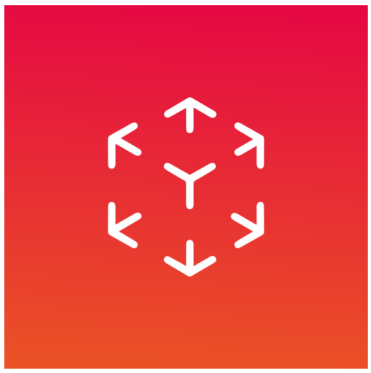
XXIII. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2020. május 21–24.

ARmore

Dinamikus kiterjesztett valóság platform és mobilalkalmazás interaktív kiadványokhoz



Szerző:

Orbán Erik

Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, II. év

Témavezetők:

dr. Simon Károly, egyetemi adjunktus

Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar

Bartus Levente, szoftverfejlesztő

Codespring

Kivonat

Az ARmore egy kiterjesztett valóságon (Augmented Reality, AR) alapuló platform és mobilalkalmazás, amelynek segítségével digitális tartalmakat lehet hozzárendelni nyomtatott kiadványokhoz. Ezek a tartalmak lehetnek hang- és videóanyagok, képek, vagy 3D-s modellek. A megjelenítés a mobil alkalmazás kameranézetében történik: ha az adott kiadványra irányítjuk telefonunkat, akkor a kiterjesztett valóságban „megelevenednek” a lapok statikus tartalmai.

A rendszer teljesen általános és felhasználóbarát: a kiadványok szerkesztői egy webes felületen keresztül könnyen kezelhetik a tartalmaikat, ehhez semmilyen programozási tudásra nincs szükségük. Egyszerűen egy referencia képet (a kiadvány adott oldalának egy részletét) kell feltölteniük minden általuk megadott AR tartalom mellé. Ezeket a képeket az ARmore alkalmazás a kamerán keresztül felismeri, és megjeleníti a hozzájuk rendelt tartalmakat. A platform dinamikusan működik: a webes felületen keresztül feltöltött tartalmak azonnal elérhetővé válnak a mobil alkalmazáson belül, ehhez nem szükséges az alkalmazás frissítése vagy újratelepítése.

Az ARmore segítségével létrehozott első interaktív kiadvány a Székelyudvarhely Polgármesteri Hivatala által az 1989-es rendszerváltás évfordulóján kiadott „30 éve szabadON” fényképalbum, és hamarosan további kiadványokkal bővül a lista.

Tartalomjegyzék

Bevezető **1**

1. Az ARmore projekt **3**

1.1. Funkcionalitások . . . 3

1.1.1. A szerver funkcionalitásai . . . 3

1.1.2. A weboldal funkcionalitásai . . . 3

1.1.3. A mobil alkalmazás funkcionalitásai . . . 4

1.2. Architektúra . . . 4

2. A szerver **6**

2.1. Adatmodell . . . 6

2.2. Adathozzáférési réteg . . . 6

2.3. Szolgáltatási réteg . . . 7

2.4. API . . . 7

2.5. Biztonság . . . 7

3. A weboldal **9**

3.1. Technológia . . . 9

3.2. Megvalósítás . . . 9

3.3. Biztonság . . . 9

4. A mobilalkalmazás **11**

4.1. Technológia . . . 11

4.2. Adathozzáférési réteg . . . 12

4.3. Szolgáltatási réteg . . . 13

4.4. AR tartalmak . . . 14

4.4.1. Képek megjelenítése . . . 15

4.4.2. Videók lejátszása . . . 16

4.4.3. Audiófájlok lejátszása . . . 17

4.4.4. 3D modellek felépítése . . . 18

4.5. Felhasználói felület . . . 19

5. Eszközök és módszerek **21**

6. Az ARmore működése **22**

7. Következtetések és továbbfejlesztési lehetőségek **23**

Bevezető

A kiterjesztett valóság (Augmented Reality, AR) olyan technológia, amelynek segítségével a körülöttünk lévő fizikai tárgyakhoz digitális kiegészítő tartalmakat rendelhetünk hozzá, és ezeket a megfelelő kijelzőkön az eredeti fizikai környezet elemeivel kombinálva jeleníthetjük meg.

Ezt a technológiát használja fel az ARmore alkalmazás, amelynek célja a nyomtatott kiadványok kiterjesztése digitális tartalmakkal. Fontos szempont volt a fiatalok körében népszerűsíteni a hagyományos nyomtatott kiadványok olvasását, vonzóbbá téve azokat a hozzájuk rendelt interaktív vizualizációkkal.

Alapvető célkitűzés volt, hogy a kiadványokhoz rendelt digitális tartalmak dinamikusan szerkeszthetőek és betölthetőek legyenek, ne kelljen minden módosítás után új verziót kiadni az alkalmazásból. Egy olyan környezet biztosítása volt a cél, amelyben a tartalomszerkesztők haladó programozási ismeretek nélkül is képesek az AR elemek kezelésére, úgy, hogy a fejlesztők beavatkozása ne legyen szükséges ehhez.

A platform koncepciója bizonyos értelemben hasonlít például a YouTube működéséhez: a szerkesztők kiadványokat publikálhatnak, dinamikusan változtatható AR tartalmakkal. Az olvasók böngészhetnek a publikációk között, és feliratkozhatnak ezekre. Ezt követően a mobileszközön keresztül hozzáférhetővé válnak számukra a kiadványokhoz társított AR tartalmak. A tartalomváltozások is azonnal elérhetőek az alkalmazáson belül, ehhez nem kell frissíteni vagy újratelepíteni az applikációt.

A dolgozat bemutatja az ARmore projektet, a fejlesztés során felhasznált technológiákat és eszközöket, részletesen ismertetve a rendszer megvalósítását és működését. Az első fejezet alrendszerekre lebontva foglalja össze a funkcionalitásokat, és ismerteti a rendszer architektúráját, amelynek megtervezésekor fontos szempont volt a dinamikus működés. A következő fejezetek az egyes alrendszereket ismertetik részletesebben.

A második fejezet a backend rész működéséről és technológiai hátteréről szól, alfejezetekre bontva mutatja be annak egyes részeit: az adatmodellt, az adathozzáférési réteget, a szolgáltatási réteget, illetve az API réteget. Ismerteti az alkalmazáslogika és kommunikáció megvalósítását, illetve az implementált biztonsági megoldásokat.

A harmadik fejezet a webes frontend részt mutatja be, amely a moderátor szerepkörben lévő felhasználóknak ad lehetőséget új tartalmak feltöltésére, illetve a jelenlegi tartalmak módosítására vagy törlésére. A felület a kiadványok szerkesztőinek készült, akik egy regisztrált azonosítóval tudnak belépni. A fejezet ismerteti a felhasznált webes technológiákat, a webes felület megvalósításának részleteit, valamint az alkalmazott biztonsági megoldásokat.

A negyedik fejezet a mobilalkalmazást ismerteti, amely talán a projekt leglényegesebb

részének tekinthető, lévén, hogy ezen keresztül jelennek meg a kiterjesztett valóság részeként a feltöltött tartalmak. Fejlesztés szempontjából is ez a rész jelentette a legnagyobb kihívást, mivel a jelenlegi AR technológiáknak még nagyon sok limitációja van. Több egyedi megoldással kellett kiegészíteni a rendelkezésre álló keretrendszert a funkcionalitások megfelelő megvalósításához. A fejezet bemutatja az alkalmazás fejlesztéséhez használt technológiákat, és külön alfejezetekben tárgyalja a jelenleg támogatott tartalom típusokat, valamint a felhasználói felület tervezésénél figyelembe vett UX design alapelveket.

