Budapesti Műszaki Szakképzési Centrum   
Petrik Lajos Két Tanítási Nyelvű Technikum

SZOFTVERFEJLESZTŐ ÉS -TESZTELŐ TECHNIKUS SZAKMA

Turbo Note

Készítette: Bató Bálint, Egri Dániel Richárd, Kékesi Bence Zoltán

Budapest, 2022.

Budapesti Műszaki Szakképzési Centrum  
Petrik Lajos Két Tanítási Nyelvű Technikum

Nyilatkozat

Alulírott Bató Bálint, Kékesi Bence Zoltán és Egri Dániel Richárd kijelentjük, hogy ez a vizsgaremek saját tudásunk, önálló munkánk terméke.

A vizsgaremek közös részeit Kékesi Bence Zoltán, Egri Dániel Richárd és Bató Bálint készítettük, ezeket pontosan jelöltük.

Aláírás Aláírás Aláírás

Bató Bálint Egri Dániel Richárd Kékesi Bence Zoltán

Budapest, 2022.

[1. Backend 1](#_Toc101391552)

[1.1. Bevezetés 1](#_Toc101391553)

[1.2. Témaindoklás 1](#_Toc101391554)

[1.3. Jellemzők 1](#_Toc101391555)

[1.4. Célok 1](#_Toc101391556)

[1.5. Backend technológiák indoklása 2](#_Toc101391557)

[1.6. Telepítés 3](#_Toc101391558)

[1.7. Adatmodell felépítése 4](#_Toc101391559)

[1.8. API végpontok 11](#_Toc101391560)

[2. Mobil 16](#_Toc101391561)

[2.1. Bevezetés 16](#_Toc101391562)

[2.2. Platform 16](#_Toc101391563)

[2.3. Az Android fejlesztésről 17](#_Toc101391564)

[2.4. Fejlesztői dokumentáció 18](#_Toc101391565)

[2.5. Felhasználói dokumentáció 33](#_Toc101391566)

[3. Asztali alkalmazás 46](#_Toc101391567)

[3.1. Bevezetés 46](#_Toc101391568)

[3.2. Java és a JavaFX 46](#_Toc101391569)

[3.3. Fejlesztői dokumentáció 47](#_Toc101391570)

[3.4. Felhasználói dokumentáció 56](#_Toc101391571)

[Web 60](#_Toc101391572)

[Irodalomjegyzék 61](#_Toc101391573)

[Ábrajegyzék 62](#_Toc101391574)

1. Backend
   1. Bevezetés

A szoftverfejlesztő képzésünk alatt csoportokba szervezve meg kellett alkotnunk egy projektmunkát egy általunk választott témáról. Választott projektünkről kellett készítsünk egy fejlesztői dokumentációt, ami tartalmazza a tapasztalatainkat és alkalmazásunk felépítését, működését és használatát. A dokumentációnak lesz egy közös része a háttérben futó szerverről és 3 egyénenként elkészített rész a különböző frontend platformokról.

* 1. Témaindoklás

Az első ötletünk egy filmeket közösen néző app volt, de ezt az ötletet hamar elvetettük. Aztán arra gondoltunk, hogy mivel tudnánk valami hasznosat készíteni amire szükség is van. A diákok manapság gyakran küszködnek az órai jegyzeteléssel lassú kézírásuk miatt. Viszont nagy részük rendelkezik okostelefonnal, amin már szinte fénysebességgel tudnak írni, mert azon kommunikálnak a barátaikkal, ott fogyasztanak minden médiát stb. Ezért úgy gondoltuk, hogy megkönnyítjük a diákok a dolgát az applikációnkkal, a TurboNote-al, röviden TNote-al.

* 1. Jellemzők

A TNote egy diákoknak tervezett jegyzetelő alkalmazás beépített órarenddel, hogy ne kelljen külön alkalmazásban vagy képen figyelni az órarendet. A jegyzetek felhőben vannak tárolva, ezért az alkalmazás működéséhez kötelező az internet kapcsolat, cserébe bármilyen eszközről elérhetjük az összes jegyzetünket és órarendünket.

* 1. Célok

Az alkalmazás céljai közé tartozik, hogy egy megbízható platformot biztosítson diákoknak, amit sikerült is megvalósítani. Ezalatt értem jegyzetek és órarendek létrehozását, szerkesztését és tárolását. Viszont fontos szempont volt egy letisztult és könnyen használható felület kidolgozása, hogy ne tántorodjanak el a fiatal diákok az alkalmazás használatától.

* 1. Backend technológiák indoklása

Mint minden stabil applikációnak úgy a miénknek is szüksége van egy stabil, megbízható backend technológiára. A technikusi képzésen a Composer csomagkezelőt és Laravel php keretrendszert tanultuk, ami egy modern, stabil és megbízható keretrendszer. A php nyelv nem arról híres, hogy modern vagy egyszerű lenne, de a Composer segítségével új életet tudunk lehelni bele. Ez az új élet ebben az esetben az előbb említett Laravel ami manapság az egyik legelterjedtebb backend keretrendszer. Modern stílusával és számtalan a fejlesztést megkönnyítő eszközével sokkal egyszerűbbé tette backend szerverünk fejlesztését. Ezen eszközök közé tartozik például az artisan command-line interface amivel egyszerűen tudunk gyakran használt kódot és fájlokat generálni, vagy például a Carbon nevű előtelepített kiegészítő ami megkönnyíti a dátumkezelést.

Idővel megtanultuk hogyan kell egy épkézláb api-t összerakni Laravelben és ezután következett a tesztelési fázis. Ezen a ponton mobilban akadt egy problémánk, mert a választott mobil kotlin keretrendszer, amit hálózati kérések kezelésére használunk nem tudott saját számítógépen futtatott backend szerverre hálózati kéréseket küldeni és fogadni. Ezért szukségünk volt egy deploy-olt, tehát a nagyvilág számára elérhető backend szerverre. Itt jön szóba a Heroku.

A Heroku egy weboldal, ami biztosít egy helyet különböző backend szervereknek egy előfizetésért cserébe. Szerencsére vannak ingyenes lehetőségek is csak limitált adatforgalommal. Ezek után szükségünk volt egy adatbázis szerverre is, amit a PostGreSQL biztosít, ami egy open-source megbízható adatbázis szerver.

A Heroku biztosít különböző kiegészítőket is, amikből van, ami elengedhetetlen. Ilyen például a Heroku PostGreSQL ami biztosítja a kapcsolatot a Heroku felhő és a PostGreSQL között. A Heroku által biztosított felhőre fel tudtuk telepíteni a backend szervert és ezek után a Heroku command-line interface-el fel tudtunk kapcsolódni a Heroku felhőre és le tudtuk futtatni az adatbázishoz tartozó migrációkat és fel tudtuk tölteni teszt adatokkal.

