Radnai Miklós
A4U58P

3. feladat(Hírportál)

Készítsük el egy online portál hírkezelő rendszerét,
ahol a munkatársak feltölthetik cikkeiket, amelyek megjelennek egy webes felületen.

1. részfeladat: A webes felület tartalmazza magát a hírportált, ahol az olvasok tetszőlegesen böngészhetik a híreket.

•A főoldalon megjelennek a legfrissebb hírek (cím és összefoglaló,dátum szerint csökkenő sorrendben, legfeljebb 10), illetve a lap tetején kiemelten a vezető cikk (cím és összefoglaló) képpel (amennyiben több kép tartozik a cikkhez, az első jelenik meg, kicsinyített méretben).

•A címet kiválasztva megjelenik a teljes tartalom(beleértve a szerző nevét, illetve a bevitel, vagy utolsó módosítás dátumát)képpel (ha van kép a cikkhez rendelve, amennyiben több kép tartozik a cikkhez, az első jelenik meg, kicsinyített méretben). A képet kiválasztva megjelenik a cikkhez tartozó képgyűjtemény, ahol egyenként lapozhatunk a képek között, illetve visszaléphetünk a cikkhez.

•A hírportál tartalmaz egy archívumot, ahol dátum szerint csökkenő sorrendben listázódnak a hírek (cím és összefoglaló). Egy oldalon legfeljebb 20 hírt láthatunk, a többiért lapozni kell. Az archívumban lehet keresni is, megadott dátumra, cím(részlet)re, vagy tetszőleges szóra a cikk tartalmából

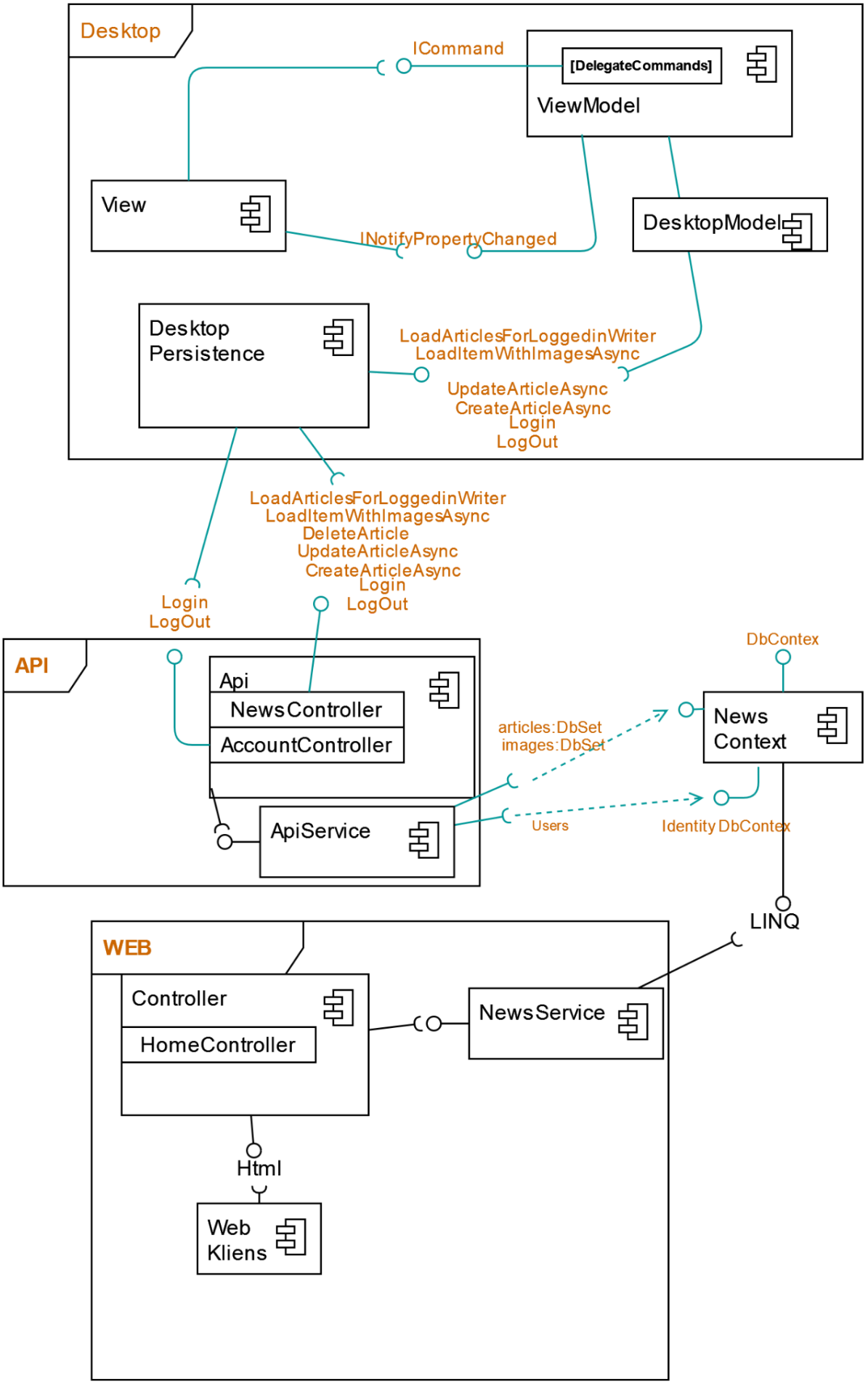
2. részfeladat:
Az asztali grafikus felületet tehát a portál munkatársai használják a cikkek írására, illetve feltöltésére.

- A program használatához először be kell jelentkeznie a munkatársnak a felhasználónév és a jelszó
megadásával. Ezt követően válnak elérhetővé a szerkesztési funkciók (illetve a kijelentkezés).
- A főablakban a saját cikkek listázódnak dátum szerint (cím, író, dátum), amelyeket módosíthatunk,
illetve törölhetünk is.
- Új cikk felvitelénél meg kell adnunk a címet, az összefoglalót (max. 1000 karakter), valamint a teljes
szöveget. Ezek kitöltése kötelező. A cikk beállítható vezető cikknek, ekkor azonban kötelező legalább egy képet feltölteni hozzá. Ezen felül feltölthetnek tetszőleges számú képet a cikkhez. A program biztosítsa, hogy egyszerre csak egy vezető cikk lehessen.
- Cikk módosításánál is ugyanezt a felületet kapjuk vissza, de már előre kitöltve.
- Cikk törlésénél a program megerősítést kér a felhasználótól.

A teljes program komponensDiagramja:

