ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

Semestrální projekt

Ovládací aplikace pro Visual Try On - Magic Mirror

Jordán Jiří

Vedoucí práce: Ing. David Sedláček, Ph.D.

Studijní program: Otevřená informatika, Bakalářský

Obor: Počítačové hry a grafika

23.5.2022

Obsah

1. Úvod

1.1. Cíl práce a motivace

Cílem této práce je vytvořit funkční prototyp ovládací aplikace k projektu Virtual Try On – Magic Mirror, který zpracovala ve své bakalářské práci Margarita Ryabová . Tento prototyp bude splňovat předem dohodnuté požadavky a bude jej možné otestovat s nezaujatými uživateli. V návaznosti na tento prototyp vznikne v rámci mé bakalářské práce kompletní aplikace, která bude uvedena v Národním muzeu společně s aplikací Margarity Ryabové, kde si budou moci návštěvníci virtuálně vyzkoušet na sobě různé historické obleční a má aplikace bude sloužit k veškerému ovládání, které bude uživatel potřebovat, aby si historické oblečení na sobě vyzkoušel. Dalším cílem je navrhnout a implementovat způsob komunikace mezi mojí aplikací a Virtual Try On – Magic Mirror.

Hlavní motivací pro zvolení právě této semestrální a do budoucna bakalářské práce byl fakt, že se jednalo o vytvoření aplikace od začátku až do jejího konce a je to projekt, který bude mít v budoucnu reálné využití a dále je zde veliký prostor se toho spoustu naučit ohledně vývoje aplikací.

1.2. Struktura práce

Následující text je rozdělen v sedmi kapitolách podle logických celků. V příští kapitole bude povrchově rozebrán projekt Virtual Try On - Magic Mirror, dále zde proběhne seznámení s funkčními a nefunkčními požadavky, které mi byly předloženy při zadávání projektu a bude zde zmíněno, jak se podobné problémy řeší v současné době v jiných muzeích.

Ve třetí kapitole uvedu návrh řešení a popíši zde, jak by aplikace měla vypadat a proč tak vypadá a představím zvolenou klient-server komunikaci.

V další kapitole bude vysvětleno, jak jsem postupoval při implementaci a ukážu, jaké technologie byly použity k jejímu vytvoření.

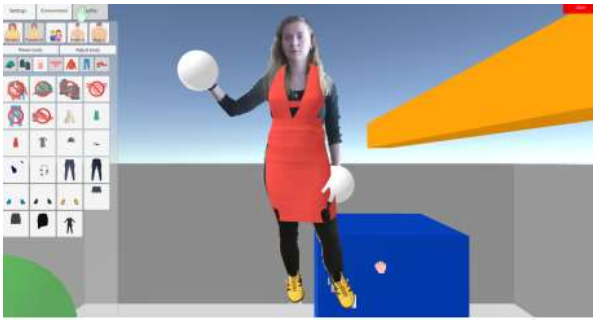
Pátá kapitola představí testování s několika uživateli, který pro účely mé aplikace simulují reálné návštěvníky muzea.

Nakonec se budu věnovat budoucímu rozšiřování aplikace, aby mohla být uvedena do stavu, kdy je připravená na každodenní provoz v muzeu a následuje krátké shrnutí a zhodnocení dosavadního progresu v projektu.

2. Analýza problému

2.1. Virtual Try On - Magic mirror

Virtual Try On – Magic Mirror (dále jako magické zrcadlo/magic mirror) je bakalářský projekt studentky Margarita Ryabová, který umožňuje na člověku vyzkoušet oblečení, aniž by ho musel fyzicky u sebe mít. Zjednodušeně to lze popsat tak, že se uživatel postaví před monitor působící jako zrcadlo, zvolí si kus oblečení a zvolené oblečení se na něm bezprostředně zobrazí.



