



TANSZÉKVEZETŐ

SZAKDOLGOZAT FELADAT

Pelles Dorottya

Mérnökinformatikus hallgató részére

Android kliens fejlesztése iskolai e-napló rendszerhez

Az iskolákban a tanulók eredményeinek, és az egyéb oktatással kapcsolatos komponensek adminisztrálására a korábbi évtizedekben papír alapú naplót használtak. Az emberi tényezőtől fakadóan így könnyen hiba csúszhatott a rendszerbe, ráadásul a naplók kézzel való vezetése igen nehézkes és időigényes volt. Mivel a naplóból csak egy példány létezett, emiatt a tanulóknak külön kellett vezetniük a jegyeiket, hogy a szüleik ellenőrizhessék előrehaladásukat.

A modern e-napló rendszereknek köszönhetően az oktatással kapcsolatos adminisztrációs teendők mára jelentősen leegyszerűsödtek, felgyorsultak. Az okostelefonok elterjedése óta a felhasználók túlnyomó többsége az e-naplót is telefonról szeretné elérni. Ennek megfelelően a legtöbb népszerű e-napló implementáció rendelkezik okostelefonos frontenddel, azonban ezek használata általában nehézkes, funkcionalitásuk töredékes.

A hallgató feladata egy már meglévő e-napló rendszer adatbázis sémáját felhasználva tanulói és oktatói e-napló mobil applikációk készítése, és ezek számára API végpontok nyújtása. A tervezett alkalmazás célja, hogy lehetőséget biztosítson a legegyszerűbb tanári adminisztratív feladatok elvégzésére. A diákok és szüleik pedig ellenőrizhetik az adott tanuló előrehaladását, illetve például az iskolai menza adott napi menüjét is.

A hallgató feladatának a következőkre kell kiterjednie:

- Mutassa be az e-napló rendszerek tipikus képességeit!
- Mutassa be a meglévő rendszer felépítését és szolgáltatásait!
- Tervezzon backend réteget a szolgáltatások REST API végpontokon történő kiajánlására!
- Tervezzon Android alapú kliens alkalmazásokat a funkciók részhalmazának mobil eszközről való elérésére!
- Készítse el a tervezett rendszerek prototípusait!

Tanszéki konzulens: Dr. Forstner Bertalan Ph. D., egyetemi docens

Budapest, 2022. október 06.

Dr. Charaf Hassan
egyetemi tanár
tanszékvezető





M Ű E G Y E T E M 1 7 8 2

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Pelles Dorottya

ANDROID KLIENS FEJLESZTÉSE ISKOLAI E-NAPLÓ RENDSZERHEZ

KONZULENS

Dr. Forstner Bertalan

BUDAPEST, 2022

Tartalomjegyzék

Összefoglaló	5
Abstract.....	6
1 Bevezetés	7
1.1 A téma eredete	7
1.2 A piacon elérhető e-napló rendszerek tipikus képességei	7
1.3 A meglévő rendszer felépítése és szolgáltatásai	8
1.4 A feladatkiírás értelmezése	9
1.5 A dolgozat felépítése	10
2 A feladat specifikálása	11
2.1 Kliensoldal	11
2.1.1 Tanulói kliens	11
2.1.2 Tanári kliens	12
2.1.3 Az alkalmazások képernyőtervei	14
2.2 Szerveroldal	19
3 Felhasznált technológiák	20
3.1 Szerveroldalon használt technológiák	20
3.1.1 ASP.NET 6	20
3.1.2 Entity framework és az adatbázis irányú megközelítés	20
3.1.3 JSON Web Token Authentikáció	21
3.1.4 SQL adatbázis	22
3.1.5 Swagger	23
3.2 Kliensoldalon használt technológiák	24
3.2.1 Android platform és a Kotlin nyelv	24
3.2.2 Retrofit 2	24
3.2.3 Android Preferences	25
3.3 Azure.....	26
4 A rendszer megtervezése	27
4.1 A rendszerszintű architektúra	28
4.2 A kliensek architektúrája	28
4.2.1 MVVM tervezési minta	29
4.3 A szerver architektúrája	29

4.3.1 MVC és Repository tervezési minták	29
4.3.2 Az adatbázis sémája	30
4.4 Kliens és szerver közötti kommunikáció	30
4.4.1 Tanulók számára nyújtott végpontok.....	31
4.4.2 Tanárok számára nyújtott végpontok.....	32
4.4.3 Egyéb végpontok	36
5 A megvalósítás érdekesebb részletei	38
5.1 Néhány felmerült probléma	38
5.1.1 Megárvult SQL felhasználók	38
5.1.2 SQL adatbázis adatainak frissítése	38
5.1.3 Hiba a RecyclerViewban felvett listenerekkel	39
5.1.4 SingleLiveEvent a LiveEvent helyett	39
5.1.5 Azure okozta teljesítménycsökkenés	40
5.2 Képek a kész alkalmazásokról	41
5.3 Tesztelés.....	52
5.4 Felhasználói élmény	52
6 Összegzés.....	53
5.5 Értékelés.....	53
5.6 Továbbfejlesztési lehetőségek	54
5.6.1 Bővítési lehetőségek	54
5.6.2 Kiterjesztés más platformokra	54
5.6.3 További alkalmazási területek	54
Irodalomjegyzék.....	56

HALLGATÓI NYILATKOZAT

Alulírott **Pelles Dorottya**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2022. 12. 07.

.....
Pelles Dorottya

Összefoglaló

Az elmúlt években a digitalizáció terjedésével életünk egyre több aspektusát tették egyszerűbbé és kényelmesebbé a technológia vívmányai. A középiskolákban a korábbi évtizedekben papír alapú naplókat használtak nemcsak a tanulók eredményeinek, de az egyéb oktatással kapcsolatos események rögzítésére is. A fejlődés előbb-utóbb elérte az iskolai adminisztrációt is, és bevezetésre kerültek az elektronikus napló rendszerek.

Az okostelefonok széleskörű elterjedését követően a mobiltelefon felhasználók számára természetessé vált, hogy a mindennapjaikban használt webes felületekhez kényelmes és felhasználóbarát telefonos frontendek is készülnek. Ez alól az e-napló rendszerek mobilos elérhetősége sem kivétel, így a ma használatos legnépszerűbb e-napló implementációkhoz mobiltelefonos applikációk is a rendelkezésünkre állnak. Probléma viszont, hogy ezek zömmel nem túl ergonomikusak, és csak az elérhető funkciók töredékét valósítják meg.

A szakdolgozat célja a Ciszterci Rend Nagy Lajos Gimnáziuma és Kollégiumának használatban lévő, de telefonos applikációval nem rendelkező e-napló rendszerének adatbázis sémáját felhasználva tanulói és tanári Android nyelvű mobil kliensek tervezése és készítése, valamint ezek számára API végpontok nyújtása. Az Android applikációk a már rendelkezésre álló webes felület releváns lehetőségeit megvalósítják, és ki is terjesztik.

A dolgozat bemutatja a jelenlegi e-napló rendszert, vázolja az elkészített kliensek elvárt működését, kitér a rendszer egyes komponenseinek megtervezésére, körüljárja a használt technológiákat, és bemutatja a megvalósítás néhány érdekesebb részletét is.

Abstract

In recent years, with the spread of digitalization, achievements of technology have made more and more aspects of our lives simpler and more convenient. In the previous decades, high schools used paper-based diaries not only to record students' results, but also to record other events related to education. Sooner or later, development reached the school administration, and electronic diary systems were introduced.

After the spread of smartphones, it became natural for mobile phone users to create convenient and user-friendly phone frontends for the web interfaces they use on a daily basis. The mobile availability of e-diary systems is no exception to this. So we also have mobile phone applications for the most popular e-diary implementations in use today. The problem, however, is that they are generally not very ergonomic and implement only some of the functions available.

The aim of the thesis is to create and design Android-language mobile clients for students and teachers using the database scheme of the e-diary system of Ciszterci Rend Nagy Lajos Gimnáziuma és Kollégiuma, which is in use but does not have a phone application, and to provide API endpoints for them. Android applications implement and extend the relevant options of the already available web interface.

The thesis presents the current e-diary system, outlines the expected functionality of the created clients, goes into the design of individual components of the system, goes around the technologies used, and goes into some of the more interesting details of the implementation.

1 Bevezetés

1.1 A téma eredete

Szakedolgozatom témájául tanárok és diákok számára elérhető Android applikációk fejlesztését választottam egy középiskola létező e-napló rendszeréhez, amelyhez a szükséges backend komponenseket is megírtam. A választott középiskola az egykori gimnáziumom, ahova 6 éven keresztül jártam. Mivel ez egy egyházi fenntartású intézmény, így számukra nincsen kötelezően előírt e-napló. Az iskolába 2012-ben vezettek be egy csak webes klienssel rendelkező naplót, ami ugyan teljes mértékben funkcionál, de designja már akkoriban is idejétmúltnak számított.

Középiskolai tanulmányaim során jutott odáig a technológia, hogy egy idő után már nem volt kézzel fogható naplója sem az osztályunknak, hanem minden kizárólagosan az e-naplóban került rögzítésre. Idővel nemcsak szinte minden diáknak volt telefonja, de tanáraimat is egyre többször láttam, hogy telefonról próbálják adminisztrálni a tanórával kapcsolatos eseményeket, mint például egy órai felelet eredményét, vagy a hiányzókat/későket, ami a webes klienssel elég nehéz, és kényelmetlen volt.

Már középiskolásként is nagyon érdekelt az informatika, így az összes ezzel kapcsolatos szakkör lelkes résztvevője voltam. Akkori informatikatanárommal, Bérczi László Bernát atyával, – aki azóta az iskola fenntartója – azóta is jó viszonyt ápolok. Vele közösen jutottunk arra a felismerésre, hogy milyen hasznos lenne mobil applikációk lefejlesztése az e-naplóhoz, innentől pedig adta magát a gondolat, hogy egyúttal legyen ez a szakedolgozatom témája is. Örömmel vállaltam a feladatot annak reményében, hogy visszaadhatok valamit egykori tanáraimnak, alma materemnek azért a fáradhatatlan, elkötelezett munkáért, amellyel a mi nevelésünket és oktatásunkat végezték, és örültem, hogy a szakedolgozatomnak lesz valami tényleges célja.

1.2 A piacon elérhető e-napló rendszerek tipikus képességei

A piacon az utóbbi években több e-napló rendszer is jelen volt, azonban ezt néhány évvel ezelőtt egységesítették, és így jelenleg az állami iskolák számára a Kréta nevű e-napló rendszer érhető el. Ez a rendszer az Android mellett az iOS és Huawei telefonokkal is kompatibilis[1].

A Krétával, és az ehhez hasonló rendszerekkel a tanárok elvégezhetik a legtöbb oktatással kapcsolatos adminisztrációt, mint például jegyek beírása, tanóra naplózása, hiányzók és késők rögzítése, órarend megtekintése, házi feladat felvétele. Az ehhez készült mobil applikációkkal pedig az e-napló rendszerek fontosabb funkcióit lehet érni, ami kiegészül a telefonos értesítések küldésével is.

Az iskola jelenleg nem tervez átállni a népszerűbb e-napló rendszerekre, ami miatt nem tudja használni ezeket a fejlettebb mobil klienseket. Ennek előnye viszont, hogy kisebb célpont lévén talán kevésbé vannak kitéve az adatlopási kísérleteknek, ami egy nagyobb rendszerrel nemrégiben elő is fordult¹.

1.3 A meglévő rendszer felépítése és szolgáltatásai

Az iskolának jelenleg is van egy működő e-napló rendszere, amelyhez a tanárok és tanulók egy-egy webes kliens segítségével tudnak csatlakozni. A kliens valamilyen módon hozzá van kötve az adatbázishoz, azonban a rendszer fejlesztőivel nem volt lehetőségem beszélni, így erről nem kaptam bővebb információt.

A webes klienssel a diákok megtekinthetik üzeneteiket, késéseiket és hiányzásaikat, figyelmeztetéseiket és elmarasztalásaikat, valamint félévre lebontva jegyeiket, amiket tárgyaként részletezhetnek is.

A tanárok többek között küldhetnek egymásnak és diákjaiknak üzeneteket, naplózhatnak hiányzásokat és késéseket, rögzíthetnek dicséreteket és figyelmeztetéseket, beírhatnak jegyeket, naplózhatnak tanórát, vagy felvehetnek tervezett dolgozatokat. Amennyiben osztályfőnökök, akkor osztályuk tagjainak rögzíthetnek egyszeri vagy időszakos igazolásokat.

¹ Telex: A fejlesztő cég megpróbálta elhallgatni a KRÉTA feltörését (hozzáférés ideje 2022.11.23): <https://telex.hu/tech/2022/11/09/kreta-rendszer-ekreta-zrt-adathalasz-tamadas-adatszivargas-elhallgatasa-naih-vizsgalat-eljaras>

1.4 A feladatkiírás értelmezése

A feladat a tervezettnél jóval nagyobb munkának bizonyult, hiszen azt hittem, hogy kiindulásnak kapni fogok egy jól dokumentált API-t, és elég lesz csak az ezzel kommunikáló Android alkalmazásokat elkészítenem. Ehhez képest a kiinduláskor egy dokumentálatlan, néhol félrevezető táblaneveket tartalmazó, de több száz táblából álló, rendkívül összetett, minta adatokat tartalmazó adatbázis állt a rendelkezésemre. Ezt kibogozni a későbbiekben elég sok fejtörés/időbefektetés árán sikerült csak.

Szükségem volt egy backendre, ami a kliensek számára REST API-t biztosít. Ezt némi utánajárás után végül ASP.NET Core technológiával és Entity Framework felhasználásával készítettem, a lekérdezésekhez pedig Linq nyelvet használtam. Korábban sem a témalaboratóriumom, sem önálló laboratórium során nem foglalkoztam ezzel a technológiával, azonban volt néhány ezzel kapcsolatos tanóránk, így egy minimális ismerettel már rendelkeztem róla. Az autentikációt és autorizációt pedig JSON Web Token segítségével oldottam meg, mert ez az utóbbi évek egyik legnépszerűbb felhasználó hitelesítésére használt technológiája.