Az ötödik fejezet az alkalmazás működését mutatja be, ezután a fejlesztés során felhasznált módszerekről és eszközökről esik szó röviden, végül a következtetések és továbbfejlesztési lehetőségek összefoglalásával zárul a dolgozat.

A projekt 2019 nyarán indult a U-Hub Mentorprogram keretein belül, a Codespring szakmai támogatásával. Három hónap fejlesztés után első prototípusa bemutatásra került a Digitális Székelyföld Konferencián. A fejlesztés folytatódott és decemberben megjelent az első kiadvány, ami felhasználja az ARmore alkalmazás adta lehetőségeket, a Székelyudvarhely Polgármesteri Hivatala által az 1989-es rendszerváltás évfordulóján kiadott „30 éve szabadON” fényképalbum. Ekkor került ki az applikáció Android verziója a Google Playstore-ba, majd március elejére sikerült kiadni az App Store-on keresztül az iOS verziót. A továbbfejlesztés jelenleg is folyamatban van, és a támogatott kiadványok listája is hamarosan bővülni fog.

1. Az ARmore projekt

Az ARmore rendszer három részből tevődik össze: egy szerver (backend), egy webes kezelői felület és egy mobilalkalmazás alkotja.

1.1. Funkcionalitások

Minden alrendszernek specifikus funkcionalitásai vannak, amelyek egymástól függnek, így az alrendszerek interfészeken keresztül kommunikálnak egymással. Az architektúra és kommunikációs felület kialakításánál fontos szempont volt, hogy az egyes alrendszerek módosítása ne legyen hatással a többi komponensre.

1.1.1. A szerver funkcionalitásai

A szerver tartja a kapcsolatot a másik két alrendszerrel. Feladata a mobilalkalmazás és a weboldal felől érkező kérések fogadása, feldolgozása és kiszolgálása. A kérések párhuzamosan kell feldolgozásra kerüljenek, ezáltal gyorsítva a folyamatot.

A mobilalkalmazás felől érkező kérések csak és kizárólag adatok lekérdezésére vonatkoznak: egy adott publikáció leírásának a lekérdezése, vagy az adott kiadványhoz tartozó digitális tartalom letöltése stb.

A weboldalon keresztül érkező kérések már nem csak lekérdezések lehetnek, hanem új tartalom feltöltésére vagy létező tartalom módosítására is vonatkozhatnak. A feltöltött adatok többnyire állományok formájában érkeznek, és a szervernek a feladata az, hogy ezeket megfelelően tárolja és hozzárendelje az adott kiadványokhoz.

Összefoglalva, a szerver egy adattároló és kiszolgáló szerepet tölt be, alapfunkcionalitása a média fájlok és más adatok fogadása, tárolása és kezelése, illetve a lekérdezések kiszolgálása.

1.1.2. A weboldal funkcionalitásai

A weboldal a tartalomszerkesztők számára biztosít funkcionalitásokat, innen érkeznek a szerver felé az adatok feltöltésére vagy módosítására vonatkozó kérések. Az oldal fejlesztésekor hangsúlyos szerepet kaptak a UX alapelvek, ezért letisztult és egyszerűen használható a felület.

A szerkesztők létrehozhatnak kiadványokat és ezekhez AR tartalmakat rendelhetnek. Utóbbi művelet esetében meg kell adniuk referencia képeket (oldalak vagy oldalrészek a kiadványból), amelyeket majd felismer a mobilalkalmazás az AR elemek megjelenítésekor. Ezekhez a referencia képekhez társulnak a hozzájuk rendelt AR elemek, amelyek lehetnek videók, hanganyagok, képek vagy 3D modellek. Mindezeknek az adatoknak a feltöltése vagy

módosítása nagyon egyszerűen, néhány gombnyomással megtörténhet a felületen, mindenféle programozási tudás nélkül.

1.1.3. A mobil alkalmazás funkcionalitásai

A mobilalkalmazás feladata a szervertől lekérni az adatokat és azokat megjeleníteni a felhasználó számára. Ezeknek az adatoknak egy részét az alkalmazás a mobileszközön is eltárolja. Ezt a folyamatot nevezzük feliratkozásnak: az applikáció megjegyzi azokat a kiadványokat, amelyeknek a tartalmát letöltöttük, és jelez, ha módosítás történt az adott kiadványnál.

Az AR tartalmak a mobilalkalmazás kameranézetében jelennek meg a kiterjesztett valóság részeként. A kiadványokban megjelölt referencia képeket az alkalmazás futás közben felismeri és lecseréli a megadott tartalomra. A statikus oldalakon így videók és három dimenziós objektumok jelennek meg, vagy hanganyagok lesznek automatikusan lejátszodva a kiadvány kameranézeten keresztüli böngészése során.

A videóanyagok lejátszása kétféleképpen történhet: a telefon lokális fájlrendszeréből betöltve a videót, vagy távoli forrásból történő stream-elés által. Utóbbi természetesen állandó internetkapcsolatot igényel, és megnövekedett adatforgalommal jár.

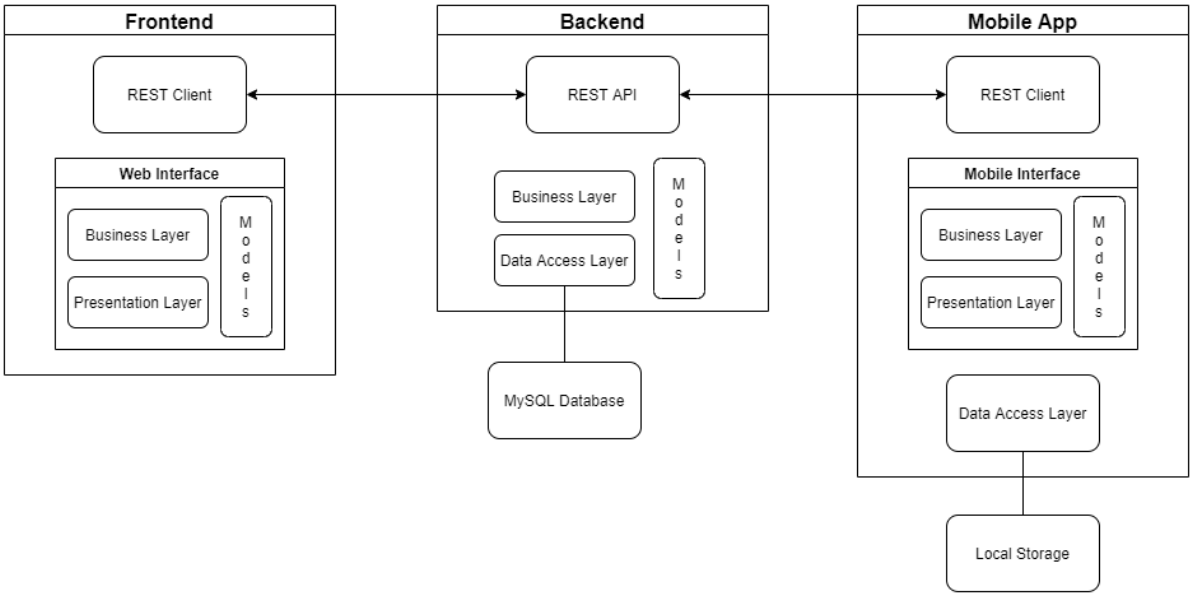
1.2. Architektúra

A szerver, a mobilapplikáció és a weboldal közötti kommunikáció REST hívásokon alapszik, amelyek kiszolgálásáért a backend felel. A backend többrétegű architektúrára épül. A Controllers modul felel a hívások fogadásáért. A HTTPS protokollon alapuló kommunikáció során az adatok JSON formátumban érkeznek, majd DTO (Data Transfer Object) objektumok épülnek fel ezekből. A DTO-k szerver oldali modell objektumokba történő átalakítása után az adatok továbbítódnak a Service modulnak, ahol a központi alkalmazáslogika implementációja található.