Projekt je realizován pomocí technologie Kinect, která na vstupu snímá pohyb člověka, který se musí postavit před senzor. Uživatel má pak možnosti svým vlastním pohybem ovládat a interagovat s aplikací, která je přizpůsobená pro tuto technologii. Jak lze vidět na obrázku vpravo, tak Margaritina aplikace má svoje vlastní UI, se kterým lze pomocí Kinectu interagovat, takže se nabízí otázka, proč je potřeba udělat speciální aplikaci na ovládání, když už v aplikaci nějaká forma UI je. Odpověď je taková, že pro účely působení aplikace v muzeu je řešení mého projektu vhodnější, neboť je mnohem přímočarejší i pro méně technicky zdatného návštěvníka a také je jednodušší a přesnější na používání. Dále skrze aplikaci můžeme UI rozšířit o další funkční prvky, které by v současné verzi UI nedávaly smysl.

Margarita si pro vývoj aplikace zvolila unity, které umožňuje vyvíjet i aplikace pro Kinect a v současné době je aplikace již k dispozici, ale její stav ještě není finální a stále probíhá zdokonalování.

2.2. Funkční a nefunkční požadavky mého projektu

Narozdíl od aplikace Magic mirror vzniká má aplikace už s tím účelem, že jednou bude sloužit jako interaktivní podpora k expozici v Národním divadle, a proto je podle toho přizpůsobena už od samotného začátku.

Hlavní funkcionalitou této aplikace je, aby si uživatel zvolil oblečení, které si chce vyzkoušet a následně se na něm zvolené oblečení obratem objevilo v magickém zrcadle, takže součástí projektu není pouze vytvoření aplikace, ale také je potřeba navrhnout oboustrannou komunikaci mezi ovládací aplikací a magickým zrcadlem. Je kladen veliký důraz na to, aby i technicky méně zdatný jedinec zvládl práci s aplikací bez jakékoliv instruktáže, tedy aplikace musí být zcela intuitivní. Další funkcionalitou aplikace je seřazení jednotlivých kusů oblečení do různých logicky souvisejících kategorií, aby se zvýšila přehlednost a také aby se uživatel dostal snadno ke kusům oblečení, které ho spíše zaujmou. Třetí významná funkcionalita aplikace je poskytnout podrobnější informace o jednotlivých kusech oblečení v případě, že to uživatele zajímá.

Významný nefunkční prvek aplikace je rychlost. Volba uživatele by se měla projevit téměř okamžitě v magickém zrcadle a není možné, aby docházelo k dlouhým prodlevám mezi kliknutím uživatele a zobrazením na monitoru, kde běží magic mirror, a proto je důležité zvolit vhodné technologie, abychom tohoto docílili. Dále aplikace musí být robusní a obsahovat pouze nezbytně nutné věci, abychom zajistili jednoduchost používání.

Vzhledem k velmi specifickému využití aplikace je jisté, že aplikace bude používána pouze na jednom typu zařízení a tím bude tablet a můžeme tedy počítat pouze s jedním typem rozložení komponent.

2.3. Existující řešení a digitalizace v muzeích

V dnešní době je digitalizace v muzeích relativně běžná věc a často se můžeme setkat s několika typy přístupů. Jednak je používaná podobná myšlenka, která je i jádrem tohoto projektu, a sice kiosk tablet, jehož pomocí lze získat dodatečné informace k expozici nebo

muzeu (příklad Touch Screen Kiosk v Royal Ontario Museum). Některá muzea se vydaly cestou vlastní aplikace, kterou si návštěvník stáhne při vstupu do muzea z Google Play nebo App Store a může sloužit jako dodatečný zdroj informací.