Frontendnek valamilyen mobil applikációt kellett készítenem, hiszen ez a projekt legfőbb célja. Mivel a kész rendszert üzembe szeretném helyezni az iskolában, így a használhatóság érdekében rengeteg funkciót kellett készítenem. Ennek következtében a projekt nagyon nagyra sikeredett, így a szakdolgozat keretein belül egyelőre csak az egyik típusú kliensek elkészítését tűztem ki célomul. Mivel az iskola tanárainak és diákjainak nagyrésze Android felhasználó, engem pedig mindig érdekelt az Android fejlesztés (habár idáig csak minimális ismeretem volt róla), és a Kotlin nyelv is, így végül emellett döntöttem.

1.5 A dolgozat felépítése

A szakdolgozatom első fejezetében bemutattam a téma eredetét és a motivációmát, a piacon elérhető többi e-napló rendszert, valamint a feladatkiírás értelmezését.

A következő fejezetben kifejttem a feladat részletes specifikációját, közte az egyes komponensek elvárt funkcionalitásával is képernyőterveken keresztül.

Ezt követően röviden ismertetem a kliens- és szerveroldalak felhasznált technológiáit, és indoklom az egyes választások okát.

A 4. fejezetben a rendszer megtervezését mutatom be. Kitérek a rendszer magasszintű architektúrájára, a kliens- és szerveroldalak architektúrájára, és az ezek közötti kommunikációra is. Bemutatom az egyes komponensek által használt tervezési mintákat, és röviden írok a kapott adatbázis szakdolgozatom szempontjából releváns tábláinak sémájáról is.

Ezután írok a megvalósítás néhány érdekesebb részletéről. Itt kitérek néhány felmerült problémára, a tesztelésre, és a bemutatom a kész applikációkat képernyőképeken keresztül.

A dolgozatomat az utolsó fejezetben az elkészült munka értékelése, és a további fejlesztési lehetőségek zárják.

2 A feladat specifikálása

Ebben a fejezetben a szerver és kliensoldali elvárt funkciók részletes specifikációja található. A rendszer legfőbb célja, hogy a hétköznapiak során használt e-napló funkciók kényelmesen elvégezhetőek legyenek Android okostelefonról is mind a diákok, mind a tanárok számára. Másik célja, hogy kiterjessze a webes kliens funkcióit, itt adatbázisszinten eleve elérhető funkciókra, vagy minimális módosítást igénylő funkciókra kell gondolni.

2.1 Kliensoldal

A kliensoldal egy tanári és egy tanulói Android alkalmazás segítségével teszi elérhetővé az általam készített backend funkcióit a felhasználók számára. Ezek specifikációját a következő két alfejezetben fejtettem ki részletesebben, és közös képernyőterveket készítettem hozzájuk.

2.1.1 Tanulói kliens

Mivel a diákok adatbázis szinten semmilyen módosító tevékenységet nem végezhetnek, ezért az itteni funkciók csak megjelenítenek. Ezek összefoglalására készítettem az alábbi, use-case-eket tartalmazó táblázatot.

Use-case	Leírás
<i>Bejelentkezés</i>	A tanuló be tud jelentkezni egy felhasználó-jelszó párossal.
<i>Üzenetek</i>	A tanuló meg tudja nézni a bejövő üzeneteket, amiket egy tanára olyan csoportnak címzett, melynek tagja.
<i>Menza menüje</i>	A tanuló meg tudja nézni a menza aznapi menüjét, és lapozgatva a korábbi és későbbi napokét is.
<i>Órarend</i>	A tanuló meg tudja nézni az adott napi órarendjét, és lapozgatva a korábbi és későbbi napokét is.
<i>Óra tanulóinak listázása</i>	A tanuló az órarend nézetén egy tanórát kiválasztva meg tudja nézni az odajáró csoporttársait.
<i>Jegyek összegző nézet</i>	A tanuló egy összegző oldalon félévre lebontva megtekintheti az egyes tantárgyi átlagait.

<i>Adott tárgy jegyei</i>	A tanuló a jegyeinek összegző nézetén egy tárgyat kiválasztva egy adott félév adott tárgyának jegyeit és részleteit tekintheti meg.
<i>Késések</i>	A tanuló kilistázhatja késéseit.
<i>Hiányzások</i>	A tanuló kilistázhatja hiányzásait és azok részleteit (Függőben van-e? Igazolt-e?)
<i>Dicséret</i>	A tanuló kilistázhatja dicséreteit.
<i>Figyelmeztetések</i>	A tanuló kilistázhatja figyelmeztetéseit.
<i>Dolgozatok</i>	A tanuló kilistázhatja a tanárai által rögzített tervezett dolgozatait.
<i>Osztályom</i>	A tanuló megtekintheti osztályának részleteit, közte az osztályfőnökét és a heteseket.
<i>Kijelentkezés</i>	A tanuló a menüben a kijelentkezésre nyomva kijelentkezhet.

2.1.2 Tanári kliens

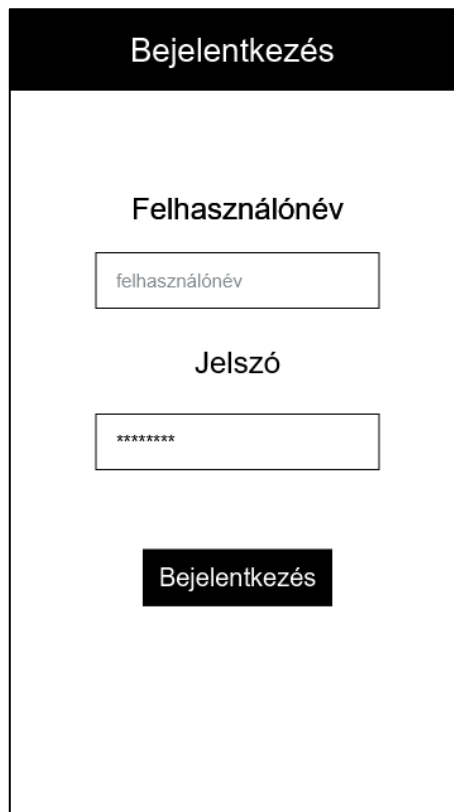
Az adatbázisban a tanárok által elérhető funkciók nagyon széleskörűek. Ezeknek a mindennapokban használt részhalmaza a mobilalkalmazásban is implementálásra fog kerülni. Összefoglalásukra készítettem az alábbi, use-case-eket tartalmazó táblázatot.

Use-case	Leírás
<i>Bejelentkezés</i>	A tanár be tud jelentkezni egy felhasználónév-jelszó párossal.
<i>Bejövő üzenetek</i>	A tanár meg tudja nézni a kapott üzeneteit.
<i>Kimenő üzenetek</i>	A tanár ki tudja listázni az elküldött üzeneteket a tanított csoportoknak és a tanártársaknak.
<i>Tanári üzenet részletei</i>	A tanár egy tanári üzenetre nyomva megtekintheti annak részleteit.
<i>Csoportos üzenet címzettjei</i>	A tanár egy csoportjának küldött üzenetre nyomva kilistázhatja az üzenet címzettjeit.
<i>Menza menüje</i>	A tanár meg tudja nézni a menza aznapi menüjét, és lapozgatva a korábbi és későbbi napokét is.
<i>Menza menü hozzáadása, törlése</i>	A tanár fel tud venni új menzai menü elemet, és törölni is tudja a meglévőket.

<i>Órarend</i>	A tanár meg tudják nézni az adott napi órarendjét, és lapozgatva a korábbi és későbbi napokét is.
<i>Óra naplózása</i>	A tanár az órarend nézetén egy tanórára nyomva naplózhatja a tanórát, rögzítheti a hiányzókat és a későket.
<i>Dicsérek megtekintése</i>	A tanár ki tudja listázni az általa beírt dicséreket.
<i>Dicséret rögzítése</i>	A tanár a dicsérek megtekintése fölön fel tud venni dicséretet egy általa tanított diáknak.
<i>Figyelmeztetések megtekintése</i>	A tanár ki tudja listázni az általa beírt figyelmeztetéseket.
<i>Figyelmeztetés rögzítése</i>	A tanár a figyelmeztetések megtekintése fölön fel tud venni figyelmeztetést egy általa tanított diáknak.
<i>Csoportos jegy beírása</i>	A tanár be tud írni jegyeket egy tanított csoportjának.
<i>Jegy beírása</i>	A tanár jegyet tud beírni egy tanított diákjának.
<i>Beírt jegyek megtekintése</i>	A tanár listázni tudja adott csoportjához és tantárgyhoz a beírt jegyeket félévre lebontva.
<i>Osztályom hetesei</i>	Ha a tanár osztályfőnök, akkor be tudja állítani az osztálya heteseit.
<i>Kijelentkezés</i>	A tanár a menüben a kijelentkezésre nyomva kijelentkezhet.

2.1.3 Az alkalmazások képernyőtervei

Az alkalmazások főbb képernyőterveit a Justinmind[2] nevű alkalmazással készítettem el a fejlesztés előtt, ezek segítségével szeretném vázolni az elképzelt alkalmazások terveit.



A képernyőterv egy mobilalkalmazás bejelentkezési felületét ábrázol. A felület fehér háttérrel rendelkezik, és a tetején egy fekete sávban a "Bejelentkezés" cím van. Középen a "Felhasználónév" cím mellett egy textmező található, amelyben a "felhasználónév" szöveg látható. Alatta a "Jelszó" cím mellett egy textmező van, amelyben a "*****" szöveg látható. A mezők alatt egy fekete gomb van, amelyen a "Bejelentkezés" szöveg áll.

2.1. ábra: Bejelentkeztető oldal

A 2.1 ábrán az alkalmazások bejelentkeztető felülete látható, ahol egy egyszerű felhasználónév-jelszó párossal jelentkezhetünk be.

Ha az alkalmazás indításakor nem vagyunk bejelentkezve, kijelentkeztünk, vagy lejár a bejelentkezésünk, akkor ez a képernyő fogad.

Ellenkező esetben ezt a képernyőt kihagyva rögtön a bejelentkezés utáni kezdőképernyőre jutunk, ami mindkét alkalmazásban a bejövő üzeneteket listázza.

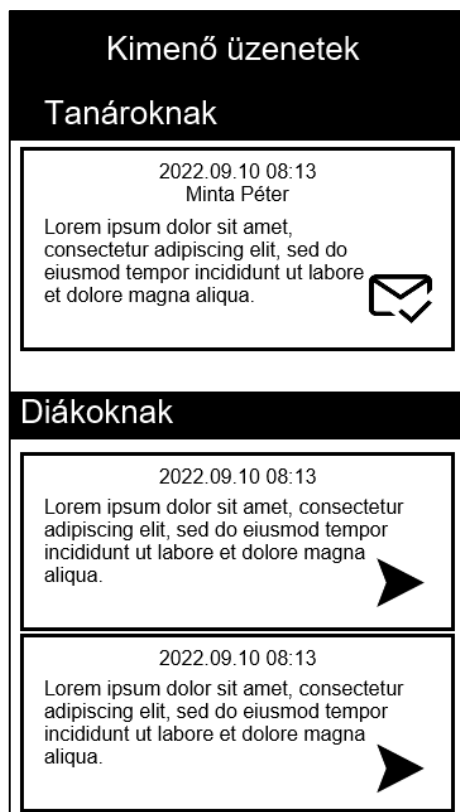
Tanulói alkalmazás	
Üzenetek	
Órarend	
Menzai menü	
Jegyeim	
Késéseim	
Hiányzásaim	
Dicséretem	
Figyelmeztetéseim	
Dolgozatok	
Osztályom	
Kijelentkezés	

2.2. ábra: Tanulói alkalmazás menüje

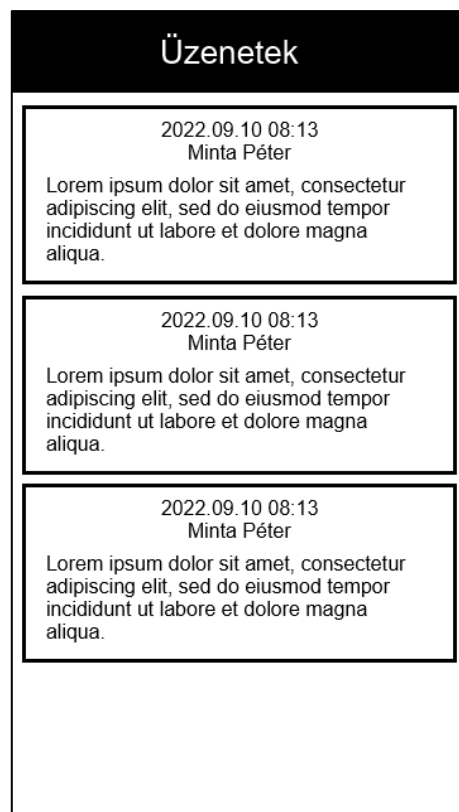
Tanári alkalmazás	
Bejövő üzenet	
Kimenő üzenet	
Menzai menü	
Menü felvétele	
Dicséret felvétele	
Figyelmeztetés felvétele	
Csoportos jegy bevitele	
Egyéni jegy bevitele	
Órarend megnézése	
Kijelentkezés	

2.3. ábra: Tanári alkalmazás menüje

Miután a felhasználó authenticálta magát, onnantól a 2.2-2.3 ábrákon látható menüket érheti el az alkalmazások fejléceiből. Mindkét alkalmazásban közös az órarend, a menzai menü, a kijelentkezés, és az üzenetek. A tanárok a diákjaiknak beírhatnak értékeléseket (például dicséretet, jegyet vagy késést), a tanulók pedig megtekinthetik ezeket.