* 1. Telepítés
     1. Rendszerkövetelmények

|  |  |
| --- | --- |
| Online backend szerver | Internet kapcsolat |
| Lokális backend szerver | 1.6 GHz vagy gyorsabb processzor  1 Gb vagy több memória |

* + 1. Telepítési útmutató

Ha az online backend szervert szeretnénk használni akkor az alkalmazás használata előtt nyissuk meg a <https://tnote-backend-app.herokuapp.com/> (A mobilalkalmazás és a webalkalmazás teszteléséhez ez kötelező)

A lokális backend szerverhez pedig:

projekt bemásolása a htdocs/www megfelelő almappájába

szükséges utasítások kiadása, ezek leírása

* composer install
* php artisan key:generate --ansi
* php artisan migrate --seed

fejlesztői szerver indítása:

Parancssorból:

* php artisan serve

POSTMAN|THUNDERCLIENT:

* http://127.0.0.1:8000{api végpont}
  1. Adatmodell felépítése
     1. Leírás

Táblák rövid leírása és szerepe:

* Users
  + Felhasználók tárolása
* Personal access tokens
  + A felhasználók bejelentkezésekor és regisztrálásakor generált autentikációhoz szükséges token-ek tárolása
* Notes
  + Jegyzetek tárolása
* Timetables
  + Órarendek tárolása
* Time table elements
  + Órarendekben létrehozott órák tárolása
* Styles
  + Stílusok tárolása
    1. Adattáblák

|  |  |  |  |
| --- | --- | --- | --- |
| **Users (users)** | | | |
| Mező név | Mező típusa | Leírás | Megkötések |
| **id** | bigint(20) | A felhasználó azonosítója | numerikus  egész |
| **name** | varchar(255) | A felhasználó neve | maximum 255 karakter |
| **email** | varchar(255) | A felhasználó email címe | maximum 255 karakter |
| **email\_verified\_at** | timestamp | Mikor hitelesítette a felhasználó az email címét | formátum:  YYYY-MM-DD HH:MM:SS |
| **password** | varchar(255) | A felhasználó jelszava, bcrypt algoritmussal titkosítva | minimum 8 karakter |
| **remember\_token** | varchar(255) | To be decided |  |
| **created\_at** | timestamp | Felhasználó regisztrációja | formátum:  YYYY-MM-DD HH:MM:SS |
| **updated\_at** | timestamp | Felhasználó adatainak utolsó változtatása | formátum:  YYYY-MM-DD HH:MM:SS |

|  |  |  |  |
| --- | --- | --- | --- |
| **Personal access tokens (personal\_access\_tokens)** | | | |
| Mező név | Mező típusa | Leírás | Megkötések |
| **id** | bigint(20) | A token egyedi azonosítója | numerikus egész |
| **tokenable\_type** | varchar(255) | A token melyik laravel modelhez készült | maximum 255 karakter |
| **tokenable\_id** | bigint(20) | A felhasználó azonosítója, amihez a token tartozik | numerikus egész |
| **name** | varchar(255) | Token neve (könnyű megkülömböztetés kedvéért) | maximum 255 karakter |
| **token** | varchar(64) | A token | maximum 255 karakter |
| **abilities** | text | Jogosultságok, amikkel a szerver foglalkozik |  |
| **last\_used\_at** | timestamp | A token utolsó használata | formátum:  YYYY-MM-DD HH:MM:SS |
| **created\_at** | timestamp | A token készítésének ideje | formátum:  YYYY-MM-DD HH:MM:SS |
| **updated\_at** | timestamp | A token utolsó szerkesztése | formátum:  YYYY-MM-DD HH:MM:SS |

|  |  |  |  |
| --- | --- | --- | --- |
| **Notes (notes)** | | | |
| Mező név | Mező típusa | Leírás | Megkötések |
| **id** | bigint(20) | A jegyzet azonosítója | numerikus  egész |
| **content** | text | A jegyzet tartalma |  |
| **ownerId** | int(11) | A jegyzet tulajdonosának az azonosítója | numerikus egész |
| **title** | varchar(255) | A jegyzet címe | maximum 255 karakter |
| **created\_at** | timestamp | Jegyzetek készítésének ideje | formátum:  YYYY-MM-DD HH:MM:SS |
| **updated\_at** | timestamp | Jegyzet utolsó szerkesztése | formátum:  YYYY-MM-DD HH:MM:SS |

|  |  |  |  |
| --- | --- | --- | --- |
| **Timetables (timetables)** | | | |
| Mező név | Mező típusa | Leírás | Megkötések |
| **id** | bigint(20) | Az órarend azonosítója | numerikus  egész |
| **userId** | int(11) | Az órarend tulajdonosának az azonosítója | numerikus egész |
| **name** | varchar(255) | Az órarend neve | maximum 255 karakter |
| **created\_at** | timestamp | Az órarend készítésének ideje | formátum:  YYYY-MM-DD HH:MM:SS |
| **updated\_at** | timestamp | Az órarend utolsó szerkesztése | formátum:  YYYY-MM-DD HH:MM:SS |

|  |  |  |  |
| --- | --- | --- | --- |
| **Time table elements (t\_t\_elements)** | | | |
| Mező név | Mező típusa | Leírás | Megkötések |
| **id** | bigint(20) | Az óra azonosítója | numerikus  egész |
| **ttid** | int(11) | Az órarend azonosítója amihez az óra tartozik | numerikus egész |
| **day** | varchar(255) | Az óra a hét melyik napján esedékes | maximum 255 karakter |
| **title** | varchar(255) | Az óra címe | maximum 255 karakter |
| **description** | varchar(255) | Az óra leírása | maximum 255 karakter |
| **start** | time | Az óra kezdete | formátum: HH:MM:SS |
| **end** | time | Az óra vége | formátum: HH:MM:SS |
| **repeating** | tinyint(1) | Az óra ismétlődik-e vagy sem | 0 vagy 1 |
| **created\_at** | timestamp | Az óra készítésének ideje | formátum:  YYYY-MM-DD HH:MM:SS |
| **updated\_at** | timestamp | Az óra utolsó szerkesztése | formátum:  YYYY-MM-DD HH:MM:SS |

|  |  |  |  |
| --- | --- | --- | --- |
| **Styles (styles)** | | | |
| Mező név | Mező típusa | Leírás | Megkötések |
| **id** | bigint(20) | A stílus azonosítója | numerikus  egész |
| **userId** | int(11) | A stílus készítőjének az azonosítója | numerikus egész |
| **description** | varchar(255) | Az stílus leírása | maximum 255 karakter |
| **style** | varchar(255) | A stílus json formában tárolva | maximum 255 karakter |
| **created\_at** | timestamp | A stílus készítésének ideje | formátum:  YYYY-MM-DD HH:MM:SS |
| **updated\_at** | timestamp | A stílus utolsó szerkesztése | formátum:  YYYY-MM-DD HH:MM:SS |

* 1. API végpontok

Users

GET /api/users

Elvárt bemenet: Nem vár bemenetet.

Elvárt kimenet: A Users táblából az összes sor összes mezőjét a password mező kivételével adja vissza JSON formátumban.

GET /api/users/{id}

Elvárt bemenet: a várt User sor id mezője.

Elvárt kimenet: A Users táblából a megadott id-t tartalmazó sor összes mezőjét a password mező kivételével adja vissza JSON formátumban.

POST /api/users

Elvárt bemenet: Egy Users sor: name, email és password mezője JSON formátumban.

Elvárt kimenet: A létrehozott User sor mezői kivéve a password mező JSON formátumban.

PATCH /api/users/{id}

Elvárt bemenet: A megváltoztatandó User sor id mezője és a megváltoztatandó mezők.

Elvárt kimenet: A megadott id mezőjű User megváltoztatott adatai.

DELETE /api/users/{id}

Elvárt bemenet : A User táblából a törlendő sor id mezője.

Elvárt kimenet: 200-as kódú http válasz.

Notes

GET /api/notes

Elvárt bemenet: Nem vár bemenetet.

Elvárt kimenet: A Notes táblából az összes sor összes mezőjét adja vissza JSON formátumban.

GET /api/notes/{id}

Elvárt bemenet: a várt Notes sor id mezője.

Elvárt kimenet: A Notes táblából a megadott id-t tartalmazó sort adja vissza JSON formátumban.

GET /api/users/{id}/notes

Elvárt bemenet: a várt Notes sor ownerId mezője.

Elvárt kimenet: A Notes táblából a megadott ownerId-val rendelkező sort adja vissza JSON formátumban.