3. Návrh řešení

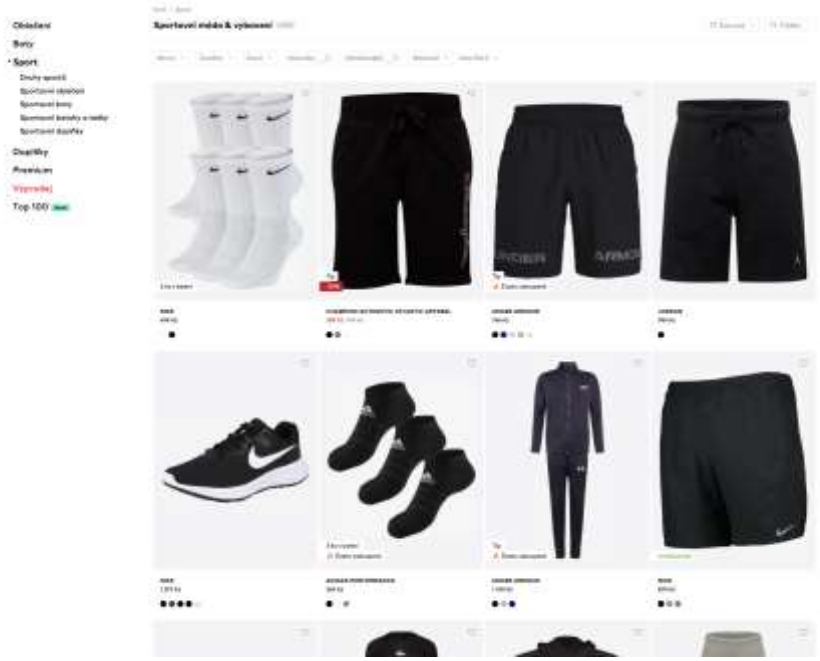
3.1. Celkový návrh

Jak již vyplývá z analytické části, tak tento projekt se skládá ze tří na sobě závislých částí. Našim primárním cílem je vytvořit aplikaci, která bude dostatečně rychle komunikovat s magickým zrcadlem a zároveň bude mít co nejjednodušší vizuální provedení, aby pro uživatele působila přívětivě, protože zde není prostor na to, aby se s ní uživatel dlouho seznamoval nebo dokonce četl nějaký návod, jak s ní pracovat. K tomu aby aplikace mezi sebou dokázaly komunikovat jsem se rozhodl zvolit prostředníka, a sice web server, který bude tvořit spojku mezi oběma koncovými aplikacemi. A neméně důležitá část bude doimplementovat potřebné funkcionality do projektu Virtual Try On – Magic Mirror, aby aplikace dokázala přijímat data ze serveru a poté s nimi pracovat.

Zároveň chceme aby aplikace nedržela žádné data a všechno se do ní dostalo ze serveru až po jejím spuštění, což se bude hodit, až se v budoucnu bude chtít Magic Mirror rozšiřovat o nové kusy oblečení. Vyhneme se tak aktualizací obou aplikací o ten samý update, ale aktualizujeme pouze tu hlavní, což je v tomto případě Virtual Try On – Magic Mirror.

Celkově jsem se snažil jít cestou, aby aplikace vizuálně připomínala e-shop s oblečení, kde nebude možné oblečení kupovat, ale pouze ho zkoušet. Návštěvník tak bude mít pocit,

že s něčím podobným již v minulosti pracoval a opět mu to pomůže se rychle zorientovat v aplikaci.



Na obrázku můžeme vidět, jakou přibližnou cestou jsem chtěl jít při vytváření UI .

3.2. Klient-Server komunikace

Velké a důležité kritérium mé aplikace je zvolení vhodného typu komunikace mezi aplikacemi, protože potřebujeme posílat informace a data oběma směry. Zvolil jsem řešení udělat jeden centrální server, který bude mít API podporující RESTové operace, které budou obě koncové aplikace znát a budou využívat příslušné endpointy. Stav, kterého chceme docílit je mít server jako spojnici, přes kterou si aplikace navzájem posílají data. Aplikace Margarity Ryabové na začátku pošle velký balík dat do mého Magic Mirror Controlleru, kde se data zpracují a zobrazí. Naopak když se návštěvník rozhodne něco nasadit nebo sundat, tak chceme poslat požadavek z Magic Mirror Controlleru (mé aplikace) na server, odkud si ho vezme druhá aplikace a požadavek provede a výsledek se projeví nasazením v magickém zrcadle. Tento způsob jsem zvolil, protože obě technologie, které využívám, mají k dispozici knihovnu, která umožňuje jednoduše komunikovat se serverem, potažmo jeho API.