A Service modul végzi el a szükséges műveleteket, majd azok eredményét a Controllers modulon keresztül továbbítja a kliensnek.

A webes adminisztrációs felületre jellemző az oldalankénti szeparáció, minden oldalért más és más vezérlő és nézet felel. A közös részek, amelyek minden oldal számára szükségesek, a Services, Helpers és Guards modulokba kerültek.

A mobilapplikációban a Handlers modul a felhasználói interakciókat figyeli, a gombok és linkek lenyomását, illetve a képernyőérintéseket kezeli. A Models modulban találhatóak az adatok reprezentációjához szükséges osztályok, míg a Controllers modul felel a műveletek végrehajtásáért. A mobilalkalmazásban található a rendszer legösszetettebb modulja is, amely



1. ábra. Az alkalmazás vázlatos architektúrája.

az AR tartalmakat kezeli. Ez a modul saját belső Controllers és Visualizers almodulokkal rendelkezik, amelyek feladata, az AR tartalmak megjelenítése és azok feldolgozása.

2. A szerver

Az adattárolást és a kliensalkalmazásoktól érkező kérések kiszolgálását végző szerver C#-ban van megírva, .NET Core [1] keretrendszer felhasználásával. A fejezetben az architekturális részei és az implementálás során felhasznált technológiák lesznek bemutatva.

2.1. Adatmodell

A rendszerhez tartozó adatmodellek a Models modulba kerültek. Ezeknek az osztályoknak nincsenek az adattagokon és hozzájuk tartozó publikus getter és setter metódusokon kívül más részei. A `ContentModel` implementálja az `IUpdateable` interfészt, mert az adatmodell tárolja az adatok legutolsó módosításának az időbélyegét, amelyet a rendszer automatikusan frissít. A modellek attribútumain annotációk találhatóak, amelyek az adathozzáférési rétegnek adnak információkat az objektum-relációs leképezésről. Ilyen jellegű annotáció például a `Key`, amely jelzi egy attribútomról, hogy az lesz az elsődleges kulcsnak megfelelő azonosító, valamint az osztályok szintjén használt `Table`, amely az osztálynak megfelelő adatbázis tábla nevének explicit meghatározását teszi lehetővé. Ezt a megközelítést nevezzük Data Annotations módszernek [7] az Entity Framework ORM keretrendszer szövegkörnyezetében.

2.2. Adathozzáférési réteg

Az adathozzáférési rétegben történik az adatbázissal való kapcsolattartás, az adatbázisműveletek végrehajtása, a lekérdezések küldése és fogadása. Az ARmore szerver MySQL adatbázist használ, és ezzel a kommunikációt az Entity Framework [21] perzisztencia keretrendszer biztosítja. Az Entity Framework egy objektum-relációs leképezést támogató technológia, amely a .NET környezet részét képzi, és mindenféle relációs adatbázissal kapcsolatos művelet elvégzésére alkalmas. Ez a keretrendszer építi fel a táblákat, lekérdezéseket, módosításokat hajt végre. Használatával egyszerűen megoldható az adatok perzisztenciája.

A `DataContext` osztály írja le az adatbázis felépítését. Ennek az osztálynak az `OnModelCreating` függvénye kiegészíti az adatmodelleken található annotációkat, mivel nem minden adatbázisra jellemző tulajdonság írható le egyszerű annotációval. Például az egy-a-többhöz kapcsolatok beállítása is itt történik. Fluent API [10] tervezési mintát használva, a Data Annotations módszerrel kombinálva építi fel az adatbázisra jellemző összes tulajdonságot.

A rendszeren belül szükség van olyan bináris adatok tárolására is, amelyre nem biztosítanak megfelelően hatékony megoldást a relációs adatbázisok. Ezek az állományok a fájlrendszerben

vannak tárolva, az adatbázisban csak elérési útvonalaikat, neveiket, típusukat és néhány további tulajdonságukat tároljuk.

Mivel a fejlesztés során a hibák előfordulásának minimalizálása rengeteg teszteléssel járt együtt, ezért egy másodlagos indítási opció is implementálva van. A szerver indításakor megadhajuk parancssori argumentumként azt az opciót, hogy tesztadatokkal töltse fel az adatbázist. Ekkor az Entity Framework nem a MySQL szerverhez csatlakozik, ahol a valódi, perzisztens adatok találhatóak, hanem a memóriában épít fel egy adatbázist és ezt tölti fel tesztadatokkal. A lokálisan tárolt bináris fájloknak a védelmére is figyeltünk, ilyen esetekben ezek egy új temporális mappába kerülnek. Ezeket a tesztadatokat JSON formátumú fájlokban tároljuk, a hozzájuk tartozó bináris fájlokkal egy helyen.

2.3. Szolgáltatási réteg

A szolgáltatási réteg tartalmazza az alkalmazás központi logikáját. A beérkező kéréseket ellemzi, vizsgálja helyességüket, majd továbbítja azokat a Services modulban található megfelelő komponenseknek. Jelenleg négy ilyen komponens van a rendszerben: a `BinaryDataService`, a `ContentService`, a `DomainService` és az `IdentityService`.

A `BinaryDataService` osztály felel a beérkező bináris fájlok létrehozásáért és módosításáért, valamint a meglévő fájlok visszatérítéséért. A `ContentService` végzi a tartalmakkal kapcsolatos adatmódosításokat. Dependency Injection módszert [8] alkalmazva futás közben kap hozzáférést az adatbázishoz, és így el tudja végezni a szükséges módosításokat. A `DomainService` működése hasonló a `ContentService`-hez, itt a tartalmakat összegyűjtő kiadványok módosítására érkező kérések feldolgozása történik. Az `IdentityService` a weboldal felől érkező bejelentkezési kérések feldolgozásáért felel.

2.4. API

A szerver a kérések fogadására és azok kiszolgálására RESTful webszolgáltatásokat használ. A REST egy olyan kommunikációs modell, amely HTTP protokollon alapuló, állapot nélküli kliens-szerver kommunikációt ír le. Ezt alkalmazza a rendszer mind a weboldalról, mind pedig a mobileszközökről érkező kérések fogadására és a válaszok küldésére: GET, POST, PUT, DELETE típusú kérések kezelését valósítja meg, a megfelelő URL használata mellett.

2.5. Biztonság

A biztonságért a rendszerben az `IdentityService` felel. Jelenlegi verziójában háromféle felhasználó típust tud megkülönböztetni: vendég, adminisztrátor és tartalomszerkesztő. Ezeknek a felhasználóknak különböző hozzáférési jogaik vannak a rendszerhez, ezek

ellenőrizve vannak a REST hívások kiszolgálása során. Amennyiben egy olyan kérést küld az adott felhasználó, amelyhez nincs joga, a szerver nem fogja azt végrehajtani, hanem egy `401 Unauthorized` üzenetet fog visszatéríteni. Bejelentkezés után a kliens egy megadott ideig érvényes JWT[12, 14] tokent kap, ennek lejárta után új bejelentkezésre van szükség.