POST /api/notes

Végpont leírása: Új jegyzet létrehozása JSON adatokból

Elvárt bemenet: Egy Notes sor: content, ownerId és title JSON formátumban megadva.

Elvárt kimenet: A létrehozott Notes sorai JSON formátumban visszaadva.

PATCH /api/notes/{id}

Elvárt bemenet: A megváltoztatandó Note ownerId mezője és a megváltoztatott title és content mezők.

Elvárt kimenet: A megváltoztatott Note ownerId, title és content mezője.

DELETE /api/notes/{id}

Elvárt bemenet : A Notes táblából a törlendő sor id mezője.

Elvárt kimenet: 200-as kódú http válasz.

Timetables

GET /api/users/{id}/timetables

Elvárt bemenet : A User táblából egy id mező.

Elvárt kimenet: A Timetable táblából azok a sorok összes mezője ahol a Timetable userId mezője megegyezik a bemenetben megadott id mezővel.

POST /api/timetables

Elvárt bemenet : A létrehozandó Timetable sor mezői, kivéve az id mezőt.

Elvárt kimenet: A létrehozott Timetable sor összes mezője.

PATCH /api/timetables/{id}

Elvárt bemenet : A Timetable táblából a megváltoztatandó sor id mezője, és megváltoztatandó mezői.

Elvárt kimenet: A megváltoztatott sor összes mezője.

DELETE /api/timetables/{id}

Elvárt bemenet : A Timetable táblából a törlendő sor id mezője.

Elvárt kimenet: 200-as kódú http válasz.

TTElements

GET /api/timetables/{id}/selectedtimetable

Elvárt bemenet : A Timetable táblából egy sor id mezője.

Elvárt kimenet: Azok a sorok a TTElement táblából ahol a ttid mező megegyezik a bemenetben megadott id mezővel.

POST /api/ttelement

Elvárt bemenet : A létrehozandó TTElement összes mezője, kivéve az id mezőt.

Elvárt kimenet:A létrehozott TTElement összes mezője.

PATCH /api/ttelement/{id}

Elvárt bemenet : A megváltoztatandó TTElement id mezője és a megváltoztatandó mezők.

Elvárt kimenet: A megváltoztatott TTElement összes mezője.

DELETE /api/ttelement{id}

Elvárt bemenet : A TTElement táblából a törlendő sor id mezője.

Elvárt kimenet: 200-as kódú http válasz.

Styles

GET /api/users/{id}/styles

Elvárt bemenet: a várt Styles sor userId mezője.

Elvárt kimenet: A Stlyes táblából a megadott userId-val rendelkező sort adja vissza JSON formátumban.

POST /api/styles

Elvárt bemenet : A létrehozandó Styles sor mezői: userId, description és style.

Elvárt kimenet: 200-as kódú http válasz.

PATCH /api/styles/{id}

Elvárt bemenet : A Styles táblából a megváltoztatandó sor id mezője, és megváltoztatandó mezői.

Elvárt kimenet: A megváltoztatott sor összes mezője.

DELETE /api/styles/{id}

Elvárt bemenet : A Styles táblából a törlendő sor id mezője.

Elvárt kimenet: 200-as kódú http válasz.

1. Mobil
   1. Bevezetés

Az Android platform nekem már általános iskola óta közel áll a szívemhez, nemcsak a telefonos játékok, hanem az Android operációs rendszer működése is. Ezért elkezdett nemrég érdekelni az Android fejlesztés is. Főleg emiatt amikor frontend platformot választottunk a csoporton belül én egyből az Android-ot választottam. Viszont egy teljes Android alkalmazás fejlesztésével most találkoztam először ezért meglepően nagy kihívást okozott az alkalmazás fejlesztése és a hibák kijavítása. Gyakran ütköztem olyan problémába, amit csak a Stackoverflow vagy online tartalom gyártók segítségével tudtam megoldani. Viszont cserébe rengeteget fejlődtem és számtalan hasznos tapasztalattal gazdagodtam. Ezek után képes voltam egy épkézláb megbízható Android alkalmazást készíteni.

* 1. Platform

Az Android a kedvenc operációs rendszerem telefonokra. A Google kezei között hamar az egyik legnépszerűbb telefonos operációs rendszer lett. Népszerűségét két fő tényezőből nyerte. Először is, az Android nagyon nyílt és mivel Linux alapú, ezért könnyen személyre szabható. Szinte az összes Android-ot futtató telefonokat gyártó cégnek van egy saját Android stílusa, ami miatt minden Android-os telefonokat gyártó cég egy egyedi Android élményt tud nyújtani. Másodszor az Android az egyetlen összetett, stabil operációs rendszer telefonokra, ami versenybe tud szállni az IOS-el.

Amikor először elkezdtem Androidra fejleszteni Java-ban mutattak példákat, viszont hamar rájöttem, hogy ez nem a legoptimálisabb nyelv. Ezért a választott programozási nyelvem a Kotlin lett. A Kotlin nyelvet a Google tulajdonában van és fejleszti, viszont a nyelv ingyenes és az is fog maradni. Az első stabil verziót 2016 februárjában adták ki és azóta folyamatosan fejlesztik. Jelenleg az 1.6.20-as verziót használják a fejlesztők és én is abban írtam meg a TNote app-ot.

A fejlesztői környezetem az Android Studio volt, 2013-óta ez a hivatalos Android fejlesztői környezet és nem is hagyott cserben. Az Android Studio szoros barátságban van a Kotlinnal, például, ha Java-ban írunk meg kódot akkor az IDE ezt le tudja nekünk fordítani Kotlinra megspórolva sok időt. A beépített emulátor pedig, szinte az összes Android verzióval tud biztosítani emulátort, amin tudjuk tesztelni az alkalmazásunkat.

Viszont a Kotlin sem egy tökéletes nyelv, ezért léteznek különböző úgynevezett dependency-k, amik segítségével meg tudunk spórolni számtalan sor fölösleges kódot. Én ezek közül például a Retrofit2 nevű dependency-t használtam, a HTTP kérések egyszerűbb megvalósítása érdekében.

Az adatbázist szerencsére nem kellett lokálisan kezelni mivel minden a felhőn van tárolva. A felhőn adatot manipuláló funkciókra kitérek később.

Az alkalmazás kinézetét az Adobe Photoshop és Adobe Xd programok segítségével csináltam.

* 1. Az Android fejlesztésről

Mielőtt elkezdeném a fejlesztői dokumentációt leírnám néhány fontos és gyakran használt osztály nevét és feladatát.

Activity -> Egy képernyő egy Android alkalmazásban.

Fragment -> Egy képernyő része, ami kitöltheti a teljes képernyőt is egy Android alkalmazásban.

RecyclerViewAdapter -> Egy kinézet csoport, ami egy RecyclerView-ban megjeleníthető módon jeleníti meg az adapternek leadott kinézeteket.

object -> Ha class helyet object szóval írunk egy új osztályt akkor egy kis motorháztető alatti varázslással a JVM gondoskodik róla, hogy csak egy példány legyen mindig életben ebből az osztályból

* 1. Fejlesztői dokumentáció
     1. Az alkalmazás felépítése

Amikor már nagyobb alkalmazásokat készítünk fontos, hogy egy átlátható és könnyen bővíthető vázat felépítsünk. Ezt én specifikus package-ekkel oldottam meg. Az alkalmazás struktúrája 6 package-ből áll össze.

ABC sorrendben a 6 package:

* adapters
* api
* logic
* models
* ui
* fragments
* util

Az adapters package-ben találhatóak a különböző objektumokhoz tartozó RecyclerView Adapter-ek, amik segítségével meg tudom jeleníteni az objektumokat RecyclerView-ban.