3.3. Uživatelské rozhraní

Nejdůležitější vlastností uživatelského rozhraní je jednoduchost, jak už bylo zmíněno dříve. Aplikace se proto skládá pouze z jednoho hlavního okna a jednoho fragmentu, který se otevře v okně jako reakce na kliknutí na konkrétního kusu oblečení, kde má uživatel možnost oblečení nasadit. Pokud takto učiní, tak se rozsvítí zelená značka, která má uživateli indikovat, že došlo k nasazení a má mu napovědět, že oblečení lze nasazovat nejen přes fragment, ale i přes zelené tlačítko, které je ve výchozím stavu šedivé a když dojde k nasazení, tak zezelená.

Druhý důležitý prvek jsou kategorie. V aplikaci jsou k dispozici dvoje kategorie, kdy vertikálně řazené kategorie na levé straně jsou nadřazené horizontálním kategoriím nad listem s oblečením. Tento princip jsem převzal z online e-shopů s oblečením, kde je tento typ hierarchicky organizovaných kategorií zcela běžnou praktikou. V našem případě jsou nadřazené ty kategorie, které filtrují oblečení na základě časového období a podřadné jsou ty, které filtrují na základě druhu oblečení.

4. Implementace

4.1. Zvolené technologie

Jak již bylo zmíněno dříve, tak tento projekt se skládá ze 3 částí a v následujících třech podkapitolách si povíme, jaké technologie byly použity k jejímu vytvoření. Je nutné říct, že se jedná pouze o prototyp, takže tento stav ještě není finální a není vyloučeno, že některá technologie bude přidána nebo nějaká stávající nahrazena jinou vhodnější technologií.

4.1.1. Magic Mirror Controller

Jelikož aplikace má velmi specifické využití, tak není potřeba, aby aplikace byla hybridní (víceplatformní) a vystačíme si s řešením, které bude funkční pouze pro jednu platformu, a tím je v tomto případě android. Z tohoto důvodu jsem se rozhodl, že hlavním programovacím jazykem této aplikace bude Kotlin. Velikou výhodou je, že

Kotlin umožňuje práci s knihovnou Retrofit, což je knihovna umožňující komunikaci s web serverem a získávat odtud data za pomoci standartních RESTových operací.

```
// following methods are used by retrofit to proceed REST operations.
interface ApiService {
    @GET( value: "/items/")
    fun fetchAllItems() : Call<List<Item>>

    @GET( value: "/categories/age/")
    fun fetchAllAgeCategories() : Call<List<Category>>

    @GET( value: "/categories/type/")
    fun fetchAllTypeCategories() : Call<List<Category>>

    @GET( value: "/general")
    fun fetchGeneral() : Call<General>

    @POST( value: "posts")
    fun createPost(@Body post: Post): Call<Post>
}
```

```
val retrofit = Retrofit.Builder()
    .baseUrl( baseUrl: "http://10.0.2.2:8088/")
    .addConverterFactory(GsonConverterFactory.create())
    .build()
val api = retrofit.create(ApiService::class.java);
```

Zde je uveden příklad toho, jak jednoduše lze pomocí Retrofitu pracovat se serverem. Pomocí proměnné api lze poté zavolat funkce v ApiService interface a počkat na odpověď ze serveru.