2.4. ábra: Tanári alkalmazás kimenő üzenetei



2.5. ábra: Üzenetek megtekintése

A 2.4-es ábrán a tanári alkalmazáshoz tartozó kimenő üzenetek képernyőterve látható. Itt az üzenetek csoportosítva vannak az alapján, hogy tanárnak vagy csoportnak szólnak-e. A tanárnak küldött üzeneteknél egy zárt vagy nyitott boríték ikon mutatja, hogy a címzett tanár már megnyitotta-e azt. A csoportnak küldötnél pedig az üzenetet kiválasztva kilistázhatjuk a címzett tanulókat.

A 2.5-ös ábrán látható a bejövő üzenetek megtekintésére szolgáló képernyőterv. Ez a Tanárok esetében szintén tartalmaz egy nyitott vagy zárt borítékot ábrázoló ikont is az alapján, hogy a címzett megtekintette-e már az üzenetet. Tanulók esetében nem, mert az adatbázis nem rendelkezik a megfelelő igaz/hamis értékű oszloppal.

Értékelés rögzítése

Csoportjaim:

12.A

Tanuló:

Minta Diák

Értékelés:

Igazgatói dicséret

Üzenet beírása

Mentés

2.6. ábra: Értékelés beírása

Értékeléseim

Szaktanári figyelmeztetés

2022.09.10

Minta Tanár

Minta Diákot szaktanári figyelmeztetésben részesítem.

Szaktanári figyelmeztetés

2022.09.10

Minta Tanár

Minta Diákot szaktanári figyelmeztetésben részesítem.

Szaktanári dicséret

2022.09.10

Minta Tanár

Minta Diákot szaktanári dicséretben részesítem.

2.7. ábra: Értékelések megtekintése

A 2.6-os ábrán az értékelések rögzítésére szolgáló, képernyőkép látható, amely a tanárok számára érhető el. Ezek a képernyők az alkalmazásban ketté lesznek bontva dicséret és figyelmeztetés fülekre, de az egyszerűség és a hasonlóság miatt közös terv készült róluk.

A tanuló számára érhető el a 2.7-os ábra képernyőterve az általa kapott értékelések (dicséret, figyelmeztetés) listázására. Ez szintén ketté lesz bontva dicséret és figyelmeztetés fülekre.

Étlap 2022.09.10	
1. Menü Minta leves Minta második Alma	
2. Menü Minta leves Minta második Alma	

A 2.8-as ábrán a tanárok és tanulók számára is elérhető, étlap megtekintésére szolgáló képernyőterv látható. Az étlap egy napra tetszőleges számú menüt tartalmazhat, viszont ez jellemzően kettő szokott lenni. A menü tetszőlegesen kiegészülhet a két fogáson kívül valamilyen extra aprósággal: egy gyümölccsel, vagy például mikulásnapon egy mikuláscsokoládéval. Mivel a menza a régi adatbázisban nem szerepelt, így az adatok betöltését muszáj hozzáadni, mint funkciót. A tanulókhöz ezt nem akartam tenni, mert jó eséllyel nem megfelelően használnák, így ez a funkció a tanárok számára érhető el.

2.8. ábra: Menzai étlap

Jegy/Késés/Hiányzás rögzítése	
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Minta Diák	1 ▾
Mentés	

A tanárok számára érhető el a csoportos jegy/késés/hiányzás rögzítésére szolgáló 2.9-es ábrán látható opció is. Késés és hiányzás esetében ezek egy tanórához tartoznak. Az órarendben egy órát kiválasztva jutunk a tanóra naplózására szolgáló képernyőre, és onnan érhetőek el a késés és hiányzás rögzítése fülek.

A csoportos jegybevitel pedig a menüből érhető el, hiszen ez nem kapcsolódik tanórához.

Késés esetén a késett percek számát lehet megadni a tanuló neve mellett, hiányzás esetén egy kapcsolót állíthatunk igazra, jegybevitelnél pedig az osztályzatok között választhatunk.

2.9. ábra: Csoport számára értékelés rögzítése

2.2 Szerveroldal

A szerveroldal feladata lesz az autentikáció és autorizáció, az adatok kinyerése és eltárolása az adatbázisban, és a mobil kliensek által megvalósításra kerülő funkciókhoz szükséges adatok kiszolgálása REST API végpontok nyújtása segítségével.

A backend a meglévő adatbázishoz az adatbázis adatait tartalmazó szöveg (connection string) segítségével fog kapcsolódni. Egyelőre mindkettő Azure-on, az üzembehelyezést követően pedig az iskola szerverén fognak futni.

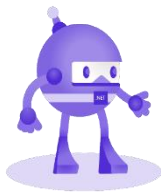
Az adatbázis adott, és rajta legfeljebb minimális, leginkább más táblákat nem érintő módosításokat (például a semmitől nem függő menzai menüt tartalmazó tábla hozzáadása) végezhetek annak érdekében, hogy kompatibilis maradjon az iskola valódi adatbázisával. A többi változtatáshoz szükségem lesz egyeztetésre az iskolával, így a nem-funkcionális bővítés nem célja a szakdolgozatomnak.

3 Felhasznált technológiák

Ebben a fejezetben szeretném bemutatni a fejlesztés során használt legfontosabb szerver és kliensoldalon használt technológiákat.

3.1 Szerveroldalon használt technológiák

3.1.1 ASP.NET 6



Az ASP.NET egy Microsoft által készített, nyílt forráskódú, szerveroldali webalkalmazás-keretrendszer, amelyet modern webszerverek fejlesztésére terveztek. Támogatja a C#, F#, és Visual Basic programozási nyelveket[3].

3.1. ábra: A .NET kabalája

forrás: Irodalomjegyzék[4]

A fejlesztés előtt több technológia is volt, amely nagyon tetszett, és szívesen megtanultam volna, mint például a Kotlin nyelvű Ktor. Mivel a .NET ismereteimet is szívesen bővítettem volna, és mert a frontendhez úgylátott Kotlin-t használtam, így végül ebben a keretrendszerben készítettem a webszerveremet, C# nyelven.

3.1.2 Entity framework és az adatbázis irányú megközelítés

Az Entity Framework (EF) egy objektum-relációs leképező (ORM) keretrendszer, amely segítségével a .NET alkalmazásunkban a relációs adatbázisunkhoz kódbeli modell osztályokat rendelhetünk.[5] Az adatbázis táblái osztályokká, az oszlopai osztály attribútumokká, a kapcsolatok pedig N-N-es kapcsolat esetében kapcsolótáblává, egyébként pedig navigációs attribútumokká (navigation property) alakulnak, amiken keresztül úgy érhetjük el a kapcsolt értékeket, mintha az osztály attribútumai lennének.

Három fő típusa az adatbázis, a modell és a kód alapú megközelítések. Az adatbázis alapú megközelítésben egy már létező adatbázist alapul véve generálhatjuk le a megfelelő modell osztályokat.[6] A modell alapú eljárás lényege, hogy egy tervező segítségével hozzuk létre az elképzelt sémát, és ebből a program elkészíti számunkra a többi. A Kód alapú megközelítés során pedig kódban megírjuk az elképzelt osztályokat, amiből így előállítható a relációs adatbázis.

Szakdolgozatom során az adatbázis alapú megközelítést alkalmaztam, hiszen volt egy kiinduló adatbázisom. Ennek esetében az volt a kényelmetlensége, hogy a relációs adatbázis táblái és oszlopai magyar nyelvűek voltak, így az elkészült modell osztályokban ezt egyesével kellett angolra fordítanom, és egységesítenem a néhol logikátlan elnevezéseket.

```
namespace enaplo.Models
{
    [Table("CsoportTagok")]
    public partial class GroupMember
    {
        [Column("Az")]
        public int Id { get; set; }
        [Column("TanuloAz")]
        public int StudentId { get; set; }
        [Column("CsopAz")]
        public int GroupId { get; set; }
        [Column("Egyeni")]
        public bool Custom { get; set; }

        public virtual Group GroupNavigation { get; set; } = null!;
        public virtual Student StudentNavigation { get; set; } = null!;
    }
}
```

3.1. kódrészlet: Egy modell osztály a generált, nem túl szerencsés elnevezésekkel

A fent látható 3.1-es kódrészlet egy egyszerűbb példa az adatbázisból készült osztályok szemléltetésére. Itt láthatóak az annotációk (Table, Column), amelyek az adatbázisban található tábla és oszlop neveit tartalmazzák. Mivel angolra fordítottam őket, így jelölnöm kellett az eredeti neveket annak érdekében, hogy az EF framework az adatbázis adatainak betöltése során össze tudja kapcsolni a megfelelő táblával, amit a repositorykba injektált DbContext segítségével végez.

3.1.3 JSON Web Token Authentikáció

A JSON Web Token (JWT) a jelenleg elérhető egyik legkorszerűbb, és legnépszerűbb felhasználók autentikálására használt technológia. A token a szerver sikeres hitelesítéskor adja ki, amit a kliensnek minden további kérése során csatolnia kell. Ebből következik, hogy tárolása is minden esetben az ő feladata. Ahhoz hasonlítható ez, mint egy garanciapapír: a meghibásodott eszközünket a szervízbe bevíve ők ellenőrzik a papír hitelességét, érintetlenségét és lejárátát, ha mindet helyesnek találják, akkor elvégzik a kért javítást.

A token 3 fő, egyenként hashelt részből épül fel, amelyeket pontok választanak egy egymástól. Az első rész a metaadatokat tartalmazza, ami nálam a token típusát és hashelés módját adja meg. A középső rész a token tényleges tartalmát jelöli, amit a token készítője adhat meg. Ilyen nálam a felhasználó azonosítója, a szerepe (tanár, diák, vagy admin), és a token lejáratási dátuma. A harmadik rész az aláírás, amihez veszi a hashelt metaadatokat és tartalmat, a titkos kulcsot, a hashelés típusát, és aláírja. Az aláírás csak akkor fog megegyezni a tokenben találhatóival, ha nem módosítottak semmit az adatokon, így ellenőrizve az érintetlenséget.[7]

Encoded

PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW11aWR1bnRpZm11ciI6IjIxMCIsImh0dHA6Ly9zY2h1bWVzLm1pY3Jvc29mdC5jb20vd3MvMjAwOC8wNi9pZGVudG10eS9jbGFpbXMvcm9sZSI6InR1YWNoZXIiLCJleHAiOjE2NzAyMjQxMDIsImV3ZWJzaXRlcy5uZXQvIiwiaXNkIjoiYHR0cHM6Ly9ubGVuYXBSby5henVyZXd1bnNpdGVzLm5ldC8ifQ.f7fxoFP_vC3vLQE8Q1rWxtR-ugt8c1pBa5nkQgpoyTk