Radnai Miklós
A4U58P

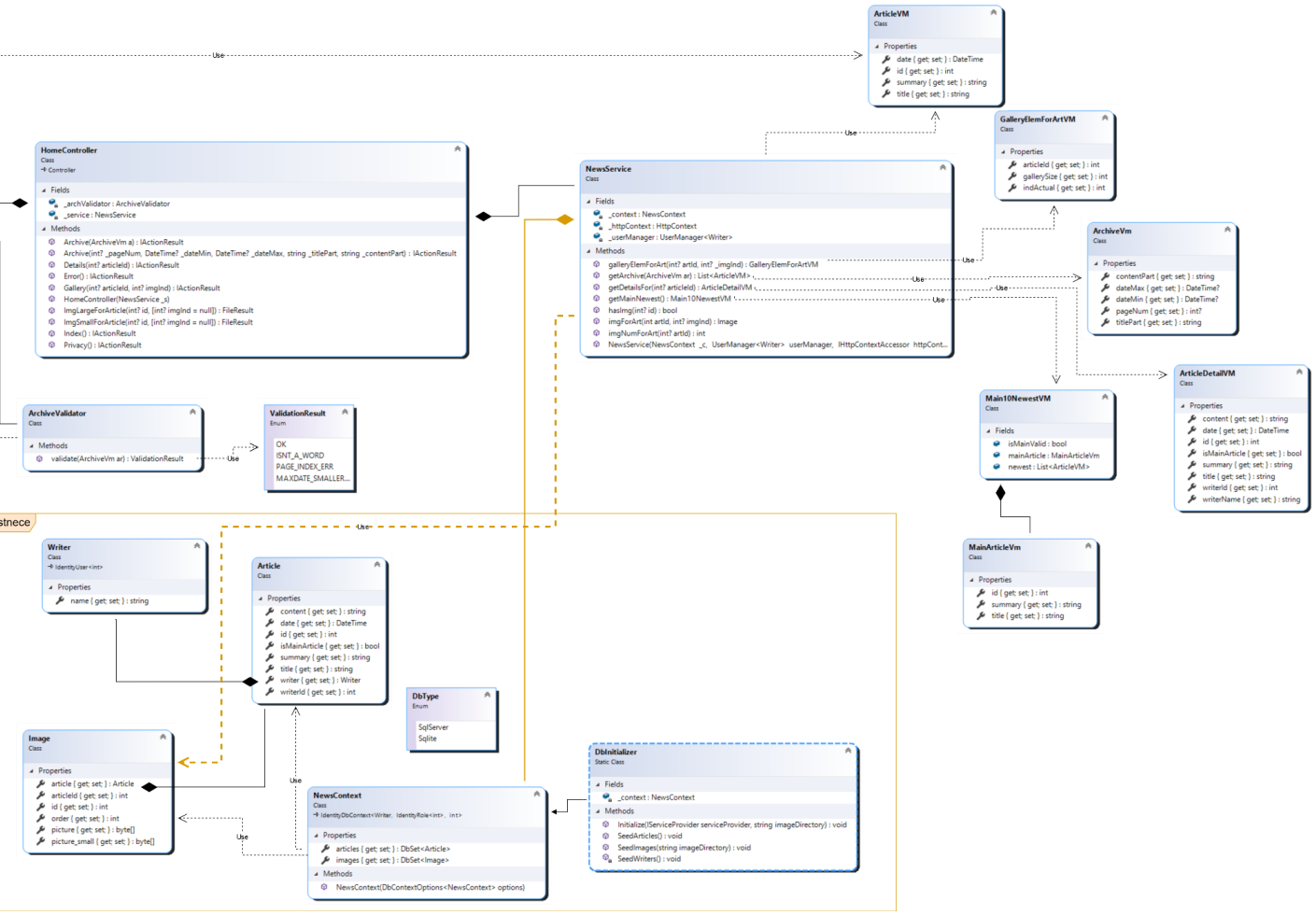
3. feladat(Hírportál)



Radnai Miklós
A4U58P

3. feladat(Hírportál)

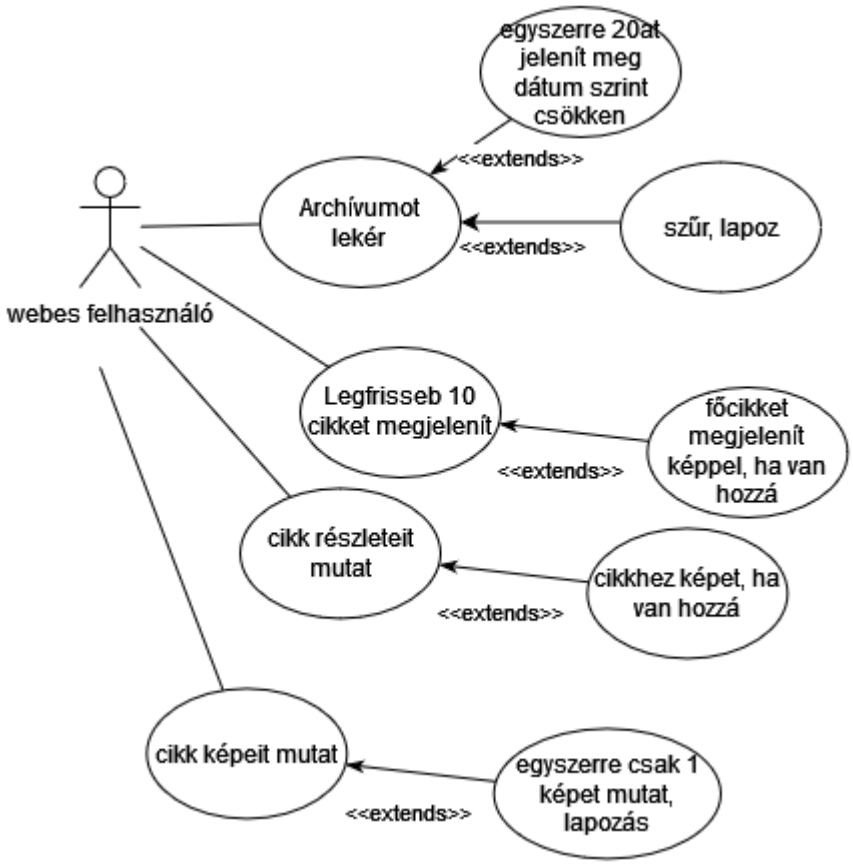
Webes felület osztálydiagramja:



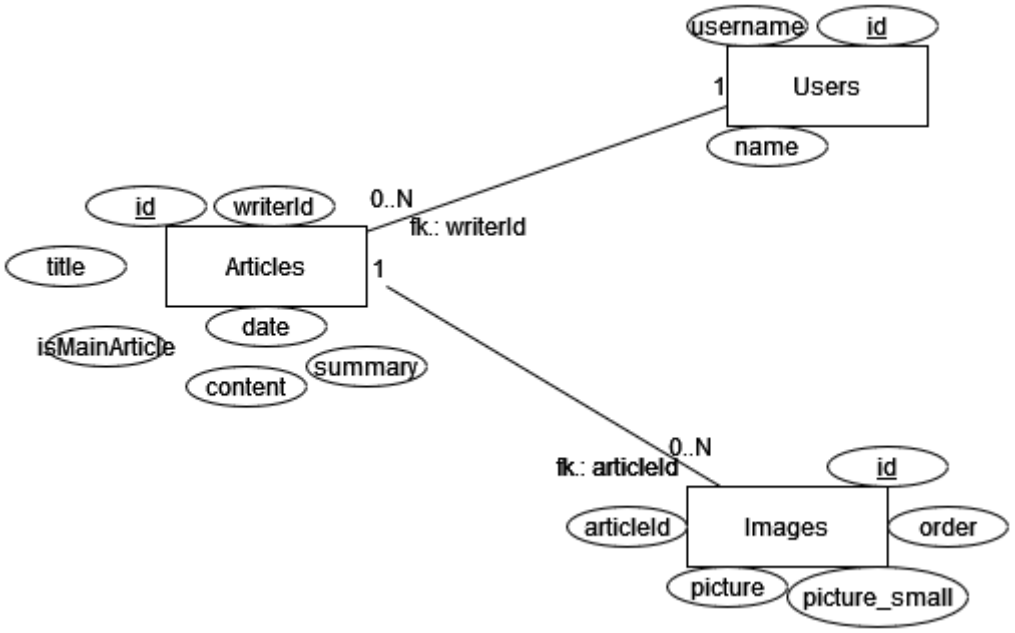
Webes alkalmazás UseCase diagramja:

Radnai Miklós
A4U58P

3. feladat(Hírportál)



Egyedkapcsolat diagram:



Radnai Miklós
A4U58P

3. feladat(Hírportál)

Elemzés:

Perzisztencia réteg:

Article

- **id :** int
- **title :** string
- **writerId :** int
- **virtual writer : Writer**
- **date :** DateTime
- **summary :** string
- **content :** string
- **isMainArticle :** bool

Egy cikk adatait tárolja:
a cikk azonosítóját, címét, a cikk íróját navigational property-ként, a cikk írásának dátumát, összefoglalóját, tartalmát, és egy a **isMainArticle** logikai változót, ami meghatározza, hogy egy cikk vezércikk-e.

