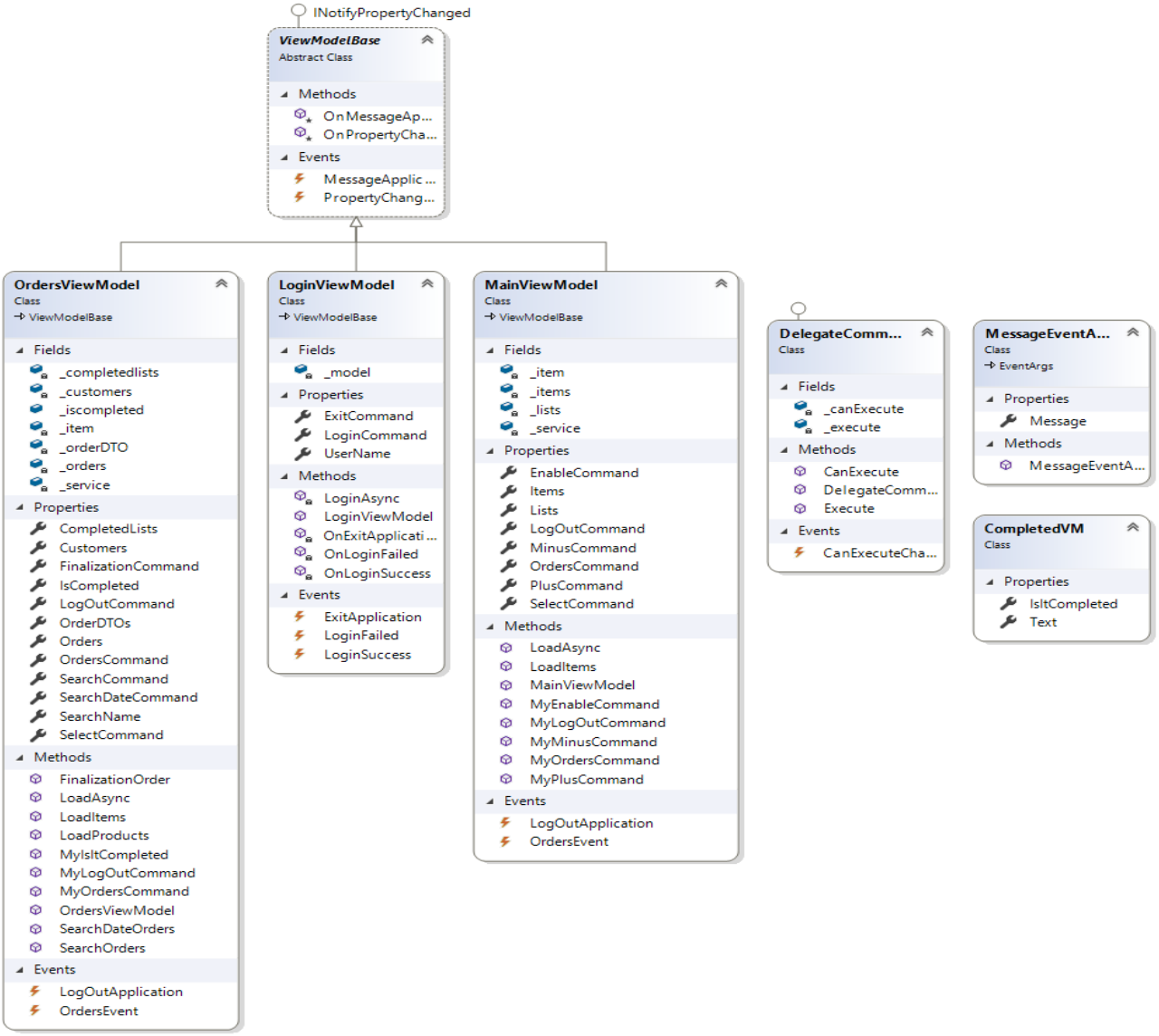
3. ábra: az adatbázis osztály diagrammja

Asztali alkalmazás

Az asztali alkalmazás **Windows Presentation Forms** alkalmazás keretében lett megvalósítva, MVVM architektúrát használva.