Jelenleg a tartalomszerkesztő rendelkezik a legmagasabb jogosultsági szinttel, lehetősége van a feltöltött tartalmak módosítására, törlésére és új tartalmak hozzáadására is. Az adminisztrátor jogköre a tartalomszerkesztő felhasználók kezelése. A vendégeknek már nincsen jogosultságuk az adatok módosítására, törlésére vagy hozzáadására, csak azok megtekintésére.

3. A weboldal

A weboldal a tartalomszerkesztők számára készült, TypeScript használatával, Angular keretrendszer [13] segítségével.

3.1. Technológia

Az Angular keretrendszer lehetővé tette a REST kérések küldését, fogadását, a biztonsági feltételek implementálását, valamint megadta az SPA (Single Page Application) [16] alkalmazásunk kereteit. Az SPA egy olyan dinamikus megközelítése a weboldalfejlesztésnek, ahol az oldalak tartalmát a böngésző futás közben cseréli. Ez a megközelítés a felhasználók számára egy sokkal jobb felhasználói élményt nyújt, menüpontváltásoknál nem szükséges a teljes oldal újratöltése, hanem csak a megfelelő elemek cserélődnek le. A módszer a tervezhetőséget is elősegíti. A különböző komponensek külön kis html fájlokba kerülhetnek, amelyek linkelhetőek egymásba egy azonosító megadásával. Ezáltal kiküszöbölhető a több ezer soros html fájlok karbantartásának szükségessége.

3.2. Megvalósítás

A megvalósításban fontos szerepet játszott az Angular keretrendszer, amely a komponensek felépítését lehetővé tette. Az oldal három alaprészből tevődik össze: a fejlécből, azaz a navigációs menüből, az oldal törzséből és a láblécből. Mindhárom komponens folyamatosan jelen van a weboldalon, tartalmuk viszont dinamikusan változhat.

A törzs komponens tartalma aszerint változik, hogy a felhasználó a navigációs menüben melyik elemre kattint. Ilyen elem például a publikációk listája, amely a törzsbe tölti be a jelenleg elérhető kiadványokat. Ha ezután a felhasználó rákattint valamelyik kiadványra, akkor betöltődik annak tartalma (a referencia képek és a hozzájuk rendelt AR elemek).

Annak érdekében, hogy az oldal még dinamikusabban működjön, szükség volt olyan TypeScript osztályok implementálására, amelyek lehetővé teszik a komponensek közti interakciókat. Ilyen például a `contentList` szervízosztály, amely a szervertől kapott tartalmakat jeleníti meg, futás közben a megfelelő számú elemet hozva létre az oldalon. Ennek az osztálynak a hatáskörébe tartozik az elemek szintjén történő módosítások kezelése is, az erre vonatkozó utasításokat REST hívások segítségével továbbítja a szervernek.

3.3. Biztonság

Az `authentication` szervízosztály és az `auth` őrzőosztály ellenőrzi az összes felhasználói interakciót a weboldalon. A szervízosztályon keresztül valósul meg a be- és kijelentkezés,

valamint a szervertől érkező token tárolása a böngésző lokális háttértárában. Ez a token hitelesíti a bejelentkezett felhasználót. A tokent a `JWT` interceptor osztály automatikusan beszúrja az összes REST hívásba.

A tokenen kívül az adott felhasználó szerepkörét is elmenti az alkalmazás. Ezt az örzőosztály ellenőrzi, amikor oldalt váltunk. Ha a felhasználó egy olyan oldalra kattint vagy egy olyan oldalnak adja meg az URL-jét, amelyhez nincs jogosultsága, az örzőosztály visszairányítja a kezdőoldalra.

4. A mobilalkalmazás

A mobilalkalmazás kódbázisa C#-ban íródott, .NET Standard 2.0 [20] keretrendszer használatával, amely magába foglalja a .NET Core 2.0, valamint a .NET Framework 4.6.1-es változatát. Ezeken kívül a Unity3D [17, 18] motor által biztosított keretrendszerek képezik az alkalmazás alapját.

Mivel a mobilalkalmazás központi elemét az AR technológia adja, mindenekelőtt ki kellett választani a megfelelő AR keretrendszert. Első körben a Google ARCore-ra [5] esett a választás. Ez a keretrendszer tette lehetővé azt, hogy a fizikai környezetről nyerjünk információkat képfeldolgozás által. Ilyen információ például egy objektum helyzete és mérete. A keretrendszer jól működött, de azt tapasztaltuk, hogy sok kép betöltése esetén lényegesen lelassul. Konkrétabban, 16 kép betöltésénél a várható idő 40 másodperc körüli volt, ami egy felhasználó számára már elfogadhatatlan lenne. A hatékonysági problémák miatt szükségesnek mutatkozott a folyamat optimalizálása.

Először azt próbáltuk elérni, hogy a képfeldolgozást optimalizáljuk, például az első végrehajtás után maradjanak eltárolva a szükséges adatok. Lényeges gyorsítást sikerült elérni a többszálúsítással is, a betöltés idejét így 15 másodpercre csökkentve.

Később azzal szembesültünk, hogy a Google ARCore keretrendszer iOS készülékeken nem képes futási időben feldolgozni a képeket. Megoldásként áttértünk az ARFoundation használatára. Ez a keretrendszer a Unity-n belül működik, a motor fejlesztői készítették, és nem egy saját képfeldolgozást valósít meg, hanem tulajdonképpen egy egységesítő réteget képez a Google ARCore és az Apple ARKit [9] felett. Ennek köszönhetően, mind az Android, mind az iOS operációs rendszert használó eszközökön megvalósítható volt az AR fejlesztés, azonos kódbázissal.

4.1. Technológia

Az AR funkcionalitások tehát az ARFoundation segítségével valósulnak meg [4], amely a háttérben a másik két keretrendszert használja. A platform kiválasztásánál dől el, hogy melyik AR technológia érvényesül: Androidos készülékek esetében a Google ARCore függvényei hívódnak meg, míg iOS készülékeken az Apple ARKit lesz a kiterjesztett valóság megalkotásáért a felelős.

Az ARFoundation egy nagyon fiatal technológia, a fejlesztési folyamat közben jelent meg a 3.0.0-preview változata, amelyet az alkalmazás jelenleg is használ. Ebben a verzióban már elérhető a képek futás közbeni feldolgozása, és bár nem végleges változat, tapasztalataink alapján elég stabilan működik. Sőt, bár a háttérben ARCore hívások történnek az Android-os eszközökön, a képek feldolgozásának ideje drasztikusan csökkent. Konkrétabban, a korábban

említett 15 másodperces betöltési idő, kevesebb mint 1 másodpercre redukálódott. Ez annak köszönhető, hogy az ARFoundation a háttérben más Unity által kiadott keretrendszereket is felhasznál a hatékonyság növelésére.

Elsősorban a Unity Job System [6], a Unity IL2CPP backend [11] és a Unity Burst Compiler [3] játszik fontos szerepet a teljesítmény növekedésében.