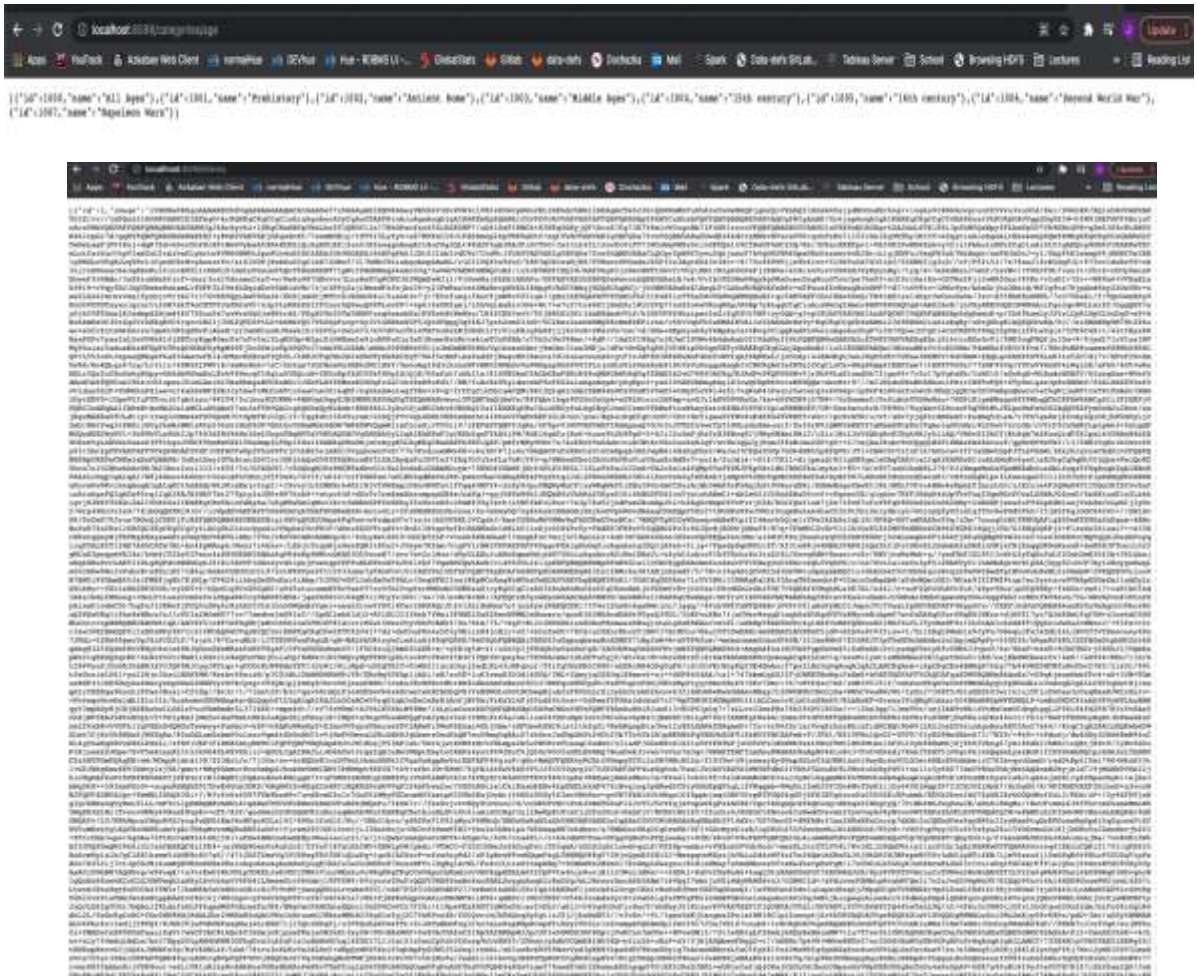
4.1.2. Server

Na vytvoření serveru jsem použil momentálně nejpoužívanější framework na vytváření webových serverů a API a tím je express.js. Tento framework je nadstavbou node.js a velmi zjednodušuje práci s web servery.

Náš server v současném stavu obsahuje jenom jednoduché API a pracuje s dvěma restovými operacemi a těmi jsou GET a POST a nic dalšího zde zatím není potřeba.

Celý tok dat je veden pomocí JSON souborů, protože práce s JSON formátem je velice lidsky čitelná a přímočará a snadno z něj dostaneme data do již nachystaných objektů v koncových aplikacích. Další nezpochybnitelná výhoda JSON formátu je, že

s jeho pomocí dokážeme posílat přes web server obrázky jednotlivých kusů oblečení. V tomto případě je ale potřeba použít Base64 kódování, které nám umožní převést binární data do textové podoby, kterou poté snadno převedeme do JSON formátu.



Na screenshotech je vidět, jak vypadají JSON data na serveru. Tyto data jsou již poslané z aplikace Virtual Try On – Magic Mirror.

4.1.3. Nová část v projektu Visual Try On – Magic Mirror

V této části jsem přidal do magického zrcadla unity skript, který ale v budoucnu bude určitě rozšířen a je možné, že bude fungovat trochu jinak. V současné chvíli je skript schopný přijímat a odesílat requesty web serveru a dále zde připravuji testovací data, která se při spuštění Visual Try On – Magic Mirror odešlou všechna na server.