1. Nézetmodell

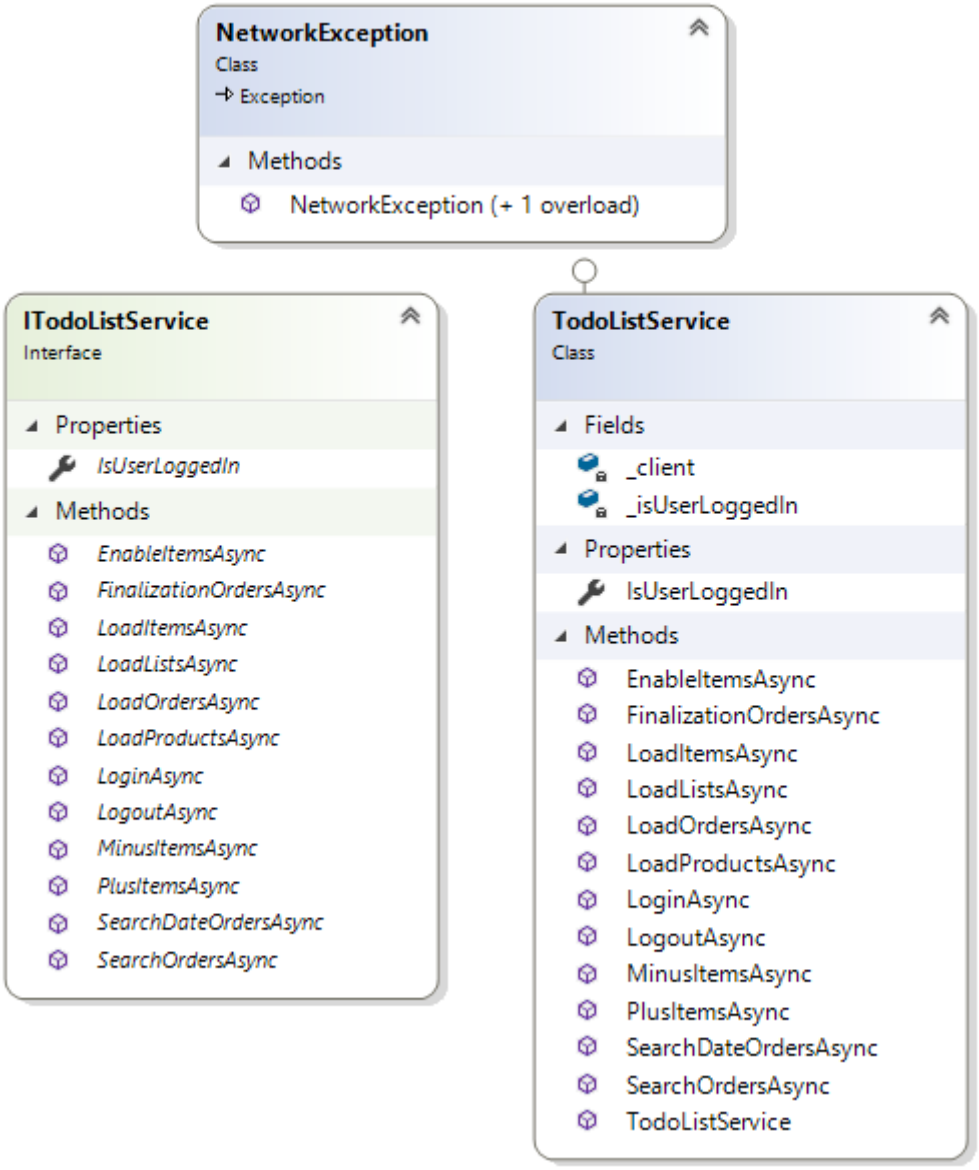
- A nézetmodell megvalósításához felhasználunk egy általános utasítás (**DelegateCommand**), valamint egy ős változásjelző (**ViewModelBase**) osztályt.
- A nézetmodellben a **LoginViewModel** és a **MainViewModel** valamint az **OrderViewModel** látja el a feladatokat, parancsok segítségével. Az előbbi felelős a bejelentkezést ellátó funkcióval, míg az alkalmazás összes további funkcióját. A nézetmodell tárol egy hivatkozást a modellről (**_service**) de csupán csak adatot továbbít neki.
- A felhasználótóság érdekében létezik egy **MessageEventArgs** osztály, ami segítségével könnyebb üzeneteket továbbítani a felhasználónak.



4. ábra: a nézetmodell osztálydiagrammja

2. Modell

- A modell lényegi részét az **AuctionPortalService** osztály valósítja meg, ami az **IAuctionPortalService** nevű interfészt implementálja.
- A modell fő feladata, hogy a webszolgáltatáson keresztül adatot módosítson az adatbázisban, direktben kommunikál a webszolgáltatással.
- A modellhez implementálva lett egy **NetworkException** osztály, ami segítségével üzenetet továbbíthatunk a felhasználónak, a webszolgáltatás elérése közben felmerülő esetleges hibákról.



5. ábra: a modell osztálydiagrammja

3. Nézet

- A nézet három képernyőt tartalmaz. A **LoginWindow** felelős a bejelentkezést kezelő ablakért, a **MainWindow**-ban kezelhetőek az alkalmazás fő funkciói, a **OrderWindow** meg a rendelések megjelenítéséért felel.

- **<u>Webszolgáltatás</u>**

A webszolgáltatás feladata egy csatornát biztosítani az asztali alkalmazás és az adatbázis között. Ehhez controllereket használ, amik az asztali alkalmazás modelljéből kapnak információt, illetve onnan lesznek kiváltva a methódusok. Három controllerrel dolgozik a webszolgáltatás. Az **AccountController** felelős a bejelentkezést lekezelő fukcióval, authentikációval. Az **ItemsController** felelős azokért a funkciókért, amik több terméket is érintenek, ilyen például az összes termék megjelenítése, illetve a kategórák. A harmadik controller az OrderController, ami előkészíti a viewnak a megjelenítendő adatokat, például esetleges névszerinti keresés vagy dátumszerinti keresés esetén. Rendelés véglegesítéséért is az OrderController felel.

- **<u>Az API végpontjainak leírása</u>**
- **POST** api/Account/Login
- Bejelentkezés
- Body: a felhasználó felhasználóneve, és jelszava
- HTTP 200: Sikeres bejelentkezés
- HTTP 401: Sikertelen bejelentkezés
- **POST** api/Account/Signout
- Kijelentkezés
- HTTP 200: Sikeres kijelentkezés
- **GET** api/Items/**CategoryId**
- Összes termék lekérdezése az adott kategóriában
- HTTP 200: Sikeres lekérdezés, terméklista JSON
- **GET** api/Items/Plus/ModelNumber
- Az adott azonosítójú termék raktárszámának növelése 1-gyel
- HTTP 200: Sikeres lekérdezés, terméklista JSON
- **GET** api/Items/Minus/DescModelNumber
- Az adott azonosítójú termék raktárszámának csökkentése 1-gyel
- HTTP 200: Sikeres lekérdezés, termémklista JSON
- **GET** api/Items/Enable/SetEnableModelNUmber
- Az adott azonosítójá termék elérhetőségének módosítása
- HTTP 200: Sikeres lekérdezés, terméklista JSON
- **GET** api/Orders/
- Az összes rendelés lekérdezése
- HTTP 200: Sikeres lekérdezés, rendelés DTO JSON

- **GET** api/Orders/Id
- Visszaadja az adott rendeléshez tartozó termékek listáját
- Body: A kiválaszott rendelés azonosítója
- HTTP 200: Sikeres lekérdezés, rendelés DTO JSON
- HTTP 400: Sikertelen lekérdezés
- **GET api/Orders/Search/searchstring**
- Vissza adja a megadott névrészletbe beleíllő rendelések listáját
- HTTP 200: Sikeres lekérdezés, rendelés DTO JSON
- **GET api/Orders/Finalization/Id**
- Véglegesíti az adott Id-vel rendelkező rendelést
- HTTP 200: sikeres lekérdezés, rendelés DTO JSON
- **GET api/Orders/SearchDate/searchdatestring**
- Az adott dátumhoz tartozó rendeléseket listázza
- Body: egy DateTime típusú dátum
- HTTP 200: sikeres lekérdezés, rendelés DTO JSON
- **GET api/Orders/Product/b**
- Visszaadja a rendelések listáját attól függően hogy teljesített(b) vagy nem teljesített (!b)
- b = Teljesített rendelés
- HTTP 200: sikeres lekérdezés, rendelés DTO JSON

- **<u>Webszolgáltatás tesztesetei:</u>**
- Tesztadatbázis inicializálásának tesztelése
- Az összes termék lekérdezése
- Kategóriák lekérdezése
- Összes termék lekérdezése
- Raktérkészlet növelésének tesztelése
- Raktárkészlet csökkentésének tesztelése
- Elérhetőség módosításának tesztelése
- Rendelések lekérdezésének tesztelése
- Dátumszerinti lekérdezés tesztelése
- Név szerinti keresés lekérdezésének tesztelése
- Rossz kategóriának lekérdezése
- Véglegesítés tesztelése
- Megfelelő termék lekérdezése