A Unity Job System lehetővé teszi a fejlesztők számára a többszálúsított alkalmazások egyszerűbb készítését. Ezt olyan módon valósítja meg, hogy kiküszöböli a különböző szálak létrehozása által előforduló hibalehetőségeket, és együttműködik a Unity más keretrendszereivel. Például korlátozza a létrehozott szálak számát, hogy ezáltal is csökkentse a szálak közötti ütközéseket az erőforrásokhoz való hozzáférésnél.

A másik két keretrendszer, a Unity IL2CPP backend és a Unity Burst Compiler a kódot optimalizálja, a C# kódbázisunkat átalakítja bytekóddá, majd azt egy erőteljesen optimalizált gépi koddá konvertálja a platform típusától függően.

Az alkalmazás készítése során még több Unity által biztosított komponens volt felhasználva, ilyenek például a képek, videók, audió fájlok feldolgozásáért felelős beépített osztályok. Fontos szerepet játszik továbbá a rendszer grafikai és fizikai modulja, ezek teszik lehetővé a kép renderelését és a digitális objektumok megfelelő elhelyezését a virtuális 3D-s térben.

A Unity3D párhuzamosan fut az alkalmazásunk által implementált függvényekkel. A kommunikáció eseményalapú, a függvények frame-enként hívódnak meg információkat szolgáltatva komponenseinknek az aktuális állapotról. Minden Unity által megjelenítendő, vagy eseményeire feliratkozó objektum leszármazottja a `MonoBehaviour` osztálynak. Amennyiben egy osztály nem tesz eleget ennek a követelménynek, a Unity nem tudja annak az osztálynak a példányait felhasználni.

4.2. Adathozzáférési réteg

Az adathozzáférési réteg feladata a kapcsolattartás a backend és a mobilapplikáció között. Ez a `Services` modul hatáskörébe tartozik. A kommunikáció HTTP kérések alapján történik, párhuzamosított formában. Az alkalmazás csak GET kéréseket intéz a szerver felé, az adatok módosítását nem teszi lehetővé a végfelhasználók számára. A lekért adatok egy részét eltárolja, egy részét viszont mindig újra lekéri a szervertől. Például a publikációk listája minden indításnál frissül, de az alkalmazás rögzíti, hogy a felhasználó melyik kiadványokra van feliratkozva. Ha egy felhasználó feliratkozik egy adott kiadványra, az azonosítóját elmenti egy konfigurációs állományba, amelyet a rendszer később figyelembe vesz minden egyes lekérdezésnél.

Jelenleg az adathozzáférési réteghez két konfigurációs állomány tartozik, amelyek egy-egy `JSON` objektumot írnak le. Az egyik ilyen objektum az `ApplicationConfig` osztály példánya,

amely a szerverről tárol adatokat. Ebben a fájlban van például a szerver IP címe, így cserélhető a kód módosítása nélkül. A másik konfigurációs fájl tartalmazza a felhasználóra jellemző adatokat. Ez az állomány teszi lehetővé például a feliratkozások rögzítését.

Amennyiben a felhasználó fel van iratkozva egy adott kiadványra, további lekérdezésekre lesz lehetősége. A kiadvány kiválasztásakor történik meg a feldolgozásra kerülő képek szerverről való letöltése, majd tárolása a lokális fájlrendszerben. A képek az azonosítójuk alapján vannak elmentve. Az azonosítók a kiadványtól függetlenek, biztosítva, hogy ne léphessenek fel ütközések több feliratkozás esetében sem.

4.3. Szolgáltatási réteg

Az alkalmazás két lehetőséget biztosít az AR részeként megjelenő médiatartalmak kezelésére: a lokális tárolóból történő, illetve a stream-elés általi lejátszást. Amennyiben a `Streaming` szolgáltatást használja a kliens, a feliratkozáskor a feldolgozandó referencia képeken kívül más adatok nem lesznek tárolva. Ha a lokális szolgáltatás aktív, akkor a képeken kívül a megjelenítendő hang-, videó-, képanyag és 3D modellek is tárolásra kerülnek. Ezeknek a fájloknak a mentése mind az azonosító alapján történik. A streaming szolgáltatás csak online működést tesz lehetővé, és ez az alapértelmezett Androidos készülékeken. A lokális szolgáltatás offline működésre is lehetőséget ad, ez az alapértelmezett iOS eszközökön. A különbség a videók lejátszásánál tapasztalt problémák következménye, és mivel ezekre még nem sikerült megfelelő megoldást találni, a jelenlegi verzióban a beállítás nem módosítható a felhasználó által.

A feliratkozás után elérhetővé válik a kiadványhoz tartozó kameranézet, lehetővé téve az AR tartalmak megjelenítését. Az AR réteg feladatkörébe tartozik a képfelismerő keretrendszer eseményeinek a kezelése: a referencia terület meghatározása; az új tartalmak megjelenítése; a már megjelenített tartalmak karbantartása, azaz frame-enkénti poziciófrissítése, méretezése; a szükségtelenné váló tartalmak törlése.

A keretrendszer nem tartja számon, hogy melyik referencia területnek történt már meg a feldolgozása, valamint azt sem, hogy milyen megjelenített tartalomról van szó. Ennek kezelésére volt implementálva az `ARTrackedImageController` osztály, amely egy szótár-alapú megoldást alkalmaz erre a feladatra. Fogadni tudja az újonnan érkező képeket, képes az információk tárolására, ezáltal követni tudja az aktív elemeket. Minden tartalomtípus különböző megjelenítővel rendelkezik, amelyet szintén a karbantartó osztály biztosít a képek számára.

4.4. AR tartalmak

A kiterjesztett valóság elemeinek kezeléséért több osztály felel, a különböző tartalomtípusoknak megfelelően. Ezeknek az alapját a `BaseVisualiserEntity` adja. Meghatározza a közös tulajdonságokat, valamint deklarálja a közös viselkedést biztosító metódusokat. Absztrakt osztály, de tartalmaz metódus implementációkat is, a nem tartalom-specifikus metódusok számára. Például, az aktiválás és deaktiválás folyamatát az alaposztály nem tudja implementálni, hiszen minden vizualizáló más és más objektumokat hoz létre, ezeknek a száma sem egyezik meg. A tartalom képre helyezését viszont implementálja: minden referencia kép esetében a maximális lefedettséget próbálja elérni.

Az osztály frame-enként ellenőrzi, hogy a kép még látható-e, és ha igen, akkor lementi annak időbélyegét. Amennyiben az időbélyeg és a jelenlegi idő között eltelt minimum két másodperc, meghívódik a deaktiválás.

A deaktiválás folyamata nem jelenti a tartalom memóriából való törlését. Csak a tartalomhoz tartozó objektumok válnak láthatatlanná, a funkciók pedig inaktívvá. Ha a kamerát újra ráviszi a felhasználó a képre, a tartalom lejátszása ugyanonnan fog folytatódni, ahol előzőleg deaktiválódott.

Az alkalmazás jelenlegi változatában, négyféle tartalom jeleníthető meg: kép-, videó-, hanganyag és háromdimenziós modell. Implementálásukért rendre az `ImageVisualizer`, a `VideoVisualizer`, az `AudioVisualizer` és a `ModelVisualizer` felel. Mindegyik megjelenítőhöz hozzá van rendelve egy kontroller osztály is, ezek felelnek a tartalmak interakcióiért.