Pro účely tohoto prototypu posílám na začátku čtyři posty: první obecné informace, další dva posílám balík s kategoriemi a poslední je velký balík s daty ohledně jednotlivých položek k obléknutí, kde drtivou většinu velikosti zabírají obrázky v Base64 kódování.

```
async void Update()
{
    timePassed += Time.deltaTime;
    if (timePassed > nextAction)
    {
        nextAction += 1f;

        var url = "http://localhost:8088/";
        UnityWebRequest www = UnityWebRequest.Get("http://localhost:8088/posts");

        www.SetRequestHeader("Content-Type", "application/json");
        var operation = www.SendWebRequest();

        while (!operation.isDone)
            await Task.Yield();

        Debug.Log(www.downloadHandler.text);
    }
}

IEnumerator Post(string url, string bodyJsonString)
{
    UnityWebRequest request = new UnityWebRequest(url, "POST");
    byte[] bodyRaw = Encoding.UTF8.GetBytes(bodyJsonString);
    request.uploadHandler = (UploadHandler)new UploadHandlerRaw(bodyRaw);
    request.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
    request.SetRequestHeader("Content-Type", "application/json");
    yield return request.SendWebRequest();
}
```

Zde je ukázka toho, jak fungují GET a POST v unity skriptu. Metoda update se volá ve frekvenci 1 vteřina a čeká, až se něco objeví na serveru. POST v tuto chvíli probíhá pouze při spuštění a pošle vše, co chceme zobrazit v android aplikaci.

4.2. Stavba aplikace Magic Mirror Controller

Magic Mirror Controller je tvořen především z jedné hlavní aktivity, kde se odehrává většina logiky aplikace. Dále je zde jeden fragment, který se vytváří jako reakce na klik v hlavní aktivitě. Další důležitou komponentou mé aplikace jsou datové třídy, které slouží k tomu, aby vytvořili vhodnou strukturu pro data, které přijme z web serveru. K vytvoření vhodného UI používám RecyclerView, které ke správnému využití zahrnuje přidání Adapteru, který tvoří prostředníka mezi backendem a frontendem aplikace a umožní naplnit

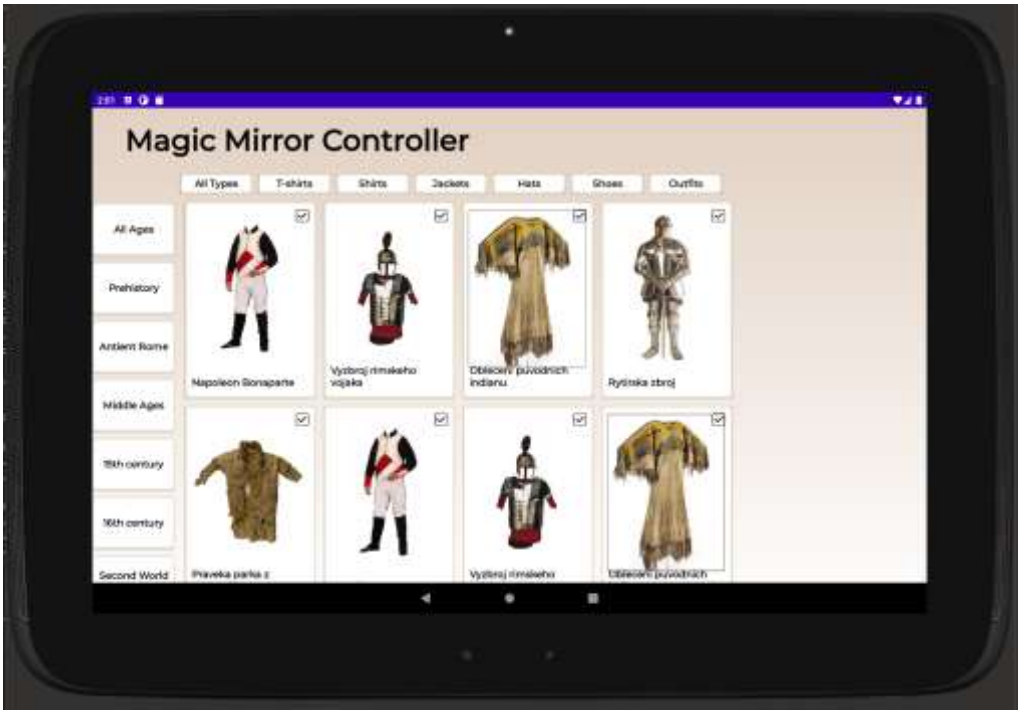
RecyclerView patřičnými daty. Poslední důležitá složka aplikace je interface s Retrofit knihovnou, který umožňuje zpracovávat RESTové operace z web serveru.