Image

- **id :** int
- **order:** int
- **articleId :** int
- **virtual article: Article**
- **picture :** Byte[]
- **picture_small :** Byte[]

Egy cikk egy képét és a kép kicsinyített mását tárolja:
az objektum azonosítóját, egy sorszámot(ahol egy cikknek több képe közül egyet kell megjeleníteni, ott a legkisebb **order**-el rendelkezőt választjuk majd),
a cikket navigational property-ként, egy képet és annak kicsinyített változatát.

Writer: IdentityUser<int**>**

- **name :** string

Az **IdentityUser**-ből örökli a userName-et, amit a név mellett használunk.
NewsContext : IdentityDbContext<Writer, IdentityRole**<**int**>,** int**>**
az identityDbContext
A **Users** DbSet-ben az írók(akik a felhasználók egyben).,
az **article** DbSet-ben a cikkek,
az **images** DbSet-ben a képek rekordjai találhatók.
A **Users** táblát az IdentityDbContext-ből örökli a NewsContex
Az identityDbContext által nyújtotta lehetőségeket (authentikációt, roleManagement-et) nem használja a program, mert az első részfeladatban nincs rá szükség.

Web réteg:

Nézetmodellek:

ArticleVM

Radnai Miklós
A4U58P

3. feladat(Hírportál)

int id
string title
string summary
DateTime date

Egy cikk adatainak tömör(nincs író, és tartalom ebben) megjelenítéséhez adja át ezt a controller a nézetnek

Main10NewestVM

List<ArticleVM> newest
bool isMainValid
MainArticleVm mainArticle

A főoldal nézetmodellje. A kontroller **newest**-be teszi a legújabb 10 cikket, **isMainValid**-ban tárolja az információt,, hogy pontosan egy vezércikk van-e az adatbázisban(tehát, hogy valid-e a vezércikk amit a **mainArticle**-be tesz, vagy null)és ezt küldi az **Index.**cshtml-nek.

MainArticleVm

int id
string title
string summary

A főoldal nézetmodellje. A főcikk tárolására használom.

ArticleDetailVM

int id
string title
int writerId
string writerName
DateTime date
string summary
string content
bool isMainArticle

A kontroller **Details** akciója a cikk részletes adatait(egyebek mellett író neve, cikk tartalma) ebben küldi a **Details**.cshtml-nek

GalleryElemForArtVM

int articleId
int indActual
int gallerySize

Egy cikkhez tartozó összes kép közötti lapozás megvalósításához kell.
A kontroller **Gallery** akciója ebben küldi a **Gallery**.cshtml-nek a képek közötti lapozáshoz szükséges adatokat. articleId és imgInd azért kell, hogy a nézet lekérhesse a kontrollertől az articleId azonosítójú cikk imgInd-edik képét (a kontroller ImgLargeForArticle(id,imgInd) akciójától), és imgInd kell a lapozás megvalósításához. GallerySize a cikkhez tartozó képek száma van.

ArchiveVm

int? pageNum
DateTime? dateMin

Radnai Miklós
A4U58P

3. feladat(Hírportál)

DateTime? dateMax
string titlePart
string contentPart

Az archívumban való szűréséhez tárol propertyket. A kontroller **Archive** akciója és az **Archive**.cshtml ezzel a nézetmodellel komunikál egymással.
a propertyk lehetnek null-ok, pageNum-ot leszámítva egy null property azt jelenti, hogy nem akarunk szűrni az adott propertyvel.
pageNum a lapozásra való(a lapozás is egyfajta szűrés).
dateMin-el a dátumokat alaulról, dateMax-al felülről lehet szűrni, ha egyik se null, akkor a két szűrődátum közötti zárt intervalumban levő cikkekre szűrünk majd..
Az **ArchiveValidator**-al validáljuk ezt a nézetmodellt, ezen kívül viszont DataAnnotation-ök is megvannak adva, hogy a kontroller külön ellenőrízhesse a nézetmodellt:
dateTime,dateMa dátum típusú kell hogy legyen,
titlePart és contentPart max 20 karakteres lehet, ezen felül contentPart-ban csak a magyar ácécé betűi lehetnek(szóköz sem lehet benne, tehát csak egy magyar szó felel meg). Viszont mindegyik property lehet null is, tehát pageInd-en kívül a null propertyk validak. ha pageInd null, akkor majd az **ArchiveValidator** validációján megbukik, de dataannotationnel nincs validálva.

Nézetek:

A **_Layout elrendező** nézet tartalmaz linket A kezdőoldalra és az Archívumra.
Minden nézet ellenőrzi, hogy null-e a nézetmodellje. ez esetben értesíti a felhasználót a sikertelen betöltésről.

Az **Index.cshtml** Main1öNewestVM nézetmodellt használ, amiből kiderül, hogy pontosan 1 cikk lett-e vezércikknek jelölve. Ha nem, akkor a weblapon a “vezércikk betöltése sikertelen” szöveg jelenik meg.
Amúgy a vezércikk(cím,öszefoglaló) jelenik meg kicsinyített képpel.
Emellett a 10 legfrissebb cikk is megtalálható a weblapon(cím,összefoglaló,dátum).
Ha egy cikkre kattintunk, akkor a cikk részleteit láthatjuk(Details.cshtml)

Details.cshtml:
A cikk cím,összefoglaló, tartalom, és amennyiben tatrozik kép a cikkhez, akkor abból az első jelenik meg kicsinyítve.
Ha van kép, akkor arra kattintva a **Gallery.cshtml**-re jutunk

Gallery.cshtml:
A cikkhez tartozó képek között böngészhetünk. Egyszerre egy kép jelenik meg, de lapozhatunk a következő és előző képre.
GalleryElemForArtVM nézetmodellben kapja meg, hogy a cikkhez hány kép tartozik, és az épp megjelenített kép ezek közötti sorszámát is. Ez a két szám a kép alatt látható.
Ha nincs kép a cikkhez, akkor a kép helyén szövegesen jelezzük ezt.
A lap alján található linkre kattintva vissza ugorhatunk a cikk részleteire(Details.cshtml).

Archive.cshtml:
Dátum szerint csökkenő sorrenben jelennek meg a cikkek(csak cím és összefoglaló)
Felül keresési paraméterek láthatóak, amikkel az archívumot szűrhetjük:

Radnai Miklós
A4U58P

3. feladat(Hírportál)

címrészlet(max 20 karakter), keresőszó(egy darab magyar szó),kezdő, és végdátum(a kezdő dátum nem lehet a végdátum utáni).
Lapozhatunk az archívumban a lent megjelenő linkekkel, illetve a szűrőparaméterek között megadhatunk oldalszámot is(az oldalszám nem lehet negatív).
Egyszerre legfeljebb 20 cikk látható.
Ha egy szűrőparamétert hibásan ad meg a felhasználó, arról értesíti a program.
A validációt az **ArchiveValidator** osztály, és a Controller beépített validátorával végezzük el.

