

Tartalomjegyzék

A program UML komponensdiagramja:	3
Auction.Desktop.....	4
Desktop alkalmazás osztálydiagramja:	4
App:.....	4
Nézet réteg:.....	4
Nézetmodell réteg:.....	5
Modell réteg:.....	6
Perzisztencia réteg:.....	7
Desktop alkalmazás UI tesztelése:.....	8
Auction.WebApi:.....	9
Webapi szolgáltatás osztálydiagramja:.....	9
A webszolgáltatás felületének leírása:.....	9
ApiService.....	11
Api kontroller unittest - Teszesetek:.....	12
Alkalmazás Perzisztencia:.....	13
Perzisztencia osztálydiagramja:.....	13
Teljes alkalmazás uml diagramja.....	15

Feladat leírása:

2.részfeladat:

Az asztali grafikus felületet az akciók hirdetői használhatják új tárgyak felvitelére, valamint alicitálás helyzetének lekérdezésére.2

- Hirdetőként bejelentkezhetünk az alkalmazásba a felhasználónév és a jelszó megadásával, majd eztkövetően lehetőségünk van a meghirdetett tárgyak követésére, új tárgy felvitelére, valamint kijelen-tkezésre.

- A meghirdetett tárgyak (a licitálás lezárási ideje szerint) listázódnak, és minden tárgynál megjelenik,hogy jelenleg mennyit ajánlottak érte.

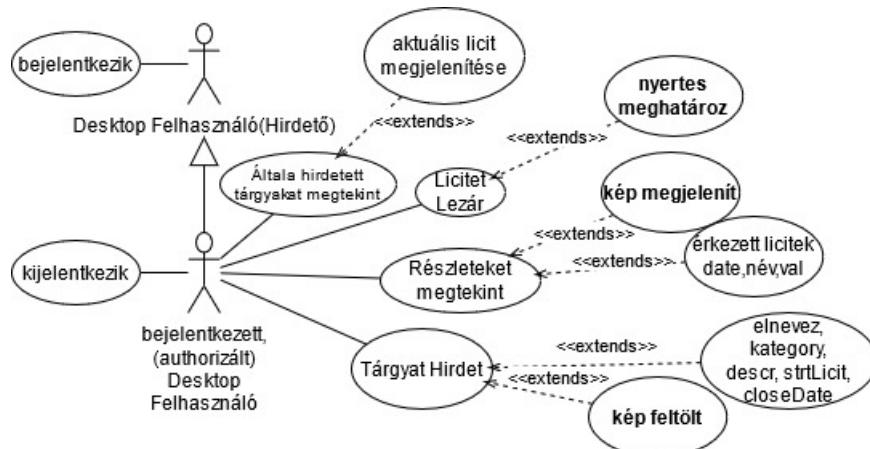
- A tárgyat kiválasztva megkapjuk az összes információt képpelgyütt, valamint az összes addigi licitet (dátum, név, összeg).

- Új tárgy felvitelkor megadjuk az elnevezést, a kategóriát, a leírást, a kezdő licitösszeget, a licitáláslezárásának dátumát, továbbá feltölthetünk egy képet is a tárgyhoz.

- Lehetőségünk van a licit azonnali lezárasára (csak ha valaki már licitált rá, ekkor az aktuális licitáló viszi a tárgyat)

Feladat elemzése:

Use Case Diagram:

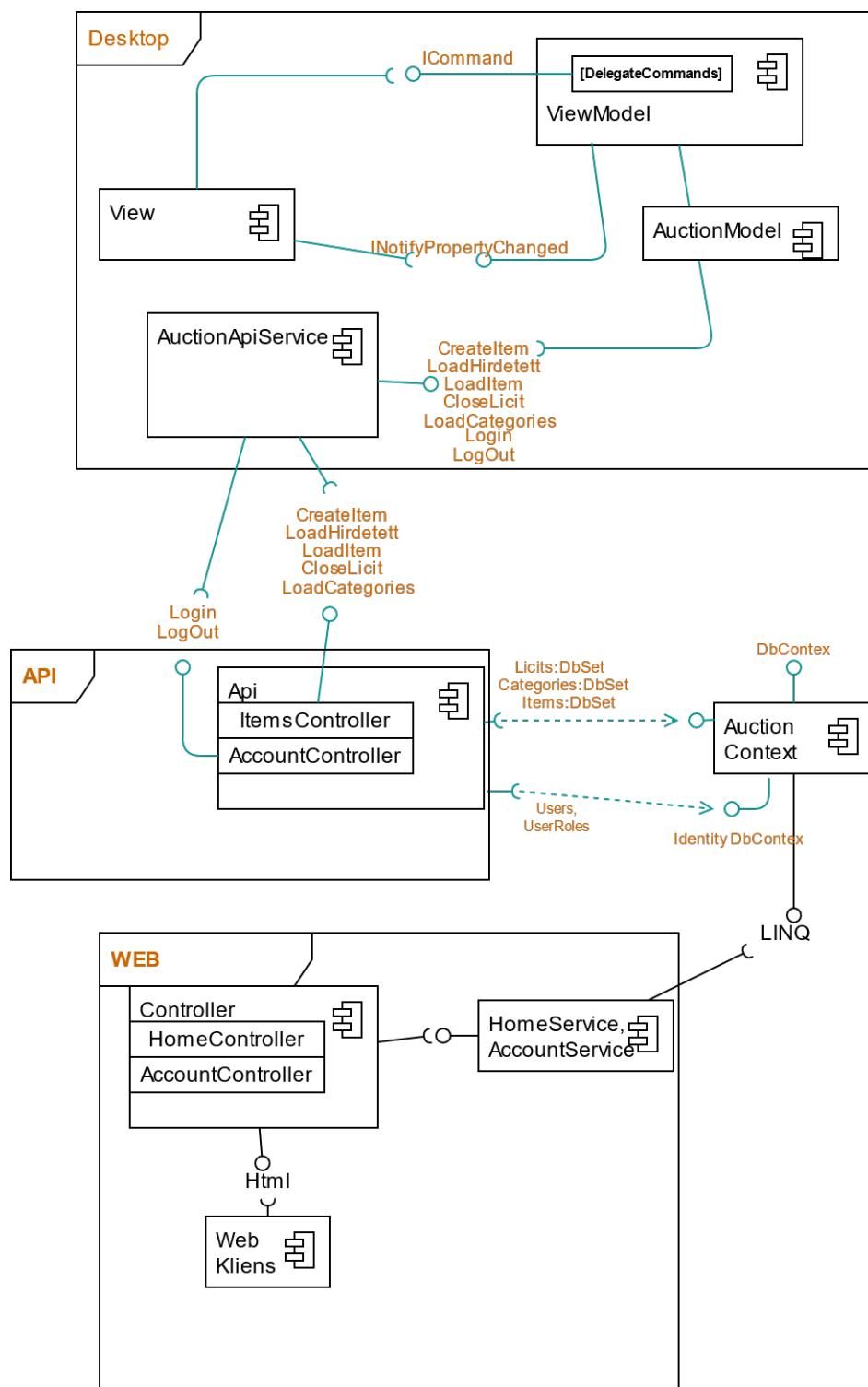


WAF ASP.NET beadandó

2. Aukciós portál

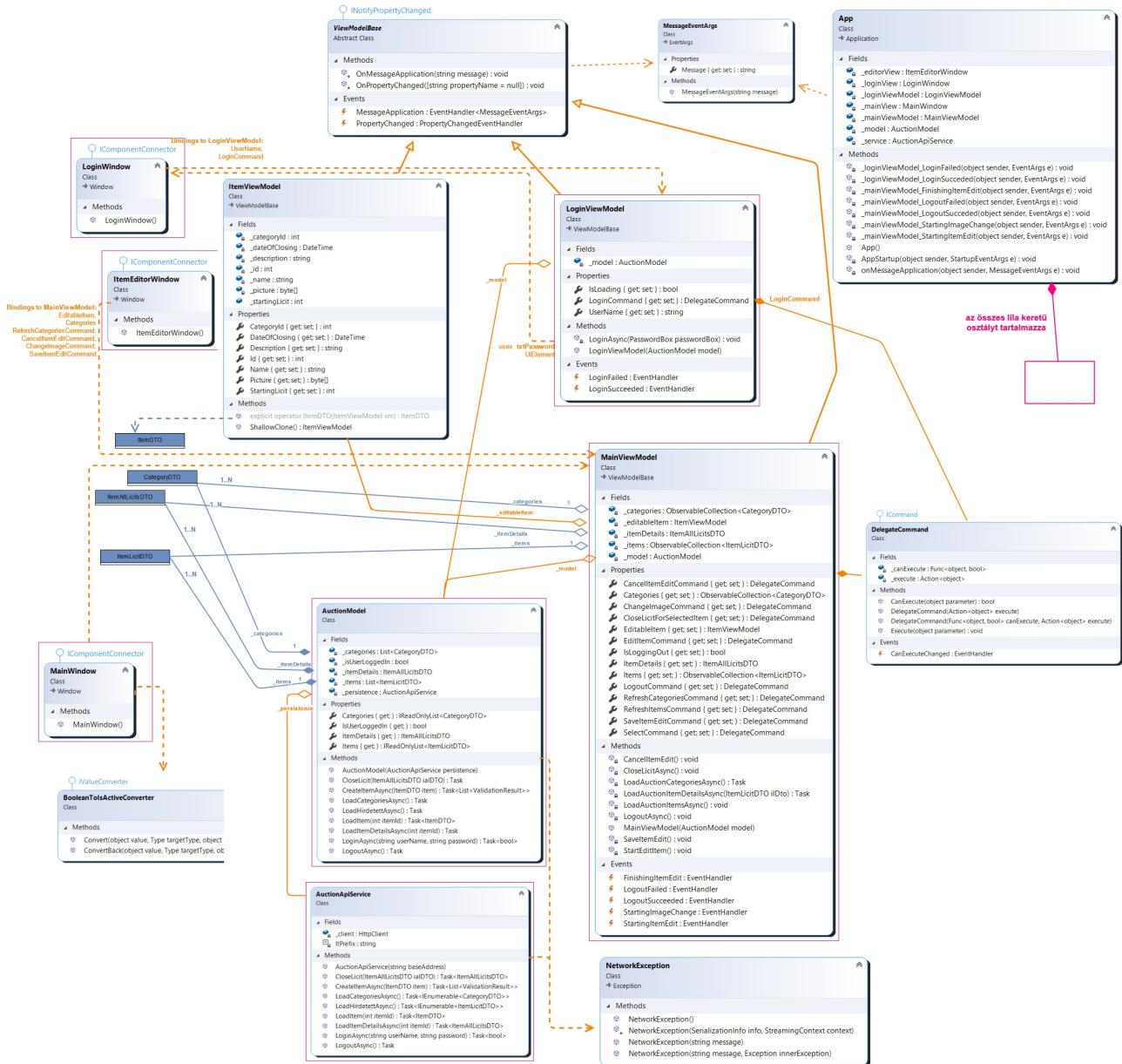
Radnai Miklós(A4U58P)

A program UML komponensdiagramja:



Auction.Desktop

Desktop alkalmazás osztálydiagramja:



App:

- létrehozza a nézeteket a nézetmodelljeikkal, model-t, perzisztenciát, és azért is felel, hogy a megfelelő nézetek legyenek aktívak(pl sikeres login után a MainWindow-t megjeleníti, a loginWindowt pedig eltünteti)

Nézet réteg:

- MainWindow.xaml**
 - Felül egy menüben található gombokkal lehet a hirdetett Tárgyakat újra lekérdezni(„refresh list”) és a kijelentkeznie a bejelentkezett felhasználónak.