4.3. Časová posloupnost vývoje

Po tom, co jsem rozhodl, jaké technologie chci používat, můj první cíl byl jakkoliv poslat data z jedné aplikace do druhé a posléze i naopak, tedy bylo na začátku potřeba vytvořit server a nějaký základ aplikace v androidu a také základ v unity skriptu. Další fází bylo poslat obrázek nějakého kusu oblečení, což znamenalo skript naučit převádět z vizuální podoby do kódování Base64, které převede obrázek do textové reprezentace a v Magic Mirror Controlleru naopak převést Base64 zpět do vizuální/jpeg podoby. Dále bylo potřeba rozmyslet si rozvržení UI a design aplikace. K tomu mi pomohla webová aplikace Figma, ve které lze v jednoduchém editoru zkoušet různé varianty.

Na obrázku vlevo je má původní myšlenka, jak aplikace měla vypadat, ale vzhledem k tomu, že finální verze bude na tabletu, tak je toto řešení nevyhovující a bylo potřeba udělat rozvržení na tablet, které je na obrázku napravo.

Jelikož aplikace Margarity je pořád ve vývoji, tak dalším krokem pro mě bylo přichystat testovací data pro kategorie a hlavně pro jednotlivé kusy oblečení a poté už následovalo doimplementovat aplikaci do takového stavu, kdy je možné ji otestovat s lidmi.



Na následujících obrazcích je vidět finální podoba aplikace, kde na spodním obrázku je zobrazen nasazený outfit Napoleon Bonaparte.

5. Testování

V této sekci otestujeme, zda aplikace splňuje zadané požadavky, jestli umí to, co jí bylo zadáno a na základě tohoto testování se budou hledat podněty ke zdokonalení, které mimo jiné rozvedu v předposlední kapitole.

Testování jsem provedl se 3 lidmi různých věkových skupin, protože nelze úplně předpokládat, že by nějaká věková skupina do muzea nechodila. Zadání pro uživatele bylo prosté, a sice, co by s aplikací dělali, kdyby v muzeu narazili na monitor (s magickým zrcadlem), před kterým by byl umístěný tablet s touto aplikací. Bohužel provést toto testování bez magického zrcadla bylo komplikovanější a myslím, že některá kritika by nevznikla, pokud by člověk viděl vizuálně, jak se na něm objevuje oblečení.

Během samotného testování všechny účastníky napadlo kliknout na kus oblečení, kde si okamžitě všimli tlačítka nasadit a oblečení tedy bez problému nasadili. Rovněž ihned zaregistrovali, že zezelenalo tlačítko u nasazeného oblečení, ale mysleli si, že je to jenom informační značka s tím, že oblečení je nasazené a nepodařilo se jim rozpoznat, že přes toto tlačítko lze nasazovat a sundavat oblečení.

Pouze jeden účastník ze tří se rozhodl využít možnosti filtrů, ale toto pokládám za důsledek toho, že testovacích dat jsem použil málo a pravděpodobně z toho důvodu nebylo potřeba filtrovat oblečení.

Původně bylo v aplikaci implementováno, že pokud uživatel klikne na „nasadit“, tak se fragment neukončí. To jsem udělal z důvodu, pokud by si uživatel chtěl dále číst informace o oblečení, tak aby mu to při nasazení nezmizelo a nemusel to posléze znovu otevřít. Nicméně jeden účastník testování právě na toto poukázal, že mu to přijde zvláštní, že fragment zůstává otevřen i po nasazení.