NewsService:

imgForArt(articleId,imgInd): **Image**
az **articleId** cikkhez tartozó képek közül visszaadja sorrendben az **imgInd**-ediket. Ha nincs ilyen, null-t ad vissza.

hasImg(articleId): bool
Visszaadja, hogy tartozik-e **articleId** azonosítójú cikkhez kép.

getMainNewest():**Main10Newest**
a nézetmodellben a főcikket, és a legújabb 10 cikket adja vissza dátum szerinti csökkenő sorrendben. Ha nem pontosan 1 cikk van vezércikknek jelölve az adatbázis,ban, akkor vzércikk helyett null-t ad vissza, és a nézetmodell **isMainValid** logikai változóban jelzi ezt.

getDetailsFor(articleId): **ArticleDetailsVM**
a **ArticleDetailsVM** nézetmodellben a **articleId** azonosítójú cikk részletit(id, tartalom,dátum,főcikk-e,öszefoglaló,cím,szerző neve) adja vissza, ha van ilyen azonosítójú cikk. amúgy null-t.

galleryForArt(articleId,imgInd): **GalleryElemForArtVM**
a **GalleryElemForArtVM** nézetmodellben **articleId-**t, a cikkhez tartozó képek számát és **imgInd**-et adja vissza.

getArchive(ArchiveVM arch): **List<ArticleVM>**
ha arch.pageInd null, akkor 0-ra cseréli, ha titlePart, contentPart null, akkor üresstringre, cseréli az eljárás.
ha dateMin null, akkor DateTim.MinValue-t, ha dateMax null, akkor DateTime.MaxValue-t használ majd az eljárás a dátumszűréshez, viszont ekkor az **arch** nézetmodell dateMin és dateMax propertyjét meghagyja null-nak.
Az eljárás **arch** propertijei alapján szűri a cikkeket(cím,tartalom, kezdőés végdátum), rendezi dátum szerint csökkenő sorrenben ezeket, és az első arch.pageNum*20 darabot átugorja, és 20 darabot tart meg a maradékból. (Így valósítja meg a lapozás funkciót).
Az így kapott cikkeketet **ArticleVM**-é alakítja (megtartja a cikkek id-jét, dátumát,összefoglalóját,címét).

ArchiveValidator:

Validate(ArchiveVM arch): **ValidationResult**
ha arch.**contentPart** tartalmaz szóközt, akkor **ISNT_A__WORD-**öt,
ha arch.**pageInd** null, vagy negatív, akkor **PAGE_INDEX_ERR**-t,

Radnai Miklós
A4U58P

3. feladat(Hírportál)

ha arch.**dateMin** > arch.**dateMax**, akkor **MAXDATE_SMALLER_THAN_MIN** hibát ad vissza, amúgy pedig **OK**-t.

HomeController:

property-jei:

_service : **NewsService**

_archValidator: **ArchiveValidator**

ImgSmallForArticle(int? id, int? imgInd = null) : FileResult
visszaadja az id azonosítójú cikk képei közül az imgIn-ediknek kcisinyített változatát ha van ilyen, amúgy null-t

ImgLargeForArticle(int? id, int? imgInd = null) : FileResult
visszaadja az id azonosítójú cikk képei közül az imgIn-ediknek a normál méretű változatát ha van ilyen, amúgy null-t

Index()
Készít egy Main10NewestVM nézetmodellt a _service.geMainNewest() függvényével és visszaadja. A ViewBag-ben elküldi a nézetnek, egy boolean formájában, hogy van-e kép a vezércikkhez.

Details(int? articleId)
A **Details.cshtml** nézetet adja vissza egy **ArticleDetailVM** nézetmodellel, amit a **_service**.**getDetailsfor**(articleId) hívással állítunk elő.
A ViewBag-ben elküldjük a nézetnek, egy boolean formájában, hogy tartozik-e kép a cikkhez.

Gallery(int? articleId, int? imgInd)
A ViewBag-ben elküldjük a nézetnek, egy boolean formájában, hogy tartozik-e kép a cikkhez.
A **Gallery.cshtml** nézetet adjuk vissza amihez a **GalleryElemForArtVM** típusú nézetmodellt a _service.**galleryElemForAr**t(articleId, imgInd) hívással állítjuk elő.

[**HttpGet**]
Archive(int? _pageNum, DateTime _dateMin, DateTime? _dateMax, string _titlePart, string _contentPart)
ha titlePart null, akkor üres string-re cseréljük
ha contentePart null, akkor üres string-re cseréljük
Ezekből elkészít egy **ArchiveVM** nézetmodellt, amit a keretrendszer validátorával(az **ArchiveVM** DataAnnotation-jei alapján) és a **_archValidator**-al is validál. Ha a utóbbi validáción bukik meg, akkor manuálisan adunk hozzá ModelError-t a nézetmodellhez:
ISNT_A_WORD: "Csak 1 szót adhat meg itt"
MAXDATE_SMALLER_THAN_MIN:"A kezdeti dátum nem lehet későbbi a végdátumnál"
PAGE_INDEX_ERR:”hibás oldalszám"
Validációs hiba esetén azonnal visszaküljük az Archive.cshtml-t a hibás nézetmodellel.
Amúgy cikkeket rendezük dátum alaján, szűrjük a nézetmodell propertijei alapján; ebből a pageNum-adik llapot adjuk vissza A szűrést(és lapozást) és rendezést a _service.getArcive(<nézetmodell>) függvény végzi el nekünk.
A ViewBag-ben küldjük, hogy lehet-e lapozni előző, illetve következő oldalra.

Radnai Miklós
A4U58P

3. feladat(Hírportál)

[**HttpPost**]
Archive(ArchiveVm arch)
ha **arch** null, akkor Index.cshtml oldalt küldi vissza, amúgy
a [HttpGet]Archive akcióra irányítjuk át a kérést és **arch** propertijeit adjuk át paraméterekként.

Startup.cs:

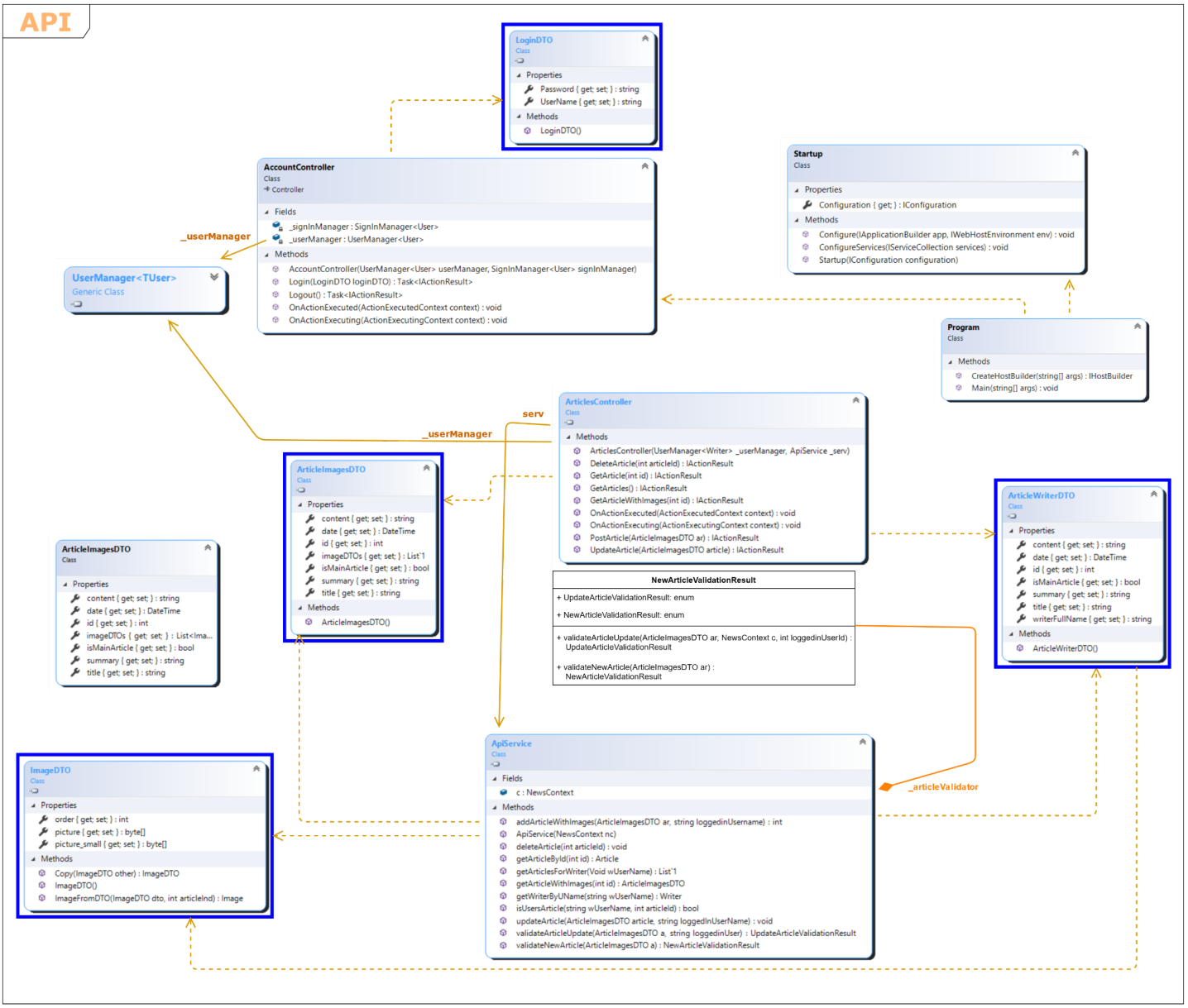
lekéri a Startup osztály **Configuration** propertyjétől a DbType-ot amit az appsettings.json fileban adtunk meg, és a tatralma alapján dönti el a program, hogy sqLite, vagy SQLServer adatázishoz kell-e csatlakozni.
A **Configure** eljárás inicializálja az adatbázist.

Radnai Miklós
A4U58P

3. feladat(Hírportál)

WebApi réteg:

WebApi osztálydiagramja:



Radnai Miklós
A4U58P

3. feladat(Hírportál)

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.

ArticleValidator:

Mezői:

- UpdateArticleValidationResult : enum
  ArticeImagesDTO típus validációs hibáinak jelzésére. Cikk frissítése előtt az új állapot validálásához
  lehetséges értékei:
  NOT_YOUR_ARTICLE, ARTICLE_NOT_FOUND, OK, NULL_TTITLE, NULL_SUMMARY, SUMMARRY_EXCEEDS_1000, NULL_CONTENT, MAIN_BUT_NOIMAGE, NULL_IMAGES, AN_IMG_IS_NULL, AN_IMG_IS_MALFORMED

- NewArticleValidationResult : enum
  ArticeImagesDTO típus validációs hibáinak jelzésére. Cikk létrehozása előtt a létrehozandó cikk validálásához.
  lehetséges értékei:
  OK, NULL_TTITLE, NULL_SUMMARY, SUMMARRY_EXCEEDS_1000, NULL_CONTENT, MAIN_BUT_NOIMAGE, NULL_IMAGES, AN_IMG_IS_NULL, AN_IMG_IS_MALFORMED

Eljárásai:

- validateArticleUpdate(ArticleImagesDTO ar, NewsContext c, int loggedinUserId) : UpdateArticleValidationResult
  Cikk frissítése előtt az új állapot validálásához
- validateNewArticle(ArticleImagesDTO ar): NewArticleValidationResult
  Új cikk létrehozása előtt annak validálásához.

ApiService:

ArticlesController használja. Az adatbázison hajt végre lekérdezéseket, módosításokat.

Mezői:

Radnai Miklós
A4U58P

3. feladat(Hírportál)

c: NewsContext

adatbáziskontextus elérése, amit a keretrendszer szolgáltat

auv: **ArticleValidator**

A cikkek validálásához. A cikkek rögzítése és módosítása előtt kell használni, hogy ellenőrízze az új/módosítandó cikk mezőit.

Eljárásai:

- **isUsersArticle**(string wUserName, int articleId) :bool

  ellenőrzi, hogy a felhasználóé-e articleId-hez tartozó cikk.
  akkor is hamisat ad vissza ha nem létezik articleId azonosítójú cikk

- **getWriterByUName**(string wUserName) : Writer

  Visszaadja a wUserName nfelhaszbn. nevű írót

- **getArticleById**(int id): Article

  visszaadja a kért cikket. ha nincs ilyen, akkor null-t

- **getArticleWithImages**(int id) : ArticleImagesDTO

  visszaadja a kért cikket és az összes hozzá tartozó képet(ImageDTO-t)

- **getArticlesForWriter**(string wUserName) : List<ArticleWriterDTO>

  a wUserName felhasználónevű író összes cikkjét adja vissza dátum szerint rendezve.
  az ArticleWriterDTO objektum tartalmazza az író teljes nevét is.

- **validateArticleUpdate**(ArticleImagesDTO a, string loggedinUser): UpdateArticleValidationResult

  a feladatot auv-nak továbbítja

- **validateNewArticle**(ArticleImagesDTO a) : NewArticleValidationResult

  a feladatot auv-nak továbbítja

- **updateArticle**(ArticleImagesDTO adto, string loggedInUserName)

  ha nincs adto.Id azonosítójú cikk, akkor InvalidOperationException hibát dobunk.
  Ha főcikké akarjuk változtatni a módosítandó cikket, akkor előbb az előző főcikket sima cikkre változtatjuk.
  adto-ban levő képeket rögzítjük(a képek **articleId** külsőkulcsát most tudjuk, ezért nem nehéz létrehozni a képeket a megfelelő articleId-val.az addArticleWithImages-nél viszont nem ilyen egyszerű)

- **deleteArticle**(int articleId)

  törli a cikket ha volt ilyen azonosítójú. amúgy InvalidOperatioException-t dob.
  Ha főcikket töröltünk, akkor 0 főcikk lesz

- **addArticleWithImages**(ArticleImagesDTO ar, string loggedinUsername): int

  Hasonló a működése az updateArticle-höz
  Ha főcikként akarjuk létrehozni cikket, akkor előbb az előző főcikket sima cikkre változtatjuk.
  létrehozzuk (mostmég csak a memóriában, nem az adatbázison) az új cikket ‘**ar**’ alapján

Radnai Miklós
A4U58P

3. feladat(Hírportál)

Ezután a **SaveChanges**-t meghívjuk, hogy a rendszer elkészítse az új cikk azonosítóját, és ezáltal létrehozhassuk ‘**ar**’-ben levő ImageDTO-kat ezzel az azonosítóval.
Ezután megkísérreljük az adatbázisba rögzíteni ‘**ar**’-ben levő imageDTO-kat.
Ha az előző **SaveChanges** lefuttt, de ez valamiért DbUpdateExceptiont dob és nem hajtja végre a képek létrehozását, akkor inkonzisztenssé válhat az adatbázis(ha főcikknek jelöltük, akkor ebben esetben maga a cikk sikeresen rögzült, de egy kép se fog hozzá tartozni).
Ezért a második **SaveChanges** esetleges hibazeneteét elkapjuk, és a catch blokkban töröljük azt, és ezt a módosítást is rögzítjük az adatbázisban.