4.4.1. Képek megjelenítése

(a) AR tartalom megjelenítése előtt (b) AR tartalom megjelenítése után

2. ábra. Képmegjelenítő komponens működése

A képek megjelenítésének implementációja volt a legegyszerűbb, a kép renderelésén kívül ehhez nem szükséges más művelet. Az ImageVisualizer az inicializálás során egy Texture2D típusú objektumot épít fel, amelyet a Unity3D motor biztosít számunkra. Az objektum felépítése után, a referencia kép alapján létrehozunk egy Quad objektumot, ami egy olyan háromdimenziós felület, amely csak egyik oldalról látható. Ennek mérete a referencia kép méretével lesz egyenlő, ráhelyeződik az eredeti képre, így teljes takarásba teszi azt. Renderelés során erre a felületre kerül a felépített Texture2D objektum, amely tartalmazza a megjelenítendő képet. Az alkalmazás aktuális változatában a JPEG és PNG formátum támogatott. Jelenleg a megjelenítés nyújtással történik: nem azonos képarány esetén a megjelenített kép valamely irányba megnyúlhat.

A kép az egyetlen olyan kiterjesztett tartalom, amely nem rendelkezik kontroller osztállyal, mivel a felhasználó számára semmilyen interakciót nem tesz lehetővé.

4.4.2. Videók lejátszása

(a) AR tartalom megjelenítése előtt (b) AR tartalom megjelenítése után

3. ábra. Videók lejátszása

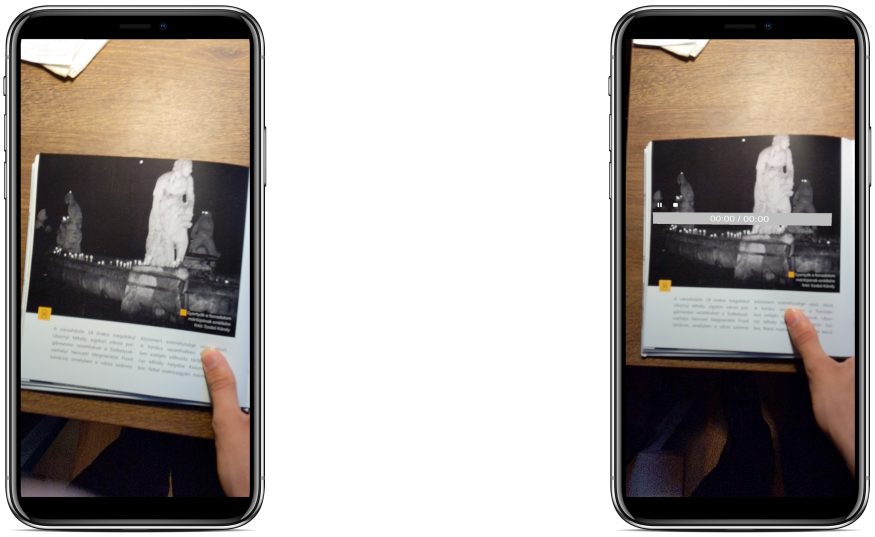
A videók megjelenítése hasonló alapokra van építve: inicializálásuk során szintén egy Quad objektum jön létre, majd ez lesz ráhelyezve a referenciaképre. A különbség a képmegjelenítőhöz képest, hogy ez a Quad objektum ki van egészítve egy videó lejátszóval, amit szintén a Unity biztosít számunkra.

A `VideoPlayer` osztály lehetővé teszi a lejátszást a lokális fájlrendszerből, és a streamelt tartalmakat is támogatja. Rendelkezik egy URL attribútummal, amelynek megadhatunk HTTP, HTTPS vagy FILE protokollnak megfelelő erőforrás azonosítókat.

A megjelenítő inicializálásának utolsó lépésében meghívódik a hozzá tartozó kontroller osztálynak a felépítése. A videót lejátszó Quad objektum fölé egy új transzparens réteg kerül. Ez tartalmazza a UI elemeket: lehetőséget ad a videó megállítására, folytatására vagy újrakezdésére. A `VideoPlayer` prepareCompleted eseményének végrehajtásakor, a gombok alatt megjelenik egy sáv, amely megengedi a videó előre- vagy hátratekerését.

A videólejátszó csak a megfelelő kódolású MP4 formátumú videókat támogatja. Ennek a szigorú megkötésnek a magyarázata, hogy az Android esetében nem minden MP4 formátum játszható le.

4.4.3. Audiófájlok lejátszása



(a) AR tartalom megjelenítése előtt (b) AR tartalom megjelenítése után

4. ábra. Hanganyagok lejátszása

Az audiófájlok lejátszása gyakorlatilag ugyanúgy működik, mint a videóké, annyi különbséggel, hogy a létrehozott Quad objektum ebben az esetben láthatatlan. A fölötte lévő UI réteg itt is megtalálható, csak elhelyezkedése változik meg: a referencia kép közepére kerül.

Az `AudioSource` komponens felel a hanganyagok lejátszásáért. A lejátszandó fájl elérésének megadása hasonló módon történik mint a már említett `VideoPlayer` esetében. Annyi a különbség, hogy az `AudioSource` nem rendelkezik URL attribútummal, a lejátszandó forrást külön fel kell építeni egy AudioClip típusú objektumként, és ezt átadni paraméterként.

Az audiófájlok szempontjából jelenleg az OGG formátum támogatott, amely teljesen szabadon felhasználható.

4.4.4. 3D modellek felépítése

(a) AR tartalom megjelenítése előtt (b) AR tartalom megjelenítése után

5. ábra. 3D modell megjelenítése

A többi megjelenítőhöz hasonlóan ebben az esetben is a referencia képre ráillesztett Quad objektum adja az alapot. Ez az objektum bár jelen van, nem látható a képernyőn, a 3D-s modell jelenik meg ezen a felületen. A felépített modell a Quad gyerek objektumává válik, ami azt jelenti, hogy bármilyen transzformálást végzünk a Quad objektumon, az a modellre is hatással lesz.

A felépítés során fontos szempont volt a méretarányosság megőrzése: a megjelenítendő háromdimenziós objektumnak alkalmazkodnia kell a referencia kép arányaihoz. Mivel bármilyen képen megjeleníthetőnek kell lennie az adott objektumnak, szükség volt egy méretezési módszer kidolgozására.

```
Bounds boundCalc = new Bounds();
var renderers = m_ModelObject.GetComponentsInChildren<Renderer>();
foreach (var renderer in renderers)
{
    boundCalc.Encapsulate(renderer.bounds);
}

var ratioCalc = Math.Max(boundCalc.size.x / transform.localScale.x,
    boundCalc.size.y / transform.localScale.y);

m_ModelObject.transform.localScale /= ratioCalc;
```

1. kódrészlet. Méretezés módszere

Mivel a modell több különálló részből tevődik össze, felépítettük azt a legkisebb keretet, ami magába

foglalja az objektum összes alkotórészét. Ismerve a keret méreteit, a Quad objektum méreteihez tudtuk arányosítani azt. Ennek a módszernek köszönhetően a megjelenítendő modell minden referencia kép esetében a rendelkezésre álló területre fog kiterjeszkedni, az arányokat megőrizve.

A vizualizáció jelenleg az OBJ formátumú fájlokat támogatja. Ez a kiterjesztés geometriai alakzatok leírását teszi lehetővé egy egyszerűsített módon. Az egyszerű szerkezete és a szabadon felhasználhatósága miatt esett erre a választás.

4.5. Felhasználói felület

(a) Kiadványok listája (b) Hamburger menü (c) Információs oldal

6. ábra. Felhasználói felület

Bár egyedi megközelítést alkalmaz a UI felépítésére, a Unity3D játékmotor UI keretrendszere sajnos nehézkessé teszi a komplexebb felületek megvalósítását.