- A menü alatti panelen listázódnak soronként a **meghirdtett tárgyak**.
- Ha rákattintunk egy meghirdetett tárgyra, akkor az **ablak alsó felében** megjelennek a kattintott **tárgy részletes adatai** képpel és a rá érkezett licitekkel.
- A kiválasztott tárgyat az ablak közepén levő „**Licit lezárása**” gombbal tudjuk lezární. Ekkor megjelenik a nyertes felhasználó felhasználóneve.
- Emellett található az „új tárgy hirdetése” gomb, amire kattintva megjelenik az „**ItemEditorWindow.xaml**” nézet, amivel létrehozhatunk egy új szavazást.
- **ItemEditorWindow.xaml**
 - Ebben új tárgyat hirdethetünk:
az ablakban megadjuk az elnevezést,kiválasztjuk a kategóriát, a leírást, a kezdő licitösszeget, a licitáláslezárásának dátumát; emellett feltölthetünk egy képet is a tárgyhoz a Gombbal.
 - A Bal alsó „**cancel**” gombbal megszakíthatjuk az új Tárgy hirdetését, ekkor visszatérünk a **MainWindow.xaml**-ra.
 - A jobbalsó „**Save**” gombbal hirdethetjük meg a tárgyat az ablak felső felében megadott adatokkal.
- **LoginWindow.xaml**
 - bejelemtkezést teszi lehetővé felhasználónév és jelszó megadásával.
Az alkalmazás felugróablakkal jelzi, ha valami hiba történt(pl ha a felhasználó nem jogosult, az adatok rosszul lettek megadva, vagy a szerveren belső hiba történt).

Nézetmodell réteg:

- Az egyes nézetmodell elemek működése nagyvonalakban:
amikor nézet meghívja a nézetmodell egyik delegáltját, akkor a delegált meghív egy eljárást, ami meghívja az **AuctionModel** egy eljárását, és annak az eredményét(sikeress, nem sikeres) a megfelelő EventHandler aktiválásával, és-vagy az OnMessageApplication() meghívásával jelezük az **App** felé,
emellett (ha a Model eljárása nem dobott hibát) frissítjük a NézetModellt, hogy az tükrözze a modelben tárolt adatokat.
- **ViewModelBase**
az INotifyPropertyChanged interface-ból öröklí a **PropertyChanged** eventhandler-t.
 - Az **PropertyChanged([CallerMemberName] String propertyName)** eljárás használja **PropertyChanged** handlert, amivel a jelzi, ha megváltozott az egyik property.
 - **MessageApplication :EventHandler<MessageEventArgs>**
Az **OnMessageApplication(String msg)** metódussal küldünk egy **MessageEventArgs** típusban tárolt üzenetet, amit az **App.xaml.cs** fog megjeleníteni.
 - **DelegateCommand:** az **ICommand** interface-ból származik, ahonnan az **Execute(object):void** és **CanExecute(object):bool** függvénydeklarációkat kapja.
A delegateComand példányokat úgy használjuk fel, hogy a nézetben parancskötjük({Binding <delegált>} kifejezéssel) őket, így amikor a nézetbeli elemet triggerelik(pl egy gomb esetén megnyomják), akkor a keretrendszer meghívja az **ICommand Execute** eljárását, így meghívva egy általunk megadott eljárást.
 - **_canExecute: Func<Object,bool>:** lambda kifejezés: ez dönti el, hogy végrekell-e hajtani az **_execute** akciót

- **_execute: Func<Object>**: ezt az akciót hajtja végre feltételesen a **Execute(Object parameter):void**: eljárásban. A feltétel pedig a **CanExecute(object)** függvény adja.
- **CanExecute(Object parameter):Bool**: Ha **_canExecute**-ot nem adtuk meg a konstruktorban(null), akkor igazat ad vissza, amúgy a **_canExecute** kifejezést kiértékeli „**parameter**” argumentumra, és visszaadja az eredményt. A nézetbeli elem, amihez parancskötjük ezt a delegáltat a **CanExecute** értéke alapján dönti el, hogy aktív legyen-e(ez lehet pl egy gomb, ami tehát csak akkor lez aktív, ha a hozzákötött delegált **CanExecute** függvénye true értékkal tér vissza)
- **MessageEventArgs: EventArgs**
Tartalmaz Egy **Message: String** property-t, amit konstruktorában a megadott szövegre beállít.
Ezta típust használják a **ViewModelBase**, hogy a leszármazottai a felhasználónak üzeneteket küldhessenek Az **App**-osztálynak (az üzenetküldés redménye egy felugró ablakban megjelenített szöveg lesz).
- **NetworkException :Exception** :
Az **Auction ApiService** ilyen hibát dob , ha egy kérést nem tudott végrehajtani.
- **MainViewModel**
 - tartalmazza az a kategóriák és meghirdetett tárgyak kollekcióját, ezen kívül:
 - **DelegateCommand**-okat:
 - **RefreshItemsCommand**: hirdetett tárgyak lekérésére
 - **RefreshCategoriesCommand** a kategóriák lekérésére
 - **SelectCommand**: a kiválasztott hirdetett tárgy részleteinek megjelenítésére
 - **CloseLicitForSelectedItem** a kiválasztott tárgyra vontakozó licit lezárására
 - **LogoutCommand**: kijelentkezéshez
 - **EditItemCommand,SaveItemEditCommand,CancellItemEditCommand**: új tárgy hirdetéséhez
 - **ChangeImageCommand** képfájl kiválasztásához kell, A **StartingImageChange** eseményt váltja ki.
 - **EventHandler**-eket, amikkel az **App**-nak jelzi, hogy melyik nézeteket kell aktiválni:
 - **LogoutSucceeded**
 - **LogoutFailed**
 - **StartingItemEdit**
 - **FinishingItemEdit**
 - **StartingImageChange**
- **ItemViewModel**
új tárgy szerkesztéséhez és létrehozásához kell. Tartalmazza az ItemDTO-nak megfelelő változókat, és egy ItemDTO-vá konvertáló operátort, hogy amennyiben befejeztük a szerkesztést tovább tudjuk küldeni az AuctionModelnek, hogy rögzítse az új tárgyat.
- **LoginViewModel**