ArticlesController leírása:

ApiService használatával validálja a kérésbeli modelleket és aztán elvégezteti az adatbázisbeli módosításokat

private ActionResult<Article> GetArticle(int id)
A PostArticle CreatedAt akciója hivatkozik erre de a Desktop alkalmazás nem használja ezt az akciót
elérés: GET api/Articles/id
<response code="200">visszaadja a kért cikket</response>
<response code="404">nincs ilyen azonosítójú cikk</response>
<response code="500">szerverhiba</response>

public ActionResult<ArticleImagesDTO> GetArticleWithImages(int id)
csak bejelentkezett íróknak.
visszaadja bejelentkezett felhasználó által kért cikket képekkel
GET api/Articles/articlewithimages/{id}
<param name="id"></param>
<returns>A bejelentkezett felhasználó által kért cikket képekkel</returns>
<response code="200">A bejelentkezett felhasználó által kért cikket képekkel</response>
<response code="404">nincs ilyen azonosítójú cikk</response>
<response code="400">van ilyen cikk, de nem a bejelentkezett felhasználóé</response>
<response code="401">sikertelen authorizáció</response>
<response code="500">szerverhiba</response>

public ActionResult<IEnumerable<ArticleWriterDTO>> GetArticles()
csak bejelentkezett íróknak.
// visszaadja A bejelentkezett felhasználó cikkjei felhasználó nevével
GET api/Articles/articles
Visszaadja a bejelentkezett felhasználó cikkjei felhasználó nevével
<response code="200">A bejelentkezett felhasználó cikkjei felhasználó nevével</response>
<response code="401">sikertelen authorizáció</response>

3. feladat(Hírportál)

<response code="500">szerverhiba</response>

public IActionResult UpdateArticle([FromBody] ArticleImagesDTO article)
csak bejelentkezett íróknak.
Lecseréli 'article'-ben található azonosítójú cikket 'article' többi adattagjával,
és lecseréli a cikkhez tartozó képeket 'article' képeire,
és szükség esetén beállítja vezércikknek a megtalált cikket, az előző vezérckket pedig sima cikként jelöli meg
Mindezt csak akkor teszi meg, ha van ilyen cikk, a bejelentkezett felhasználóé, minden adattag valid,
az összefoglaló 1000karakter vagy kevesebb, és amennyiben főcikknek jelöltük meg ezt, akkor kell hogy legyen legalább egy kép is hozzá
PUT() api/Articles [fromBody] ArticleImagesDTO article
<param name="article"></param>
<returns>egy http választ</returns>
<response code="200"></response>
<response code="404">Nem létezik 'article'.id azonosítójú cikk</response>
<response code="400">létezik 'article'.id azonosítójú cikk, de 'article' egyik adattagja hibás</response>
<response code="401">sikertelen authorizáció</response>
<response code="500">szerverhiba</response>

public ActionResult DeleteArticle(int articleId)
csak bejelentkezett íróknak.
törli a bejelentkezett felhasználó 'articleid' azonosítójú cikkjét.
Delete api/Articles/{articleId}
<param name="articleId">a törlendő cikk azonosítója</param>
<returns>http választ</returns>
<response code="200">törlés sikeres</response>
<response code="401">sikertelen authorizáció</response>
<response code="404">Nem létezik 'articleid' azonosítójú cikk</response>
<response code="400">Nem a bejelentkezett felhasználóé a cikk</response>
<response code="500">szerverhiba</response>

public ActionResult<ArticleWriterDTO> PostArticle(ArticleImagesDTO ar)
<summary>
csak bejelentkezett íróknak.
Létrehozza 'ar' cikket,
és rögzíti a benne levő képeket,

Radnai Miklós
A4U58P

3. feladat(Hírportál)

/ és szükség esetén beállítja vezércikknek létrehozott cikket, az előző vezérckket pedig sima cikként jelöli meg.
Mindezt csak akkor teszi meg, ha minden adattag valid,
az összefoglaló 1000karakter vagy kevesebb, és amennyiben főcikknek jelöltük meg ezt, akkor kell hogy legyen legalább egy kép is hozzá
POST api/Articles [FrommBody]ArticleImagesDTO ar
</summary>
<param name="ar">a rögzíteni kívánt cikk és képei</param>
<response code="201">a léátrehozott cikket adja vissza</response>
<response code="400">'ar' egyik adattagja hibás</response>
<response code="401">sikertelen authorizáció</response>
<response code="500">szerverhiba</response>
csak bejelentkezett iroknak

A NewsArticle kontroller egységtesztje:

GetArticleWithImagesTest1:
-1 indexű cikket kérjük le
Elvárás:
NotFound üzenettel tér vissza a kontroller

GetArticleWithImagesTestOK:
A kiválasztjuk w1 írónak egy cikkjét, amihez 4 kép tartozik
Elvárás:
A kontroller isszaadja a megfelelő cikket egy ArticleWithImagesDTO objektumban a hozzátartozó 4 képpel.

GetArticlesTestOK1:
“w3” nevű felhasználót bejelentkeztetjük és az ArticlesController GetArticles eljárásával lekérdezzük a cikkjeit
Elvárás:
Visszakapjuk mind a 47 cikkjét ArtileWriterDTO-k Listájában.
A cikkeket dátum szerint növekvő sorrenben kapjuk vissza

GetArticlesTestOK2:
“w1” nevű felhasználót cikkjeit lekérjük az ArticlesController GetArticles eljárásával.
Elvárás:
w1 nevű felhasználó által írt 2 cikket kapjuk vissza dátum szerint rendezve egy IEnumerable<ArticleWriterDTO> formájában.

Radnai Miklós
A4U58P

3. feladat(Hírportál)

mindkét ArticleWriterDTO “writerFullName” propertyje helyesen be van állítva “writer1”-re (“writer1” a “w1” felhasználónevű író teljes neve).

UpdateArticlesTest1():
w2 megpróbálja módosítani w1 cikkjét,
Elvárás:
BadRequest üzenettel tér vissza a kontroller, és a cikk nem változik meg

UpdateArticlesTest2():
nem létező (-1) azonosítójú cikket próbáljuk módosítani,
Elvárás:
NotFound üzenettel tér vissza a kontroller

UpdateArticlesTest3():
A title1 nevű cikket próbáljuk megváltoztatni, és minden adatot validan töltünk ki, kivéve hogy főcikknek jelöljük, de nem töltünk fel vele egy képet sem,
Elvárás:
BadRequest üzenettel tér vissza a kontroller, és a title1 nevű cikk nem változik meg. A két hozzátartozó kép se változik meg.

UpdateArticlesTestOK():
a title1 nevű cikket módosítjuk úgy hogy minden adattagját változtatjuk
Elvárás:
a módosítás után megváltozik az adatbázisban tárolt cikk,
főcikk lesz,
pontosan egy cikk lesz főcikk,
az előző 2 imageDTO helyett mostmár csak az újonnan feltöltött ImageDTO fog a cikkhez tartozni,
az író és a készítés dátuma nem változik meg.

DeleteArticleTest1():
w1 nevű felhasználó megpróbálja törölni a “title” című cikket, amit nem ő írt
Elvárás:
BadRequestObjectResult
se a cikket, se a hozzá tartozó képek nem törlődnek

DeleteArticleTest2():
w1 egy nem létező (-1 azonosítójú) cikket próbál törölni
Elvárás:

Radnai Miklós
A4U58P

3. feladat(Hírportál)

NotFoundObjectResult

DeleteArticleTestOK():
w1 a saját “title1_2” című cikkjét akarja törölni a DeleteArticle akcióval
Elvárás:
a cikk törlődik az adatbázisból a hozzátartozó 4 képpel együtt

PostArticleTest1()
a w1 nevű felhasználó új cikket akar létrehozni, amit főcikknek akar megjelölni, de nem tölt fel hozzá egy képet sem
Elvárás:
a NewsController BadObjectResult-al tér vissza, amiben küldi a “MAIN_BUT_NOIMAGE" hibaszöveget
a cikk nem jön létre.