Mint minden más a játékmotorban, a UI elemek is háromdimenziós objektumok, amelyek felhasználnak bizonyos speciális osztályokat. Ilyen például a `Canvas` osztály, amely a UI renderelésével kapcsolatos információkat tartalmazza. Ezeknek az osztályoknak köszönhetően nyeri el az objektum a UI elemekre jellemző tulajdonságokat.

A UI felépítésének rendszere valamilyen szintű hasonlóságot mutat a HTML leíró nyelvhez. Az elemek itt is egy fastruktúrát építenek fel, összeségük adja a felületet. Az elemek között szülő-gyerek kapcsolatok fognak felépülni, és a szülő objektumok transzformálása, skálázása hatással lesz a gyerek komponensekre. Ezeket a műveleteket a `RectTransform` osztály valósítja meg.

A HTML-ben megtalálható tag-ek helyett itt osztályokat tudunk egy adott elemhez hozzáadni. Bár a Unity nem rendelkezik olyan sok opcióval mint a HTML, a lényegesebb funkciókat képes ellátni ezeknek az osztályoknak a segítségével. Ilyen osztály például a `VerticalLayoutGroup`, ami az elem gyerekeit egy vertikális listává alakítja. A lista szerkezetét és kinézetét az osztály attribútumainak segítségével tudjuk módosítani.

A Unity UI keretrendszerének limitációi miatt sokszor a fejlesztőnek kell egyes komponenseket, funkciókat implementálnia. Az ARmore alkalmazásban is találhatóak ilyen elemek. Például, a hivatkozási link megjelenítése és kattintásra való reagálásának megvalósítása, vagy a hamburger menü

implementációja, a hozzátartozó animációval.

```
 serviceProvider.GetDomainService().GetDomains((domains) =>
{
    for (int i = 0; i < domains.Length; ++i)
    {
        GameObject button = Instantiate(buttonTemplate) as GameObject;
        button.transform.SetParent(buttonTemplate.transform.parent, false);
        button.GetComponent<DomainButtonController>().Initialize(domains[i]);
        SearchBarController.Instance.AddItem(
            new SearchItemModel(domains[i].Title, button));
    }
});
```

2. kódrészlet. UI felépítése

A 2. kódrészletben látható a kiadványok listájának feltöltése, ami szemlélteti azt is, hogy a UI keretrendszer olykor kiegészítésekre szorul. A bemutatott kódban, a buttonTemplate nevű objektum van sokszorosítva, majd a SetParent metódussal beillesztve a fastruktúrába.

A 6. ábra mutatja be a mobilalkalmazásban megtalálható felhasználói felületet. Egyik részén megfigyelhető a fentiekben említett kódrészlet végrehajtásának eredménye. A másik két részen láthatóak a UI elemek: a hamburger menü és az általános információkat leíró oldal.

A UI keretrendszer - limitációinak ellenére - rendelkezik nagyon lényeges pozitív tulajdonságokkal is. A vizualizációkon található UI rétegek, amelyek a kiterjesztett tartalmakra igazodnak, könnyen megvalósíthatóak voltak a rendszernek köszönhetően. Mivel az elemek háromdimenziós objektumok, így a Quad objektum gyerekeiként hozzárendelhetőek voltak a tartalmakhoz, és tudták követni a mozgatásokat, skálázásokat. Mindez a `Canvas` osztályban található renderelési funkciók felhasználásával történhetett meg.

5. Eszközök és módszerek

Az ARmore projekt fejlesztése során a verziókövetést a Git biztosította [19]. A projekt komponensei külön tárolókba kerültek, minden funkcionlitás implementálása külön feature ágakon valósult meg. Az implementációt követően a kódbázis módosításai egy review folyamaton mentek keresztül. Minden funkcionalitás ellenőrizve volt, és ezután a stabilnak tekintett develop ágba volt beintegrálva (merge). A master ágba kerültek a kiadott változatok.

A tároló menedzsmentjére a GitLab biztosított egy webes felületet, és a folyamatos integrációt is segítette. Minden módosításnál automatikusan újrafelépült a projekt, így a hibákat egy gyors és egyszerű módon lehetett kiszűrni.

A projekt egységeinek összeillesztését és a kiadásra alkalmas változatok elkészítését a Docker tette lehetővé. A MySQL szerver egy külön konténerben kapott helyet, elválasztva a backendtől, így az adatbázis különállóan is karbantarthatóvá vált. A folyamatos integráció segítségével a létrehozott konténer egy egyszerű parancs végrehajtásával elérhetővé tehető bárki számára.

A TypeScript, HTML, CSS, C# forráskódok és a tulajdonságokat leíró konfigurációs állományok szerkesztésében a Microsoft által fejlesztett Visual Studio Code segített.

A frontend fejlesztése során felhasznált külső függőségek kezeléséért és a környezet konfigurálásának egyszerűsítéséért az npm csomagkezelő felelt [2]. A hasonló szerepet betöltő NuGet csomagkezelőt is alkalmaztuk a .NET keretrendszeren alapuló backend résznél [15].

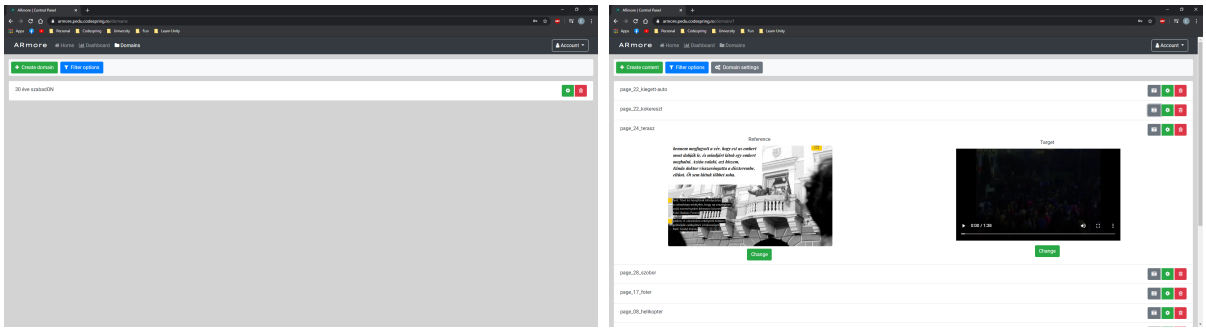
A kitűzött célok elérésében a Scrum agilis szoftverfejlesztési módszer segített [22]. A fejlesztés kéthetes sprintekben történt. A feladatok nyilvántartását és követését szintén a GitLab webes felülete tette lehetővé.

6. Az ARmore működése

Az alkalmazás indítása után egy animált kezdőképernyő jelenik meg az alkalmazás logójával. A két másodperces animációt követően a kiadványok listája válik elérhetővé.

A lista tartalmazza a jelenleg elérhető publikációkat, minden elemhez hozzá van rendelve egy rövid leírás, valamint lehetőség van adott kiadványokra feliratkozni. A feliratkozást követően a kiadvány tartalma letöltődik a lokális tárolóba. Ezután a kiadványon történő módosítások automatikusan megtörténnek a fájlrendszerünkben is.