Az api package-ben található a Retrofit2-vel létrehozott kapcsolat a szerverrel és az Api interface a hálózati kérésekkel.

A logic package-ben találhatók a ViewModel osztályok, amik az Activity-nek és Fragment-nek biztosítják az adatokat, és kezelik az Api kéréseket.

A models package-ben találhatóak az osztályok az adatmodellt alkotják, és ahol szükséges egyedi response-t és request-et példányosítanak.

A ui package-ben találhatók a bejelentkezésért és regisztrációért felelős Activity-k,

és az alkalmazás nagy részéért felelős ApplicationActivity ami tárolja a ui/fragments-ben található Fragment-eket.

A ui/fragments package-ben találhatók a jegyzetek, táblázatok és jelenlegi felhasználó megjelenítéséért és szerkesztéséért felelős Fragment-ek.

A util package-ben találhatók azok az osztályok amik az egész alkalmazásban használt adatok eléréséért, és a Fragment-ek állapotáért felelősek.

* + 1. Feladatspecifikáció

Megjelenítés

MainActivity.kt:

Leírás: Az első képernyő, amit az alkalmazás első megnyitásánál látunk

|  |  |
| --- | --- |
| Funkciók | isLoggedIn() -> lefut egy ellenőrzés, és ha már volt bejelentkezve felhasználó aki nem jelentkezett ki, akkor automatikus átkerülünk az ApplicationActivity-re  btnToLogin.setOnClickListener -> átkerülünk a bejelentkező képernyőre  btnToRegister.setOnClickListener -> átkerülünk a regisztráló képernyőre |

LoginActivity.kt:

Leírás: A képernyő, ahol be tudunk jelentkezni

|  |  |
| --- | --- |
| Funkciók | btnBackFromLogin.setOnClickListener -> vissza kerülünk a kezdőképernyőre  btnLogin.setOnClickListener -> meghívja a LoginViewModel login() funkcióját  loginViewModel.loginActivityState.observe() -> megfigyeljük a LoginViewModel loginActivityState nevű változóját és kezeljük az összes állapotát  egyéb a töltő icon-t megjelenítő és eltüntető funkciók |

LoginViewModel.kt

Leírás: A bejelentkezés hálózati kérés végrehajtása és válasz átküldése a LoginActivity-nek

|  |  |
| --- | --- |
| Változók | loginActivityState -> egy mutableLiveData típusú változó ami a login() funkció válaszának tárolásáért felelős |
| Funkciók | login() -> meghívja az ApiInstance api változójának a login() funkcióját |

RegisterActivity.kt:

Leírás: A képernyő ahol tudunk regisztrálni

|  |  |
| --- | --- |
| Funkciók | btnBackFromRegister.setOnClickListener -> visszakerülünk a kezdőképernyőre  btnRegister.setOnClickListener -> meghívja a RegisterViewModel register() funkcióját  viewModel.registerActivityState.observe -> megfigyeljük a RegisterViewModel registerActivityState nevű változóját és kezeljük az összes állapotát  egyéb a töltő icon-t megjelenítő és eltüntető funkciók |

RegisterViewModel.kt:

Leírás: A regisztrálás hálózati kérés végrehajtása és a válasz átküldése a RegisterActivity-nek

|  |  |
| --- | --- |
| Változók | registerActivityState -> egy mutableLiveData típusú változó ami a register() funkció válaszának tárolásáért felelős |
| Funkciók | register() -> meghívja az ApiInstance api változójának a register() funkcióját |

ApplicationActivity.kt:

Leírás:

|  |  |
| --- | --- |
| Változók | viewModel -> ApplicationViewModelFactory() segítségével létrehozunk egy speciális ApplicationViewModel osztályt amelyben paraméterként meg kell adnunk egy SessionManager példányt  navHostFragment -> adunk nevet az xml-ben appNavHostFragment id-jű elemnek  navController -> külön elmentjük a navHostFragment navigáció menedzserét  appBarConfiguration -> létrehozunk egy új navigáció konfiguráció osztályt a létrehozott navigációs gráf és az xml-ben drawerLayout id-jű elemek segítségével |
| Funkciók | navigationView.setupWithNavController() -> a navController segítségével beállítjuk a menü elemeit  setupActionBarWithNavController() -> beállítja a navigáció menedzser viselkedését  onSupportNavigateUp() -> felülírja mi történjen amikor másik menü elemet választunk |

ApplicationViewModel.kt:

Leírás: Az összes jegyzetekkel és órarendekkel kapcsolatos hálózati kérés végrehajtása és ezek állapotának átadása a Fragment osztályoknak

|  |  |
| --- | --- |
| Változók | notesListFragmentState -> egy mutableLiveData típusú változó ami a getNotes() funkció válaszának tárolásáért felelős  timetablesListFragmentState -> egy mutableLiveData típusú változó ami a getTimetables() funkció válaszának tárolásáért felelős  accountFragmentState -> egy mutableLiveData típusú változó ami a getCurrentUser() funkció válaszának tárolásáért felelős  timetableFragmentState -> egy mutableLiveData típusú változó ami a getSelectedTimetable() funkció válaszának tárolásáért felelős  ttElementFragmentState -> egy mutableLiveData típusú változó ami a getSelectedTTElement() funkció válaszának tárolásáért felelős |
| Funkciók | getNotes() -> meghívja az ApiInstance api változójának a getNotes() funkcióját  createNote() -> meghívja az ApiInstance api változójának a newNote() funkcióját  updateNote() -> meghívja az ApiInstance api változójának updateNote() funkcióját  deleteNote() -> meghívja az ApiInstance api változójának a deleteNote() funkcióját  getTimetables() -> meghívja az ApiInstance api változójának a getTimetables() funkcióját  getSelectedTimetable() -> meghívja az ApiInstance api változójának a getSelectedTimetable() funkcióját  createTimetable() -> meghívja az ApiInstance api változójának a createTimetable() funkcióját  deleteTimetable() -> meghívja az ApiInstance api változójának a deleteTimetable() funkcióját  getSelectedTTElement() -> meghívja az ApiInstance api változójának a getSelectedTTElement() funkcióját  createTTElement() -> meghívja az ApiInstance api változójának a createTTElement() funkcióját  updateTTElement() -> meghívja az ApiInstance api változójának a updateTTElement() funkcióját  deleteTTElement() -> meghívja az ApiInstance api változójának a deleteTTElement() funkcióját  getCurrentUser() -> meghívja az ApiInstance api változójának a getUser() funkcióját  logout() -> meghívja az ApiInstance api változójának a logout() funkcióját |

Az összes Fragment osztály:

leírás: Változók és funkciók amik minden Fragment osztályban megtalálhatók

|  |  |
| --- | --- |
| Változók | viewModel -> a törzs-Activity viewModel változójából létrehozunk egy példányt  sessionManager -> egy SessionManager példány |
| Funkciók | a töltő ikont megjelenítő és eltüntető funkciók |

AccountFragment.kt:

Leírás: A felhasználó adatainak megjelenítése és kijelentkezés

|  |  |
| --- | --- |
| Funkciók | btnLogout.setOnClickListener -> kijelentkezés, a telefon belső tárhelyén mentett autentikációs adatokat töröljük és vissza kerülünk a kezdőképernyőre  accountFragmentState.observe() -> megfigyeljük az ApplicationViewModel accountFragmentState nevű változóját és kezeljük az összes állapotát |

NotesListFragment.kt:

Leírás: A felhasználó jegyzeteinek listázása, jegyzet törlése, új jegyzet létrehozása

|  |  |
| --- | --- |
| Változók | notesAdapter -> egy NotesAdapter típusú változó |
| Funkciók | setupRV() -> a RecyclerView alap tulajdonságainak beállítása  delete -> egy Note elem törlése  createDeleteDialog() -> egy ablak létrehozása amivel meg tudjuk erősíteni a törlést  notesAdapter.setOnItemClickListener -> navigáció a NoteFragment–re az elemre kattintott elem adataival  notesAdapter.setOnItemLongClickListener -> elem törlés megerosítő ablak megejelnítése  btnNewNote.setOnClickListener -> navigáció a NoteFragment–re egy üres Note példánnyal  notesListFragmentState.observe -> megfigyeljük az ApplicationViewModel noteFragmentState nevű változóját és kezeljük az összes állapotát |

NoteFragment.kt:

Leírás: A kiválasztott jegyzet szerkesztése és a változtatások mentése

|  |  |
| --- | --- |
| Változók | args -> Navigáció által kapott adatok  note -> Az args-ból kinyert Note példány |
| Funkciók | btnSaveShownNote.setOnClickListener -> meghívja a saveNoteChanges() funkciót  saveNoteChanges() -> a note objektum id-jától függően készít egy új vagy frissít egy meglévő note objektumot |

TimetableListFragment.kt:

Leírás: A felhasználó órarendjeinek listázása, törlése, és új órarend létrehozása

|  |  |
| --- | --- |
| Változók | timetablesAdapter -> egy TimetablesAdapter típusú változó |
| Funkciók | setupRV() -> a RecyclerView alap tulajdonságainak beállítása  getData() -> a TImetable lista lekérése  addNewTimetable() -> új órarend felvétele  delete() -> órarend törlése  createDeleteDialog -> egy ablak létrehozása amivel meg tudjuk erősíteni a törlést  timetablesAdapter.setOnItemClickListener -> navigáció a TimetableFragment-re az elemre kattintott elem id-jával  timetablesAdapter.setOnItemLongClickListener -> törlés megerősítő ablak megjelenítése  timetablesListFragmentState.observe -> megfigyeljük az ApplicationViewModel timetablesListFragmentState nevű változóját és kezeljük az összes állapotát |

TimetableFragment.kt:

Leírás: A kiválasztott órarendben található TTElement típusú órák megjelenítése, órák törlése, új óra létrehozása

|  |  |
| --- | --- |
| Változók | args -> Navigáció által kapott adatok  timetableId -> Az args-ból kinyert lepasszolt órarend id-ja |
| Funkciók | setupRV() -> RecyclerView alap adatainak beállítása  getData() -> A TTElement lista lekérése  delete() -> TTElement törlése  createDeleteDialog() -> törlés megerősítő ablak létrehozása  ttElementsAdapter.setOnItemClickListener -> Navigáció a TTElementFragment-re az elemre kattintott elem id-jával  ttElementsAdapter.setOnItemLongClickListener -> törlés megerősítő ablak megjelenítése  fabNewTTElement.setOnClickListener -> Navigáció a TTElementFragment-re 0 értékű id-val  timetableFragmentState.observe -> megfigyeljük az ApplicationViewModel timetableFragmentState nevű változóját és kezeljük az összes állapotát |

TTElementFragment.kt:

Leírás: A kiválasztott óra összes adatának megjelenítése, szerkesztése és mentése

|  |  |
| --- | --- |
| Változók | args -> Navigáció által kapott adatok  spAdapter -> ArrayAdapter<String> típusú változó  hour -> Integer típusú változó  minute -> Integer típusú változó  timeSetterFlag -> String típusú változó  selectedElementId -> Integer típusú változó |
| Funkciók | setupSpAdapter() -> a napválasztó lista elemeinek és kinézetének beállítása  fillData() -> beállítja a mezők adatait a TTElement alapján  getDefaultTime() -> az hour és a minute változók mostani órára és percre állítása  onTimeSet() -> a timeSetterFlag értéke alapján az óra kezdetének vagy végének beállítása és kiiratása  saveTTElementChanges() -> selectedElementId értéke alapján új TTElement létrehozása vagy meglévő TTElement szerkesztése  getData() -> adatok betöltése  btnSelectStart.setOnClickListener -> timeSetterFlag megfelelő beállítása és időválasztó ablak megnyitása  btnSelectEnd.setOnClickListener -> timeSetterFlag megfelelő beállítása és időválasztó ablak megnyitása  ttElementFragmentState.observe -> megfigyeljük az ApplicationViewModel ttElementFragmentState nevű változóját és kezeljük az összes állapotát |

Hálózati logika

ApiInstance.kt (Object):

Leírás: Az első referenciánál létrehoz egy példányt az osztályból és onnan fogva azt az 1 referenciát használva tudunk API kéréseket küldeni

|  |  |
| --- | --- |
| Változók | api -> A Retrofit osztály Builder() funkciójával tudunk egy építeni egy osztályt ami tud kommunikálni egy API-val. A következő funkciókban tudunk definiálni egy alap URL-t, egy osztályt, ami konvertálja a JSON válaszokat osztályokká, majd végül egy Interface osztályt, amiben definiáltuk hogyan kommunikáljunk az API-val. |

Constants.kt (Object):

Leírás: Az API-val való kommunikációhoz szükséges URL-eket tárolja String változókban

TNoteApi.kt (Interface):

Leírás: A Retrofit-el készített API kommunikációs osztályok működéséhez szükséges egy Interface osztály, amiben definiáljuk hogyan kommunikáljunk az API-val. Ehhez ún. Annotation-okkal kell ellátnunk a funkciókat használunk, hogy a Retrofit felismerje, hogy ezek speciális funkciók. Az Annotation-ben kell megadnunk a cél URL-t is.

A funkciók előtti Annotaion-ok lehetnek:

* @GET()
* @POST()
* @PATCH()
* @DELETE()

Ezek fogják megadni a HTTP kérés típusát. Viszont egy HTTP kérésben sok változó lehet. Ezek közül, amikre nekem szükségem volt az autorizáló token, egy specifikus számra végződő URL-re küldés és a HTTP kérésnek a body-ja (tartalma). Ezeket ugyan úgy Annotation-okkal lehet megadni, viszont a funkció konstruktorába kell beírni.

A három konstruktorban használt Annotation-om volt:

* @Path()
* @Header()
* @Body

Ezek közül a Path() Annotation-el lehet például egy változóként kezelhető id-t az URL-be. A Header() Annotation-el lehet A HTTP kérés fejlécébe megadni adatokat.

A Body Annotation-nel lehet megadni tartalmat a HTTP kérésnek.

|  |  |
| --- | --- |
| Funkciók | getNotes() -> küld egy HTTP kérést amivel vissza kapjuk a felhasználó jegyzeteit  newNote() -> küld egy HTTP kérést amivel létrehoz egy új jegyzetet  updateNote() -> küld egy HTTP kérést amivel frissít egy jegyzetet  deleteNote() -> küld egy HTTP kérést amivel töröl egy jegyzetet |

Segéd osztályok

ApplicationViewModelFactory.kt:

Leírás: Ennek az osztálynak a segítségével létre tudunk hozni egy különleges ApplicationViewModel példányt, ami elfogad egy SessionManager típusú paraméter a konstruktorába.

Resource.kt:

Leírás: Ebben a fájlban 4 osztály található amiből 1 zárt 3 publikus. Az 1 zárt osztály egy ősosztályt biztosít a 3 publikus osztálynak. Ezt a 4 osztályt használjuk arra, hogy egyszerűen kezeljük a HTTP kéréseket. A ViewModel-re végződő osztályokban, ahol megkapjuk a HTTP válaszokat, egy Resource-ból származtatott példányban tároljuk el őket. Így az Activity-k tudni fogják a HTTP kérés állapotát a Resource-ból származtatott osztály alapján.