PostArticleTest2()
a w1 nevű felhasználó új cikket akar létrehozni, de invalid képet tölt fel vele

Elvárás:
a NewsController BadObjectResult-al tér vissza, amiben küldi a “AN_IMG_IS_MALFORMED" hibaszöveget
se a cikket, se a vele együtt feltöltött képet nem rögzítjük az adatbázisba.

PostArticleTest1()
a w1 nevű felhasználó új cikket akar létrehozni, amit főcikknek akar megjelölni, de nem tölt fel hozzá egy képet sem
Elvárás:
a NewsController BadObjectResult-al tér vissza, amiben küldi a “MAIN_BUT_NOIMAGE" hibaszöveget
a cikk nem jön létre.

PostArticleTestOK1()
a w1 nevű felhasználó új cikket akar létrehozni és egy valid képpárt is feltölt hozzá
Elvárás:
a NewsController CreatedAtActionResult-al tér vissza, amiben küldi a létrehozott cikket.
A vele feltöltött képpár is rögzül az adatbázisba

Radnai Miklós
A4U58P

3. feladat(Hírportál)

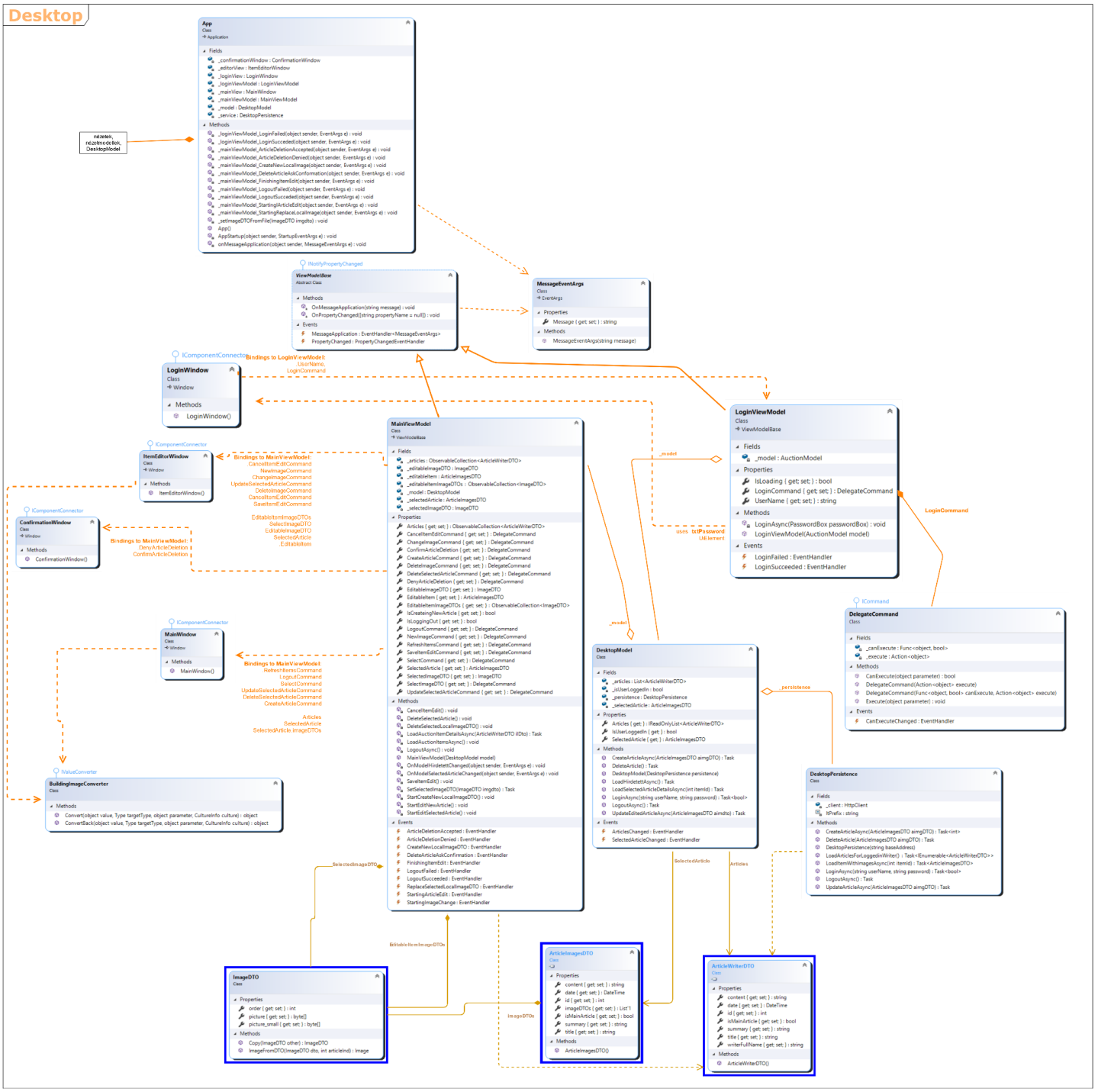
PostArticleTestOK2()
a w1 nevű felhasználó új cikket akar létrehozni, amit vezércikknek szán, és 2 valid képpárt is feltölt hozzá
Elvárás:
a NewsController CreatedAtActionResult-al tér vissza, amiben küldi a létrehozott cikket.
az új cikk az egyetlen vezércikk.
A létrehozandó cikkel feltöltött 2 képpár is rögzül az adatbázisba

Radnai Miklós
A4U58P

3. feladat(Hírportál)

Desktop alkalmazás

Desktop alkalmazás UML osztálydiagramja:



Radnai Miklós
A4U58P

3. feladat(Hírportál)

App:

◦ létrehozza a nézeteket a nézetmodelljeikkel, 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), a képek filból olvasásáért is ez felel.

Nézet réteg:

- **LoginWindow**
  Bejelentkezhet itt a felhasználó a felhasználó nevével és passwordjével
- **MainWindow**
  - Megjeleníti a bejelentkezett felhasználó cikkjeit
  - Egy gombbal az egy cikket kijelölni.
  - Egy másikkal pedig az ItemEditorView nézetre lehet navigálni, hogy szerkesszük a kijelölt cikket
  - mégegy gomba van új cikk írására, ami szintén az ItemEditorWindow-ra vezet bennünket.
  - A cikk törlése gomb a ConfirmationWindowra küld, ami megerősítést kér a cikk törlése előtt.
  - Kijelentkezés gomb a fejlécből érhető el
- **ItemEditorView**
  a MainWindowban kijelölt cikket, vagy az épp elkészítendő cikket lehet eben a nézetben szerkeszteni:
  - képeket lehet feltölteni a cikkhez, és az elkészült cikket menteni lehet.
- **ConfromationWindow**
  a **tovább** gombbal lehet törölni a kijelölt cikket, a **mégse** gommbal vissza lehet vonni a cikk törlését

Nézetmodell réteg:

Az nézetmodell működése nagyvonalakban:
parancskötés:
amikor nézet egy eleme(pl gomb) meghívja a nézetmodell egyik delegáltját, akkor
a delegált meghív egy eljárást,ami
meghívja az **AuctionModel** egy eljárását, és annak az eredményét(sikeres, nem sikeres) a megfelelő EventHandler aktiválásával, és-vagy az OnMessageApplication() meghívásávál jelezzük az **App** felé,
emellett (ha a Model eljárása nem dobott hibát) frissítjük a NézetModellt, hogy az tükrözze a modelben tárolt adatokat.