Z výsledku mého testování vyplývá, že hlavní myšlenka tohoto prototypu se podařila, a sice všichni účastníci dokáží nasadit oblečení. Nicméně jakmile bude možnost vyzkoušet aplikaci společně s magickým zrcadlem, tak je potřeba udělat testování znovu, na jehož základě teprve lze udělat nějaký závěr.

6. Budoucí rozšíření

Vzhledem k tomu, že se momentálně jedná pouze o prototyp aplikace, tak je zde prostor pro zdokonalování. První důležitá věc je zdokonalit unity skript, který bude říkat, jaké oblečení se má nasadit. V současné době neznám finální podobu reprezentace, ve které budou data k nasazení uchovávána, takže první důležitá věc bude se s Margaritou dohodnout na tom, v jaké formě budu mít data, která se budou dále posílat na server a také na tom, co přesně ode mě bude chtít dostávat za informaci, aby se oblečení nasadilo.

Server by bylo vhodné připravit tak, aby se spouštěl pokaždé společně s Virtual Try On – Magic Mirror aplikací a nebylo nutné spouštět pokaždé separátně server a poté až koncové aplikace.

V aplikaci bude potřeba trochu upravit architekturu, aby byl kompletně oddělený view od logiky aplikace. Dalším krokem bude dotažení designu aplikace, který v současné formě není úplně přesný a určitě by to chtělo doladit a připravit na to, že aplikace může běžet na tabletech o různém rozlišení. Funkční prvek, který bude muset být přidán, je volba jazyka. V budoucnu by aplikace měla umět přepínat mezi jazyky, minimálně mezi češtinou, angličtinou a němčinou, aby ji mohli využívat i zahraniční návštěvníci.

Na základě testování a výpovědi účastníku jsem došel k závěru, že by bylo vhodnější filtrovat data podle logických celků než nutně podle časového období. (např. Indiáni, Napoleonská éra, ale i časové jako třeba Středověk, Druhá světová válka apod.)

Nejdůležitější věc, která musí být upravena je zelené tlačítko, jež musí lépe indikovat, že za jeho pomoci můžeme nasazovat a sundávat oblečení a není tedy nutné pokaždé kliknout na kus oblečení a pak ještě jednou na tlačítko nasadit. Fragment se zvoleným oblečením by měl spíše sloužit jako zdroj podrobnějších informací.

7. Závěr

V této práci jsem zabýval návrhem komunikace mezi aplikacemi a také vývojem samotné aplikace. Nejprve bylo nutné si něco nastudovat o problematice síťové komunikace a také zajistit si vhodné technologie k vývoji samotného Magic Mirror Controlleru. Na druhou stranu aplikace je v prototypové fázi a tento semestrální projekt lze považovat za výkop směrem k cíli a je na tomto projektu ještě spoustu práce, která bude náplní mé bakalářské práce.

Zdroje a v definice:

RESTové operace: Je to architektura rozhraní, která umožňuje jednotný a snadný přístup ke zdrojům. REST definuje čtyři základní metody pro přístup, které se jmenují po tom, co opravdu znamenají. Jsou to GET (= Retrieve), POST (= Create), DELETE a PUT (= Update)

Endpoint: Koncový bod. Tento pojem se využívání u API. Může to být třeba http://localhost:8088/categories/age a pokud by se místo „age“ na konci bylo něco jiného, tak mluvíme o jiném endpointu.

Bakalářská práce Margarity Ryabové: https://dspace.cvut.cz/handle/10467/99196

Xbox wiki kinect: https://xbox.fandom.com/wiki/Kinect

Mobilní aplikace v muzeu: https://www.smb.museum/en/whats-new/detail/a-new-app-a-comprehensive-education-and-outreach-programme-and-over-60000-visitors-to-beyond-compare-art-from-africa-in-the-bode-museum/

REST API: https://zdrojak.cz/clanky/rest-architektura-pro-webove-api/

API: https://cs.wikipedia.org/wiki/API

Retrofit: https://square.github.io/retrofit/