SessionManager:

Leírás: Az Android SharedPrefences osztály egyszerűbb menedzselése miatt van rá szükség.

NotesAdapter :

Leírás: Egy Note osztályokat tároló listából egy RecyclerView-ban megjeleníthető listát csinál

TimetablesAdapter :

Leírás: Egy Timetable osztályokat tároló listából egy RecyclerView-ban megjeleníthető listát csinál

TTElementsAdapter :

Leírás: Egy TTElement osztályokat tároló listából egy RecyclerView-ban megjeleníthető listát csinál

* 1. Felhasználói dokumentáció

A következő rész a felhasználóknak szól. Néhány teszteset-ben bemutatom az alkalmazás helyes működését és megnézünk pár hibás tesztesetet is.

Az alkalmazás egy virtuális Pixel 4 típusú eszközön lett tesztelve.

* + 1. Alkalmazás megnyitása

Bejelentkezés

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * Létező fiók | 1. Login gombra kattintás 2. Helyes felhasználó adatok megadása 3. (Új) Login gombra kattintás | * Egy lista a létező jegyzetekről |
|  | | |

Regisztráció

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat | 1. Register gombra kattintás 2. Helyes regisztrációs adatok megadása 3. (Új) Register gombra kattintás | * Egy üres lista a jegyzetekről |
|  | | |

Kijelentkezés

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók | 1. A bal felső menüre kattintás 2. Az Account (Fiók) opció kiválasztása 3. Logout gombra kattintás | * A kezdő képernyő |
|  | | |

* + 1. Jegyzetek

Jegyzetek megjelenítése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók | 1. Bejelentkezés vagy regisztráció után ez az első, amit látunk | * A létrehozott jegyzetek egy listában |
|  | | |

Új jegyzet

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók | 1. A jegyzet lista képernyőn a bal alsó plusz gombra kattintás 2. Cím (Title) megadása kötelező 3. A bal alsó mentés gombra kattintás | * Egy felugró üzenet „Note saved” felirattal * Az új jegyzet megjelenik a listában |
|  | | |

Jegyzet Szerkesztése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * Létező jegyzet | 1. A szerkeszteni kívánt jegyzetre kattintás 2. Változtatni kívánt adatok megadása 3. A bal alsó mentés gombra kattintás | * Egy felugró üzenet „Note saved” felirattal * Az szerkesztett jegyzetre kattintva az új adatokat látjuk |
|  | | |

Jegyzet törlése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * Létező jegyzet | 1. A szerkeszteni kívánt jegyzetre hosszú kattintás 2. A felugró ablakban Igenre (Yes) kattintás | * Egy felugró ablak „Deleted” felirattal * A törölt jegyzet eltűnik a listából |
|  | | |

* + 1. Órarendek

Órarendek megjelenítése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók | 1. A bal felső menüre kattintás 2. A Timetables (Órarendek) opció kiválasztása | * A létrehozott órarendek egy listában |
|  | | |

Új órarend

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók | 1. Az órarend lista képernyőn, a fenti szöveg bemeneti mezőbe az új órarend nevének beírása 2. Az Add (Hozzáadás) gombra kattintás | * Megjelenik a listában az új órarend |
|  | | |

Órarend törlése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * létező órarend | 1. A törölni kívánt órarendre hosszú kattintás 2. A felugró ablakban Igenre (Yes) kattintás | * Egy felugró ablak „Deleted” felirattal * Eltűnik a listából a törölt órarend |
|  | | |

* + 1. Órák

Órák megtekintése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * létező órarend | 1. A megtekintendő órarendre kattintás | * Megjelenik egy lista az órarendben található órákról |
|  | | |

Új óra

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * létező órarend | 1. Az órarend képernyőn a jobb alsó plusz gombra kattintás 2. Kívánt adatok megadása 3. A jobb alsó mentés gombra kattintás | * Egy felugró üzenet „Event saved” felirattal * Az szerkesztett jegyzetre kattintva az új adatokat látjuk |
|  | | |

Óra szerkesztése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * létező órarend * létező óra | 1. A szerkeszteni kívánt órára kattintás 2. Változtatni kívánt megadása 3. A jobb alsó mentés gombra kattintás | * Egy felugró üzenet „Event saved” felirattal * Az szerkesztett jegyzetre kattintva az új adatokat látjuk |
|  | | |

Óra törlése

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * létező órarend * létező óra | 1. A törölni kívánt órára hosszú kattintás 2. A felugró ablakban Igenre (Yes) kattintás | * Egy felugró ablak „Deleted” felirattal * Eltűnik a listából a törölt óra |
|  | | |

* + 1. Hibás tesztesetek

Bejelentkezés rossz adatokkal

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat | 1. Login gombra kattintás 2. Helytelen felhasználói adatok megadása 3. (Új) Login gombra kattintás | * Egy felugró ablak az elkövetett hibának feliratával |
|  | | |

Jegyzet cím nélkül

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók | 1. A jegyzet lista képernyőn a bal alsó plusz gombra kattintás 2. A bal alsó mentés gombra kattintás | * Egy felugró ablak „Title required” felirattal |
|  | | |

Órarend cím nélkül

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók | 1. Az órarend lista képernyőn, a fenti szöveg bemeneti mező mellett az Add gombra kattintás | * Egy felugró ablak „Name required” felirattal |
|  | | |

Óra cím nélkül

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * Létező órarend | 1. Az órarend képernyőn a jobb alsó plusz gombra kattintás 2. Kívánt adatok megadása (Title mező kihagyása) 3. A jobb alsó mentés gombra kattintás | * Egy felugró ablak „Title required” felirattal |
|  | | |

Elemek többszöri mentése létrehozáskor

|  |  |  |
| --- | --- | --- |
| Szükséges környezet | Lépések | Elvárt eredmény |
| * Internet kapcsolat * A bejelentkezés vagy regisztrálás végrehajtása * Létező fiók * Létező órarend | 1. Bármelyik új elem létrehozásának lépéseinek követése 2. A létrehozás végén kattintsunk a mentés gombra több mint egyszer | * Ahányszor rákattintottunk a mentés gombra annyi elemet hozott létre |
|  | | |

1. Asztali alkalmazás
   1. Bevezetés

A projektmunka kiosztásakor még nem volt semmi ötletem, hogy mobilt, webes alkalmazást vagy az asztali alkalmazást válasszam és mivel nem volt semmi ellenvetésem egyikkel sem ezért arra a döntésre jutottam, hogy majd dolgozok azzal, amit a csapattársaim meghagynak. Így kaptam meg az asztali alkalmazást.

Középiskolában a C# nyelvvel ismerkedtem a tanórákon ezért az első ötletem platformmal kapcsolatban a WPF (Windows Presentation Foundation) volt, de egy kis utánanézéssel rájöttem, hogy valószínűleg jobban járok a JavaFX-szel. Az első fontos pont volt, hogy a projektmunka készítése alatt Javát tanultunk az iskolában, ami miatt nem kellett pluszban egy programozási nyelvet tanulnom vagy legalábbis felfrissítenem. A másik pont a különböző fórumokon talált válaszok voltak, amik elmagyarázták, hogy Javára sokkal több függőség (ún. dependency) és könyvtár (ún. library) található, amik nagyban megkönnyítik a program készítését.

* 1. Java és a JavaFX

Java az egyik legnépszerűbb és legelterjedtebb programozási nyelv a világon, megközelítőleg 10 millió fejlesztővel és 13 milliárd Javát futtató eszközzel.