Modell réteg:

Tartalmazza a Perzisztenciát, a hirdetett tárgyak listáját, a választható kategóriák listáját,
public event EventHandler ItemDetailsChanged:

szól a mainViewModelnek, hogy a Modelben megváltozott a kiválasztott tárgy, ezért frissítenje kell a saját ItemDetails példányát.

public event EventHandler HirdetettChanged:

szól a mainViewModelnek, hogy a hirdetett tárgyak lista megváltozott, úgyhogy a modellbeli **Items** lista alapján frissítse a saját ObservableCollection-jét.

Ezekre a handlerekre, a mainViewModel fog feliratkozni a konstruktorában,

A model továbbítja a NézetModell kéréseit a perzisztencia felé, és a perzisztencia válasza alapján frissíti a saját property-jeit(**items**, **itemDetails**, **Categories**), és az ItemDetailssChanged és HirdetettChanged handler invoke-olásával jelzi a nézetnek, ha frissítéssel kell a property-jeit, illetve ha a perzisztencia válasz helyett hibát dobott (pl mert olyan tárgyra akartuk lezárni a licitét, amire nem érkeztek licitek), akkor ezt a hibát a Model továbbadja a nézet felé.

A **CreateItemAsync(item)** metódusa bizonyos validációs hibákat (ha voltak validációs hibák) stringek listájaként továbbít a nézetmodellnek, hogy az majd jelezhesse ezeket a felhasználónak.

AuctionModel

tartalmazza a hirdetett tárgyak és kategóriák listáját, a kiválasztott tárgy (ha ki lett már választva egy) részleteit, és a következő függvényeket:

- LoadHirdetettAsync(),
- LoadItem(Int32 itemId),
- LoadItemDetailsAsync(Int32 itemId),
- CloseLicit(ItemAllLictsDTO ialDTO),
- CreateItemAsync(ItemDTO item),
- LoadCategoriesAsync(),
- LoginAsync(string userName, string password),
- LogoutAsync()

Perzisztencia réteg:

AuctionApiService

HttpClient objektum segítségével kéréseket intéz az Api felé, és az esetleges hibastatuszkódokat NetworkException formájában továbbítja a Model rétegnek.

Task<IEnumerable<ItemLicitDTO>> **LoadHirdetettAsync()**:

az Items kontroller getGirderett akciójától lekéri a fehasználó által meghirdetett tárgyak listáját.

public async Task<ItemDTO> **LoadItem**(Int32 itemId):

az Items kontroller getItem akciójától lekéri az itemId aznosítúval rendelkező tárgyat.

public async Task<ItemAllLictsDTO> **LoadItemDetailsAsync**(Int32 itemId):

az Items kontroller getItemDetails akciójától lekéri az itemId aznosítóval rendelkező tárgyat.

public async Task<ItemAllLictsDTO> **CloseLicit**(ItemAllLictsDTO ialDTO):

az Items kontroller felé put kérésben küldi az ialDTO objektumot, hogy az majd az ialDTO.item.Id azonosítóval rendelkező tárgyra lezárja a hirdetést.

4 -féle hibakódot küldhet az Items Kontroller, ezeket kezeli:

404: NetworkException-t dob „nem létezik ilyen tárgy üzenettel” (ez azt jelenti, hogy az ialDTO.items.Id rosszul volt beállítva)

401 NetworkException-t dob „csak az ön által hirdetett tárgyat zárhatja le”

405: NetworkException-t dob „a licit még nem indult el, vagy már lezártult” üzenettel.

403: NetworkException-t dob „nem licitáltak még a tárgyra” üzenettel.

Ha más státuszkódot küld az Items kontroller, akkor Network exceptionben dobuk a státuszkódot.

A **CreateItemAsync(ItemDTO item)** eljárásába validációs hibák listájával tér vissza, ha „item” ben rosszul adtuk meg a lezárási dátumot, vagy nem a bejelentkezett hirdetőé a tárgy.

Desktop alkalmazás UI tesztelése:

nem hirdető felhasználó bejelentkezettése:

eredmény:

rossz felhasználónév/jelszó esetén „Sikertelen Bejelentkezés” üzenetet dob az alkalmazás(és nem jelentekezteti be a felhasználót)

ha helyes adatokat ad meg akkor sikeresen bejelentkezik, megjelenik a MainWindow, de hibaüzenetet dob (egy abalban szöveget jelenít meg) az alkalmazás a hirdetett tárgyak lekérésekpor és új tárgy hirdetésének kísérletekor. Kijelentkezhet.

Hirdető felhasználó bejelentkezettése:

eredmény:

rossz felhasználónév/jelszó esetén „Sikertelen Bejelentkezés” üzenetet dob az alkalmazás(és nem jelentekezteti be a felhasználót).

ha helyes adatokat ad meg akkor sikeresen bejelentkezik, megjelenik a MainWindow.

Ha már be volt jelentkezve egy felhasználó, és átjelentkezünk, akkor az új felhasználó tárgyai jelennek meg.

Hirdető által hirdetett tárgyak listázása:

eredmény:

megjelenik az összes általa hirdetett tárgy névvel, kezdő és záródátummal, hirdető nevével, és az aktív licettel.

Hirdető által új tárgy hirdetése:

megjelenik Az ItemEditorView, ahol megadhatjuk a nevet, leírást, kezdőlicitet, zárás dátumát, kiválaszthatunk egy képet, és egy kategóriát(az aktuális kategóriátk listáját gombbal frissíthetjük)

ha a név, leírás mezőt üresen hagyjuk, a kezdő lictet 1nél kisebbre állítjuk, a zárás idejét mostantól egy napon belülire állítjuk, akkor

„név emgadása kötelező”, „leírás megadása kötelező”, „kezdőlicit nem lehet egynél kisebb”, „zárás dátuma legkorábban mosttól 24 óra múlva kell legyen”,

és nem küldi el a tárgyat, hanem hagyja tovább szerkeszteni.

A képet nem kötelező kiválasztani, de ha nem igazi képet választunk ki(kipróbált teszteset: txt fájl .png kiterjesztésre átnevezve), akkor „hiba lépett fel a kép feltöltésekor” üzenetet dob az alkalmazás, és nem választja ki a képet.

Ha az előbbi adatokat helyesen megadjuk, akkor elküldhetjük a tárgyat a „Save” gombbal.

Ezután eltűnik az ItemEditorView, és a refresh List gombra kattintva megjelenik az új tárgy is.

Egy Tárgy Kiválasztása Hirdető által:

megjelenik a kiválasztott tárgy neve, a licitit vezető felhasználó (ha még nincs rá licit, akkor üresen marad ez a mező), ha még aktív a licit, akkor a „A licit Aktív”, ha nem aktív a licit akkor a „Licit már/még nem aktív” szöveg jelenik meg,

megjelenik a tárgy képe(ha van, amúgy üresen marad), kezdőlicitje,nyitás és zárás dátuma, kategória neve, ayz aktuális licit értéke, és alul megjelennek a kiválasztott tárgyra érkezet licitek időponttal, névvel, licit értékkel és azonosítóval.

A felhasználó kijelentkezhet és egy másik felhasználóval bejelentkezhet, ekkor újratöltődnek a hirdetett tárgyak.

Egy Tárgyra a szavazás lezárása Hirdető által:

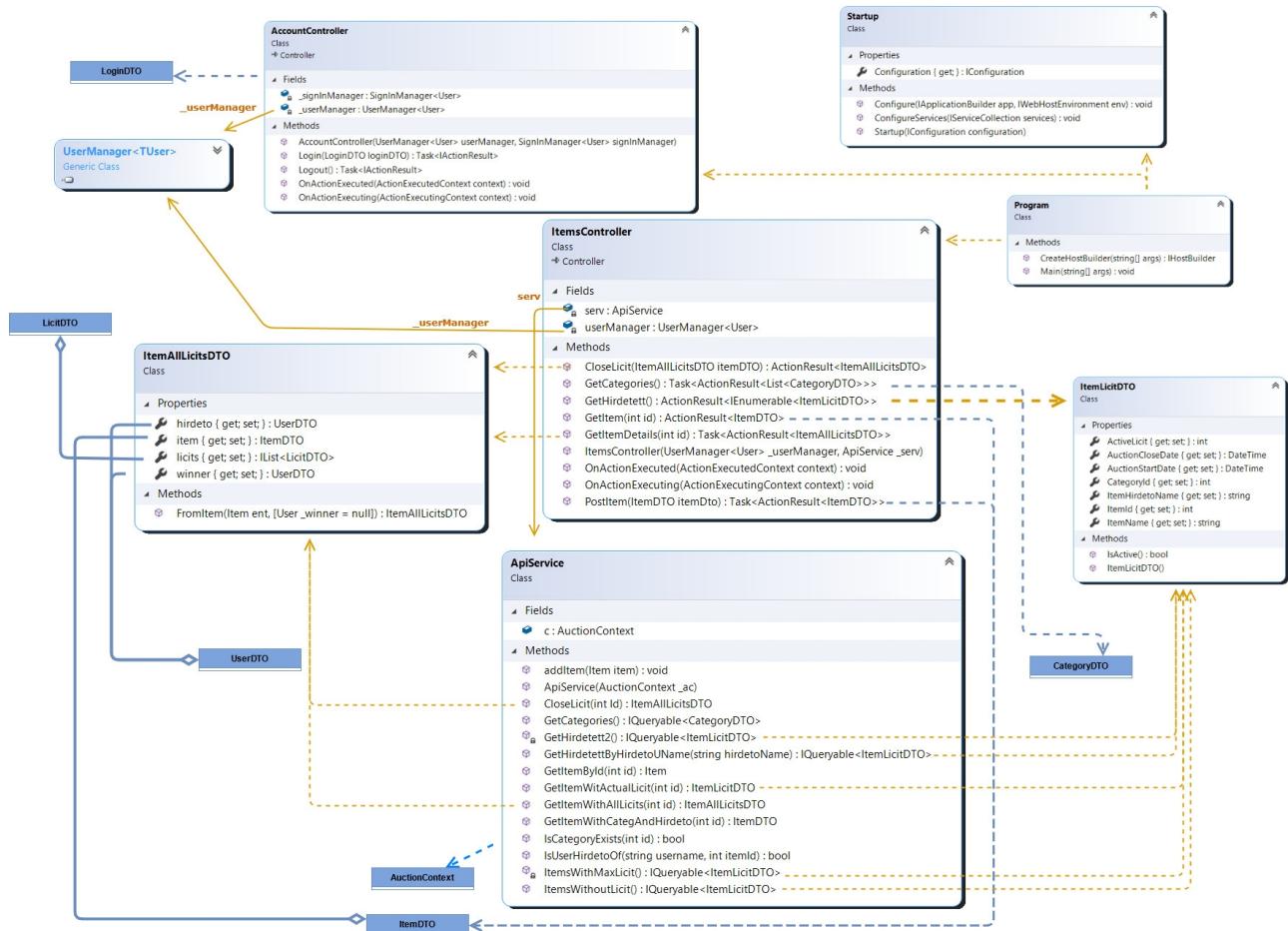
a „kiválasztott licit lezárása” gomb megnyomására a következők történhetnek:

ha érkezett a kiválasztott tárgyra rá licit, akkor a licit lezárul, és a program egy ablakban megjeleníti, hogy melyik felhasználó nyerte a licitet.

Ha nem volt a tárgyra licit, akkor „nem licitáltak a targyra, ezért a licit nem zarható le” üzenetet dob fel az alkalmazás, és a licitit nem zárja le.

Auction.WebApi:

Webapi szolgáltatás osztálydiagramja:



A webszolgáltatás felületének leírása:

AccountController eljárásai:

- **public async Task<IActionResult> Login(LoginDTO):**
Bejelentkezés. Sikertelen bejelentkezés esetén Forbidden Státuszkódot küldünk vissza.
Adatbázis hiba esetén InternalServer Error500 Kóddal térünk vissza
POST kérést vár, uri: /api/Account/Login

- **public async Task<IActionResult> Logout():**
Kijelentkezés.
Kijelentkeztetjük az aktuális felhasználót, sikertelenség esetén InternalServer Error500 Kóddal térünk vissza
uri: /api/Account/Logout
authentikációhoz kötött: csak bejelentkezett felhasználók érhetik el
Get kérést vár.

ItemsController eljárásai:

- `public ActionResult<IEnumerable<ItemLicitDTO>> GetHirdetett():`
a hirdető által hirdetett tárgyak listázása.
Get kérést vár, Uri: api/Items/hirdetett
authorizál: csak bejelentkezett Hirdetők érhetik el
Adatbázis hiba esetén InternalServer Error500 Kóddal térünk vissza, amúgy 200as kóddal tér vissza, és küldi a hirdetett tárgyakat
 - `async Task<ActionResult<ItemLicitDTO>> GetItem(Int32 id):`
a felhasználó lekérheti az általa meghirdetett egyik tárgyat(kategóriával és képpel) azonosító alapján.
ha nem létezik a tárgy, akkor 404-el térünk vissza:
ha nem ő hirdette, akkor unauthorized státussal térünk vissza
Get kérést vár, Uri:api/Items/{id}
 - `public async Task<ActionResult> GetItemDetails(Int32 id):`
a megadott azonosítóval rendelkező tárgy részletes adatai képpel, és az összes ráérkezett licettel(dátum, név, összeg) együtt.
Get kérést vár, Uri: api/Items/details/{id}
authorizál: csak bejelentkezett Hirdetők érhetik el
a hirdető csak általa hirdetetett tárgyhoz kérheti le a részleteket, ha más tárgyhoz tartozik a megadott id, akkor Unauthorized státussal tér vissza
ha az id-hez nem tartozik , tárgy, akkor NotFound üzenettel tér vissza
 - `public async Task<ActionResult<List<CategoryDTO>>> GetCategories():`
kategóriák
Get kérést vár, Uri:api/Items/Categories
 - `public async Task<ActionResult<ItemDTO>> PostItem(ItemDTO):`
új targy felvitelle
POST kérést vár, Uri: api/Items/
authorizál: csak bejelentkezett Hirdetők érhetik el.
Ha a kezdőlicit 1nél kisebb, vagy a zárás dátuma 24órán belüli, vagy a megadott kategória null vagy nem létezik, akkor BadRequest válaszban küldi ezeket validációs hibák formájában.
ha a tárgy felvétele sikeres, akkor visszatér a tárggyal és a `GetItem(Int Id)` akcióra mutató URL-el amin lekérhető a létrejött tárgy.
 - `public async Task<ActionResult> CloseLicit(ItemAllLictsDTO):`
a licit azonnali lezárasa(csak ha valaki már licitált rá, ekkor az aktuális licitáló viszi a tárgyat)
HttpPut kérést vár a lezárandó tárgy adataival.(csak az .Items.Id property-t használjuk fel a küldött tárgyból
Uri: /api/Items/closeLicit/
authorizál: csak bejelentkezett Hirdetők érhetik el
ha nincs ilyen azonosítójú tárgy → Status404NotFound.
Ha a licit még nem indul el, vagy már lezárult → 405
Ha a tárgy más felhasználóé, vagy id-je nem található -> Unauthorized,
Ha nem licitáltak rá -> 403Forbidden státusz
Ha adatbazis frissítése sikertelen -> status500

Az **ItemsController** az **ApiService** nevű segédosztályt használja, ami kezeli az adatbázist.

ApiService

- **GetHirdetettByHirdetoUName(String userName):**
Visszaadja a „userName” felhnevű felhasználó által hirdetett tárgyakat.
- **addItem(Item item): void:**
 - A tárgyat rögzíti az adatbázisban.
 - Az **ItemsController** használja ezt az eljárást új tárgy hirdetésekor. (**PostItem**)
- **GetItemWithCategAndHirdeto(int Id): ItemDTO**
 - Hibát dob, ha nincs ilyen tárgy.
 - A Tárgyat kikeresi, feltölti a **Category**, **Hirdeto** propertyjeit. És ItemDTO-vá konvertálja.
- **IsUserHirdetoOf(String uname, int itemId):**
 - ha nem létezik ilyen azonosítójú tárgy, vagy nem az **uname** nevű felhasználó hirdette, akkor hamisat ad vissza, amúgy pedig igazat.
- **IsCategoryExists(int id): bool**
 - megmondja, hogy létezik e kategória ilyen azonosítóval
- **GetItemWithAllLicits(Int Id): ItemAllLicitDTO**
 - hibát dob, ha nincs ilyen azonosítójú tárgy
 - az id azonosítójú tárgyat visszaadja **ItemAllLicitsDTO**-vá konvertálva, miután feltöltötte a tárgy licitjeinek listáját
- **CloseLicit(int Id):**
 - ha nem találja a tárgyat, akkor **InvalidOperationException**-t dob(.Single()).
 - megkeresi az id azonosítójú tárgyat, feltölti a navigational property-jeit,
 - Ezután megkeresi a licitet vezető felhasználót(itt feltételezzük, hogy licitáltak a tárgyra, ezt a controllerben előbb ellenőrizni kell)
 - lezáraja a szavazást(item.DateOfClosing beállításával);
 - A tárgy sikeres frissítése esetén ItemAllLicitDTO típusban küldjük vissza a lezárt licitálást és a nyertes felhaasználó adatait,
 - ha nem sikeres, akkor DbUpdateException-t dob

Api kontroller unittest - Tesztesetek:

```
GetHirdetettTest1
elvárt: visszakapjuk neil1234 által hirdetett 3 darab tárgyat
GetHirdetettTest2
elvárt: visszakapjuk bruce11 által hirdetett 4 darab tárgyat
GetHirdetettTest3
elvárt: a neil1234 által meghirdetett 3 darab tárgyhoz tartozó aktuális licitét visszaadja a
GetHirdetett(), amik rendre: 20005 8006 8000

GetItemDetailsTest1Async
elvárt: visszaadja az 1-es id-vel rendelkező tárgyat ItemAllLicitDTO formájában az 5 darab
rá érkezett licittel együtt
GetItemDetailsTest2
elvárt: visszaadja az 2-es id-vel rendelkező tárgyat ItemAllLicitDTO formájában a 6 darab
rá érkezett licittel együtt
GetItemDetailsTest3
elvárt: unauthorized-nak kell az eredménynek lennie:
GetItemDetailsTest4
nem letezik, ezért az eredménynek notfound-nak kell lennie:

PostItemTest1
elvárt: sikeresen rögzíti a tárgyat:
PostItemTest2
elvárt: badrequest hibás azonosítójú kategória miatt:
PostItemTest3
elvárt: badrequest hibás keydo licit miatt
PostItemTest4
elvárt: badrequest letelt zarodatum miatt

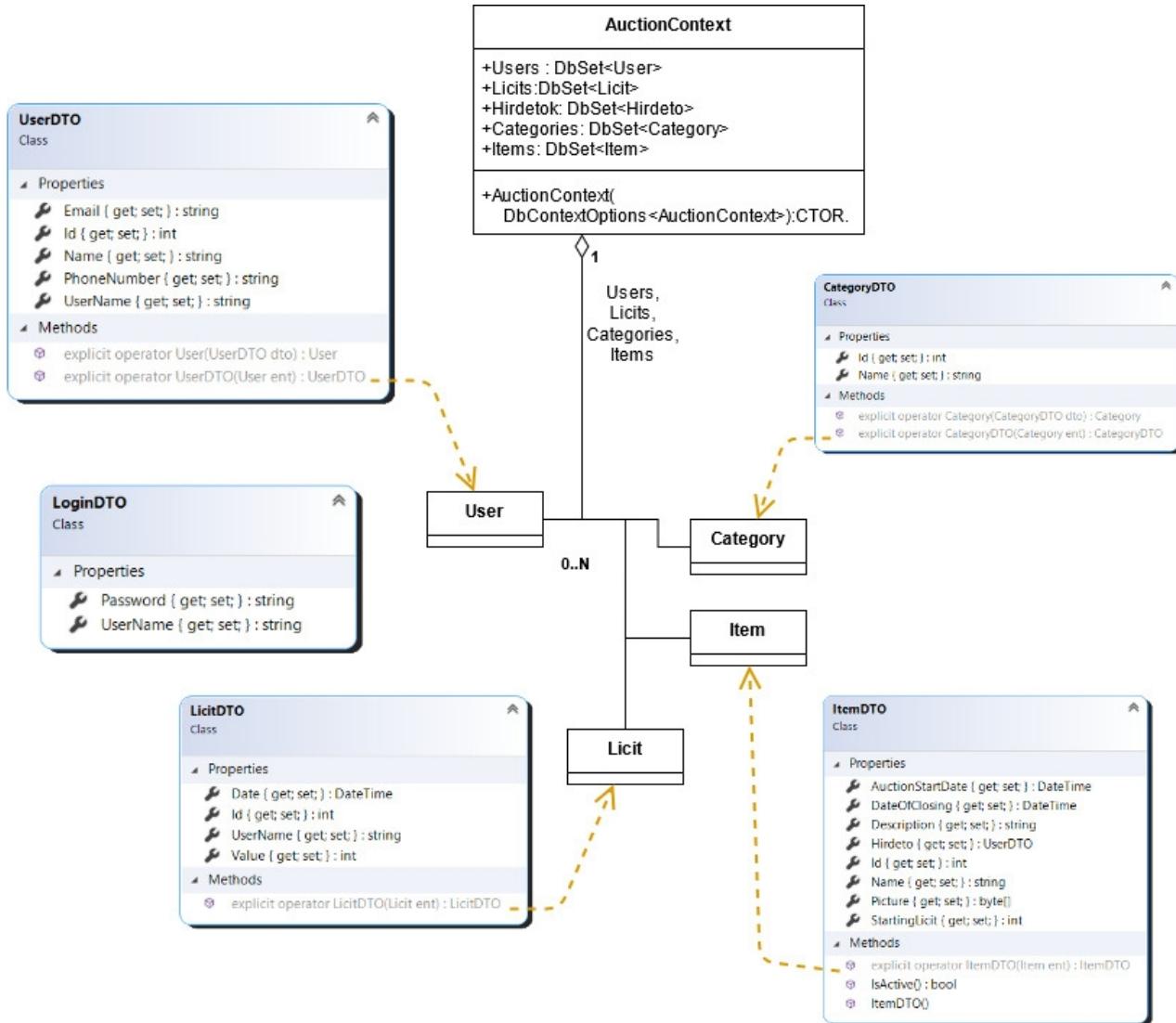
CloseLicitTest1
elvárt: sikeresen lezárja a licitjét, és a legnagyobb licitet letevő user nyer:
CloseLicitTest2
elvárt: nem zarhatja le, mert nincs rá licit
CloseLicitTest3
elvárt: unauthorized, mert már leyarult a licit
CloseLicitTest4
elvárt: unauthorized, mert nem zarhat le más által hirdetett licitet

GetItemTest1Async
elvárt: sikeresen visszaadja a kért azonosítójú tárgyat
GetItemTest2
elvárt: nem az Ő tárgya, ezért unauthorized az eredmény:
GetItemTest3
nem letezik, ezért az eredménynek notfound-nak kell lennie:

GetCategoriesTest1
elvárt: a GetCategories() visszatér a 3 kategóriaát tartalmazó listával
GetCategoriesTest2
elvárt: mivel először a tesztelés elején egy új kategóriát, a GetCategories() az új
kategóriát is visszaküldi a listában
```

Alkalmazás Perzisztencia:

Perzisztencia osztálydiagramja



DTO-K:

szeríalizálható objektumok, az api és a Desktop alkalmazás Perzisztencárésze ilyen obejktumok küldésével kommunikál egymással.

CategoryDTO, ItemDTO, LicitDTO, LoginDTO, UserDTO: a hasonló nevű entitásmodellek adatait tárolják(navigációs propertyk nélkül), és tartalmaznak egy-egy konverziós operátort is hogy DTO-vá lehessen az entitásokat konvertálni.

ItemLicitDTO:

a desktop alkalmazásban a meghirdetett tárgyak listája ilyen típusú obejktumokat tartalmaz.

Az Item típus adatain kívül egy **ActiveLicit** property-t is tartalmaz. Ez a a tárgy lictálására vonatkozó legnagyobb licit értékét tárolja.

ItemAllLicitDTO:

a desktop alkalmazáshoz a kiválasztott tárgy részletes adatainak elküldésére szolgál.

a következő DTO-kat tartalmazza magában:

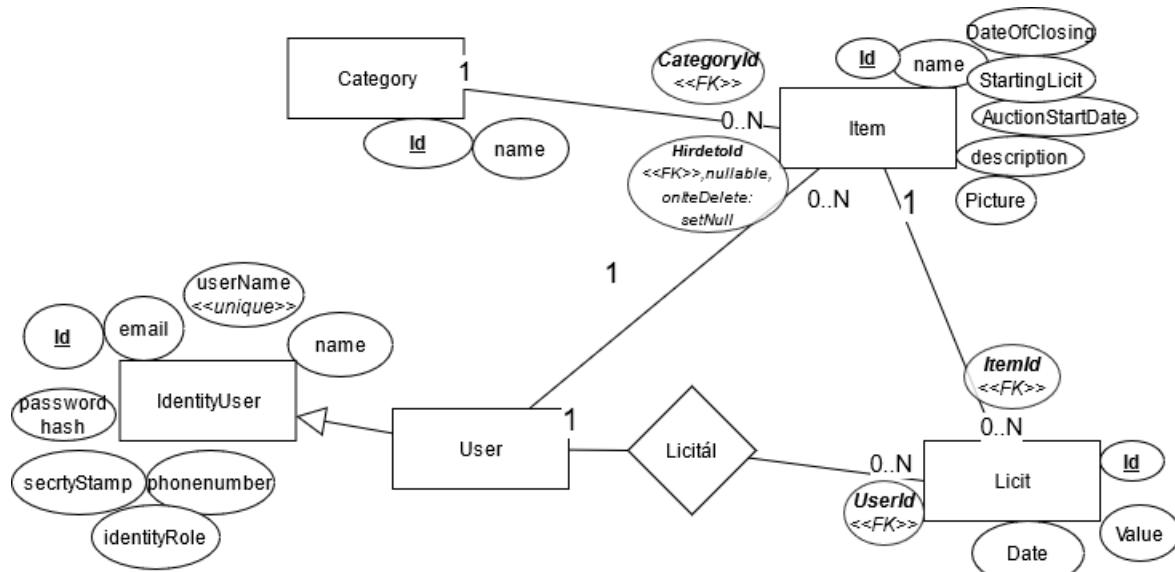
- item: ItemDTO,
- licits: Ilist<LicitTO>
az összes licit, ami a meghirdetett tárgyra érkezett
- hirdeto: UserDTO
- winner: UserDTO
ha nem licítáltak a tárgyra, akkor null, amúgy
a licitet vezető felhasználó(aki egyben a nyertes is amennyiben lezártult a licit)

tartalmaz egy eljárást, amivel egy Item és Egy User objektummal létrehozhatunk egy **ItemAllLicitDTO**-t. A User megadásával a winner propertyt tölthetjük fel.

Adatbázis:

- **DBSET<USER> USERS**
regisztrált felhasználók(közülük pár „hirdeto” role-t kap, így ők hirdethetnek is)
- **DbSet<Category> Categories**
meghirdetett tárgyakhoz választható kategóriák(bútor, ingatlan, egyéb)
- **DbSet<Item> Items**
licitálásra meghirdetett tárgyak
- **DbSet<Licit> Licits**
meghirdetett tárgyakra leadott licitek

Az adatbázis EgyedKapcsolat diagramja:



Megjegyzés:

Az Item tábla User felé mutató Hirdetőd külsőKulcsa nullázható, és A User(aki hirdette a tárgyat) törlésekor a default(„ON DELETE CASCADE”) rekurzív törlés helyett Null-ra állítódik.

Ez azért kell, mert nem enged olyan ER struktúrát kialakítani, amiben törléskor több tábla felé is el kéne indulnia a kaszkádosításnak, márpédig Egy user törlésekor a Licitjeit, és Hirdetett Tárgyait is törölné a rendszer az on delete cascade szabályai szerint.

Emiatt egy User törlése előtt „manuálisan” kell törölni az általa hirdetett tárgyakat, hogy az adatbázisban ne legyenek null foreignKey-ek.(bár az alkalmazás nem is ad lehetőséget felhasználók törlésére, ezért ebből nincs probléma)

Teljes alkalmazás uml diagramja