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier": "210",

  "http://schemas.microsoft.com/ws/2008/06/identity/claims/role": "teacher",
  "exp": 1670224102,
  "iss": "https://nlenaplo.azurewebsites.net/",
  "aud": "https://nlenaplo.azurewebsites.net/"
}
```

VERIFY SIGNATURE

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

titkos kulcs

)

☐ secret base64 encoded

3.2. ábra: Egy Tanári JWT token

A 3.2-es ábrán látható egy a szerver által generált tanári token, amit a JWT hivatalos weboldala[7] által dekódoltam.

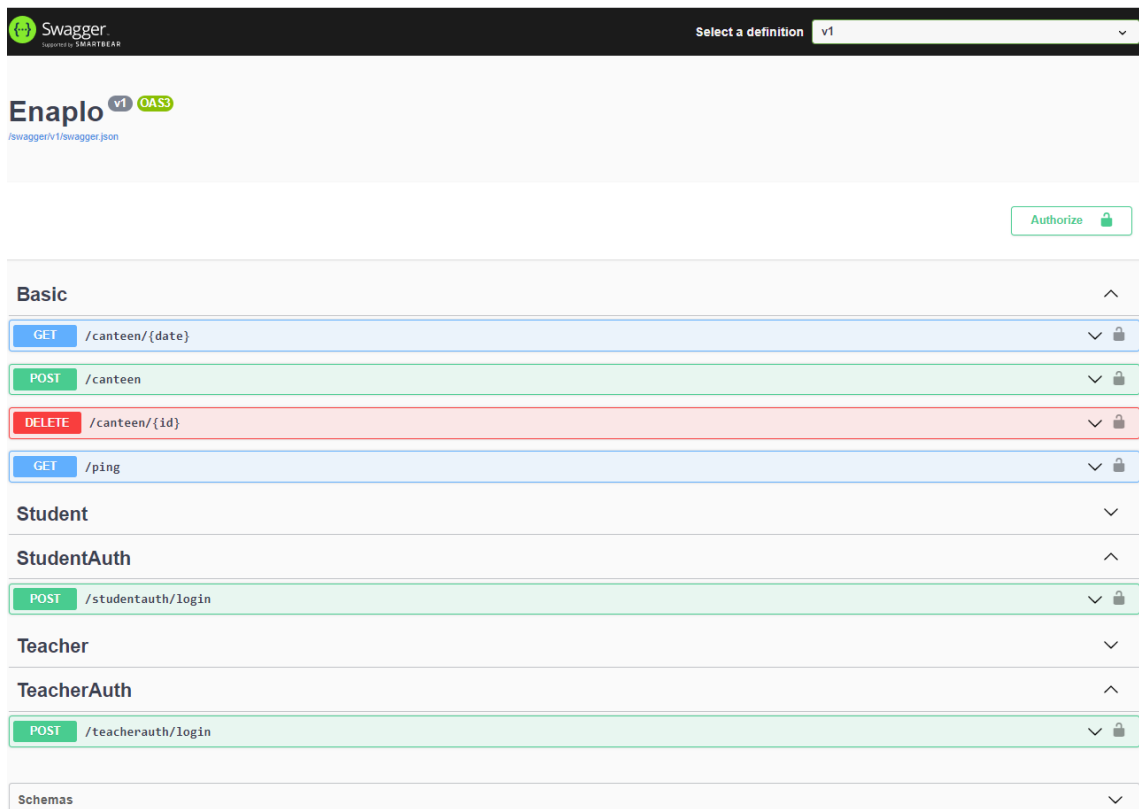
3.1.4 SQL adatbázis

Az adatok eltárolásához szükségem volt egy adatbázisra is. Ez egy Microsoft SQL Server (MSSQL) típusú minta relációs adatbázis, amit az iskola fenntartójától kaptam. Ebben az adatbázisban az adatok teljesen anonimizálásra kerültek, és az alkalmazások elkészítéséhez ezeket vettem alapul. Kezdetben több, mint 100 táblából állt, és dokumentáció nem tartozott hozzá. A számomra most nem szükséges táblákat töröltem belőle, és így 30 táblára sikerült lecsökkentenem a számukat.

3.1.5 Swagger

A Swagger egy könnyen használható API-fejlesztői eszközcsoomag, ami az API teljes életciklusa során segíti a fejlesztést: segítséget nyújthat például a tervezésben, a dokumentációban, a tesztelésben, és akár az üzembehelyezésben is.[8]

A Swagger segítségével generált UI-on keresztül autentikálhatjuk magunkat, és kipróbálhatjuk a végpontokat. Erről a felhasználói felületről látható egy kép a 3.3-as ábrán, ahol az átláthatóság kedvéért a végpontok kategóriáinak nagyrésze összezárt állapotban látható.



3.3. ábra: Swagger UI

Miután a képen is látható tanulói, vagy tanári bejelentkeztető végpontok valamelyikén (studentauth/login vagy teacherauth/login) hitelesítjük magunkat, azután a jobb felső sarokban látható zöld, „Authorize” feliratú gombra kattintva bemásolhatjuk a kapott tokent egy „Bearer ” prefixummal ellátva (jelölve ezzel a hitelesítő token felhasználási típusát), és így a weboldal felé intézett további kéréseink során a swagger automatikusan csatolja azt.

3.2 Kliensoldalon használt technológiák

3.2.1 Android platform és a Kotlin nyelv

Az Android egy Linux alapú mobil operációs rendszer. Fejlesztése 2005-ben kezdődött meg és 2008-ban jelent meg az első Androidos okostelefon. Az Android az összipiaci viszonylatban egy 2021 évvégi felmérés szerint 71%-os részesedéssel rendelkezett, megelőzve ezzel az Apple és a többi gyártó mobil operációs rendszereit.[9]

Az Android alkalmazások fejlesztése a JetBrains cég által létrehozott Android Studio nevű fejlesztőkörnyezet segítségével történik. A fejlesztés nyelve régebben a Java volt, viszont a JetBrains néhány éve létrehozta a Kotlin nyelvet a Java kiváltására, és a Google 2017-ben az Android fejlesztés hivatalos nyelvéné tette. [10] A nyelv egyik nagy előnye a Javával szemben, hogy sokkal kevesebb kóddal írhatjuk le benne ugyanazt, valamint, hogy a ma népszerű nyelvek legkedveltebb képességei is mind-mind megtalálhatóak benne.

Az elkészült Androidos alkalmazásainkat a Google Play áruházba feltöltve tudjuk a nagyközönség elé tárni, ahol jelenleg már körülbelül ötmillió alkalmazás elérhető.[11]

3.2.2 Retrofit 2

A Retrofit 2 egy típusbiztos REST kliens kódkönyvtár Androidra és Javára, amellyel a HTTP API-t Java interfésszé alakíthatjuk.[12] Használatával az API-hívásokat egyszerű Java vagy Kotlin metódushívásokként kezelhetjük, ahol csak azt kell pluszban meghatároznunk, hogy mely URL-re akarunk és milyen típusú kérést küldeni, valamint annotációval jelölhetünk például POST kérés esetén, hogy annak belsejében utazzon-e az adat. A JSON vagy XML átalakítást is teljes mértékben rábízhatjuk, melyhez a projekt a Retrofit2 Moshi nevű konverterét használja.

```
@GET("teacher/absence")
suspend fun getMissingStudents(
    @Query("LessonId") lessonId: Int,
    @Query("DividendId") dividendId: Int
): List<MissingResponse>

@POST("teacher/absence")
suspend fun postMissingStudents(
    @Body missingRequest: MissingRequest
): StringResponse
```

3.2. kódrészlet: példa az API-hívásokra

A 3.2-es kódrészleten egy tanárok számára elérhető GET és POST típusú példa látható az API-hívásokra. A GET hívás segítségével egy adott tanórához kérhetjük le a tanulók listáját és tanulónként igaz/hamis értéket az alapján, hogy van-e már könyvelve hiányzásuk arra az alkalomra. A kliensen oldalon a tanulókat egymás alá listázva, mellettük egy kapcsolóval érhetjük el. Miután a kapcsolón bejelöltük a hiányzókat a kódrészleten látható POST hívás segítségével visszaküldhetjük azt a webszervernek, amely ez alapján elvégzi az adatbázis frissítését.

3.2.3 Android Preferences

Az Android Preferences segítségével primitív típusokat (például szám, szöveg) tárolhatunk el a kliensoldalon kulcs-érték párokban.[13] Ebben tárolom el bejelentkezés után a JSON Web Token és törölöm a kijelentkezéskor.

A 3.3-as kódrészleten a UserPreferences osztály és metódusai láthatóak, amelyek segítségével a JWT eltárolható és törölhető a kliensoldalon.

```
class UserPreferences(  
    context: Context  
) {  
  
    private val applicationContext = context.applicationContext  
    private val datastore: DataStore<Preferences> =  
        applicationContext.createDataStore(name = "local_data" )  
  
    val jwtToken: Flow<String?>  
        get() = datastore.data.map { preferences ->  
            preferences[JWT_KEY]  
        }  
  
    suspend fun saveJwtToken(jwtToken: String) {  
        datastore.edit { preferences ->  
            preferences[JWT_KEY] = jwtToken  
        }  
    }  
  
    suspend fun cleareJwtToken() {  
        datastore.edit { preferences ->  
            preferences.clear()  
        }  
    }  
  
    companion object {  
        private val JWT_KEY = preferencesKey<String>("jwt_key")  
    }  
}
```

3.3. kódrészlet: A JWT eltárolását végző osztály

3.3 Azure

Az adatbázist, és az ezzel kommunikáló webszervert az üzembe helyezés után az iskola a saját szerverén fogja tárolni. Addigis kellett valamilyen átmeneti megoldás ahhoz, hogy a fejlesztés alatt a mobil applikációk el tudják érni a minta adatbázis adatait, és ez a választás az Azure-ra esett. Ez egy felhőszolgáltatásokat nyújtó felhőplatform[14], amelynek az adatbázis és a webszerver szolgáltatásait használtam.

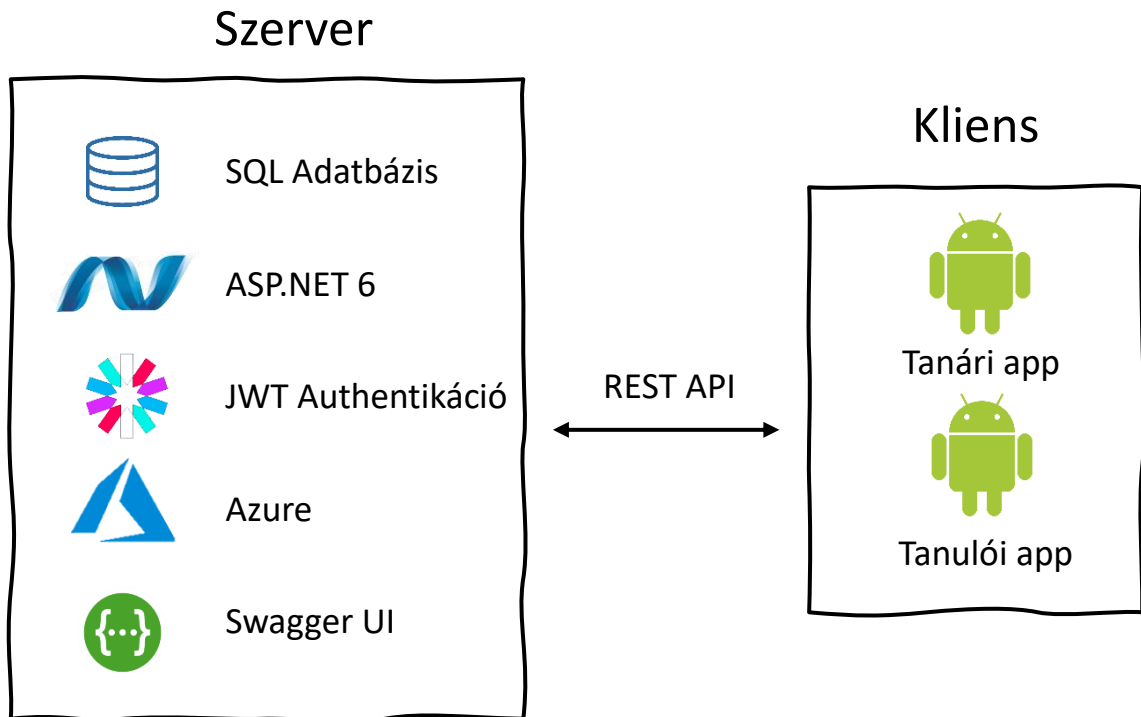
4 A rendszer megtervezése

Mivel korábban sohasem készítettem a feladatkiírásnak megfelelő komplexitású szoftvert, így a tervezés elején fontosnak tartottam, hogy a témáról kicsit tájékozódjak. Több tervezési minta tanulmányozása után végül szerveroldalon az MVC-re (Model-View-Controller), kliensoldalon pedig az MVP-t elvetve annak modern változatára, az MVVM-re (Model-View-ViewModel) esett a választásom.

A tervezési minták alkalmazása egy ilyen méretű szoftver esetén rendkívül kifizetődő. Felgyorsítják a tervezési folyamatot, hiszen a felmerülő problémákra bevett fejlesztési paradigmákat kínálnak, amelyek felmerülésekor így nem kell újra feltalálnunk a megoldást, hanem jóeséllyel a választott minta beépítetten tartalmazza azt[15].

Egy ilyen minta használatának előnye még, hogy egy a projektet nem, de a mintát ismerő programozó sokkal gyorsabban kiismerheti magát a kódban, mert az egyes moduljait meglátva egyből tudja azok logikai funkcióját. Használatával elérhetjük, hogy a projekt egyes komponensein belül erős függés, más komponensekkel pedig laza csatolás alakuljon ki. A laza csatolás következtében az egyes modulok cseréje könnyű, így elősegítve például a tesztelést, vagy a UI egyszerű cseréjét.

4.1 A rendszerszintű architektúra



4.1. ábra: A magasszintű architektúra

A 4.1-es ábrán látható a rendszer magasszintű architektúrája, amellyel célom egy átfogó képet nyújtani a rendszer egyes komponenseiről és az azok közötti kommunikáció irányáról. Az ábrán felhasznált logók az irodalomjegyzékben található [7], [8], [16], [17], és [18] weboldalakról származnak.

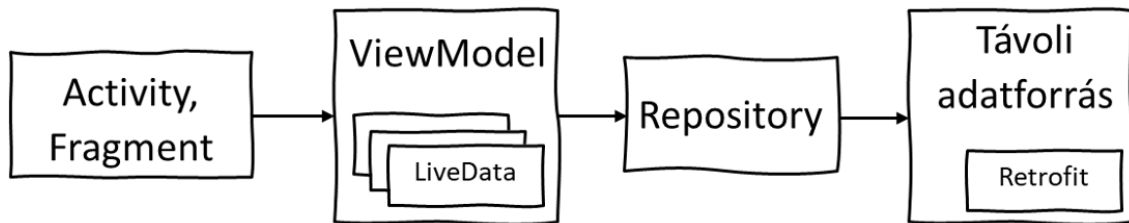
A kliens applikációk a szerverrel REST API-n keresztül kommunikálnak oda-vissza irányba, amelynek leírása a Swagger UI segítségével könnyen értelmezhető, az adatbázissal való kommunikációt pedig a webszerver végzi. Ha az egyik kliens adatmódosító kérést küld a szerver felé, akkor a webszerver ez alapján módosítja az adatbázist, és a másik kliens frissítést követően már a frissített értékeket látja.

4.2 A kliensek architektúrája

A kliensoldalon egy tanári és egy tanulói Android applikációt készítettem, amelyek viszont azonos architektúra alapján készültek, így ezt ebben a fejezetben összevontan, egyszerre mutatom be.

4.2.1 MVVM tervezési minta

Az MVVM tervezési minta jelenleg a Kotlin nyelvű Android applikációk által leggyakrabban használt tervezési minta. Három fő komponense a Model, a View és a ViewModel.



4.2. ábra: MVVM felépítése

A 4.2-es ábra az MVVM mintát mutatja be egy REST API-val kommunikáló Android alkalmazás esetén, amilyen az enyém is. Az Activityk és Fragmentek az alkalmazásunk UI részét valósítják meg, ami a mintában a Viewnak felel meg. Az adatok megjelenítéséhez adatkötésre (data binding) van szükség, amely a ViewModelben került megvalósításra, így a UI csak a ViewModeltől függ az adatok megszerzésének érdekében, a mélyebb rétegektől nem. A ViewModel pedig ugyanilyen függéssel rendelkezik a Repository irányába, mert onnan szerzi meg a UI számára továbbított adatokat. A Repository a Retrofit nevű könyvtár segítségével küld kéréseket a webszerverünk felé, ezzel lekérve onnan az adatokat.

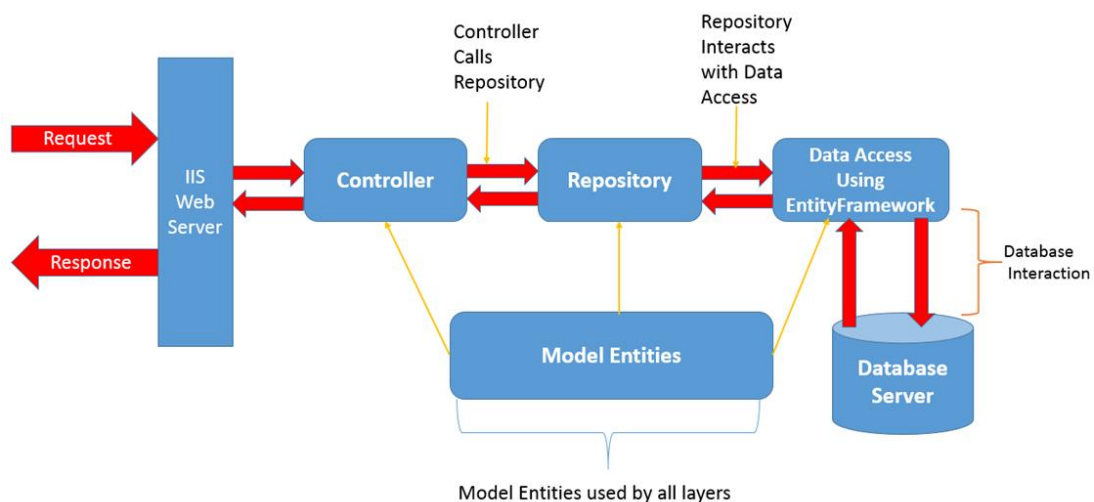
Az ábra és az alkalmazások architektúrájának tervezéséhez és elkészítéséhez egy angol nyelvű internetes videósorozatot vettem alapul.[19]

4.3 A szerver architektúrája

4.3.1 MVC és Repository tervezési minták

A Model-View-Controller tervezési mintával alkalmazhatjuk a „Separation of Concern” (azaz függőségek szétválasztása) tervezési elvet, hiszen szétválasztjuk egymástól a UI-t (View), az adatelérési réteget (Model) és az üzleti logikát (Controller), amelyhez a végpontok elérési útvonalait és feltételeit köthetjük.[20]

A Repository minta alkalmazása során egy köztes réteget helyezünk az üzleti logika, és az adatelérési réteg közé. Az üzleti réteg ilyenkor tipikusan egy interfészt ismer, amelyet a beékelte Repository osztály valósít meg. Ennek köszönhetően elrejtethetjük az adatelérés részleteit az üzleti logika elől, így könnyítve például a tesztelhetőséget, vagy biztosítva az adatelérés részleteinek egyszerű cseréjét az üzleti logika módosítása nélkül. A modularizálásnak köszönhetően könnyebben menedzselhetővé is válik a kódunk.



4.3. ábra: Repository minta az MVC-ben, forrás: [21]

A fent látható 4.3-as ábra az MVC és Repository minták együttes alkalmazásakor előálló architektúrát mutatja be.

4.3.2 Az adatbázis sémája

Mint korábban említettem, az adatbázis rendkívül összetett. Kezdetben száznál is több táblát tartalmazott, amiből a számomra most használaton kívülieket elhagyva harmincra sikerült leredukálnom számukat. Az adatbázis sémája nagyon nagy, és rendkívüli szövevényessége miatt teljesen átláthatatlan. Nem én készítettem, és biztonsági okokból nem is kaptam engedélyt a meglévő rendszer adatbázis sémájának bemutatására, ezért nem csatolok róla ábrát.

4.4 Kliens és szerver közötti kommunikáció

A kliens és szerver közötti kommunikáció a webservertől biztosított REST API végpontokon keresztül, JSON típusúvá alakított adatok segítségével történik. Mivel a mobil kliensek funkcionalitása eléggé gazdag, így a végpontok száma is jelentős, közel

ötven darab. Az adatbázis eléggé összetett, így a végpontok logikái is elég bonyolultak, ezek megírása a szakdolgozat készítésének egy jelentős részét tette ki.

4.4.1 Tanulók számára nyújtott végpontok

A tanulói kliens csak GET típusú, adatlekérő végpontokat érhet el, ezek szemléltetésére készítettem az alábbi táblázatot. A táblázatban található minden végpont a <https://nlenaplo.azurewebsites.net/student> útvonal után fűzve érhető el, és mivel mindegyik egységesen GET kérés, így ezt nem tüntettem fel mellettük.

Kérés típusa, útvonala	Funkciója
/name	A tanuló lekérdezheti a nevét, ami a menüben a név megjelenítéséhez kerül használatra.
/late	A tanuló listázhatja a késéseit.
/messages	A tanuló listázhatja bejövő üzeneteit. (Küldeni pedig nem tud, mert az adatbázis sémája nem teszi lehetővé).
/absence	A tanuló lekérdezheti hiányzásait idő szerint csökkenő sorrendben.
/admonitory	A tanuló lekérdezheti mulasztásait.
/propitious	A tanuló listázhatja dicséreteit.
/exams	A tanuló lekérheti a tanárai által rögzített tervezett dolgozatait.
/sumgrades	A tanuló listázhatja tantárgyait a hozzátartozó jegyek átlagával, félévre lebontva.
/grades	Egy adott tantárgyat és félévet kiválasztva a tanuló ezen keresztül megtekintheti jegyeit, és azok részleteit (például mire, kitől kapta).
/timetable/{date}	Egy adott dátumot megadva a tanuló lekérheti az adott napi órarendjét.

/groupmembers/{group}	A tanuló egy csoportját megadva listázhatja annak tagjait.
/class	A tanuló lekérheti osztályát és annak részleteit, például osztályfőnökét, és a heteseket.

Példaképpen az alább látható 4.1-es kódrészleten szeretném megmutatni a 3568-as azonosítójú tanuló <https://nlenaplo.azurewebsites.net/student/sumgrades> útvonalra küldött GET típusú kérésének eredményét leszűrve csak a magyar irodalom és nyelvtan jegyeire (az osztályzatok szintén véletlen értékekre lettek cserélve az anonimizálás érdekében):

```
[
  {
    "semester": "I. félév",
    "subject": "Magyar irodalom",
    "average": 1.6666666666666667
  },
  {
    "semester": "I. félév",
    "subject": "Magyar nyelv",
    "average": 4.7
  }
  {
    "semester": "II. félév",
    "subject": "Magyar irodalom",
    "average": 3.857142857142857
  },
  {
    "semester": "II. félév",
    "subject": "Magyar nyelv",
    "average": 4.8
  },
  ...
]
```

4.1. kódrészlet: Egy tanuló magyar jegyei /sumgrades végpont válaszában.

4.4.2 Tanárok számára nyújtott végpontok

A tanári mobil kliens már jóval szélesebb körű funkcionalitással bír, mint a tanulói, hiszen ez a már meglévő adatok listázása mellett sokféle módosító kérést is küldhet a webszervernek. Ezeket a kéréseket részletezi az alábbi táblázat, ahol a kérések egyaránt a <https://nlenaplo.azurewebsites.net/teacher> útvonalhoz fűzve érhetőek el.

Kérés típusa, útvonala	Funkciója
GET /name	A tanár lekérheti a nevét, amit a menüben a név megjelenítéséhez szükséges.
GET /timetable/{date}	A tanár lekérheti egy adott napi órarendjét.
GET /outmessages/teacher	A tanár lekérheti tanártársainak küldött üzeneteinek előnézetét.
GET /outmessages/group	A tanár listázhatja tanított csoportjainak küldött üzeneteit.
GET /outmessages/group/{messageid}	A tanár egy csoportüzenet azonosítóját megadva listázhatja az üzenet címzettjeinek listáját.
GET /messages/{messageid}	A tanár egy tanári üzenet azonosítóját megadva lekérheti a teljes üzenetet részleteivel.
GET /teachers	A tanár lekérheti tanártársainak listáját, azaz mindenkiét a sajátján kívül.
POST /message/teacher	A tanár üzenetet küldhet egy tanártársának.
GET /messages	A tanár lekérheti tanártársaitól kapott üzeneteinek előnézetét.
GET /groups	A tanár lekérheti tanított csoportjait.
POST /message/group	A tanár üzenetet küldhet egy csoportjának.
GET /propitious/types	A tanár lekérheti a dicséretes lehetséges típusait.
GET /admonitory/types	A tanár listázhatja a figyelmeztetések lehetséges típusait.

POST /propitious/	A tanár dicséretet rögzíthet egy diákja számára.
POST /admonitory	A tanár figyelmeztetést írhat be egy diákjának.
GET /admonitory	A tanár listázhatja idáig beírt figyelmeztetéseit.
GET /propitious	A tanár megjelenítheti eddig rögzített dicséreteit.
GET /groupmembers/{groupid}	A tanár egy csoport azonosítóját megadva lekérheti a csoport tagjait. Ehhez nem kell, hogy tanítsa őket.
GET /registerlesson	A tanár lekérheti egy tanóra naplózási adatait, amennyiben a tanóra még nem került naplózásra, akkor automatikusan létre is jön.
POST /registerlesson	A tanár mentheti a tanóra naplózási adatain végzett módosításokat.
GET /absence	A tanár egy tanóra naplózási azonosítóját, és a tanórafelosztás azonosítóját megadva lekérheti egy tanóra hiányzóit.
POST /absence	A tanár a GET /absence által kapott adatok módosításait visszaküldheti a szervernek, így mentve azt.
GET /late	A tanár egy tanóra naplózási azonosítóját, és a tanórafelosztás azonosítóját megadva listázhatja egy tanóra későit.
POST /late	A tanár a GET /late által kapott adatok módosítása után mentheti azokat.
GET /grade/types	A tanár lekérheti a lehetséges osztályzatok típusait

POST /grade	A tanár egyéni jegybevitelt csinálhat annak részleteit megadva.
POST /grades	A tanár csoportos jegybevitelt hajthat végre az osztályzatok, és azok részleteinek megadásával.
GET /subjects/{groupid}	A tanár egy adott csoportjához lekérdezheti azokat a tárgyakat, amelyekből tanítja őket.
GET /sumgrades/{dividendid}	A tanár a tanórafelosztás azonosítóját megadva listázhatja az egyes tanulókat a hozzájuk tartozó átlagokkal, félévekre lebontva.
GET /grades	A tanár egy félévet, tantárgyat és tanulót kiválasztva listázhatja annak jegyeit, és jegyeinek részleteit.
GET /class	Amennyiben a tanár osztályfőnök, vagy helyettes, akkor listázhatja osztályának adatait, köztük a heteseket.
POST /class	Amennyiben a tanár osztályfőnök, vagy helyettes, akkor a GET /class kérés eredményében a heteseket átírhatja és ezen a végponton visszaküldheti, így rögzítve azokat.

Példaképpen az alább látható 4.2-es és 4.3-as kódrészleteken szeretném megmutatni a késők listázását és rögzítését. A 4.2-es ábrán egy POST kérés látható, amit a <https://nlenaplo.azurewebsites.net/teacher/late> útvonalra küldtem, alatta a 4.3-as ábrán pedig egy GET kérés található, amit ugyanezen adatok lekérdezésére küldtem el a <https://nlenaplo.azurewebsites.net/teacher/late?lessonId=393925÷ndId=6576> útvonalra.

```
{
  "lessonId": 393925,
  "dividendId": 6576,
  "date": "2022-11-25",
  "lates": [
    {
      "int": 2913,
      "string": "Antal Richárd",
      "len": 5
    },
    {
      "int": 2890,
      "string": "Balog Borbála",
      "len": 6
    },
    {
      "int": 2285,
      "string": "Berki Eliza",
      "len": null
    },
    ...
  ]
}
```

4.2. kódrészlet: Egy tanóra késéseinek beállítása a /late útvonalra küldött POST kérésben

```
[
  {
    "int": 2913,
    "string": "Antal Richárd",
    "len": 5
  },
  {
    "int": 2890,
    "string": "Balog Borbála",
    "len": 6
  },
  {
    "int": 2285,
    "string": "Berki Eliza",
    "len": null
  },
  ...
]
```

4.3. kódrészlet: Egy tanóra késői a /late útvonalra küldött GET kérés eredményében

4.4.3 Egyéb végpontok

Az átláthatóság kedvéért a végpontokat néhány főbb tag alá soroltam. A teacher és student tagek elég jelentősek, ezért ezeket külön-külön mutattam be. Ebben az alfejezetben szeretném bemutatni a többi tag alá tartozó végpontokat. Ezek összefoglalására is készítettem egy táblázatot, amely itt látható.

Kérés típusa, útvonala	Funkciója
GET /ping	Authentikáció nélkül érhető el ez a végpont, amelynek eredménye egy JSON-be ágyazott „pong!” üzenet ezzel jelezve a küldő számára, hogy a webszerver elérhető.
POST /studentauth/login	Authentikáció nélkül érhető el ez a végpont, ahol az üzenet belsejében egy helyes tanulói felhasználó-jelszó párossal megkaphatjuk a JWT, amit a további kéréseinkhez csatolhatunk, így „bejelentkezve”.
POST /teacherauth/login	Megegyezik a fenti „/studentauth/login”-ra küldött POST üzenet leírásával, de itt tanárként lehet bejelentkezni.
GET /canteen/{date}	Tanulóként vagy tanárként egy dátumot megadva lekérhetjük az adott napi menzai menüt.
POST /canteen	Tanárként egy adott menüelemet módosíthatunk annak adatait megadva.
DELETE /canteen/{id}	Tanárként egy adott menüelemet törölhetünk annak azonosítóját megadva.

Példaképpen szeretném bemutatni egy menzai menü lekérése érdekében küldött GET kérés eredményét a <https://nlenaplo.azurewebsites.net/canteen/2022-11-25> útvonalra, amely a 4.4-es kódrészleten látható:

```
[
  {
    "id": 324,
    "date": "2022-11-25T00:00:00",
    "firstMeal": "Bableves",
    "secondMeal": "Rántott sajt sültkrumplival",
    "extra": "Dió"
  },
  {
    "id": 325,
    "date": "2022-11-25T00:00:00",
    "firstMeal": "Gombaleves",
    "secondMeal": "Zöldséges tésztasaláta",
    "extra": ""
  }
]
```

4.4. kódrészlet: /canteen útvonalra kiadott GET kérés eredménye

5 A megvalósítás érdekesebb részletei

Ebben a fejezetben a megvalósítás néhány érdekesebb részletét szeretném bemutatni. Röviden írok a fejlesztés során felmerült néhány nehezebben megoldható problémáról, bemutatom a kész alkalmazást képernyőképen keresztül, és írok az elkészült munka teszteléséről is.

5.1 Néhány felmerült probléma

5.1.1 Megárvult SQL felhasználók

A kapott adatbázis Azure-ra történő telepítése során belefutottam egy olyan hibába, hogy bizonyos „orphaned user”-ek, azaz olyan felhasználók maradtak az adatbázisban, akikhez már nem kapcsolódtak bejelentkezési adatok. Ezeket a felhasználókat az irodalomjegyzékben lévő linken[22] található script lefuttatásával töröltem ki, és ezután további hibák nélkül ment az Azure-ra történő telepítés.

5.1.2 SQL adatbázis adatainak frissítése

Az anonimizált adatbázis eredendően az iskola tavalyi adatbázisa volt. Emiatt a mobil alkalmazásokban például az órarendet megnyitva nem kaptunk adatot az adott napra, csak ha fél évet visszalapoztunk. Ez így használhatatlan volt a fejlesztéshez, teszteléshez, így az összes dátumhoz hozzáadtam egy évet.

Ez egyrészt azt eredményezte, hogy jelenleg olyan mintaadatok is betölthetnek a kliensekben, amelyek keltezésük alapján még létre sem jöttek. Mivel a valós adatbázison ez nem fordulhat majd elő, és így néhány képernyő (például a jegyek összegző nézete) kifejezőbb, ezért nem változtattam rajta.

A másik hozadéka, hogy ideiglenesen elromlott néhány funkció, mert az adatbázis néhány táblájában a dátumon kívül egy hét napja oszlop is el van tárolva, amire szükség van a dátum egyértelmű azonosításához (az egyik táblában a dátum csak a hetet határozza meg). Ahogy egy évet hozzáadtam a dátumokhoz, így hibára futottak a végpontok. A hét napja oszlop a valósághoz képest egy nap csúszásban volt, de ezt a napok csúsztatásával, és a szombatok hétfőre cserélésével könnyedén orvosolni lehetett.

5.1.3 Hiba a RecyclerViewban felvett listenerekkel

A RecyclerViewban több esetben is az elemekhez listenereket kellett felvennem. Például a késés beírására szolgáló képernyőn az EditText típusú elemhez szerettem volna a szöveg megváltozásához regisztrálni egy listenert.

Ez el is készült, viszont a kész képernyőt fel-le görgetve idővel minden tanuló mellé valamilyen szám íródott be hibásan. Rövid hibakeresés és interneten való kutakodás után arra jutottam, hogy a hiba forrása az, hogy az onBindViewHolder metódus minden alkalommal beregisztrált egy új listenert az elemhez, amihez így a régi, más adatra mutató listenerek is hozzákapcsolva maradtak, ezek írták be a valótlan értékeket.[23]

A megoldás egyrészt az volt, hogy a listenert a ViewHolder létrehozásakor regisztráltam az onBindViewHolder metódus helyett, illetve hogy az elemek újrahasználatosságát is letiltottam a ViewHolder IsRecyclable tulajdonságának hamisra állításával.

5.1.4 SingleLiveEvent a LiveEvent helyett

A már elkészült, jól működőnek hitt alkalmazás sokadik tesztelése során vettem észre azt az elég nagy problémát, hogy a legtöbb POST műveletet végző képernyőre nem lehetett visszatérni az első mentés után. Tehát például amikor tanárként egyszer már frissítettem a heteseket az „Osztályom” menüpontban, és újra az „Osztályom” menüpontra próbáltam lépni, akkor látszólagos navigálás nélkül csak „A hetesek frissítése sikeres volt!” üzenetet láttam felugrani.

A problémát az okozta, hogy a LiveEventre illesztett observer nemcsak egyszer kapott értesítést az elküldött POST kérés eredményéről, hanem az első válasz után folyamatosan azt jelezte, hogy a válasz megérkezett. Mivel a válasz megérkezése esetén a program feladata a képernyő bezárása és a válaszüzenet feldobása volt, ezért láttam úgy, mintha a képernyő meg sem nyílt volna.

A megoldást a LiveEventek helyett a SingleLiveEvent használata jelentette.[24] Mivel ezt az osztályt a beépített könyvtárak nem tartalmazzák, ezért kódját kimásoltam az internetről[25], és felvettem egy ilyen nevű osztályt. Ezután pontosan azt történt, amit eredendően is vártam: az egyes POST üzenetekre a válasz üzenetét csak egyszer jelezte az observer, utána gond nélkül vissza tudtam lépni a képernyőre.

5.1.5 Azure okozta teljesítménycsökkenés

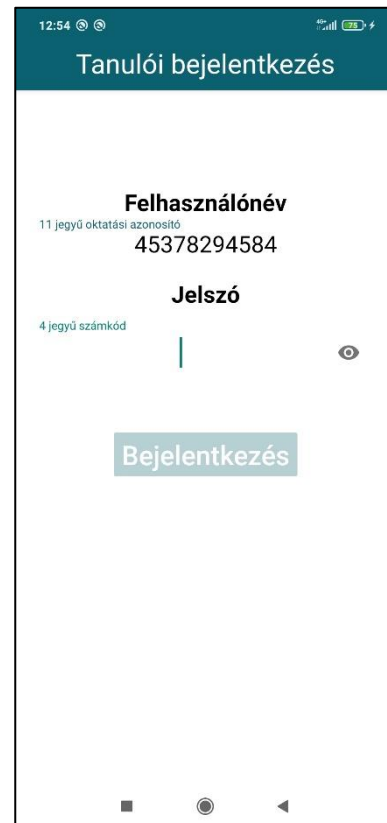
Amennyiben hosszú ideig nem kap kért egy Azure-on futó webservert, akkor bizonyos szolgáltatási szinteken az infrastruktúra felügyelő rendszer leállíthatja. Ezt a „Mindig bekapcsolva” funkcióval lehet szabályozni, de a választott ingyenes webservert csomagban ennek bekapcsolása nem engedélyezett. Ez azt eredményezi, hogy mikor használni akarjuk a mobil klienseket, akkor az első kérésre nagyon sok időt (akár egy percet) is várni kell, hiszen meg kell várni ameddig a szolgáltató felébreszti. Ezután viszont a többi kérés már a szokott, maximum néhány másodperces válaszidővel működik.

5.2 Képek a kész alkalmazásokról

Ebben az alfejezetben az elkészült mobilalkalmazásokat szeretném képernyőképeken keresztül bemutatni. Az elkészült alkalmazásokat megnyitva elsőként a bejelentkeztető oldalra jutunk. Itt egy felhasználónév-jelszó párossal jelentkezhetünk be, és a „Bejelentkezés” gomb csak az adatok beírása után válik aktívvá. Az 5.2-es és 5.3-as képernyőképeken ezek a bejelentkeztető felületek láthatóak.



5.2. ábra: Tanári bejelentkező felület

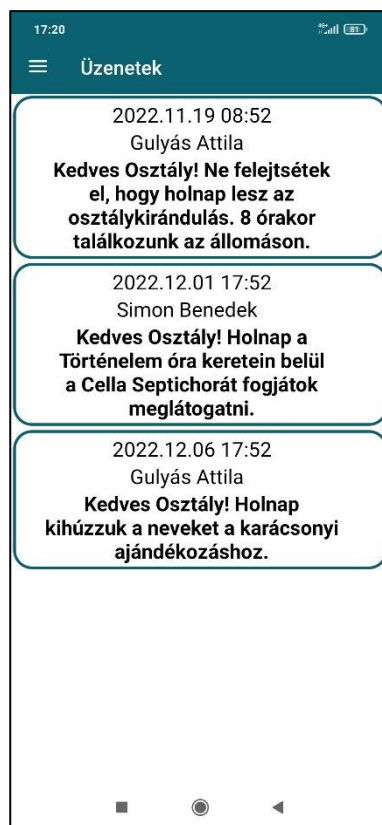


5.3. ábra: Tanulói bejelentkező felület

A bejelentkezést követően mindkét applikáció a felhasználót a bejövő üzenetek földre navigálja. Amennyiben már korábban bejelentkeztük az alkalmazásba és ez még nem járt le, akkor az alkalmazás megnyitásakor egyből ez a képernyő fogad bennünket. Ezek a képernyők láthatóak az alábbi, 5.4-es (tanári), és 5.5-ös (tanulói) ábrákon. A tanári alkalmazásban az üzenetek mellett látható egy zárt vagy nyitott borítékot ábrázoló ikon is az alapján, hogy az üzenetet a címzett megnyitotta-e már.



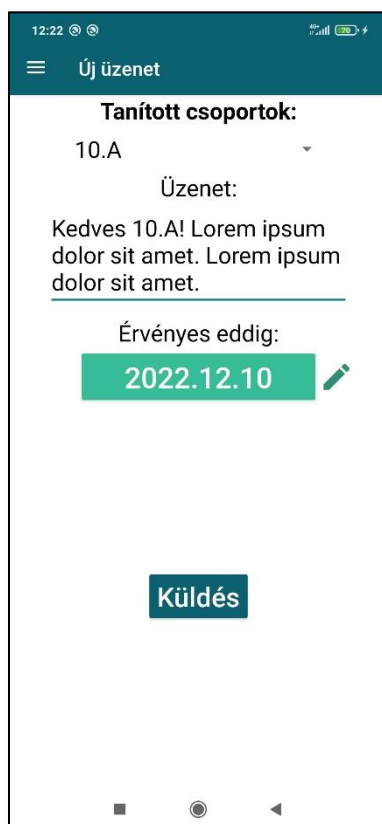
5.4. ábra: Tanári bejövő
üzenetek képernyő



5.5. ábra: Tanulói bejövő
üzenetek képernyő



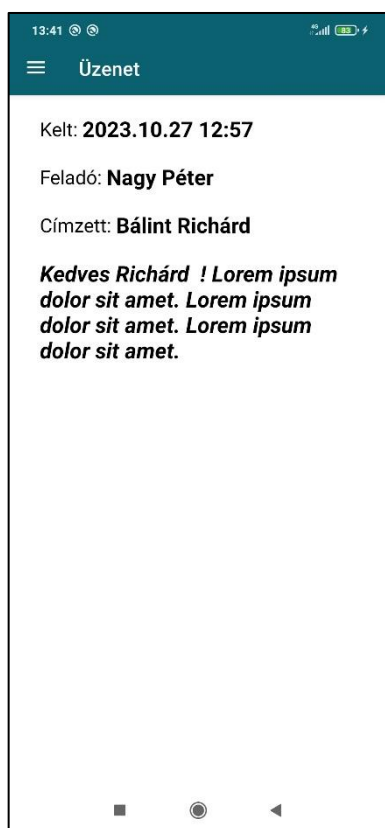
5.6. ábra: Tanári kimenő
üzenetek képernyő



5.7. ábra: Tanár tanulói csoportnak
üzenetek küldő képernyő

A tanulói oldalon az adatbázis hiányosságai miatt nincsen lehetőség üzenetek küldésére, sem az üzenetek olvasottnak jelölésére. A tanárok számára viszont lehet üzenetet küldeni, és a kimenő üzeneteket megtekinteni. Az elküldött üzenetek megtekintésére szolgáló képernyő látható az 5.6-os ábrán. Ha az 5.6-os ábrán a „Tanulói csoportnak” cím jobb oldalán tanálható „Új” feliratú gombra nyomunk, akkor az 5.7-es ábrán látható képernyőre jutunk. Itt adhatjuk meg a tanított csoportnak küldött üzenet részleteit. Ez a képernyő nagyon hasonlít arra, amellyel a tanártársaknak küldhetünk üzenetet, ezért arról nem csatolok külön képet.

Az 5.4-es ábrán a bejövő üzeneteknél, és a 5.6-os ábrán a tanári kimenő üzeneteknél is csak az üzenet egy részlete látható. A teljes üzenetet az üzenetre nyomva tekinthetjük meg, amelyre egy példa az 5.8-as ábrán látható. A tanulói csoportnak küldött üzenetet kiválasztva pedig kilistázhatjuk az üzenet címzettjeit. Ezt az alább látható 5.9-es ábra szemlélteti.

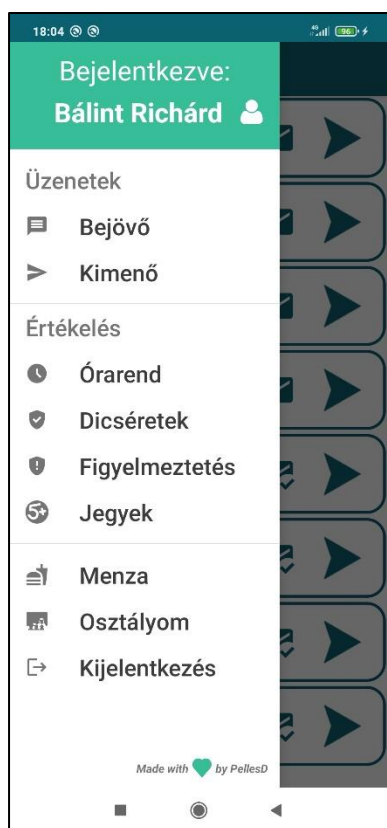


5.8. ábra: Tanári üzenet részleteinek megtekintése

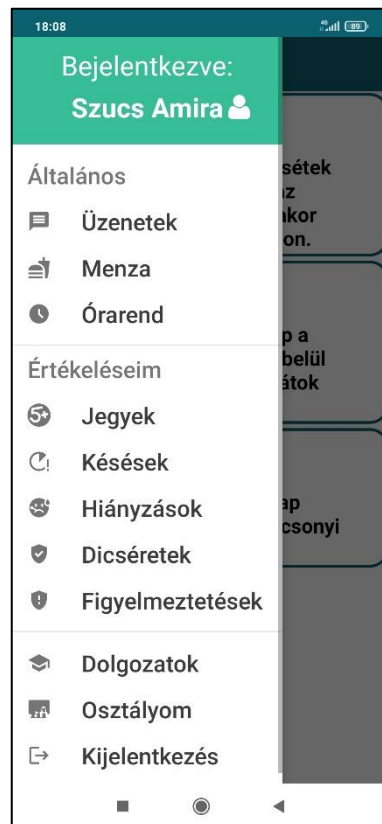


5.9. ábra: Tanulói üzenet címzettjeinek listázása

Mindkét alkalmazáshoz tartozik egy-egy menü, ahol az alkalmazások egyes funkciói között válogathatunk. Az 5.10-es ábrán a tanári, az 5.11-es ábrán a tanulói applikáció menüje látható, a menük tetején pedig a bejelentkezett személy neve olvasható.



5.10. ábra: Tanári alkalmazás menüje



5.11. ábra: Tanulói alkalmazás menüje

A menün az „Órarend” lehetőséget kiválasztva a tanárok és diákok esetében a következő oldalon látható 5.12-es és 5.13-as oldalakra jutunk. Azt, hogy melyik nap órarendjét szeretnénk megtekinteni, az mindkét esetben a képernyő alján található nyilak segítségével módosítható/lapozgatható 1-1 napot előre, vagy hátra.

A tanulói órarend esetén egy adott órát kiválasztva az 5.9-es ábrához hasonló képernyőn listázhatjuk az órára járó diákok névsorát. A tanári órarend már ennél jóval összetettebb, itt az egyes tanórák mellett három ikon látható. Az első ikonon néhány csillagos ötös osztályzatot ábrázol, amelyet kiválasztva az adott órához tartozó csoportos jegybevétel képernyőre léphetünk, amit az 5.14-es ábra jelenít meg. A második ikonon egy darab csillagos ötös osztályzat látható, ezzel a tanórához kapcsolódó egyéni jegybevételi képernyőre navigálhatunk, ami az 5.15-ös képernyőképen látható. A könyv ikonnal pedig a tanóra naplózását végezhetjük el, amelyet az 5.16-os ábra mutat be.



5.12. ábra: Tanári órarend



5.13. ábra: Tanulói órarend



5.14. ábra: Csoportos jegybevétel



5.15. ábra: Egyéni jegybevétel



5.17. ábra: Hiányzók rögzítésére szolgáló képernyő



5.18. ábra: Késők rögzítésére szolgáló képernyő



5.16. ábra: Tanóra naplózására szolgáló képernyő

Az 5.16-os ábrán látható képernyőn végezhető el a tanóra naplózása. Itt módosítható a tanóra néhány alapbeállítása is, valamint megadható a tanórához tartozó cím. A „Hiányzások”, illetve „Késések” feliratú gombokra nyomva a tanár tovább navigálhat egy-egy képernyőre, melyekkel a tanóra hiányzóit és későit rögzítheti. Az 5.17-es ábrán a hiányzókat jelölhetjük be az egyes kapcsolók állítgatásával, az 5.18-as képernyőképen látható oldalon pedig a késéseket naplózhatjuk a késett percek bevitelével.

A tanuló beírt késéseit a „Késések”, beírt hiányzásait és azok státuszát pedig a „Hiányzások” menüpontot kiválasztva érheti el, amelyeket az 5.19-es és 5.20-as ábrákon szemléltetnek. Az egyes hiányzások mellett megtekinthető annak státusza, és a státusz szerinti színe a keresetőség megkönnyítésének érdekében. A hiányzások, valamint a késések időpont szerint csökkenő sorrendben helyezkednek el.

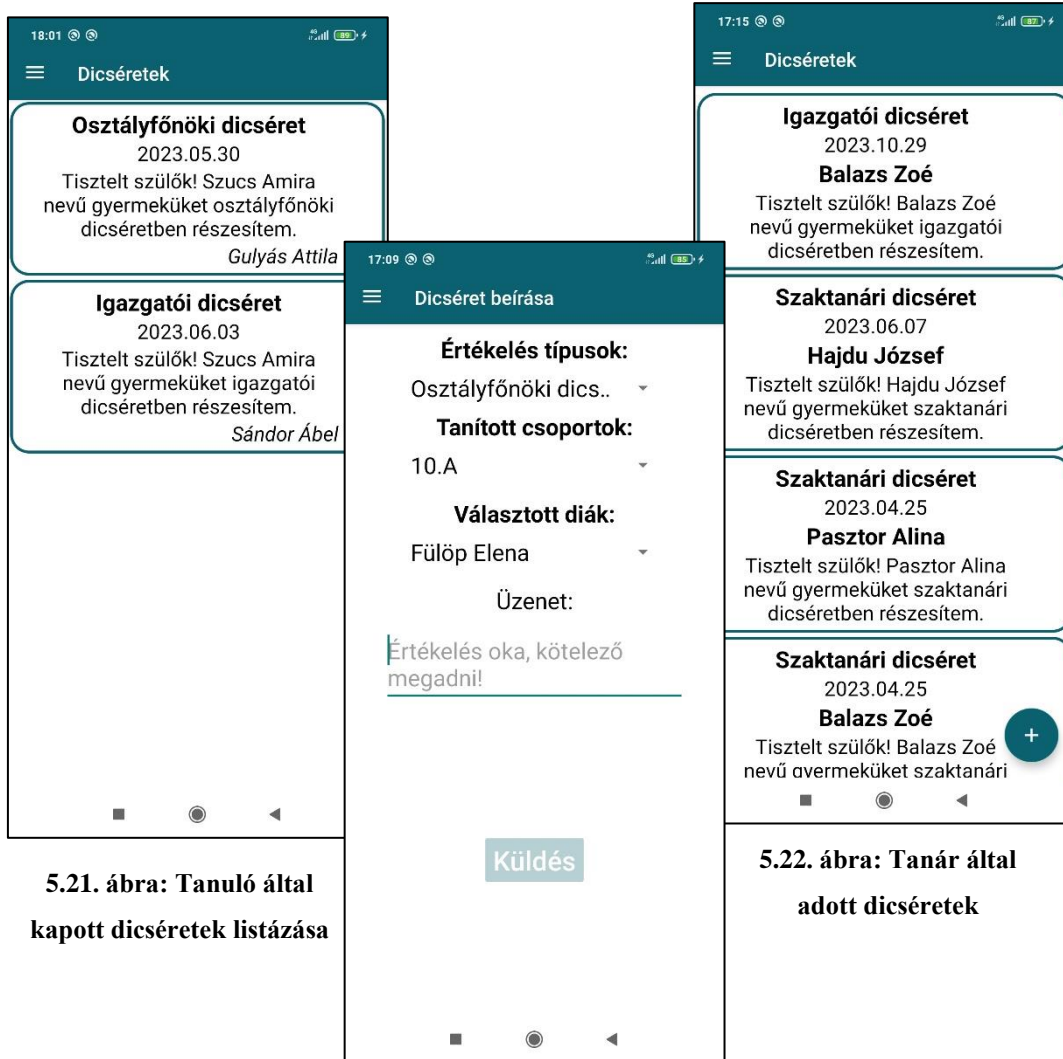


5.19. ábra: Tanuló késései



5.20. ábra: Tanuló hiányzásai

A menüben a „Dicsérek” opciót kiválasztva a tanuló esetén az 5.21-es ábrán látható képernyőre jutunk, ahol a tanuló által kapott dicséreteket és azok részleteit listázhatjuk ki. Tanárok esetén az 5.22-es ábrán megjelenített képernyőre navigálunk, ahol az adott tanár által rögzített dicsérek, és azok részletei találhatók. A jobb alsó sarokban található hozzáadás ikonnal az 5.23-as ábrán látható képernyőre jutunk, ahol a tanár egy tanított diákja számára adhat dicséretet.



A „Figyelmeztetések” menüpontot kiválasztva dicsérek helyett figyelmeztetésekkel végezhetjük el ugyanezeket a műveleteket, ezért ezt nem részletezem külön.

A „Menza” menüpontot kiválasztva az 5.24-es ábra képernyőjére jutunk, ahol az iskolai menza adott napi menüit láthatjuk. A tanárok esetében az egyes menük mellett a szerkesztést és törlést lehetővé tevő ikonok találhatóak, a képernyő alján pedig egy menüelem hozzáadását biztosító gomb helyezkedik el. Mivel a menzai menü funkciót én hoztam létre, így a webes kliensekben ez nem létezik, tehát a mobil kliensben biztosítani kellett a menüelemek szerkesztését/hozzáadását/törlését, amit a tanárok számára tettem lehetővé. A tanulók ugyanezt a képernyőt érik el, de csak megtekinteni tudnak.

A jobb oldali 5.25-ös ábrán a menüelem szerkesztését/hozzáadását szolgáló dialógusablak látható. A tanárok itt felvehetnek/szerkeszthetnek első-, és második fogást, valamint opcionálisan valami extra dolgot, ami az ebédhez járhat (például alma).



5.24. ábra: Menzai menü



5.25. ábra: Menzai menü szerkesztése/hozzáadása

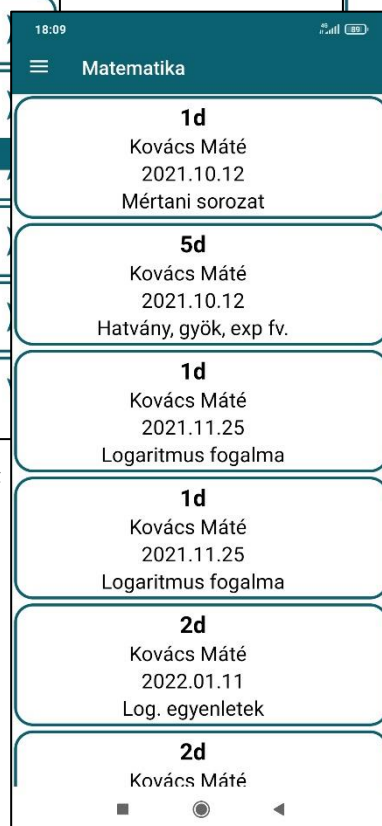
A menüben a „Jegyek” opciót választva tanárok esetén az 5.26-os ábrán látható, tanulók esetén pedig az erre nagyon hasonló 5.27-es ábrán látható képernyőre érkezünk.



5.26. ábra: Tanított csoport jegyei félévenként



5.27. ábra: Tanuló jegyei félévenként, tárgyanként



5.28. ábra: Jegyek részletező nézet

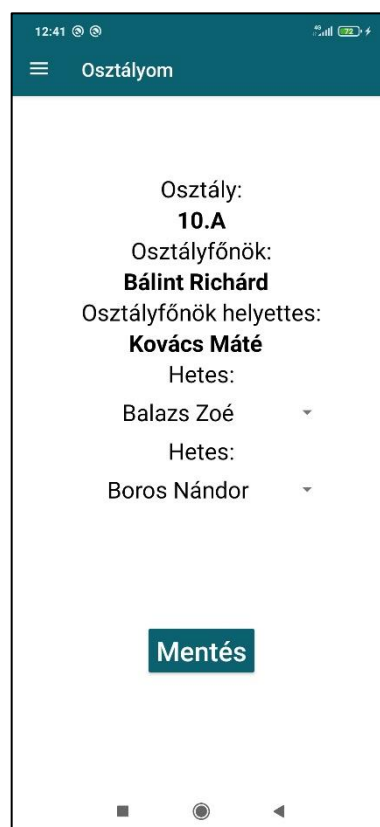
A tanári változatban a képernyő tetején található egy összecsukható menü, amelyben kiválaszthatunk egy csoportot, és egy tantárgyat. Ennek kiválasztása után az adott csoport diákjainak megfelelő tantárgyból szerzett, két tizedesjegyre kerekített tantárgyi átlagait láthatjuk félévekre lebontva, mellette a megfelelő tanuló nevével. A tanulói változatban pedig a bejelentkezett tanuló tantárgyi átlagait az adott tantárgy nevével ellátva láthatjuk félévekre lebontva.

A két képernyő valamelyikén egy elemet kiválasztva az 5.28-as ábrán látható képernyőre jutunk, ahol a kiválasztott tanuló választott félévhez és tantárgyhoz tartozó jegyeinek részletes listáját láthatjuk. Itt az egyes rövidítések a jegy típusára utalnak, mint például „d” a dolgozat és „td” pedig a témazáró dolgozat rövidítése.

A tanulói klienshez tartozó menüben a „Dolgozatok” lehetőséget választva a tanuló tanárai által rögzített tervezett dolgozatai láthatóak, amire egy példát az 5.29-es ábra tartalmaz. Ez a lehetőség a tanári kliensben korábban is létezett, azonban a diákok nem láthatták a felvett értékeket, így valójában értelmetlen volt használni.



5.29. ábra: Tanuló betervezett dolgozatai



5.30. ábra: Osztály és a hetesek megtekintése

Az osztályfőnökök vagy helyetteseik a menüben az „Osztályom” lehetőséget választva az 5.30-as ábra képernyőjére jutnak, ahol beállíthatják és menthetik a heteseket. A tanulók számára szintén elérhető ez az opció, viszont ők nem tudják módosítani a hetest. Az adatbázisnak ez is egy olyan funkciója volt, amely már eleve létezett, viszont a diákok számára nem volt megtekinthető, így nem használták.

Végül, de nem utolsósorban a „Kijelentkezés” menüpontot kiválasztva az alkalmazás kijelentkezteti a felhasználót és a bejelentkező oldalra dobja. A háttérben itt megtörténik a JWT törlése a kliensoldal lokális adatbázisból.

5.3 Tesztelés

Tesztelés gyanánt az úgynevezett User Acceptance Test (UAT) módszert választottam. Ennek a tesztelési módszernek az alkalmazása során a szoftvert a kiadása előtt a készítő teszteltetik jövőbeli potenciális felhasználókkal, és a visszajelzések alapján még a kiadás előtt módosíthatnak a terméken. Az UAT általában manuálisan történik, a felhasználók elkezdik használni az alkalmazást, és tesztelik, hogy hogyan reagál a szoftver. A tesztesetek forgatókönyvei automatizálhatók is, szimulálva a felhasználói élményt.[26]

Az UAT manuális típusú tesztelését választottam. Az alkalmazások készülése közben folyamatosan, minden egyes funkció helyes működését ellenőriztem, és az alkalmazás elkészülte után egészben is többször megnéztem a működést.

Mivel én már nem tudtam „friss szemmel” nézni rá, így környezetemet is megkértem, hogy próbálják ki az alkalmazást. Ennek során volt is két nagyobb hiba, amit feltártak, valamint érkezett néhány apróbb módosítási javaslat. Ilyen módosítás volt például, hogy azoknál a felsoroló oldalaknál, ahol az egyes elemekhez tartozó nyilakra nyomva részletező képernyőre lehet jutni, ott elég legyen az elemre nyomni az átlépéshez. Az egyik feltárt hiba akkor merült fel, amikor megkezdtünk egy tanórát regisztrálni, átléptünk egy másik képernyőre, majd innen vissza. Ekkor az alkalmazás megpróbálta újratölteni az adatokat, de nem kapta meg, hogy mely tanórához tartoznak, így az összeomlott, és leállt. A másik nagy hibát feljebb részleteztem. [5.1.4]

5.4 Felhasználói élmény

A felhasználói élmény (User Experience) növelése érdekében igyekeztem minden felhasználói interakció hatására valamilyen eseményt kiváltani, és éjszakai módot is készítettem. Ha valaminek a betöltésére vár az alkalmazás, akkor egy töltődés ikon jelenik meg. Adatrögzítést végezve az oldal alján felugrik egy szöveges üzenet, ami kiírja annak eredményét. Amennyiben pedig nincs elérhető adat, akkor az adatok helyén egy ezt jelző felirat jelenik meg. Ha internetkapcsolati vagy más jellegű hiba lép fel, azt a képernyő alján jelzi az alkalmazás. A főbb hibákat külön részletezi, ilyen az internethiba, a lejárt munkamenet, vagy ha a szerver túl hosszú ideig nem válaszolt. A többi hibát pedig hibakóddal együtt jeleníti meg. Hiba esetén a legtöbb esetben az üzenet mellett egy „Újra” felirat is megjelenik, amire rányomva megkísérélhető a kérés újraküldése.

6 Összegzés

5.5 Értékelés

A szakdolgozatom célja az volt, hogy volt középiskolám tanárai és diákjai számára elkészítsem a jelenlegi e-napló rendszer Android mobil klienseinek prototípusát, ezáltal kényelmesebbé és gördülékenyebbé tegyem annak használatát, és kiterjesszem funkcionalitását.

A kliensek és a webszerver sikeresen elkészültek. Miután a webszervert az iskola saját szerverére feltelepítjük és hozzákötjük a valódi adatbázishoz, azután az applikációkat ténylegesen üzembehelyezhetjük majd. Ekkor látható lesz, hogy az alkalmazások jelenlegi funkciói mennyire felelnek meg a tényleges igényeknek, és a beérkező javaslatok alapján módosíthatóak és bővíthetőek lesznek.

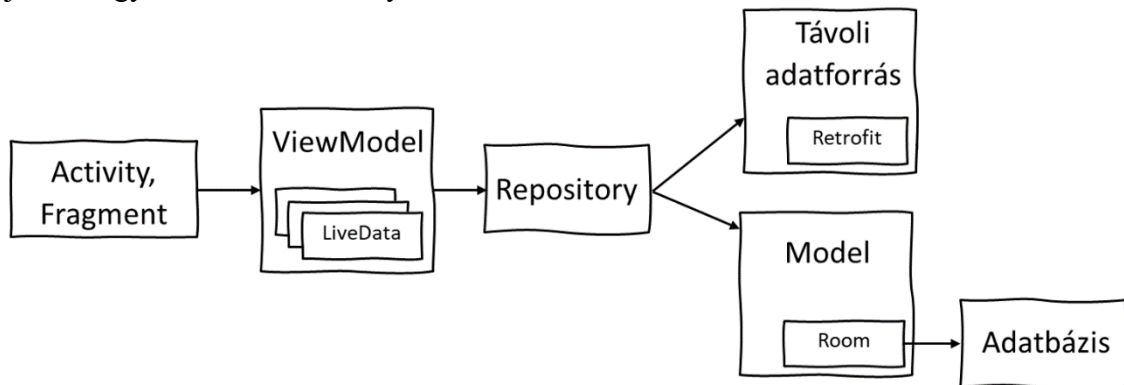
A projekt tervezése és elkészítése során a legfontosabb szempont az volt, hogy átlátható és könnyen módosítható legyen, hiszen várhatóan még sok további fejlesztéssel fogom bővíteni a jövőben. Emiatt többek között tervezési minták alkalmazásával próbáltam minél modularizáltabbá tenni a kódomat. Az összetartozó funkcionalitásokat közös csomagokba helyeztem, az egyes modulok között laza csatolás kialakítására törekedtem. A készített rendszer backendből és frontendből is állt, így megismerkedtem a .NET keretrendszerrel, mélyítettem az Android tudásomat, és a technológiák és tervezési minták keresgélése során rengeteg új ismeretre tettem szert. Egyaránt láthattam bele az aszinkron működő webszerverek világába, és a frontend fejlesztés sokszor nagyon időigényes folyamataiba is. A fejlesztés során elég sokszor futottam bele megoldhatatlannak tűnő problémákba. Ilyenkor órákig tartó kutakodás és vesződések után végül mindig ráleltem a megoldásra, és úgy érzem, hogy ezekkel tanultam a legtöbbet.

A szakdolgozatom elkészítése hatalmas munkának bizonyult, amivel mindeközben rengeteget tanultam is. Mivel végig az a cél lebegett a szemem előtt, hogy az iskola tényleg használni tudja majd a munkámat, ezért végig motivált voltam és sok örömet leltem a fejlesztésben.

5.6 Továbbfejlesztési lehetőségek

5.6.1 Bővítési lehetőségek

Az alkalmazások jelenleg a JSON Web Token kivételével nem tárolnak adatot, hanem minden képernyőre lépés alkalmával újra lekérnek mindent a webszervertől, így internetkapcsolat nélkül nem használhatóak. Ezt javítani lehetne, ha a letöltött adatokat eltárolnánk a kliensoldalon egy lokális adatbázisban. A lapra lépve először mindig innen tölthetnénk be az adatok, és ezzel párhuzamosan elindulna egy kérés a webszerver felé. Amikor a kérés visszaérkezne, akkor az új adatok alapján megtörténne az oldal frissítése. Így el lehetne érni, hogy az alkalmazás internet nélkül is használható legyen, illetve jelentős gyorsítást is eredményezne.



6.1. ábra: Kliensoldal továbbfejlesztett architektúrája

Ehhez a kliensoldal 4.2-es ábrán látható architektúráját ki kellene egészíteni azzal, hogy a Repository ne csak a webszerverhez, hanem a lokális adatbázishoz is forduljon adatokért. A módosított felépítést az 6.1-es ábra hivatott szemléltetni.

5.6.2 Kiterjesztés más platformokra

Az alkalmazások jelenleg csak Androiddal kompatibilisek. Mivel a piacon más mobil operációs rendszerek is elérhetőek, így a választott iskolában jelenleg nem fogja tudni mindenki használni a mobil klienseket. A jövőben mindenképpen szeretném megoldani az iOS-sel való kompatibilitást, mert ez a két operációsrendszer egy 2022 októberi kutatás szerint lefedi az európai mobilpiac 99,5%-át.[27]

5.6.3 További alkalmazási területek

Mivel az alkalmazás funkciói az adatbázis képességein alapulnak, így ez eléggé megköti a szoftver jelenlegi kiterjeszthetőségét. Ilyen probléma például, hogy a diákok

nem tudnak üzenetet küldeni, vagy hogy ha a diákok megnyitnak egy üzenetet, akkor ennek ténye az adatbázisban nem kerül rögzítésre (így nem lehet egységes „látta” értéket megjeleníteni az alkalmazásban minden típusú üzenetnél). Ameddig az ilyen jellegű hiányosságok nem lesznek az adatbázisban, és így a mobilkliensben is javítva, addig elég nehézkesnek gondolom a kiterjeszthetőséget.

Ha a jövőben az iskola részéről kérést kapok arra, hogy további funkciókkal bővítsen, és ehhez hajlandóak az adatbázist is módosítani, akkor ez könnyedén megtehető. Ilyennek látom például a menza befizetését, a kollégiummal kapcsolatos adminisztrációt, délutáni sporttevékenységeket, iskolaszintű hirdetések közzétételét vagy az értesítések küldését a telefonra.

Az iskola fenntartójának van még további 4 intézménye (3 gimnázium és egy középiskola), amelyek ugyanezt az e-naplót használják, ugyanilyen adatbázissal. Ha a mobilkliensek helytállnak majd az iskolában, akkor be szeretnénk vezetni a fenntartó többi iskolájában is. Ez csak nagyon minimális módosítással járna, és így potenciálisan több, mint kétezer ember (mínusz az iOS felhasználók) tudná majd használni.

Irodalomjegyzék

- [1] *Kréta Mobil applikációk (hozzáférés ideje: 2022.11.10):*
<https://tudasbazis.ekreta.hu/pages/viewpage.action?pageId=4065001>
- [2] *Free prototyping tool for web & mobile apps - Justinmind (hozzáférés ideje 2022.11.30):* <https://www.justinmind.com/>
- [3] *What is .NET? (hozzáférés ideje 2022.11.20):* <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet>
- [4] *Github Dotnet-bot (hozzáférés ideje 2022.11.20):* <https://github.com/dotnet-bot>
- [5] *Entity Framework (hozzáférés ideje 2022.11.21):* <https://learn.microsoft.com/en-us/aspnet/entity-framework>
- [6] *Database-First(Schema First) Approach in Entity Framework (hozzáférés ideje 2022.11.21):* <https://www.javatpoint.com/database-first-approach-in-entity-framework>
- [7] *JSON Web Tokens (hozzáférés ideje 2022.11.21):* <https://jwt.io/>
- [8] *About Swagger (hozzáférés ideje: 2022.11.20):* <https://swagger.io/about/>
- [9] *Introduction to Android Development (hozzáférés ideje: 2022.11.17):*
<https://www.geeksforgeeks.org/introduction-to-android-development/>
- [10] *Discover the History of Kotlin (hozzáférés ideje: 2022.11.20):*
<https://openclassrooms.com/en/courses/5774406-learn-kotlin/5930526-discover-the-history-of-kotlin>
- [11] *Google Play Store Statistics (hozzáférés ideje 2022.11.20):*
<https://www.appventurez.com/blog/google-play-store-statisticshttps://www.appventurez.com/blog/google-play-store-statistics>
- [12] *Getting Started with Retrofit 2 (hozzáférés ideje 2022.11.24):*
<https://howtodoinjava.com/retrofit2/retrofit2-beginner-tutorial/>
- [13] *Android Preferences Example (hozzáférés ideje 2022.11.22):*
<https://www.javatpoint.com/android-preferences-example>
- [14] *What is Azure? (hozzáférés ideje: 2022.11.20):*
<https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/>
- [15] *Design Pattern – Overview (hozzáférés ideje 2022.11.23):*
https://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm
- [16] *Microsoft Net Framework Logo (hozzáférés ideje 2022.11.23):*
<https://rwandi.blogspot.com/2020/07/microsoft-net-framework-logo.html>

- [17] *Microsoft Azure Logo (hozzáférés ideje 2022.11.23):* <https://logos-marques.com/microsoft-azure-logo/>
- [18] *Android-logo (hozzáférés ideje: 2022.11.23):* <https://myadtech.mx/android-logo/>
- [19] *Android Login/Signup with MVVM - YouTube (hozzáférés ideje 2022.10.25):* <https://www.youtube.com/playlist?list=PLk7v1Z2rk4hgmIvyw8rvpiEQxIAbJvDAF>
- [20] *ASP.NET MVC Pattern | .NET (hozzáférés ideje 2022.11.27):* <https://dotnet.microsoft.com/en-us/apps/aspnet/mvc>
- [21] *ASP.NET MVC 5 (hozzáférés ideje 2022.11.27):* <https://www.dotnetcurry.com/aspnet-mvc/1155/aspnet-mvc-repository-pattern-perform-database-operations>
- [22] *Script to Drop All Orphaned SQL Server Database Users (hozzáférés ideje 2022.11.24):* <https://www.mssqltips.com/sqlservertip/3439/script-to-drop-all-orphaned-sql-server-database-users/>
- [23] *RecyclerView .addTextChangedListener gives multiple positions - Stack Overflow (hozzáférés ideje 2022.11.20):* <https://stackoverflow.com/questions/54234730/recyclerview-addtextchangedlistener-gives-multiple-positions>
- [24] *How to Observe Event Only Once Using SingleLiveEvent in Android? - GeeksforGeeks (hozzáférés ideje 2022.11.25):* <https://www.geeksforgeeks.org/how-to-observe-event-only-once-using-singleliveevent-in-android/>
- [25] *SingleLiveEvent class in kotlin (github.com) (hozzáférés ideje 2022.11.25):* <https://gist.github.com/JoaoGeniselli/220ca15fb99f614c4e06d6643d41e05a>
- [26] *What is UAT? Breaking Down User Acceptance Testing (hozzáférés ideje 2022.11.23):* <https://www.codecademy.com/resources/blog/what-is-user-acceptance-testing/>
- [27] *Mobile Operating System Market Share Europe | Statcounter Global Stats (hozzáférés ideje 2022.11.28):* <https://gs.statcounter.com/os-market-share/mobile/europe/>