Sikerességét annak is köszönheti, hogy rengeteg operációs rendszeren elfut, ami nagyban megkönnyíti a fejlesztők munkáját. Ezek mellett előnyei közé sorolható az egyszerűsége más nyelvekhez hasonlítva, az objektum orientált programozás lehetősége és műveletek több szálon való futtatásának lehetősége. Igaz más nyelvekhez képest több memóriát használ és egy picit lassabban, de összességében egy nagyon hatékony és kedvelt nyelv.

JavaFX az egyik legújabb Javára épülő GUI keretrendszer a sok közül. Jelenleg ez bizonyul az egyik legkönnyebbnek és legstabilabbnak. Igaz nincs annyi GUI eleme, mint az egyik régebbi Java GUI keretrendszernek, a swingnek és nem tűnnek ezek az elemek olyan modernnek, mint mondjuk a .NET alapú WPF-ben de a feladatok jelentős részét meg lehet oldani relatív könnyen.

* 1. Fejlesztői dokumentáció
     1. Fejlesztői eszközök
* Nyelv: Java
* Keretrendszer: JavaFX
* Fejlesztői környezet: IntelliJ IDEA 2021.2.2
* Függőségek (dependency):
  + [Simple-JSON](https://github.com/fangyidong/json-simple)
    - JSON fájlok olvasása és kezelése

Fejlesztés alatt a **Java 17 SDK**-t használtam.

* + 1. Feladatspecifikáció

Controller

Leírás: Ebből az osztályból származik a többi Controller (Controller szót a nevében tartalmazó osztály). Közös funkciók leegyszerűsítésére szolgál.

MainController

Leírás: A main-view.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Feladata | Az alkalmazás fő ablaka, ezen belülre töltődik be a többi fxml. A program indításakor vagy betölti a regisztrációs / bejelentkezős ablakot vagy ha a felhasználó nem jelentkezett ki akkor betölti a dashboard-ot. |

DashboardController

Leírás: A dashboard-view.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Fontos funkciók | Az alkalmazás különböző ablakai közötti váltásra nyújt lehetőséget, ami az ablak bal oldalán lévő gombokkal valósítható meg. Az ablak rendelkezik még egy görgethető tárolóval, ahova a többi ablak tölthető be (pl.: jegyzet szerkesztő). |
| Fontos funkciók | * logoutBtnClick() -> kijelentkezés * noteBtnClick() -> jegyzetek listáját jeleníti meg * profileBtnClick() -> a bejelentkezett felhasználó profiljához kapcsolódó információkat jeleníti meg * settingsBtnClick() -> az alkalmazáshoz kapcsolódó beállításokat jeleníti meg |

ReglogController

Leírás: A reglog-view.fxml fájlhoz tartozó controller. A bejelentkezés és regisztráció grafikus részét kezeli.

|  |  |
| --- | --- |
| Feladata | A regisztrációhoz és bejelentkezéshez kéri be és szűri meg az adatokat (pl.: név, jelszó). Ezek mellett a GUI-t kezeli. |
| Fontos funkciók | * submitBtnClick() -> elindítja a bejelentkezést vagy regisztrációt * checkInputs() -> ellenőrzi, hogy a bemeneti mezőkbe beírt adatok megfelelnek-e * emailFormatCheck() -> ellenőrzi, hogy az email az megfelelő formátumú-e |

NoteListController

Leírás: A note-list-view.fxml fájlhoz tartozó controller. A jegyzetek kilistázásáért felelős.

|  |  |
| --- | --- |
| Feladata | Elindíja a jegyzetek lekérését az adatbázisból majd a kapott jegyzeteket megjeleníti egy gomb formájában. |
| Fontos funkciók | * loadNotes() ->elindítja a jegyzetek lekérését majd a kapott jegyzetekből gombokat készít és a képernyőre helyezi |

NoteEditorController

Leírás: A note-editor-view.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Feladata | Betölti a kiválasztott jegyzetet egy szerkesztőbe valamint gombokat kínál a jegyzet törlésére és mentésére |
| Fontos funkciók | * saveBtnClick() -> vagy menti a szerkesztett jegyzetet vagy elmenti az újonnan létrehozott jegyzetet * passData() -> a NoteListControllerből ezen keresztül érkezik a kiválasztott jegyzet |

ProfileController

Leírás: A profile-view.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Feladata | Az éppen bejelentkezett felhasználónak írja ki a nevét, az email címét és a regisztrációjának időpontját |

SettingsController

Leírás: A settings-view.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Feladata | A beállításokat tartalmazza ami jelenleg csak az alkalmazásból való kilépéskor való kijelentkezés állításából áll |

TimetableController

Leírás: A timetable-view.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Feladata | Megjeleníti a felhasználó órarendjét |
| Fontos funkciók | * loadTimetables() -> elindítja a órarend elemek lekérését valamint miután megkapja, betölti azokat a képernyőre |

TTElementEditorController

Leírás: A ttelement-editor.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Feladata | Az órarend elemek szerkesztésére vagy létrehozására szolgál |
| Fontos funkciók | * loadDataToInputs() -> ha szerkesztünk egy órarend elemet akkor annak az adatai betölti a megfelelő mezőkbe * saveBtnClicked() -> elindítja az órarend eleme mentését * deleteBtnClicked() -> elindítja az órarend elem törlését * titleTyped() -> ellenőrzi a megadott cím helyességét és ha az nem megfelelő azt jelzi a felhasználónak * startTyped() -> ellenőrzi a megadott kezdő időpont helyességét és ha az nem megfelelő azt jelzi a felhasználónak * endTyped() -> ellenőrzi a megadott vég időpont helyességét és ha az nem megfelelő azt jelzi a felhasználónak * comboboxAction() -> a nap megadására szolgáló lenyíló menüben megadott adat helyességét ellenőrzi és ha az nem megfelelő azt jelzi a felhasználónak |

TestViewController

Leírás: A test-view.fxml fájlhoz tartozó controller.

|  |  |
| --- | --- |
| Feladata | Tesztelés szempontjából lett létrehozva. Ez egy üres oldal, amit új elemek fejlesztése alatt használtam. Úgy érhetjük el, ha az alkalmazás indításakor az indítási argumentumok első helyére megadjuk a „--test” szöveget. Ez az egyetlen módja, hogy elérjük ezt a lapot ezért a kész alkalmazásban nincsen gomb vagy parancs amivel ezt elérhetjük. |

Api

Leírás: A backend szerverrel való kommunikáláshoz kapcsolódó osztály. Minden tagja statikus ezért nem kell példányosítani.

|  |  |
| --- | --- |
| Feladata | Tárolja az összes parancsot és alap változót (pl.: api szerver címe) ami a backend szerverrel való kommunikációra szolgál. |
| Fontos funkciók | * register() -> megpróbál létrehozni egy új felhasználót * login() -> megpróbál bejelentkezni a megadott adatokkal * logout() -> megpróbálja kijelentkeztetni a felhasználót * processReceivedAuthData() -> a regisztráció és bejelentkezés után ez a parancs állítja be az aktuális felhasználót a User osztályban * getNotes() -> visszaadja a felhasználó összes jegyzetét egy Note típusú listában * addNote() -> megpróbál létrehozni egy új jegyzetet * saveNote() -> megpróbál elmenteni egy már létező de módosított jegyzetet * deleleteNote -> megpróbálja törölni a megadott azonosítójú jegyzetet * getTimetableElements() -> visszaadja a felhasználó összes órarend elemét * addTTElement() -> megpróbál létrehozni egy új órarend elemet * saveTTElement() -> megpróbál elmenteni egy már létező de módosított órarend elemet * deleteTTElement() -> megpróbálja törölni a megadott azonosítójú órarend elemet * checkStatuseCode() -> a megadott kapcsolaton megnézi a kapcsolat státusz kódját és ha az nem 2-essel kezdődik akkor dob egy új Exceptiont aminek a tartalma a kapott hibaüzenet * getResponse() -> megpróbálja visszaadni a megadott kapcsolat válaszát, elég kezdetleges és ha már korábban olvastunk be adatot a kapcsolatból akkor Exception-t dob * sendPostData() -> elküldi a megadott kapcsolatra a megadott adatot JSON formátumban POST metódussal * sendPatchData() -> elküldi a megadott kapcsolatra a megadott adatot JSON formátumban PATCH metódussal * sendDefaultData() -> az e fölött lévő kettő funkciónak készíti fel a kapcsolatot * defaultConn() -> létrehoz egy nagyon alapszintű kapcsolatot, magában nincs használva, csak segéd funkció a többi kapcsolat felépítéséhez * getConn() -> vissza tér egy GET metódusú kapcsolattal * postConn() -> vissza tér egy POST metódusú kapcsolattal * patchConn() -> vissza tér egy PATCH metódusú kapcsolattal * deleteConn() -> vissza tér egyDELETE metódusú kapcsolattal * setBearer() -> a megadott kapcsolathoz hozzáfűzi a megadott tokent * readJson() -> a megadott kapcsolatból kiolvassa és visszaadja szövegként a választ * getJSONArrayFromString() -> elkészít és vissza ad egy JSON tömböt a megadott szövegből * getJSONArray() -> elkészít és vissza ad egy JSON tömböt a megadott kapcsolatból * getJSONObjectFromString() -> elkészít és vissza ad egy JSON objektumot a megadott szövegből * getJSONObject() -> elkészít és vissza ad egy JSON objektumot a megadott kapcsolatból |

WindowManager

Leírás: Az ablakok közötti váltások és egyéb az ablak kezeléséhez tartozó funkciók osztálya.

|  |  |
| --- | --- |
| Feladata | Az ablakok váltására szolgáló osztály |
| Fontos funkciók | * setInnerScrollpane() -> ezzel a funkcióval lehet megadni az osztálynak hogy melyik görgethető tárolóba rakja be a későbbiekben a megfelelő elemeket * getRootContainer() -> visszaadja a legkülső tárolót amibe elemeket tölt be * changeToRegLog() -> átvált a regisztráció / bejelentkezés fülre * changeToDashboard -> átvált az alkalmazás kezdőoldalára * changeToNoteList() -> betölti a görgethető listába a  note-list-view.fxml-t * changeToNoteEditor() -> betölti a görgethető listába a  note-editor-view.fxml-t * changeToTimetable() -> betölti a görgethető listába a  timetable-view.fxml-t * changeToFxml() -> betölti a megadott elérési útvonalú fxml-t * setScrollPaneScrollSpeed() -> átállítja a görgethető tároló görgetési sebességét * scrollDown() -> legörget a görgethető tároló legaljára * scrollUp() -> felgörget a görgethető tároló legtetejére * checkInnerScrollPane() -> vissza ad igaz vagy hamis értéket attól függően, hogy be lett-e állítva már a görgethető lista |

Popup

Leírás: Az ablakon felugró elemek megjelenítésére szolgál

|  |  |
| --- | --- |
| Feladata | Felugró elemek megjelenítéséről és animálásáról gondoskodik |
| Fontos funkciók | * getContentContainer() -> visszaadja a felugró ablak fő tárolóját * isCloseOnWindoClick() -> vissza adja, hogy a felugró ablak becsukódik-e * show() -> megjeleníti a felugró ablakot * hide() -> elrejti a felugró ablakot * hideAll() -> elrejti az összes megjelenített felugró ablakot * blink() -> egy animációt játszik le a felugró ablakon * setText() -> beállítja a felugró ablak címét * setColor() -> beállítja a felugró ablak színét * setTextColor() -> beállítja a felugró ablak szövegének a színét * withBlur() -> beállítja, hogy a háttér legyen homályos amig a felugró ablak meg van jelenítve * addBody() -> a megadott VBox-ot (tárolót) hozzáfűzi a felugró ablak aljához * setCloseTimer() -> milliszekundumban megadott idő után automatikusan bezárja a felugró ablakot * withFadeInAndOut() -> beállítja, hogy átmenetesen jelenjen meg és tűnjön el a felugró ablak * setFadetransitionTime() -> beállítja, hogy meddig tartson a megjelenítési és eltüntetési animáció * setDistanceFromTop() -> beállítja, hogy hány pixel távolságra legyen az ablak tetejétől * placeToCenter() -> a felugró ablak nem felül hanem az ablak közepén jelenik meg |

Leírás: A bejelentkezett felhasználó adatait tárolja. Minden tagja statikus ezért nem kell példányosítani.

Note

Leírás: Jegyzetek tárolására szolgáló osztály.

|  |  |
| --- | --- |
| Feladata | A jegyzetek tárolására és a jegyzet különböző adatainak beállítására valamint lekérésére szolgál |

User

Leírás: az aktuális felhasználó adati

|  |  |
| --- | --- |
| Feladata | A bejelentkezett felhasználó adatainak tárolására szolgál |

* 1. Felhasználói dokumentáció

Az alkalmazás alapvető működését bemutató tesztesetek leírása

* + 1. Tesztesetek

Regisztrálás

|  |  |  |
| --- | --- | --- |
| Szükséges körülmények | Lépések | Elvárt eredmény |
| - internetkapcsolat | 1. A középső panel tetején a regisztráció kiválasztása  2. Helyes adatok megadása  3. Regisztráció gomb megnyomása | Bedob az alkalmazás főoldalára ami a jegyzetek listázása ;s fel-l megjelenik egy sikerek regisztráció felirat |
|  | | |

Bejelentkezés

|  |  |  |
| --- | --- | --- |
| Szükséges körülmények | Lépések | Elvárt eredmény |
| - internetkapcsolat | 1. A középső panel tetején a bejelentkezés kiválasztása  2. Helyes adatok megadása  3. Bejelentkezés gomb megnyomása | Bedob az alkalmazás főoldalára ami a jegyzetek listázása |
|  | | |

Kijelentkezés

|  |  |  |
| --- | --- | --- |
| Szükséges körülmények | Lépések | Elvárt eredmény |
| - internetkapcsolat  - felhasználó be van jelentkezve | 1. az ablak bal alsó sarkán rányomunk a kijelentkezés gombra | Bedob regisztrációs és bejelentkezéses oldalra |
|  | | |

Új jegyzet felvétele

|  |  |  |
| --- | --- | --- |
| Szükséges körülmények | Lépések | Elvárt eredmény |
| - internetkapcsolat  - felhasználó be van jelentkezve | 1. megnyomjuk a jegyzetek feliratú gombot  2. Rányomunk az új jegyzet feliratú gombra  3. Megadunk egy címet  4. Rányomunk a mentés gombra | Az ablak tetején megjelenik egy sikeres mentés feliratú szöveg |
|  | | |

Meglévő jegyzet szerkesztése

|  |  |  |
| --- | --- | --- |
| Szükséges körülmények | Lépések | Elvárt eredmény |
| - internetkapcsolat  - felhasználó be van jelentkezve | 1. megnyomjuk a jegyzetek feliratú gombot  2. Rányomunk az általunk választott jegyzetre  3. Szerkesztjük a jegyzetet  4. Rányomunk a mentés gombra | Az ablak tetején megjelenik egy sikeres mentés feliratú szöveg |
|  | | |

Web

Irodalomjegyzék

Ábrajegyzék