Egy kiadvány AR tartalmainak megtenkintéséhez előzetesen szükséges a feliratkozás, ezután elérhetővé válik az indítás funkció is. A kiadvány elindítása után a kameranézet jelenik meg. A referencia képek fölé helyezve a telefont, a kiterjesztett tartalmak megjelennek a kép felületén.



(a) Domains menüpont (b) Tartalmak listája

7. ábra. Weboldal

Amennyiben rendelkezünk tartalomszerkesztő fiókkal, a weboldalon keresztül létrehozhatunk új publikációkat, vagy a rendelkezésre álló tartalmakon végezhetünk módosításokat. Bejelentkezés után, elérhetővé válik a Domains menüpont, ami tartalmazza a kiadványok listáját. (7a. ábra)

Ha a kiadvány tartalmain szeretnénk végezni módosításokat, akkor ezt a fogaskerék ikonra kattintva tehetjük meg. Ennek hatására megjelenik a publikáció tartalmainak listája (7b. ábra), és egy-egy kattintás segítségével hozzáadhatunk új tartalmat, vagy módosíthatunk, illetve törölhetünk tartalmakat. A módosítás művelete a tartalom típusától független: mind a videók, mind a 3D-s modellek hozzáadása megoldható egyetlen gomb lenyomásával.

A végrehajtott módosításokról azonnal visszajelzést kapunk, az újonnan feltöltött tartalmak megjelennek a weboldalon.

7. Következtetések és továbbfejlesztési lehetőségek

A U-Hub Mentorprogram keretein belül sikerült egy olyan stabil prototípust elkészíteni az ARmore projekthez, amely minden tervezett alapfunkcióval rendelkezik. A kiterjesztett tartalmakat dinamikus módon lehet hozzárendelni a kiadványokhoz, így sikerült megfelelni a projekt szempontjából legfontosabb követelménynek.

Emellett fontos szempont volt még az alkalmazás használatának egyszerűsége: probáltuk mindig előtérbe helyezni a felhasználóbarát megközelítéseket. Ennek köszönhetően mind a felhasználók, mind a tartalomszerkesztők számára egy letisztult felület áll rendelkezésre.

A projekt rengeteg finomításon és optimalizáláson esett át, így például lényegesen javult a dinamikus tartalmak betöltésének hatékonysága.

Egy új mérföldkő volt a projekt számára, az alkalmazás kiadása a Google Play platformon, majd a későbbiekben az iOS verzió megjelenése az App Store-on. Ezzel párhuzamosan Székelyudvarhely Polgármesteri Hivatala kiadta az első ARmore rendszer által támogatott kiadványt, a "30 éve szabadON" albumot.

A továbbfejlesztési lehetőségek közül a legfontosabbak:

- a streaming alapú, illetve a lokális állományokat használó szolgáltatások között lehessen váltani minden kiadvány esetében, platformtól függetlenül;
- a képeken kívül képgalériák is megjeleníthetőek legyenek;
- a 3D-s modellek legyenek megfelelően textúrázva, és legyenek támogatva esetükben is az egyszerű alapműveletek (nagyítás, forgatás);
- a 3D-s modellekhez lehessen animációkat hozzárendelni;
- a jogosultsági rendszer továbbfejlesztése: a tartalomszerkesztők csak saját kiadványaikhoz férjenek hozzá a felületen (multi-tenancy)
- az adminisztrátorok legyenek moderátori szerepkörrel is felruházva: például tudják eltávolítani a nem megfelelő (pl. szerzői jogokat sértő) tartalmakat

Hivatkozások

[1] *.NET Core documentation*. URL: https://docs.microsoft.com/en-us/dotnet/core/about/ (utolsó elérés dátuma: 2020. márc. 24.)

[2] *About npm*. URL: https://docs.npmjs.com/about-npm/ (utolsó elérés dátuma: 2020. ápr. 20.)

[3] Charles Amat. *Unity Burst Compiler: Performance Optimization Made Easy*. 2018. URL: http://infalliblecode.com/unity-burst-compiler/ (utolsó elérés dátuma: 2020. ápr. 17.)

[4] *AR Foundation*. URL: https://unity.com/unity/features/arfoundation (utolsó elérés dátuma: 2020. ápr. 13.)

[5] *ARCore overview*. URL: https://developers.google.com/ar/discover (utolsó elérés dátuma: 2020. ápr. 11.)

[6] *C# Job System Overview*. URL: https://docs.unity3d.com/Manual/JobSystemOverview.html (utolsó elérés dátuma: 2020. ápr. 17.)

[7] *Data Annotations Attributes in EF 6 and EF Core*. URL: https://www.entityframeworktutorial.net/code-first/dataannotation-in-code-first.aspx/ (utolsó elérés dátuma: 2020. márc. 25.)

[8] *Dependency Injection*. URL: https://www.tutorialsteacher.com/ioc/dependency-injection (utolsó elérés dátuma: 2020. márc. 26.)

[9] *Dive into the world of augmented reality.* URL: https://developer.apple.com/augmented-reality/ (utolsó elérés dátuma: 2020. ápr. 13.)

[10] *Fluent API Configurations in EF 6*. URL: https://www.entityframeworktutorial.net/code-first/fluent-api-in-code-first.aspxx (utolsó elérés dátuma: 2020. márc. 25.)

[11] *How IL2CPP works*. URL: https://docs.unity3d.com/Manual/IL2CPP-HowItWorks.html (utolsó elérés dátuma: 2020. ápr. 17.)

[12] *Introduction to JSON Web Tokens*. URL: https://jwt.io/introduction/ (utolsó elérés dátuma: 2020. márc. 26.)

[13] *Introduction to the Angular Docs*. URL: https://angular.io/docs (utolsó elérés dátuma: 2020. márc. 27.)

[14] J. Bradley M. Jones és N. Sakimura. *JSON Web Token (JWT)*. URL: https://tools.ietf.org/html/rfc7519 (utolsó elérés dátuma: 2020. márc. 26.)

[15] Alfred Myers Mike Jones Christopher R. Gill és Andy Zivkovic. *An introduction to NuGet*. 2019. URL: https://docs.microsoft.com/en-us/nuget/what-is-nuget (utolsó elérés dátuma: 2020. ápr. 20.)

[16] Neoteric. *Single-page application vs. multiple-page application*. 2016. URL: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58 (utolsó elérés dátuma: 2020. ápr. 23.)

[17] Josh Petty. *What is Unity 3D & What is it Used For?* URL: https://conceptartempire.com/what-is-unity/ (utolsó elérés dátuma: 2020. ápr. 10.)

[18] Adam Tuliper. *Unity: Developing Your First Game with Unity and C#*. 2014. URL: https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/august/unity-developing-your-first-game-with-unity-and-csharp (utolsó elérés dátuma: 2020. ápr. 10.)

[19] Jatin Varlyani. *Unity Burst Compiler: Performance Optimization Made Easy*. 2019. URL: https://levelup.gitconnected.com/what-is-git-how-to-use-it-why-to-use-it-explained-in-depth-76a5066abaaa (utolsó elérés dátuma: 2020. ápr. 20.)

[20] *What is .NET Standard?* URL: https://dotnet.microsoft.com/platform/dotnet-standard (utolsó elérés dátuma: 2020. ápr. 10.)

[21] *What is Entity Framework?* URL: https://www.entityframeworktutorial.net/what-is-entityframework.aspx (utolsó elérés dátuma: 2020. márc. 25.)

[22] *What is Scrum?* URL: https://www.scrum.org/resources/what-is-scrum (utolsó elérés dátuma: 2020. ápr. 23.)