- **MainViewModel**
  a MainWindow és ItemEditorWindow által megjelenítendő adatokat taralmazza
  propertyjei:
  - ObservableCollection Articles tárolja a felhasználó cikkjeit, amit DelegateCommand RefreshItemsCommand hatására kér le a Model-től

Radnai Miklós
A4U58P

3. feladat(Hírportál)

- ObservableCollection **SelectedArticle** tárolja a kijelölt cikket, amit DelegateCommand SelectCommand hatására kér le a Model-től
- ObservableCollection **EditableItem** tárolja a szerkesztés alatt levő cikket. Ezt az ItemEditorView használja.
  A CreateArticleCommand és **UpdateSelectedArticleCommand** DelegateCommand-ok hatására jön létre a kitöltetlenül illetve az Update.. command esetén a kijelölt cikkből kapja az adatokat.
- ObservableCollection **EditableItemImageDTOs** tárolja a szerkesztés alatt levő cikk képeit.
- ObservableCollection **SelectedImageDTO** tárolja a szerkesztés alatt levő cikk képei közül azt amellyiket a felhasználó kijelölte.
- ObservableCollection **EditableImageDTO** tárolja a szerkesztés alatt levő cikk képei közül azt amellyiket a felhasználó épp szerkeszti.

DelegateCommand-ok:

- DelegateCommand RefreshItemsCommand
  a mainWindow ezzel utasítja a nézetmodellt, hogy kérje le újra az író cikkjeit
- DelegateCommand SelectCommand
  a mainWindow ezzel utasítja a nézetmodellt, hogy kérje le SelectedArticle-be a Command paraméteréhez(egy ArticleWriterDTO-hoz) tartozó képeket
- DelegateCommand **DeleteSelectedArticleCommand**
  itemEditorWindow használja. hatására a nézetmodell jelzi az app-nak hogy jelenítsen meg egy megerősítést kérő ablakot. ha a flehasználó megerősíti, akkor a megjelnített ConfirmationView ablak parancskötéssel meghívja a ConfirmArticleDeletion DelegateCommand-ot.
- DelegateCommand **LogoutCommand**
  mainWindow használja. kijelentkezés kérése
- DelegateCommand **CreateArticleCommand**
  itemEditorWindow használja. a nézetmodell ennek a hatására létrehoz egy üres ArticleImageDTO-t és a EditableItemImageDTOs, EditableImageDTO-t null-ra állítja. **IsCreateingNewArticle** igazra állításával jelezzük, hogy ha a felhasználó készen van, és menteni szeretné a módosításokat, akkor lecseélni kell egy cikket, és nem pedig újat megkísérelni létrehoznia a modellnek
- DelegateCommand UpdateSelectedArticle:
  hasonló a feladat mint CreateArticleCommand meghívása után. A különbség, hogy Is**CtreatingNewArticle-t** hamisra állítjuk, és a kiválasztott cikkadatavial fel kell tölteni **EditableItem**-et
- DelegateCommand SaveItemEditCommand
  EditableItem isCreatingNewArticle logkai értéke alapján vagy egy frissítendő cikk új adatait,vvagy egy létrehozandó cikk adatait tartalmazza. így isCreatingNewArticle értéke alapján létre próbálunk hozatni egy új cikket a

Radnai Miklós
A4U58P

3. feladat(Hírportál)

Modellel, vagy EditableItem adataival szeretnénk módosítani (editableItemben tárolt id alapján) a megfelelő cikket.

- DelegateCommand CancelItemEditCommand
  itemEditorWindow használja. a nézetmodell megszakítja az létrehozandó/módosítandó cikk szerkesztését.
- DelegateCommand **ConfirmArticleDeletion**
  utasítja a Modellt- hogy törölje a kijelölt cikket. ehhez a Modell a saját ÜselectedArticle mezőjét használja
- DelegateCommand **DenyArticleDeletion**
  a ConfirmationWindow használja. hatására az appnak jelezzük, hogy fedje el a ConfirmationWindow-t.(a cikk törlését visszavonta a felhasználó)
- DelegateCommand **SelectImageDTO:**
  az itemEditorWindow használja. htására a nézetmodell a parancs paraméterére állítja SelectedImageDTO-t, és EditableImageDTO-t is.
- DelegateCommand NewImageCommand
  az itemEditorWindow használja. új kép létrehozása
- DelegateCommand DeleteImageCommand
  az itemEditorWindow használja. kép törlése

Modellodell réteg:

DesktopModel:

Mezői:

- **_persisence**: DesktopPersistence
  A modelltől elfedi Webapit. A modell persistencenek a LoadArticlesForLoggedinWriter, LoadItemWithImagesAsync, DeleteArticle, UpdateArticleAsync, CreateArticleAsync függvényeit hívhatja, ami erre html kéréseket intéz a z Api réteg felé, hogy az az Adatbázison elvégezze a módosításokat.
- **Articles**: IReadOnlyList<ArticleWriterDTO>
- **SelectedArticle**:ArticleImagesDTO

Eseményvezérlői:

_persistence megfelelő eljárásait használják, és továbbítják a hibaüzenteket

- **SelectedArticleChanged**
  jelzi a nézetmodellnek, hogy megváltozott a kijelölt cikk, és frissítse magának a modelben tárolt alapján
- **ArticlesChanged**
  jelzi a nézetmodellnek, hogy megváltozotak a felsorolt cikkek és frissítse magának a modelben tárolt alapján

Eljárások:

3. feladat(Hírportál)

sikertelenség esetén NetworkException formájában továbbítják az okát a viewModel felé.

sikeresség esetén frissítik a Modell állapotát

- LoadHirdetettAsync:
  _persistence-től lekéri a cikkeket, és Articles-be tölti
- LoadSelectedArticleDetailsAsync:
  _persistence-től lekéri a cikkhez a képeit
- UpdateEditedArticleAsync:
  _persistence-tnek jelez, hogy változtassa meg a cikket
- DeleteArticle:
  _persistence-től kéri, hogy törölje a cikket
- CreateArticleAsync
  _persistence-től kéri, hogy hozza létre a cikket

Perzisztencia réteg(**DesktopPersistence**):

Mezők:

readonly **_client:** HttpClient
a html kérések küldésére a Webapi felé

Eljárások:

_client használatával kéréseket intéznek az api-hoz. az api sikertelenséget jelző státuszkódjait hibaüzenetekre fordítják, és NetworkExceptionbe csomagolva továbbítják a Modell felé

- async Task<IEnumerable<ArticleWriterDTO>> LoadArticlesForLoggedinWriter()
  az apitól lekéri a cikkeket
- async Task<ArticleImagesDTO> LoadItemWithImagesAsync(Int32 itemId)
  az apitól lekéri a tárgyat a képeivel
- async Task UpdateArticleAsync(ArticleImagesDTO aimgDTO)
  Az apinak Put klérést küld, hogy frissítse a aimdto adataival az aimdt-.id azonosítójú cikket.
- async Task DeleteArticle(ArticleImagesDTO aimgDTO)
  apihoz delete lkérést intéz aimgdto.id azonosítóval
- async Task<int> CreateArticleAsync(ArticleImagesDTO aimgDTO)
  apinak post kérésben küldi aimgDTO-t. ha sikeres, akkor visszaadja az új cikk azonosítóját
- async Task<bool> LoginAsync(string userName, string password)
  post kérésben megkísérli bejelentkeztetni a felhasználót
- async Task LogoutAsync()
  Az api Logout akcióját hívja meg , hogy azkijelentkeztesse a Desktop felhasználót..

Radnai Miklós
A4U58P

3. feladat(Hírportál)

Desktop + Api felhasználói esetei: