GenericCrudOperator.cs

* Minden controller osztály rendelkezik egy crud operator attribútummal. Ez az osztály olyan generikus metódusokat tartalmaz, amik segítségével alapvető adatbázis műveletek hallathatóak végre. Minden metódus vár egy típust. Álltalában megkeresi az ilyen típusú rekordokat tartalmazó táblát. Crud műveleteket haj végre.
* Metódusok:
  + GetDbsetForGeneric 🡪 Visszatér a kapott típust tartalmazó dbset-tel amit a databasecontext tartalmaz.
  + GetRowById 🡪 Visszatér egy rekorddal aminek az id-je a paraméterben kapott string és a típusa megeggyezik a megadottal.
  + GetMultipleRowsByForeignId 🡪 Visszatér a rekordok listájával, amiknek a a paraméterként kapott property nevének értéke megegyezik a paraméterként kapott stringgel.
  + DeleteRowById 🡪 Kitörli a rekordot az adatbázisból, aminek az id-je a paraméterként kapott strring.
  + InsertNewRow 🡪 Beszúr egy sort az adatbázis azon táblájába, aminek a típus a kapott típus. A sor értékeit paraméterként kapja.
  + GetRowByForeignId 🡪 Ugyanaz, mint a GetMultipleRowsByForeignId, csak nem elemek listájával tér vissza, hanem csak egy darab rekorrdal.
  + SaveDatabaseChanges 🡪 Elmenti az adatbázisban létrejött változásokat.
  + UpdateRow 🡪 Frissíti az adatbázisban a parméterként kapott rekordot.

BaseEntityController.cs

* Absztrakt ős a kontrollereknek. Példányosítja a GenericCrudOperator-t, DI-vel kapott UserManagert és DatabaseContext-et attribútumként tárolja. Minden controller minden metódusa http válaszokba csomagolt rekordokkal/szövegekkel tér vissza, mivel a controllerek metódusai majdnem mind Api Endpointok, melyeket http kérésen keresztül hív meg a kliens.

Services

* Különböző servicek, amik egyes kontrollerek műveleteit bizonyos műveleteit tartalmazzák. Csak azok az eljárások vannak servicekbe kiszervezve, melyek több kontrollerben is használandóak. Ezeknek a segítségével nem kell kódot duplikálni, hanem azokba az osztályokba lehet injektálni, amelyek ezeket a műveleteket használják. Az injektáláshoz az ASP.NET Core Dependeny injection-jét használjuk. Ehhez a program.cs fájlban regisztrálni kell a serviceket. Erre egy pálda: 🡪 builder.Services.AddScoped<IRestaurantService, RestaurantService>();. Minden servicehze tartozik egy interfész, és az azt megvaolósító osztálly. Minden servicebe kiszervezett művelet visszatérési értéket egy http válasz objektum, mert van ahol ezt kell használni. Ahol nem ott pl. a listát ki kell ebből csomagolni. Minden ilyen metódus kap még egy GenericCrudOperator-t hogy tudjon olvasni/írni az adatbázisból/ba.
* OwnerService 🡪
  + ListRestaurantsByOwner 🡪 A metódus kilistázza azokat az éttermeket, amelyeknek a UserId nevű propertije megegyezik a paraméterként kapottal. A megfelelő éttermekből létrehozott DTO-kal tér vissza. Ezen felül még kislistázza azokat a nyitvatartási időpontokat, ahol a RestaurantId property a restaurant id-je. Ezt is a Dto egyik értéke.
* LikedRestaurantService 🡪
  + DeleteLikedRestaurantByIdForLoggedUser 🡪 A paraméterként kapott id-jű LikedRestaurant rekordot kitörli az adatbázisból.
  + GetLikedRestaurantByLoggedInUser 🡪 Kilistázza azokat a LikedRestaurant rekordokat, ahol a UserId property a paraméterként kapott értékkel egyenlő.
* ReservationService 🡪
  + DeleteReservationByIdForLoggedUser 🡪 A paraméterként kapott id-jű Reservation rekordot kitörli az adatbázisból. Ezek kívül a Reservation-hoz tartozó Restaurant NumOfFreeSeats propertyjének értékét a foglalás létszámával növeli.
  + GetReservationsByLoggedInUser 🡪 A paraméterként kapott user id-jének megfelelő foglalásokat listázza ki. Ezekből a Reservationokból létrehozott Dto-k listájával tér vissza.
* RestaurantService🡪
  + DeleteRestaurantById 🡪 A paraméterként kapott id-jű Restaurant rekordot kitörli az adatbázisból. Ezen felül kitörli az étterem összes függőségét, hogy azok se maradjanak az adatbázisban ezek a következőek: Categories, Menu, Foods, Reservations, OpeningHours.

ImageToByteArrayConverter

* A Food entitások Image property-je byte tömbként tárol képeket. Ez az osztály tartalmaz 2 statikus metódus, melyek segítségével képeket lehet byte tömbbé konvertálni.
* Metódusok:
  + PathToByteArray 🡪 A példa adatok képeit az Assets mappában tároljuk. Ezzel a metódussal a kapott paraméter (ami egy fájlnév) szerinti fájlt lehet byte tömbbé konvertálni. Mivel minden kép ugyanott van, így csak a név változik az elérési útvonalban, ezért ezt a metódus előre, minden esetben azonos módon meghatározza.
  + FileToByteArray 🡪 A paraméterként kapott IForFile típusú paramétert konvertálja byte tömbbé. Ezt minden esetben http kérésen keresztül, a kliensben kiválasztott fáljként kapjuk meg.

UserController

* A User Entitiyhez tartozó kontroller. A BaseEntityController leszármazottja, onnan örökli az ott definiált attribútumokat. Ezen kivűl rendelkezik még attribútumokkal a következő osztályokból: SignInManager (ASP.NET Identity álltal biztosított osztály a bejelentkezéshez), IRestaurantService, IOwnerService, IReservationService, ILikedRestaurantService. A különböző serviceket DependenyInjection segítségvel kapja meg a controller, melyeket korábban regisztráltunk a Program.cs-ben.

Metódusok:

* + Register 🡪 A paraméterként kapott UserRegisterDto adatai alapján regisztrál egy felhasználót. A felshasználó létrehozásához a UserManager-t használja. Ezen kívül a UserManager segítségével regisztrálja a felhasznákó UserRole-ját.
  + Login 🡪 A paraméterként kapott UserLoginDto adataival beléptet egy felhasználót. Ha sikeres a belépés, akkor az adatok alapján beállítja a dto userrole-ját és a http válaszban visszatér vele.
  + LogOut 🡪 A SignInManager használatával kilépteti a felhasználót.
  + GetLoggedInUsersData 🡪 UserManager-rel meghatározzuk a beléptetett felhasználót, majd létrehozunk az adatok alapján egy UserDatasDto-t, majd visszatér vele a http válaszban.
  + GetAllUsers 🡪 Kilistázza az összes User-t egy listába, majd visszatér vele.
  + DeleteUserByIdWithAdmin 🡪 Adminként kitörli a paraméterként kapott id-jű felhasználót az adatbázisból. Meghívja a DeleteUserDependencies metódust, mely kitörli a felhasználó függőségeit.
  + UpdateUserForLoggedInUser 🡪 Kitörli az Api hívást meghívó usert. Szintén meghívja a DeleteUserDependencies metódust, mely kitörli a felhasználó függőségeit.
  + DeleteUserDependencies 🡪 Kitörli az adatbázisból a paraméterként kapott felhasználó függőségeit. Ez két féle lehet: Owner vagy Customer. Owner esetén a RestaurantService segítségével kitörli a tulajdonos éttermeit. Customer esetén a ReservationService és a LikedRestaurantService segítségével kitörli a felhasznákóhoz köthető értékeléseket és étterem kedveléseket.

RestaurantController

* A Restaurant entityhez tartozó kontroller. Attribútumként tárol egy restaurantservice objektumot, melyet DI segítségével kap meg.
* Metódusok:
  + ListAllRestaurants🡪 Kilistázza az adatbázisban található összes éttermet.
  + GetRestaurantById 🡪 Megkeresi a paraméterként kapott restauranId-hez tartozó étterem rekordot, és az adataiból létrehozott dt-val tér vissza http válasz formájában.
  + GetRestaurantById 🡪 Megkeresi a paraméterként kapott restauranId-hez tartozó étteremet és visszatér a hozzá tartozó Menu rekorddal.
  + UpdateRestaurantById 🡪 A paraméterként kapott restaurantid-hez tartozó étterem adatait frissíti a szintén paraméterként kapott UpdateRestaurantDto adatai alapján.
  + DeleteRestaurantById 🡪 Kitörli a paraméterként kapott id-hez tartozó étterem rekordot az adatbázisból.

ReservationController

* Reservation entitáshoz tartozó kontroller. Attribútumként tárol egy ReservationService objektumot, melyet DI segítségével kap meg.
* Metódusok:
  + GetReservationsByLoggedInUser 🡪 A UserManager segítségével megkeressük az endpoint-ot meghívott, bejelenetkezett usert, és kilistázzuk azokat a Reservation rekordokat, ahol a UserId idegen kulcs megegyezik a user id-jével.
  + GetReservationsByRestaurantId 🡪 Kilistázza azokat a Reservation rekordokat, melyek RestaurantId idegen kulcsa megegyezik a paraméterként kapott id-vel.
  + DeleteReservationByIdForLoggedUser 🡪 Kitörli azt a rekordot, melynek id-je megegyezik a paraméterben kapottal. Ezt a ReservatioinService metódusának meghívásával teszi meg.
  + ReserveTableForLoggedUser 🡪 Az endpoint-ot meghívó bejelentkezett user foglalhat ezzel asztalt, amely a paraméterben kapott CreateReservation dto-ban kapott adatok alapján tesz meg. Ezen kívül lecsökkenti a dto-ban lévő restaurantId-hez tartozó étterem szabad férőhelyeinek számát. A foglalás csak akkor jön létre, ha az étteremben van elég szabad hely a foglaláshoz.

ReviewController

* Review entitáshoz tartozó kontroller. Nincs semmilyen service attribútuma, mert az osztályban található összes endpoint/fv egyedi erre a kontrollerre.
* Metódusok:
  + GetAvargeRatingByRestaurantId 🡪 A paraméterben kapott Restaurant id-hez tartozó étterem értékeléseinek az átlagát számolja ki.
  + GetReviewsForRestaurantById 🡪 A paraméterben kapott Restaurant id-hez tartozó étterem értékeléseit lehet lekérdezni vele.
  + CreateNewReviewForRestaurant 🡪 Új értékelést lehet vele létrehozni a paraméterként kapott id-hez tartozó étteremhez. Az értékelés adatait a szintén paraméterként kapott CreateReviewDto-ban lehet megadni, és a létrehozott Review rekord innen kapja a property értékeket.
  + DeleteReviewById 🡪 A paraméterként kapott ReviewId-hez tartozó értékelés rekordot lehet vele kitörölni az adatbázisból.

LikedRestaurantController

* LikedRestaurant entitáshoz tartozó kontroller. Rendelkezik egy LikedRestaurantService attribútummal, melyet a regisztrálása után DI segítségével kap meg.
* Metódusok:
  + GetLikedRestaurantByLoggedInUser 🡪 A bejelentkezett user id-jét megkeressük a UserManager-rel, majd az id-t paraméterként átadva meghívja a LikedRestaurantService GetLikedRestaurantByLoggedInUser metódusát, ami kilistázza a userhez tartozó étterem kedveléseket, melyekből Dto-kat hoz létre.
  + LikeRestaurantForLoggedInUser 🡪 A bejelentkezett usert megkeressük a UserManager-rel majd a paraméterként kapott restaurantId-hez tartozó étteremhez létrehozunk egy rekordot a user nevében. Azért, hogy netudjon 2x belikeolni egy éttermet ugyanaz a user, ezért kigyüjtük azokat az éttermeket, amiket a user már likeolt, és csak akkor hozzuk létre a rekordot, ha azt az éttermet a user még nem kedvelte.
  + DeleteLikedRestaurantByIdForLoggedUser 🡪 A paraméterként kapott LikedRestaurantId-hez tartozó kedvelés rekordot kitörli, erre a LikedRestaurantService DeleteLikedRestaurantByIdForLoggedUser metódusát hívja meg.

OwnerController

* Owner entitáshoz tartozó kontroller. Rendelkezik egy ownerService attribútummal, melyet a regisztrálása után DI segítségével kap meg.
* Metódusok:
  + ListRestaurantsByOwner 🡪 HTTP GET metódus. A bejelentkezett tulajdonos által kezelt éttermek listázása. Csak az "Owner" szerepkörrel rendelkező felhasználók számára érhető el. A bejelentkezett user id-jét megkeressük a UserManager-rel, majd az id-t paraméterként átadva meghívja az ownerService ListRestaurantsByOwner metódusát, ami kilistázza az ownerhez tartozó éttermeket, melyekből Dto-kat hoz létre. Visszatérési értéke a tulajdonos által kezelt éttermek listája
  + CreateRestaurantWithOwner 🡪 HTTP POST metódus. Csak az "Owner" szerepkörrel rendelkező felhasználók számára érhető el. Paraméterben megkapja az új étterem adatait. Feladata egy új étterem létrehozása. Először lekéri a bejelentkezett usert, majd a Dto-ban kapott új étterem adatait le elleőrzi, hogy nem üresek-e. Ezek után létrehozza az új étterem példányát és a hozzá tartozó nyitvatartási időket. Helyes végrehajtásnál visszatérési értéke az újonnan létrehozott étterem 200 OK kóddal, hiba esetén hibajelzés.
  + AddCategoryToMenu 🡪 HTTP POST metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az étterem azonosítóját és a kategória nevét. Feladata egy új kategória hozzáadása a menühöz az adott étterem azonosító alapján. Először lekéri az éttermet és a hozzá tartozó menüt, majd, ha a menü nem null, akkor létrehozza a menühöz tartozó új kategóriát. Visszatérési értéke az újonnan létrehozott kategória vagy hibajelzés.
  + AddFoodToCategory 🡪 HTTP POST metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja a kategória azonosítóját, az étel nevét, leírását és árát. Feladata egy új étel hozzáadása egy kategóriához az adott kategória azonosító alapján. Először lekéri a kategóriát, majd, ha a kategória nem null, akkor létrehozza a kategóriához tartozó új ételt. Visszatérési értéke az újonnan létrehozott étel vagy hibajelzés.

TableController

* Table entitáshoz tartozó kontroller, amely az éttermek asztalainak kezelésére szolgál
* Metódusok:
  + ListTablesByRestaurantId 🡪 HTTP GET metódus. A bejelentkezett tulajdonos által kezelt éttermek listázása. Csak az "Owner" és "Customer" szerepkörrel rendelkező felhasználók számára érhető el. Paraméterben megkapja az étterem azonosítóját. Feladata egy étteremhez tartozó asztalok listázása. Először lekéri egy étterem asztalait az étterem azonosítója alapján, majd, ha léteznek asztalok, akkor azokat konvertálja TableDto formátumba és egy listába teszi ezeket. Visszatérési értéke az étteremhez tartozó asztalok listája vagy hibajelzés.
  + DeleteTableById 🡪 HTTP DELETE metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az asztal azonosítóját. Feladata egy adott asztal törlése az azonosító alapján. Visszatérési értéke sikeres törlés üzenet vagy hibajelzés
  + AddTableToRestaurant 🡪 HTTP POST metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az étterem azonosítóját és az új asztal adatait. Feladata egy új asztal hozzáadása egy adott étteremhez. Először lekéri az éttermet azonosító alapján, majd, ha az étterem nem null, akkor létrehozza az új asztalt, és növeli az étterem szabad helyeinek számát. Visszatérési értéke az újonnan létrehozott asztal vagy hibajelzés.

CategoryController

* Category entitáshoz tartozó kontroller, amely az éttermek menüjéhez tartozó kategóriák kezelésére szolgál
* Metódusok:
  + AddCategory 🡪 HTTP POST metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az étterem azonosítóját és az új kategória adatait. Feladata egy kategória hozzáadása egy adott étteremhez. Először lekéri az éttermet azonosító alapján, majd, ha az étterem nem null, akkor létrehozza az új kategóriát, és hozzáadja az étterem menüjéhez. Visszatérési értéke az újonnan létrehozott kategória vagy hibajelzés.
  + ListCategoriesByRestaurantId 🡪 HTTP GET metódus. Csak a "Owner" és "Customer" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az étterem azonosítóját. Feladata az adott étterem menüjéhez tartozó kategóriák listázása. Először lekéri az éttermet azonosító alapján, majd, ha az étterem nem null, akkor listába gyűjti a kategóriákat, majd ezeket egy új listába átkonvertálja CategoryDto formátumba. Visszatérési értéke az étterem menüjéhez tartozó kategóriák listája vagy hibajelzés.
  + DeleteCategoryById 🡪 HTTP DELETE metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja a kategória azonosítóját. Feladata egy adott kategória és a hozzá tartozó ételek törlése az azonosító alapján. Visszatérési értéke sikeres törlés üzenet vagy hibajelzés

FoodController

* Food entitáshoz tartozó kontroller, amely az éttermek menükategóriáihoz tartozó ételek kezelésére szolgál
* Metódusok:
  + GetMenuItems 🡪 HTTP GET metódus. Csak a "Owner" és "Customer" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az étterem azonosítóját. Feladata egy étteremhez tartozó ételek listázása. Először lekéri az éttermet azonosító alapján, majd a hozzá tartozó menü kategóriáit és a kategóriákhoz tartozó ételeket. Visszatérési értéke az étterem menüjéhez tartozó ételek listája FoodDto formátumban vagy hibajelzés.
  + GetMenuItemsByCategoryId 🡪 HTTP GET metódus. Csak a "Owner" és "Customer" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja a kategória azonosítóját. Feladata egy kategóriához tartozó ételek listázása. Először lekéri az ételeket a kategória azonosítója alapján, majd ezeket konvertálja a FoodDto formátumába. Visszatérési értéke egy kategóriához tartozó ételek listája FoodDto formátumban vagy hibajelzés.
  + DeleteFoodByID 🡪 HTTP DELETE metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az étel azonosítóját. Feladata egy adott étel törlése az azonosító alapján. Visszatérési értéke sikeres törlés üzenet vagy hibajelzés
  + AddNewFoodToCategory 🡪 HTTP POST metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja a kategória azonosítóját és az új étel adatait. Feladata egy étel hozzáadása egy adott kategóriához. Először lekéri a kategóriát azonosító alapján, majd, ha az nem null, akkor létrehozza az új ételt, és hozzáadja a kategóriához. Visszatérési értéke az újonnan létrehozott étel vagy hibajelzés.
  + AddImageToFood🡪 HTTP PUT metódus. Csak a "Owner" szerepkörrel rendelkező felhasználók számára elérhető. Paraméterben megkapja az étel azonosítóját és feltöltendő kép fájlt. Feladata egy kép hozzáadása egy adott ételhez. Először sorosítja a kép fájlt, majd lekéri az ételt és hozzáadja a képet. Visszatérési értéke sikeres kép feltöltés vagy hibajelzés.