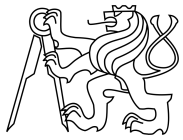
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Aplikace pro mobily a chytré televize pro neziskovou TV
Student:	Bc. Michael Drdlíček
Vedoucí:	Ing. Tomáš Vondra
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

CATVUSA je nezisková TV propagující ČR v USA. Nabízí tři televizní kanály (cestopisný, jazykovědný a kuchařský), dvě radiové stanice a mapu zajímavých míst v ČR.

Ve spolupráci s producentem a technikem televizní stanice sepište požadavky na mobilní aplikaci. Požadavky analyzujte a navrhněte jejich řešení na mobilní platformě. Aplikace by měla splňovat doporučení pro tvorbu grafického rozhraní Android a podporovat širokou škálu zařízení, včetně Android TV. Aplikaci implementujte. Otestujte funkčnost aplikace a proveďte test použitelnosti nebo beta test s dalšími přispěvateli CATVUSA.

Funkčnost zahrne streaming videa, rádio se zvýhodněním podporovatelů nadace, mapu míst s možností uložení oblíbených a kvízy o ČR. Ve spolupráci s technikem řešícím web projektu navrhněte REST API, které bude umožňovat především stahování nejnovějších seznamů videí všech kategorií a míst zájmu, a implementujte stranu klienta. Veškerý obsah je již připraven v databázi webu, která nemá HTTP API.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 1. července 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Aplikace pro mobilní telefony a chytré televizory pro neziskovou TV

Bc. Michael Drdlíček

Vedoucí práce: Ing. Tomáš Vondra

4. ledna 2017

Poděkování

Na tomto místě bych rád poděkoval zejména vedoucímu práce Ing. Tomáši Vondrovi za čas strávený vedením práce a poskytováním rad v celém průběhu tvorby práce. Dále bych chtěl poděkovat Johnu Hornerovi, vedoucímu projektu CATVUSA, za dodávání potřebných materiálů, testování a zpětnou vazbu při vývoji. Rád bych také poděkoval Michalu Ozogánovi za vytvoření webového API. Poděkování patří také Kateřině Králové za revizi textů a všem spolupracovníkům CATVUSA, kteří se podíleli na testování aplikace. Nakonec bych rád poděkovat svým rodičům za podporu během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. ledna 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2017 Michael Drdlíček. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Drdlíček, Michael. *Aplikace pro mobilní telefony a chytré televizory pro neziskovou TV*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá návrhem a implementací aplikací pro mobilní zařízení a TV s operačním systémem android pro projekt CATVUSA. Tento projekt propaguje ČR v USA, zejména pomocí videí o ČR. Aplikace zprostředkovávají webovou část projektu uživatelům s OS android. Práce se skládá z analýzy podobných aplikací, návrhu aplikací, jejich implementace a otestování.

Klíčová slova Android, CATVUSA, videopřehrávač, propagace ČR

Abstract

This dissertation is concerned with a draft and an implementation of an application for mobile phones and televisions with the Android operating system for the CATVUSA project. This project promotes the Czech Republic in USA particularly with the help of videos about the Czech Republic. The applications mediate web part of project to users with the Android operation system. The dissertation consists of an analysis of similar applications, draft of applications, their implementation and testing.

Keywords Android, CATVUSA, video player, propagation of the Czech Republic

Obsah

Úvod **1**

1 Cíl práce **3**
1.1 Požadavky na mobilní aplikaci 3
1.2 Požadavky na aplikaci pro TV 5

2 Analýza **7**
2.1 Analýza existujících řešení 7
2.2 Analýza možností přehrávání videa 12
2.3 Přehrávání rádia 14
2.4 Aplikace pro TV 14

3 Návrh **15**
3.1 API 15
3.2 REST 15
3.3 Návrh webového API 17
3.4 Návrh databáze 20
3.5 Návrh vzhledu 22
3.6 Návrh TV 28

4 Realizace **31**
4.1 Moduly 31
4.2 Build gradle 32
4.3 Manifest 34
4.4 Struktura modulů 35
4.5 Komunikace se serverem 37
4.6 Naplnění databáze 39
4.7 Fragment využívající API 42
4.8 Cache 42
4.9 Implementace jednotlivých částí modulu mobile 44

4.10 Implementace jednotlivých částí modulu tv 50

5 Testování **53**
5.1 Alfa testování 53
5.2 Beta testování 54

Závěr **55**

Literatura **57**

A Seznam použitých zkratek **59**

B Obsah přiloženého CD **61**

Seznam obrázků

1.1 Use case diagram 6

2.1 Detail města v aplikaci CzechTurism 8
2.2 Možnosti zábavy v aplikaci Google Trips 9
2.3 Hlavní stránka města v aplikaci Guides by Lonely Planet 11
2.4 Přehrávač v aplikaci Travel Video Collections 12

3.1 Schéma databáze 21
3.2 Vzhled menu a rádia 23
3.3 Vzhled seznamu a detailu videa 24
3.4 Vzhled mapy a seznamu míst 25
3.5 Vzhled seznamu a spuštění kvízu 26
3.6 Vzhled otázky a výsledků 26
3.7 Vzhled přihlášení a registrace 27
3.8 Vzhled uživatelského účtu a nastavení 28
3.9 Vzhled TV aplikace 29

4.1 Diagram průběhu načtení informací ze serveru 40
4.2 Diagram naplnění databáze 42
4.3 Vývojový diagram načtení dat do fragmentu 43

Seznam tabulek

4.1 Přehled modulů v aplikaci 32
4.2 Knihovny použité v aplikaci 33
4.3 Přehled verzí androidu, s využitím větším než 0.1% 34
4.4 Balíčky v aplikaci 36
4.5 Základní rozdělení zdrojů 37

Úvod

Neziskový projekt Czech-American TV(CATVUSA) se zabývá propagací České republiky v USA. Projekt se snaží šířit povědomí o české historii, památkách, zvycích, tradicích, přírodě atd. Jedná se o šíření zejména formou videí.

CATVUSA vysílá každý týden půlhodinový pořad v Americké kabelové televizi. Pořad je vždy zaměřen na konkrétní kraj, město, památku, obyčej apod. Tento pořad je následně vystaven ve webové částí tohoto projektu, kde je uživatelům kdykoliv dostupný pro přehrání. Videa jsou tvořena ve spolupráci s jednotlivými kraji ČR. Na internetových stránkách je dále možné poslouchat internetové rádio, přehrávat video kurz češtiny, přehrávat videa o české kuchyni nebo třeba spustit kvíz o jednotlivých krajích ČR.

Tato práce se bude zabývat návrhem a vytvořením mobilní aplikace a TV aplikace pro operační systém android, které zprostředkují obsah webové části projektu uživatelům tohoto systému. Zejména se bude jednat o přehrávání videa, poslech rádia, zobrazení zajímavých míst na mapě a možnost vyzkoušet si naučné kvízy o ČR.

Chytrá mobilní zařízení jsou stále rozšířenější, a uživatelé je využívají čím dál tím více. Spoustu činností, na které uživatelé dříve potřebovali počítač, zvládnou dnes z mobilního zařízení. Proto je potřeba jim obsah přehlednou formou zprostředkovat na jejich zařízeních.

Ačkoliv by se mohlo zdát, že tvorba aplikace zprostředkovávající obsah webu je poněkud zbytečná, přináší aplikace uživatelům různé výhody. Zejména jednodušší ovládání, protože i mobilní web se hůře ovládá, než mobilní aplikace. Zvláště pokud aplikace zprostředkovává pouze důležitý obsah webu. Další výhodou jsou například notifikace díky nimž se uživatel automaticky dozví o novém obsahu.

Kapitola 1

Cíl práce

Cílem této práce je navrhnout a vytvořit aplikaci pro operační systém android. Aplikace bude jednoduše a přehledně zprostředkovávat obsah webové části projektu. Bude vytvořena ve dvou verzích. První verze bude pro mobilní telefony a tablety. Tato verze bude zprostředkovávat uživateli i další funkcionalitu oproti webu, jako vědomostní kvíz a mapu zajímavých míst v České republice. Druhá verze bude určena pro televizory, ta bude naopak pro jednoduchost ovládání dálkovým ovladačem obsahovat pouze přehrávač videí a rádia.

Mobilní verze bude optimalizována jak pro mobilní telefony, tak i tablety. Bude podporovat oba režimy otočení obrazovky a bude dostupná pro co největší počet verzí operačního systému.

Aplikace bude vytvořena podle standardů a doporučených postupů pro tvorbu aplikací pro operační systém android. Jak uživatelských, tak programátorských, dostupných z [15] a [16]. Díky tomu bude zajištěno, že se uživatelé snadno a rychle v aplikaci zorientují, a vše jim bude zprostředkováno pro ně známým designem.

Aplikace bude alespoň základně fungovat i bez připojení k internetu. V tomto stavu však bude obsahovat pouze textové informace. Pro videa, radio, mapu a veškeré obrázky bude potřeba internetové připojení, aby tento obsah zbytečně neplýtval pamětí zařízení.

1.1 Požadavky na mobilní aplikaci

1.1.1 Přehrávání videí

Hlavní funkcí aplikace bude přehrávání videí dostupných na webu. Videa budou roztříděna do několika kategorií (broadcast, class, cooking). Pomocí těchto kategorií bude možné videa filtrovat. Zároveň bude možné videa vyhledávat podle názvu. Při zobrazení detailu videa se mimo možnosti přehrát video také

zobrazí doprovodné texty k videu. Pokud bude k videu dostupný vzdělávací kvíz, bude spuštěn ihned po přehrání videa.

1.1.2 Přehrávání rádia

Aplikace bude stejně jako web umožňovat přehrávání dvou radiových stanic (folk, klasik). Při přehrávání bude ověřeno, zda je uživatel do aplikace přihlášen a zda má zaplaceno členství. Pokud ano, tak bude moci poslouchat neomezeně, pokud ne, tak bude po 10-ti minutách vyzván k přihlášení a případnému zaplacení přes PayPal nebo zaplacení pomocí in-app nákupu zprostředkovaného Googlem.

1.1.3 Členství

Aplikace bude umožňovat přihlášení do členské sekce a případné finanční podpoření projektu. Stejně tak bude možné zaregistrovat nového uživatele.

1.1.4 POI

Aplikace umožní uživatelům vyhledávat a zobrazovat zajímavá nebo pro turisty důležitá místa v České republice (POI).

Zobrazena budou místa, která se podaří sehnat ve spolupráci s producentem TV (informační centra, turistická centra, turistické památky, památky UNESCO, ...). V detailech daného místa bude vždy vypsáno v přehledné formě vše zjištěné o daném místě.

Všechna místa, o kterých bude dostatek informací pro zobrazení na mapě, budou tuto možnost využívat.

Aplikace bude obsahovat seznam POI (případně základní filtr, což záleží na tom, kolik míst a jakého typu bude k dispozici). Dále bude možné místa vyhledat na mapě nebo pomocí textového vyhledávání.

1.1.5 Vědomostní kvíz

Aplikace bude obsahovat vzdělávací kvízy. Tématem kvízů budou jednotlivé kraje České republiky. Aplikace bude zobrazovat seznam všech dostupných kvízů, ze kterého bude možné jednotlivé kvízy spouštět. Stejně tak se kvíz spustí automaticky po skončení videa, pokud se video týká některého z krajů.

Otázky budou vždy typu 1 z N. Sadu otázek dodá zadavatel. Každý kvíz bude obsahovat více otázek. Po vybrání odpovědi bude uživateli zobrazeno, zda odpověděl na otázku správně (obarví se zeleně) nebo špatně (obarví se červeně). Při špatné odpovědi se uživateli zeleně zobrazí i správná odpověď na otázku. Otázky nebude možné přeskakovat a pokračovat na další bude možné vždy, až po kliknutí na některou z odpovědí. Po dokončení kvízu se uživateli zobrazí vyhodnocení.

1.1.6 Sdílení a sociální sítě

Z aplikace bude možné sdílet videa a POI. U videa bude sdílena URL adresa videa. U POI půjde o adresu na mapě. Tyto adresy bude možné sdílet mezi všechny aplikace nainstalované v uživatelově zařízení, které sdílení obsahu umožňují.

Dále bude možné z aplikace přejít na stránky organizace na jednotlivých sociálních sítích, které využívá ke své propagaci.

1.1.7 Oblíbené

Videa a POI bude možné uložit do oblíbených. Aplikace pak bude obsahovat sekci, ve které budou zobrazeny všechny uživatelovy oblíbené položky. V této sekci bude možné přepínat mezi zobrazením seznamu oblíbených videí a oblíbených POI.

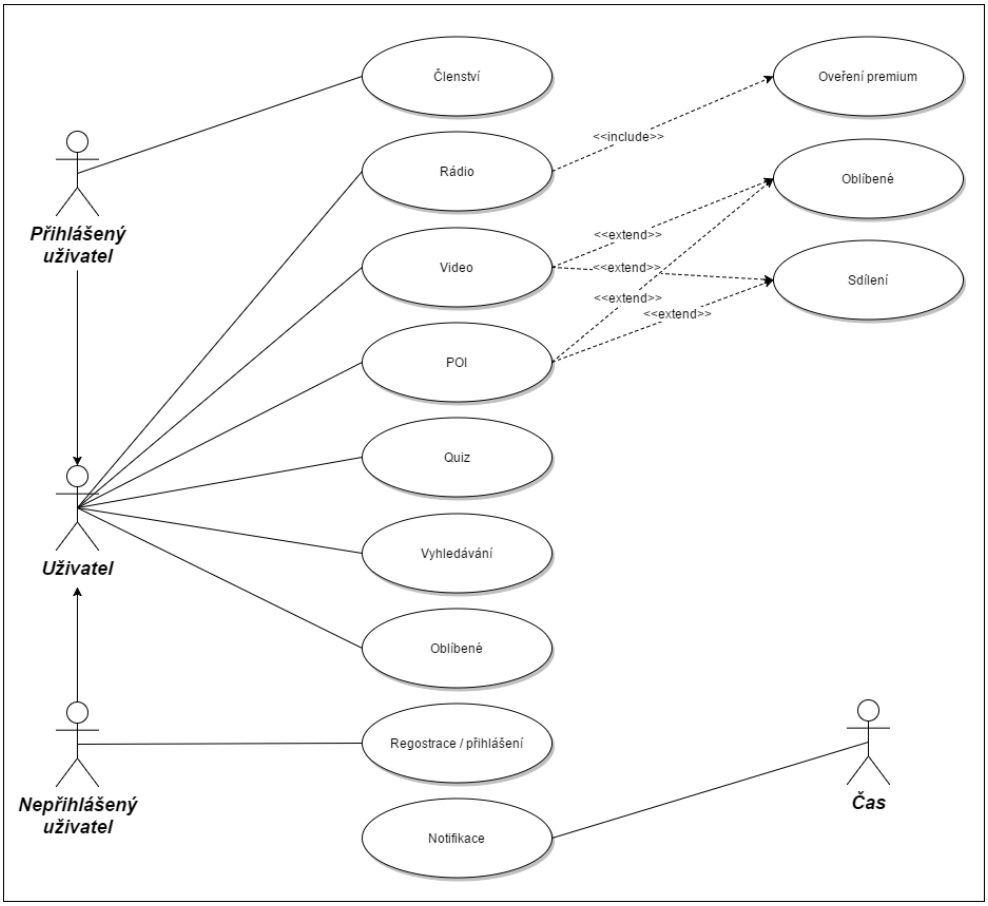
Kategorie oblíbených bude zobrazena i přímo v jednotlivých sekcích.

1.1.8 Notifikace

Aplikace bude kontrolovat, zda jsou dostupná nová videa. Pokud uživatel zapne možnost automatické kontroly, bude aplikace na pozadí v denních intervalech kontrolovat, zda jsou dostupná nová videa. Pokud se tak stane, bude uživatel upozorněn pomocí notifikace.

1.2 Požadavky na aplikaci pro TV

Jak bylo zmíněno výše, aplikace bude obsahovat pouze přehrávač videí a rádia. Videa budou roztříděna do kategorií stejně jako v mobilní aplikaci.



Obrázek 1.1: Use case diagram

Kapitola 2

Analýza

Na úvod této kapitoly bude provedena analýza podobných aplikací. Vybrány jsou aplikace pro platformu android, které se zabývají podobnou tématikou. Tedy podporou turistiky a přehráváním naučných videí. Dále budou popsány a srovnány po technické i uživatelské stránce možnosti samotného přehrávače videí a rádia.

2.1 Analýza existujících řešení

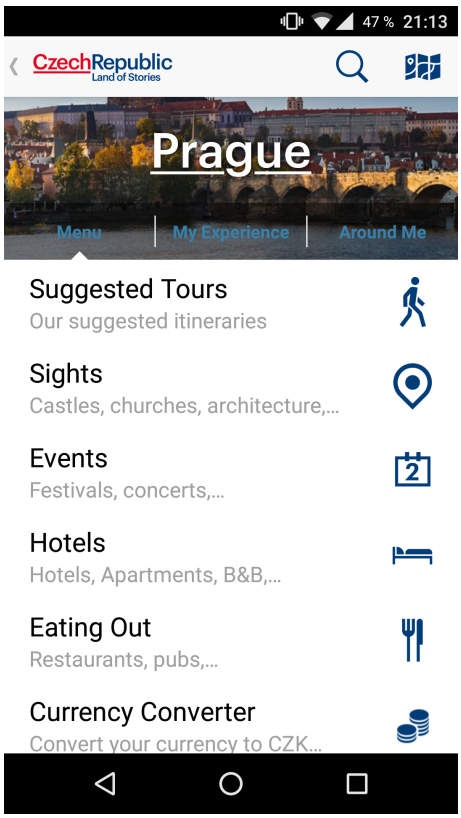
2.1.1 Czech Republic Land of stories

Jako první byla vybrána aplikace[1] od organizace CzechTurism. Jedná se o příspěvkovou organizaci provozovanou přímo Ministerstvem pro místní rozvoj ČR. Jedná se v podstatě o jakousi oficiální aplikaci ČR pro rozvoj zahraniční turistiky u nás. Na úvodní stránce je zobrazen seznam 13cti měst. Jedná se sice o turisticky zajímavá místa, nicméně o ostatních městech se uživatel nedozví vůbec nic. Ke každému městu jsou zobrazeny tři sekce. V první jsou nejzajímavější tipy na výlety, u kterých si lze popis nechat přečíst pomocí Text-to-Speech. Dále seznam památek v okolí včetně ceny vstupného, informace o hotelech a restauracích. Každé město obsahuje sekci pro převod měny z korun na měnu zvolenou v nastavení aplikace. Převod je však všude stejný, proto by bylo lepší tuto sekci přesunout na úvodní stránku aplikace.

Velmi zajímavá je sekce obsahující uživatelovy zážitky. Je možné například vytvořit fotografii, napsat k ní poznámku, zobrazit ji na mapě, ukládat všechna místa do oblíbených a sdílet na sociálních sítích. Uživatel si tak může z dovolené odvést deník v elektronické podobě. Poslední sekce u každého města obsahuje seznam nejbližších zajímavých míst. Sdílet lze z aplikace pouze na sociální sítě, nikoliv však přímo odeslat pomocí zprávy dalšímu uživateli.

To, že je aplikace zaměřená na zahraniční turisty dokládá i fakt, že je celá v základu v angličtině. V nastavení lze přepnout na mnoho světových jazyků,

čeština mezi nimi však není. Aplikace nemá příliš stažení, zato má ze všech popsaných nejlepší hodnocení 4.4/5.



Obrázek 2.1: Detail města v aplikaci CzechTurism

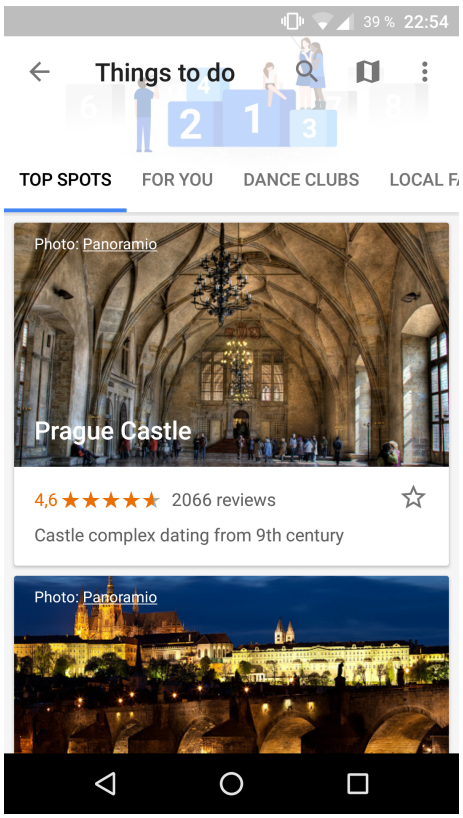
2.1.2 Google Trips

Další vybranou aplikací je aplikace[2] společnosti Google. Google stojí za celým androidem a tvoří standardy pro tvorbu aplikací. Poslední guidelines[15], jak tvořit vzhled aplikací, se nazývá Material design. Tato aplikace je v tomto stylu samozřejmě napsaná.

Jednoduchá úvodní obrazovka zobrazuje pouze slide show několika hlavních měst. Důležitější je zde však vyhledávací pole, do kterého uživatel napíše město, které chce navštívit. Po vyhledání města může uživatel vytvořit výlet. Tato možnost přidá zvolené město na úvodní stránku a pokud uživatel chce, stáhne mu všechny další informace v aplikaci z daného města do zařízení pro offline použití. V době testování aplikace však stahování nefungovalo a vždy skončilo chybovou hláškou, informující, že se stažení nepovedlo. Sekce rezervace zobrazuje informace o rezervacích vytažených z gmail účtu uživatele. Asi nejlepší funkcí je zobrazení možností zábavy roztříděné do mnoha kategorií. V

každé kategorii je pak přehledný seznam míst se základními informacemi. Po otevření daného místa jsou zobrazeny podrobnější informace včetně recenzí dalších uživatelů. Na mapě lze zobrazit jak celý seznam míst, tak jednotlivá místa. Místa je možné ukládat do oblíbených. Sekce denní plán obsahuje seznam tras na celý den. Po vybrání trasy je vidět vše propojené na mapě včetně popisu jednotlivých místa. Jak dlouho se jde mezi místy, otevírací doba a jak dlouho se typický uživatel na daném místě zdrží. Sekce jídlo a pití obsahuje informace o typických jídlech pro dané místo, ale také seznam nejlépe hodnocených podniků roztříděný do kategorií podle využití. Poslední dvě sekce obsahují všeobecně důležité informace o tom, jak to na daném místě funguje.

Seznamy míst v této aplikaci jsou inspirací při tvorbě seznamu videí v aplikaci, jež je předmětem této práce. Dále pokud bude k dispozici dostatek informací o zajímavých místech v ČR, bude tato aplikace inspirací pro využití těchto dat.



Obrázek 2.2: Možnosti zábavy v aplikaci Google Trips

2.1.3 Guides by Lonely Planet

Nejprve byla vybrána pro analýzu aplikace TouristEye - Travel Guide[3]. Ta však ihned po spuštění otevřela dialog, ve kterém vybízela ke stažení této aplikace[4], že s ní bude dosaženo lepších výsledků. V aplikaci je na úvodní stránce vyžadováno přihlášení Lonely Planet účtem. To lze však přeskočit a uživatel se dostane na seznam vybraných míst. Pokud chce jiné místo, může ho na stejné stránce vyhledat. Velkou nevýhodou je, že si z vybraného města nelze nic prohlédnout dokud nestáhneme celého městského průvodce. Zároveň nás aplikace neinformuje o tom, jak je průvodce velký, což je zejména na roamingu zásadní informace. Naštěstí se v rámci průvodce nestáhnou mapy ani obrázky, ale pouze veškeré textové informace, tím pádem by velikost stahování neměla být příliš velká.

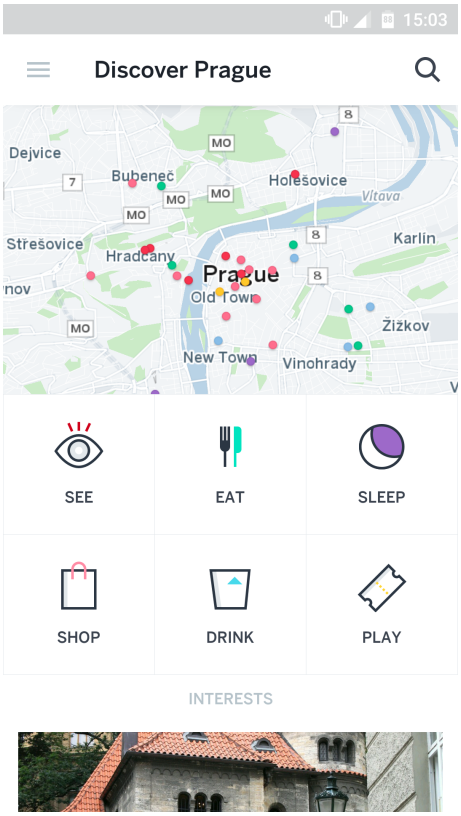
Po stažení a zobrazení konkrétního města, se dostaneme na jeho detail. Zde největší část obrazovky zabírá mapa, která je určena spíše řidičům než turistům. Mapové podklady zobrazují zejména čísla silnic. To je však pro turistu, který se většinu času pohybuje bez vlastního automobilu, zbytečné a bylo by lepší zobrazit užitečnější informace. Mapové poklady ani alespoň vrstvy nelze změnit.

Dále je zde několik základních kategorií jako jídlo, nákupy, zajímavosti, atd. Po zobrazení kategorie se objeví seznam míst, opět pod mapou zobrazující daná místa. Na detailu místa jsou zobrazeny všechny kontakty, popis, mapa, atd. Místo si lze uložit do oblíbených. Při přihlášení lze pak oblíbená místa zobrazit i na dalších zařízeních. Oblíbená místa se zobrazují ke každému městu zvlášť a lze se k nim dostat z levého vysouvacího menu. Místo lze sdílet do všech aplikací podporujících sdílení. Velmi užitečná je informace o tom, jakým spojem městské hromadné dopravy se dá k místu dostat. Škoda, že si na mapě nelze zobrazit, kudy daný spoj jede, nebo jak se na něj dostat. Užitečná je také informace o výši vstupného.

Dále je na na hlavní stránce města seznam několika zajímavých kategorií pro danou lokaci, ty jsou u každého města jiné, vždy pak následně obsahují seznam a detaily míst.

Poměrně matoucí je levé menu. Mezi uživateli systému android je zvykem, že toto menu obsahuje rozcestník celé aplikace. Zde je však v menu hlavně navigace v rámci daného města. Jako poslední položka úplně dole se lze přepnou zpět na seznam všech měst. Menu mimo zobrazení hlavní stránky a oblíbených míst v daném městě obsahuje navíc sekci důležitých informací o daném městě. Zejména ceny potravin a služeb, jež by mohl turista potřebovat.

I přes velký počet stažení a průměrné hodnocení 4.1/5 je celá aplikace poměrně pomalá a při testu několikrát přestala úplně reagovat. Zároveň uživatelské rozhraní celé aplikace je neintuitivní a matoucí, aplikace tedy nebude použita jako inspirace při tvorbě práce.



Obrázek 2.3: Hlavní stránka města v aplikaci Guides by Lonely Planet

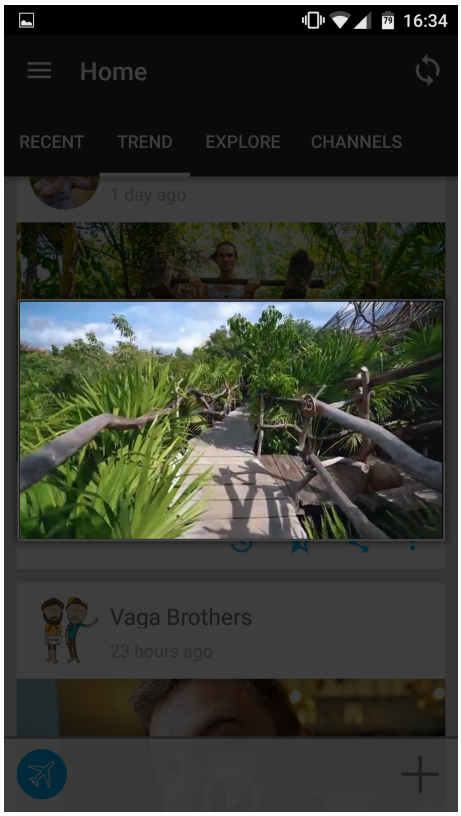
2.1.4 TVA: Travel Video Collections

Tato aplikace[5] byla vybrána, protože na rozdíl od všech předešlých obsahuje i videa. Aplikace funguje jako agregátor největších cestovatelských Youtube kanálů. Základní obrazovka obsahuje seznam videí včetně náhledu a hlavních informací. Video je možné uložit na později, hodnotit, sdílet nebo přidat do oblíbených. Na detailu videa je mimo předchozího navíc podrobný popis videa vytažený z Youtube. Z hlavní stránky se lze dostat do sekce, kde uživatel může odebrat některé kanály. Kanálů je 25 a nelze přidávat další. Další zajímavou funkcionalitu aplikace neobsahuje.

Při hodnocení aplikace se narozdíl od běžného hodnocení pomocí hvězdiček nebo čísel používá slovní hodnocení, což je odchylka od standardů v systému android.

Samotné video streamuje přímo se serverů Youtube. K přehrávání je použito Android Player API[6]. Programátor zde určí pouze velikost a pozici přehrávače. Veškerý obsah včetně UI pak přidá do přehrávače přímo API a UI není možné měnit. V aplikaci není možné přehrát video na celou obrazovku. Při otočení zařízení do režimu na šířku se sice okno přehrávače zvětší, stále má

ale okolo sebe okraj s pozadím z původní aplikace. Při přehrávání na výšku je video vycentrované a vše na pozadí je mírně ztmavené.



Obrázek 2.4: Přehrávač v aplikaci Travel Video Collections

2.2 Analýza možností přehrávání videa

2.2.1 Android MediaPlayer

Jedná se o nejjednodušší možný přehrávač, který je dostupný přímo v Android SDK a není tudíž potřeba žádná další knihovna. Nemá téměř žádné možnosti nastavení. Největší nevýhodou je nemožnost přehrávat šifrovaná videa. To však není případ této aplikace.

Přehrávač umožňuje přehrávání video souborů ve formátu mp4 ze serveru, stačí nastavit URL videa. Zároveň je přehrávač poměrně rychlý a úsporný co se týče kódu. Video se přehrává v defaultním *VideoView*. K němu lze přidat základní ovladač, ten umožňuje posun videa do libovolného času a play/pause. Případně se dá defaultní ovladač upravit podle potřeb nebo vytvořit vlastní.

2.2.2 ExoPlayer

Jedná se o knihovnu[7] oficiálně podporovanou Googlem, která vznikla, aby mohla být rychle vyvíjena a aktualizována a byla nezávislá na aktualizacích Android SDK, které jsou méně časté. Knihovna podporuje širokou škálu formátů.

Umožňuje přehrávání šifrovaných videí pomocí Digital Rights Management (DRM). DRM zprostředkovává technické metody, jejichž účelem je ovládat či omezovat používání obsahu digitálních médií. Používá se zejména pro ochranu autorských práv a dodržování licenčních podmínek vztahujících se k obsahu.

Další výhodou je technologie Dynamic adaptive streaming over HTTP (DASH), která video přehrává jako řadu segmentů. Při změně rychlosti připojení může přehrávat další segmenty s jiným datovým tokem pomocí změny rozlišení a zachovat tak plynulé přehrávání.

Přehrávač využívá vlastní *ExoPlayerView*, ke kterému lze přidat defaultní Android ovladač nebo lze nastavit vlastní.

ExoPlayer podporuje také Smooth Streaming, což je stejná technologie jako DASH, ale vytvořená společností Microsoft.

Vzhledem k tomu, že videa jsou na serveru uložena pouze v jednom rozlišení a kvalitě, DASH nemá smysl využít. Stejně tak videa nejsou nijak chráněna pomocí DRM, tudíž je použití této knihovny zbytečné.

2.2.3 JW Player

Přehrávač dostupný z [23] byl doporučen producentem CATVUSA, protože ho používají na webu a existuje i jeho SDK pro Android. Přehrávač nabízí podobné možnosti jako výše zmiňovaný ExoPlayer, ale na rozdíl od něj je placený.

Podstatnou výhodou, kterou přináší, je možnost nahrání videí v různé kvalitě přímo na servery JW playeru a následné využití DASH.

Zadavatel má sice zaplacenou licenci, ale pouze na starší verzi přehrávače. V současné době však výrobce požaduje měsíční platby za využívání služeb. Vzhledem ke značně omezenému rozpočtu organizace nepřichází využití této možnosti v úvahu.

2.2.4 JieCao Video Player

Další možností je tato open-source knihovna[8]. Její použití je velmi jednoduché, nabízí pouze základní přehrání souboru z URL adresy a základní změny view. To pro potřeby tohoto projektu bohatě stačí.

V podstatě se jedná o rozšíření základního Android MediaPlayeru o další funkcionalitu týkající se zejména jednodušší implementace funkcí, které by pro správné fungování aplikace musely být doimplementovány.

Hlavní výhodou tohoto řešení je zjednodušená práce s přehráváním na celou obrazovku. Základní ovladač, který je umístěn ve view přehrávajícím

video, obsahuje tlačítko pro celou obrazovku a na rozdíl od ostatních řešení si knihovna sama pamatuje aktuální čas videa. Tudíž odpadá práce s ukládáním stavu videa a není potřeba tvořit další fragmenty pro přehrávání na celou obrazovku.

Další výhodou je podpora gest, kterou knihovna umožňuje při přehrávání na celou obrazovku. První gesto je horizontální přejetí po displeji čímž lze měnit hlasitost. Vertikální přejetí po displeji pak umožňuje rychlé posouvání ve videu.

Tato knihovna splňuje všechny podmínky pro funkčnost aplikace, přináší i další výhody a navíc značně zjednodušuje implementaci, proto bude při tvorbě aplikace použita.

2.3 Přehrávání rádia

U rádia, na rozdíl od videa, jsou potřeba pouze základní ovládací prvky. Tedy tlačítko pro přehrávání a pauzu, ovládání hlasitosti a informace o aktuální písničce. Stejně tak zde není nic jako DASH nebo DRM kvůli nimž by bylo potřeba používat sofistikovanější řešení přehrávače rádia.

Pro tyto účely bude stačit základní Android MediaPlayer, ten splňuje vše, co je potřeba pro tuto sekci.

Rádiové stanice se liší mimo žánru zejména ve struktuře uložení souborů na serveru a v systému přehrávání. Aby se zamezilo opakovanému přehrávání stejné skladby a naplno se využily všechny soubory, bude aplikace v lokální databázi obsahovat informace o tom, které skladby již byly přehrány.

2.4 Aplikace pro TV

Pro přehrávání videa v této aplikaci bude také bohatě stačit defaultní Android MediaPlayer. Již z principu toho, že jde o televizní aplikaci zde bude přehrávání vždy na celou obrazovku. Nebude se zde řešit ani rotace obrazovky a další podobné problémy spojené s přehráváním v mobilní verzi, proto stačí využít základní MediaPlayer a není potřeba k tomuto projektu připojovat externí knihovnu, která by v tomto případě zbytečně zvětšovala velikost projektu.

Kapitola 3

Návrh

V této kapitole bude nejprve teorie ohledně webového API. Poté bude proveden návrh webového API pro komunikaci se serverem a databáze pro ukládání dat v zařízení. Oba návrhy spolu úzce souvisí, protože v databázi budou uložena zejména data ze serveru. Na závěr bude proveden přibližný návrh vzhledu.

3.1 API

API[9] označuje rozhraní pro programování aplikací. Jde o sadu tříd, metod, nebo protokolů nějaké knihovny, webu, jiné aplikace, ale i jádra operačního systému, které může programátor použít.

API lze rozdělit do těchto základních kategorií:

- obecné - jedné se o celé API, například programovacího jazyka
- konkrétní - řeší konkrétní problém, například Google Maps API
- jazykově závislé - lze použít pouze v daném programovacím jazyce, využívá syntaxy a základní prvky tohoto jazyka
- jazykově nezávislé - lze volat z různých programovacích jazyků, například webové API

3.2 REST

Representational State Transfer (REST)[10] je architektura rozhraní představená v roce 2000 v disertační práci Roye Fieldinga, jednoho se spoluautorů HTTP. REST je datově orientován, určuje, jak přistupovat k datům(zdrojům) a zajišťuje k nim jednotný a snadný přístup. Všechny zdroje musí být jednoznačně identifikovány. REST definuje 4 základní metody pro přístup ke zdrojům. Pro jejich popis metod se bude pracovat s fiktivním seznamem videí.

3.2.1 GET

Jedná se o základní metodu. Ta pouze získává data ze zdroje, nikterak s nimi na straně zdroje nemanipuluje. Na rozdíl od ostatních metod, defaultně nepotřebuje autentizaci. Autentizaci potřebuje pouze pokud chceme získání dat zprostředkovat pouze někomu.

```
GET /videos.formát
GET /videos/1234.formát
GET /videos.formát?count=10&rating=4
```

První GET požadavek vrátí seznam videí, druhý vrátí konkrétní video. Oba požadavky vrátí odpověď v daném formátu. Filtrování vracených výsledků lze vidět na třetím příkladu, používají se zde běžné query parametry, které se dávají za otazník.

3.2.2 POST

Pomocí těchto požadavků se do zdroje přidávají nová data. Protože identifikátor určuje až server, odesílají se nově vkládaná data bez něj. Na příkladu je vidět vytvoření nového videa, kde data musí být v serverem podporovaném formátu. Pokud se jedná o webové API a vše proběhne v pořádku, vrátí se v odpovědi identifikátor, pod kterým nově vložená data nalezneme.

```
POST /videos/video
{data}
```

3.2.3 PUT

Tato metoda slouží k editaci již vytvořených dat. Je podobná metodě POST, s tím rozdílem, že požadavek je již volán na konkrétní identifikátor. Následující příklad aktualizuje video s daným identifikátorem.

```
PUT /videos/1234
{data}
```

3.2.4 DETELE

Touto metodou lze zdroje mazat. Způsobem požadavku je podobná metodě GET.

```
DELETE /videos/1234
```

3.3 Návrh webového API

Webové API bylo navrženo ve spolupráci s webmasterem webových stránek organizace. Většina požadavků je pouze typu GET. Pouze uživatelská sekce vyžaduje i jiné metody pro manipulaci se zdroji.

Již při návrhu zjištění možností, jak API na straně serveru realizovat, bylo zjištěno jedno zásadní omezení. Web je vytvořen pomocí redakčního systému WordPress. Ten ukládá data do databáze velmi komplikovaným způsobem. Ruční vytvoření API pomocí dotazů do databáze například pomocí jazyka PHP bylo tedy kvůli složitosti dotazů ihned zavrhnuto. Bylo tedy navrhnuto řešení webové části API pomocí pluginu. Vybrán byl plugin WordPress REST API(Version 2)[11]. Ten byl vybrán vzhledem k dobrým předchozím zkušenostem webmastera s tímto pluginem. Zároveň plugin splňoval všechny požadavky pro vytvoření API k této aplikaci, avšak vzhledem k určitým omezením pluginu, bylo API navrhnuto co nejjednodušeji.

Co se formátu dat týče, používají se pro webové API zejména formáty JSON a XML. XML je vhodnější pro složitější struktury a tam, kde je kladen důraz na datové typy. Oproti tomu JSON má jednodušší syntaxi a celkově se snadněji používá. Pro toto API byl vybrán formát JSON, protože v API nebudou žádné složité struktury pro jazyk Java, v němž se píší aplikace pro android existují knihovny pro převod na objekty a navíc vybraný plugin pro WordPress podporuje pouze tento formát.

Samotná komunikace mezi API a aplikací bude tedy probíhat tak, že aplikace vyšle požadavek na určitou URL a jako odpověď se vrátí data ve formátu JSON. Data budou následně uložena v aplikaci.

V následujících kapitolách jsou popsány jednotlivé API metody, včetně všech parametrů a struktur, které vrací.

3.3.1 API pro video

```
http://catvusa.com/api/video?api=1
http://catvusa.com/api/video?api=1&type=vybrana_kategorie
```

První adresa vrací seznam všech videí dostupných na serveru jako pole ve formátu JSON. Pro filtrování jednotlivých kategorií lze použít druhý odkaz. Vzhledem k tomu, že uživatel nijak neovlivňuje videa na serveru, je zde pouze metoda GET.

V poli jsou uloženy všechny informace o každém videu, které jsou potřebné pro funkčnost aplikace.

- id - identifikátor videa
- type - kategorie (broadcast, class, cooking)
- name - název videa

- date - datum přidání videa
- url – adresa příspěvku s videem
- direct url – adresa mp4 souboru
- image – hlavní obrázek k videu
- chapters – pole s popisky a obrázky
- excerpt – základní popis videa

Jednotlivé položky chapters obsahují:

- img – adresa doprovodného obrázku
- desc – popis k obrázku

3.3.2 API pro radio

```
http://catvusa.com/api/radio?api=1
```

Na adrese výše je dostupný seznam všech písniček a doprovodných informací k nim. Uživatel opět nijak neovlivňuje obsah serveru v této sekci. Tudíž je zde opět pouze GET.

- description - popis písničky
- file - adresa mp3 souboru
- intro - intro ve formátu mp3 k písničce
- genre - žánr
- image – obrázek k písničce

3.3.3 API pro POI

```
http://www.catvusa.com/data/turist.json
```

Na této adrese je dostupný seznam adres s turistickými informačními centry a veškerými informacemi, které se k nim podařilo získat. Producent televize i autor práce se snažili získat co nejvíce dat. Bohužel žádná dostupná instituce nemá více než seznam turistických center s jejich adresou a polohou. Turistických center je sice před 600, ale lepší by však bylo mít data například k památkám, muzeím, zkrátka oblíbeným turistickým místům. Stejně tak by bylo lepší mít více kontaktních informací jako otevírací dobu, telefon nebo webovou adresu. Nicméně taková data se nepodařilo sehnat a při realizaci budou tedy ke každému místu dostupné tyto informace:

- id - identifikátor místa
- name - název místa
- city - město ve kterém se místo nachází
- street - ulice místa
- postal code – poštovní směrovací číslo daného místa
- latitude – zeměpisná šířka
- longitude – zeměpisná délka

Jedná se opět pouze o GET metodu, protože uživatel neovlivňuje data na serveru.

3.3.4 API pro kvíz

```
http://www.catvusa.com/api/quizzes/?api=1
```

Ani zde nejsou ovlivňována data v serverové databázi z aplikace, proto se jedná o GET metodu. Metoda vrací seznam všech kvízů a veškerá data pro sestavení kvízu. Kvíz je definován pomocí:

- info - informace o kvízu, v aplikaci je využit z informací pouze název
- questions - pole jako seznam otázek

Každá otázka pak obsahuje:

- a - pole odpovědí na otázku, každá odpověď je textově zadaná, navíc jedna z odpovědí obsahuje prvek correct označující správnou odpověď
- q - textové zadání otázky

3.3.5 API pro uživatelskou sekci

```
http://www.catvusa.com/wp-json/wp/v2/users/me
```

Tato adresa slouží pro přihlášení uživatele. Jde opět o GET metodu a vrací dva možné formáty odpovědi. Je zde ale do hlavičky požadavku potřeba přidat autorizaci. Plugin bohužel umí pracovat pouze s basic authentication[12] neboli jednoduchým ověřením. Tato metoda ověření funguje tak, že se ze jména a hesla vytvoří jeden textový řetězec, ten se zakóduje metodou base64 a výsledek je přidán do hlavičky HTTP požadavku. Při tvorbě řetězce jsou jméno a heslo odděleny dvojtečkou.

Metoda base64 zajistí to, že lze přenést i znaky, které nejsou povoleny v HTTP. Neprobíhá zde však žádné zašifrování dat. Jméno i heslo lze rozkódovat bez jakéhokoliv klíče nebo podobného zabezpečení, jež obsahují jiná autorizační metody. Obecně je tedy potřeba, aby spojení mezi klientem a serverem bylo bezpečné.

Pokud uživatel zadal špatné heslo nebo jméno, vrátí se ze serveru odpověď obsahující návratový kód 401, neboli neoprávněný přístup, a v JSON v odpovědi je chybová hláška. Při správně zadaných údajích je vrácen návratový kód 200 OK a tělo odpovědi obsahuje informace o uživateli. Jedinou informací z této odpovědi, která je využitelná v aplikaci, je jméno uživatele. Dále jsou zde informace jako obrázek a url profilu. Obrázek je však pro všechny uživatele stejný a adresa url neposkytuje žádné přidané informace, proto v aplikaci nejsou tato data využita.

3.4 Návrh databáze

Použita bude SQLLite databáze, kvůli dostupnosti přímo v systému android bez použití dalších knihoven. Do databáze se uloží veškerá data, která přijdou z API a budou využita v aplikaci. Data v DB jsou uložena i po ukončení aplikace nebo restartování zařízení. Díky tomu vše uložené bude fungovat i v režimu offline. Kvůli úspoře místa v zařízení a dat při komunikaci s API, budou uloženy pouze textové informace. Videa, písničky, obrázky a mapové podklady budou vyžadovat internetové připojení.

K dalším úsporám, zejména co se týká limitů přenesených dat, poslouží cachování obsahu. To v podstatě znamená, že při opětovném návratu do již stažené kategorie, která vyžaduje data ze serveru, bude ověřeno stáří dat v zařízení. Podle stáří se buďto použijí již uložená data nebo se stáhnou data nová. Délka uložení dat se také nazývá cache time. Vzhledem k tomu, že u různých kategorií se v různých intervalech aktualizuje obsah na webu, bude i interval v aplikaci různý pro různé kategorie.

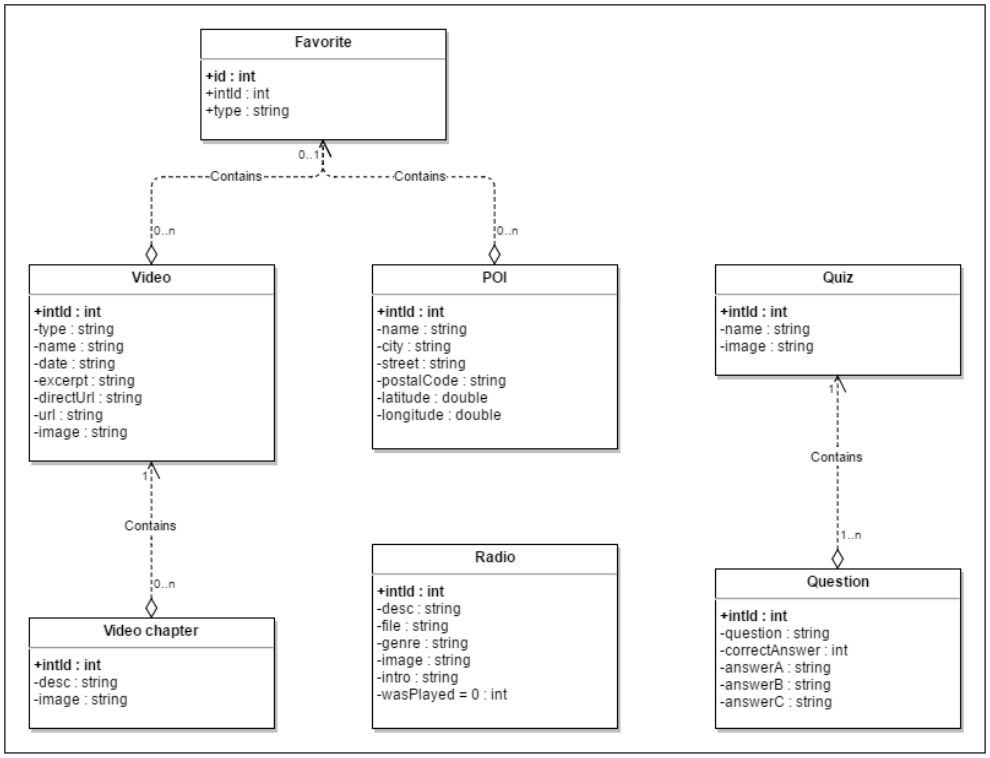
Nejčastěji obnovovanou kategorií jsou videa. I ta se však přidávají maximálně jednou týdně. Tato kategorie je však z celé aplikace nejdůležitější, proto je potřeba zajistit, že se nové video dostane k uživatelům co nejdříve. Cache time této sekce bude tedy nastaven na jeden den. Naopak v sekci rádio se hudba skoro nemění. Cache time zde bude tedy nastaven na 14 dní. Vhledem k náhodnému přehrávání písniček a k tomu, že aplikace stejně neobsahuje seznam všech písniček, uživatel stejně ihned nepozná, že jsou staženy nové písničky. Navíc je to kategorie, kde je vrácen největší počet výsledků z API, tudíž je na výkon i data nejméně úsporná. U POI a kvízů se pravděpodobně data také nebudou skoro obnovovat. Pokud by však bylo potřeba dostat nějakou změnu mezi uživatele, bude cache time nastaven na týden. Poslední kategorií, kde se komunikuje s API je uživatelská sekce. Zde nebude žádný cache time. API se bude vždy volat, když to bude potřeba, kvůli zajištění vždy aktuálních

dat. Toto API bude voláno při přihlášení, odhlášení, změně údajů a ověření v rádiu, že má uživatel zaplaceno premium členství. Zde je vždy potřeba mít aktuální informace ze serveru.

Cache time a informace o uživateli nebudou uloženy v databázi, ale v takzvaných *SharedPreferences* (SP). Ty využívají ukládání pomocí klíč-hodnota a lépe se hodí pro ukládání jednoduchých, málo objemných a nestrukturovaných dat, typicky primitivních datových typů. Stejně jako databáze, přežije tato paměť ukončení aplikace i restart zařízení.

Mimo dat z API je dále potřeba uchovat informace o uživatelových oblíbených položkách a o tom, jaká písnička již byla přehrána. Obrázek 3.1 zobrazuje schéma databáze. Vzhledem k delšímu cache time u rádia bude informace o přehrání uložena přímo v tabulce se všemi písničkami a při obnově dat ze serveru budou písničky opět vybírány z celé množiny. Naproti tomu oblíbené položky mají vlastní tabulku, aby při obnovení tabulky ze serveru nebyly položky ztraceny.

Mohlo by se zdát, že společná tabulka oblíbených pro videa i POI může způsobit problém, pokud v ní bude video i POI se stejným ID. Videa mají však ID až od 2000 a více, oproti tomu POI jsou číslovaná od 1, je jich cca 600 a další nebudou přibývat. Pokud by se tak do budoucna dělo, stačí vytvořit druhou tabulku se stejnou strukturou a ukládat oblíbené zvlášť.



Obrázek 3.1: Schéma databáze

3.5 Návrh vzhledu

3.5.1 Wireframe

Pro prvotní návrh vzhledu byly použity takzvané wireframy[13], někdy též nazývané mock. Ty definují rozmístění prvků na obrazovce zařízení. Pro každou výrazně různou obrazovku aplikace by tedy měl být vytvořen wireframe. Nejedná se o grafický návrh, neobsahuje ani obrázky, pokud to není nezbytně nutné, tak je tento návrh černobílý. Jedné se o skicy, které mají být ideálně tvořeny pouze pomocí základních tvarů a texty. Každý vetší projekt by je měl obsahovat a měly by obsahovat všechny funkční požadavky zadavatele. Wireframy se tvoří ještě před samotným vývojem aplikace a dávají se schvalovat zadavateli. Díky tomu jsou pak jasně vymezeny požadavky zadavatele i práce programátora. Nejedná se však o přesný vzor jak má výsledná aplikace vypadat a zejména velikosti prvků se mohou výrazně lišit.

Wireframy se dají kreslit ručně na papír nebo ve speciálním k tomu určeném software, ale i například vystříhat z papíru a skládat jako skládačka. Na návrhy k této práci byl použit SW Balsamiq Mockup 3[14]. Tento SW obsahuje komponenty pro tvorbu návrhů různým druhů. Co se týče mobilních platforem, obsahuje v základu pouze prvky pro iOS. Nicméně, pomocí knihoven se do něj dají přidat další UI prvky. Byla tedy přidána knihovna pro android a v kombinaci s dobrými zkušenostmi s tímto SW v různých dalších projektech vznikl ideální nástroj pro tvorbu návrhů. Z programu se dají wireframy exportovat do různých formátů. Pro tuto práci byl zvolen formát do PNG. Po vytvoření návrhů vzhledu proběhla konzultace jak s producentem TV, tak s vedoucím práce, která vedla ke schválení návrhů.

3.5.2 Zvyklosti systému android

Při návrhu aplikace bylo potřeba dbát na zvyklosti v tomto OS dostupných z [15]. Ty se řídí zejména doporučeními Googlu, ale i dlouhodobou zvyklostí uživatelů tohoto systému na určité prvky a jejich chování. Při nedodržení těchto konvencí může být uživatel zmatený, špatně se v aplikaci orientovat nebo špatně chápat její chování. Z toho důvodu se raději podívá po nějaké alternativě a nebo aplikaci špatně hodnotí.

V následujících kapitolách jsou popsány jednotlivé obrazovky aplikace, včetně zvyklostí na nich zakomponovaných. Popisky jsou také doplněny wireframy pro lepší představu vzhledu.

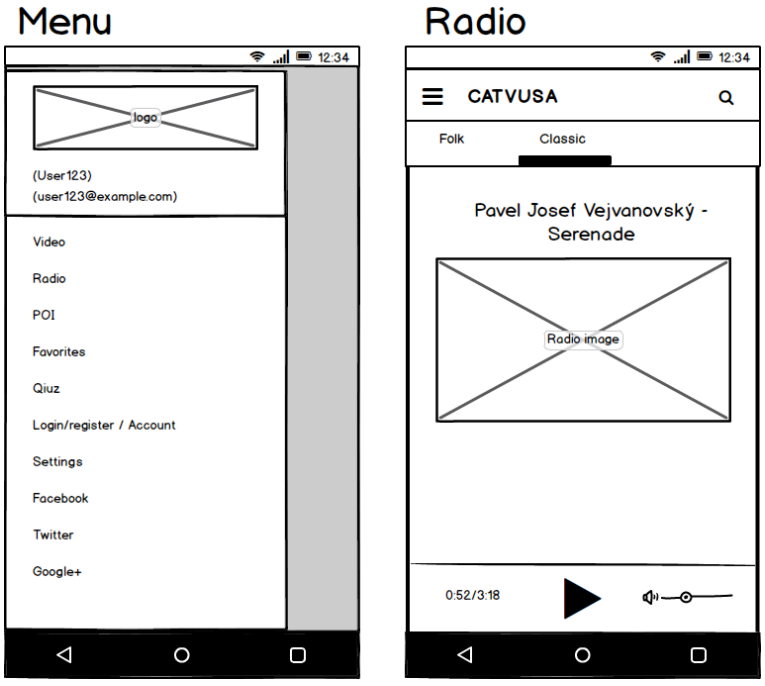
3.5.3 Menu

Menu je základní rozcestník celé aplikace. Zde je hned vidět základní zvyklost systému android. Menu vyjíždí vždy z levé strany displaye a to přetažením prstu od levého okraje obrazovky nebo kliknutím na tři vodorovné čárky umístěné vždy v levém rohu. Menu v horní části bude zobrazovat logo společnosti

a pod ním základní údaje o uživateli (pokud je přihlášen). Následovat bude seznam všech hlavních částí aplikace mezi kterými lze přepínat. Menu bude přítomno na všech hlavních obrazovkách jednotlivých kategorií.

3.5.4 Rádia

Rádio obsahuje dva žánry. Mezi nimi bude možné přepínat buďto kliknutím na název žánru nebo přejetím prstu do strany. Tomuto prvku se v systému android říká *ViewPager* a uživatelé jsou na jeho chování navyklí. Stránka žánru pak bude obsahovat jméno autora, název písničky a obrázek. Ve spodní části pak bude umístěno ovládání rádia a aktuální stav skladby. Tyto prvky budou přítomny dole z důvodu jednoduchého ovládání jednou rukou. Horní část, ve které bude umístěno menu obsahuje dále název aplikace nebo název sekce pro lepší orientování v aplikaci. V této části bude také tlačítko pro vyhledávání. To je v sytému android ve většině aplikací umístěno právě v pravém horním rohu. Celá tato horní lišta se nazývá *ActionBar* a uživatelé jsou zvyklí v ní najít menu a základní ikony pro ovládání aplikace.



Obrázek 3.2: Vzhled menu a rádia

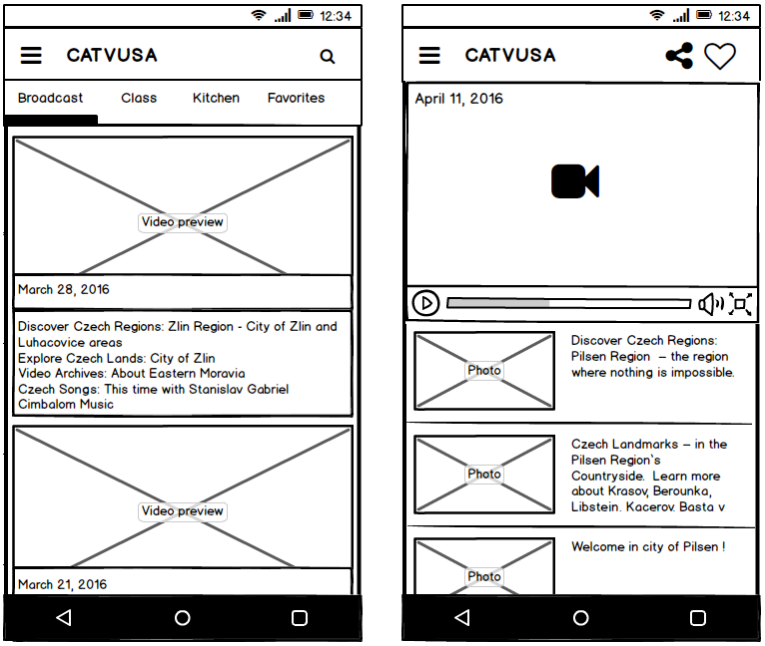
3.5.5 Seznam videí

Video bude obsahovat tři základní kategorie a sekci oblíbených. Mezi sekcemi bude možné přepínání stejně jako u rádia. Seznam pak bude vždy mít stejný

vzhled, pouze bude obsahovat různá videa. Videa budou zobrazená jako seznam, kde u každého videa bude úvodní fotografie, název a to hlavní, o čem video pojednává.

3.5.6 Detail videa

Hlavní část této obrazovky bude zabírat video přehrávač, v jeho horní části bude název videa. Pod přehrávačem bude seznam doprovodných textů a obrázků k nim. V horní části budou namísto tlačítka pro vyhledávání přítomna tlačítka pro sdílení videa a jeho uložení do oblíbených.



Obrázek 3.3: Vzhled seznamu a detailu videa

3.5.6.1 Mapa POI

Hlavní část obrazovky zde zabere mapa. Ta bude zobrazovat všechna dostupná místa. Po kliknutí na dané místo se zobrazí informace o místě spolu s možností spustit navigování do místa (pomocí externí aplikace).

3.5.7 Seznam POI

Tato sekce bude zobrazovat seznam všech abecedně seřazených míst. U každého místa bude uveden název a adresa. Pomocí tlačítek bude možné místo buďto zobrazit na mapě popsané výše nebo do něj opět pomocí externí aplikace zahájit navigaci. Dále půjde POI sdílet a uložit do oblíbených. Ty bude možné případně zobrazit na další záložce této kategorie.

Obrázek 3.4: Vzhled mapy a seznamu míst

3.5.8 Seznam kvízů

Zde bude seznam všech dostupných kvízů s reprezentovaných názvem.

3.5.9 Spuštění kvízu

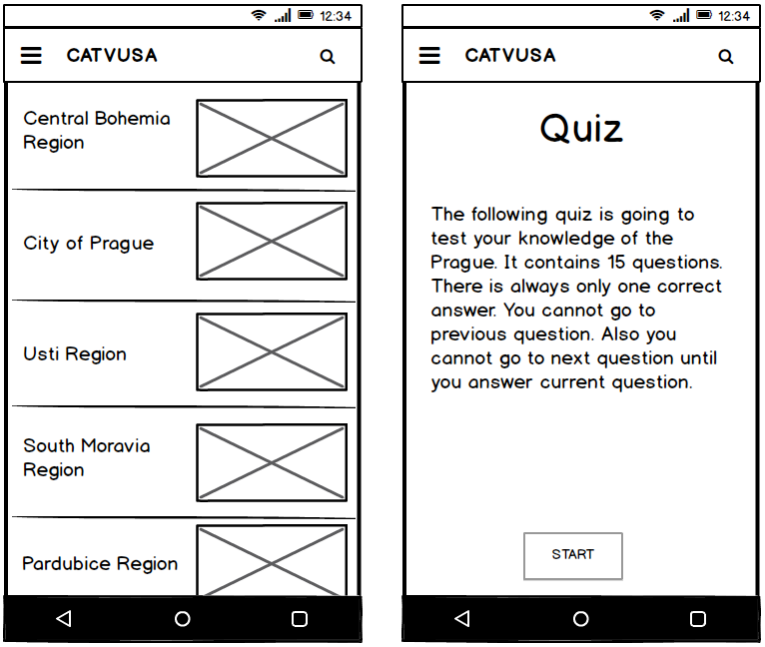
Na této obrazovce bude zobrazeno vysvětlení pravidel o průběhu kvízu. Bude zde také název kraje, kterého se kvíz týká. Ve spodní části se bude nacházet tlačítko pro spuštění kvízu. Celá sekce kvízu od této obrazovky až po jeho ukončení bude stejná jak pro spuštění ze seznamu, tak automatické spuštění po skončení videa.

3.5.10 Kvízová otázka

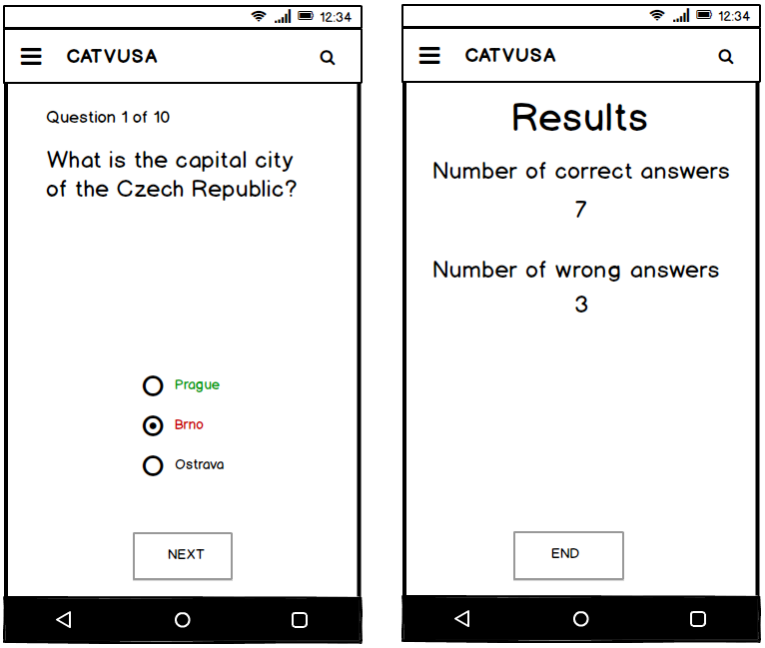
V horní části bude číslo aktuální otázky, pod ním text otázky. Dále budou následovat tři možné odpovědi. Spodní tlačítko pro pokračování na další otázku se zobrazí až po kliknutí na některou z odpovědí. Ve stejnou chvíli se uživateli zeleně obarví správná odpověď a pokud odpověděl špatně, tak červeně špatná odpověď.

3.5.11 Konec kvízu

Na této obrazovce bude pouze vyhodnocení kvízu a tlačítko pro jeho ukončení. Vyhodnocení bude obsahovat počet správných a špatných odpovědí.



Obrázek 3.5: Vzhled seznamu a spuštění kvízu



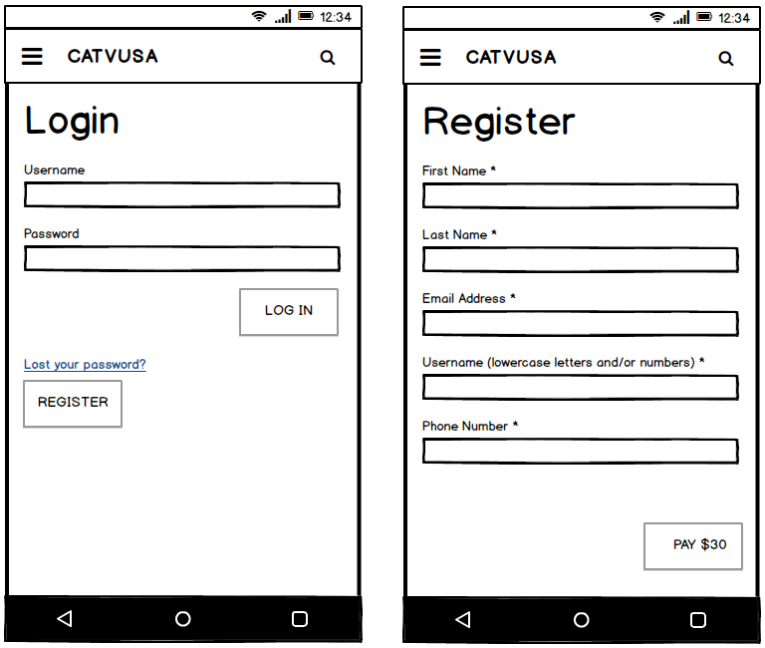
Obrázek 3.6: Vzhled otázky a výsledků

3.5.12 Přihlášení

Tato sekce bude obsahovat formulář pro přihlášení obsahující prvky pro vložení emailu a hesla. Tlačítko pro přihlášení spustí proces přihlášování. Tlačítko pro zapomenuté heslo otevře webový prohlížet s adresou na stránku pro obnovu hesla. Tlačítko registrace spustí příslušnou sekci.

3.5.13 Registrace

Zde bude formulář obsahující všechny informace, které jsou potřeba pro zaregistrování. Informace budou stejné jako na webu projektu. Následně bude možné zaplatit 30 dolarů přes PayPal.



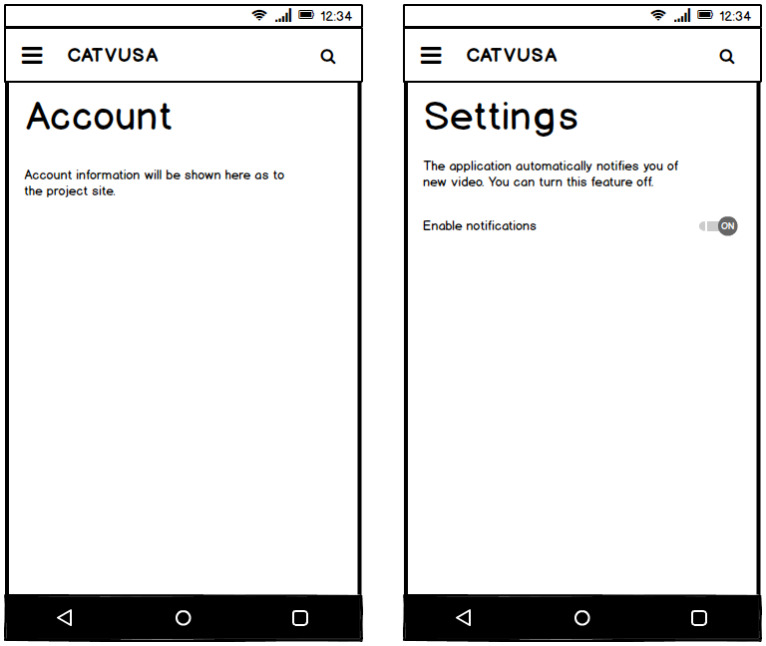
Obrázek 3.7: Vzhled přihlášení a registrace

3.5.14 Účet

Jak již bylo popsáno výše, z API pro uživatelskou sekci je jediná použitelná informace jméno. Proto zde bude zobrazeno společně s emailem, který uživatel zadal při přihlášení. Ve spodní části obrazovky pak bude tlačítko pro odhlášení.

3.5.15 Nastavení

V sekci nastavení bude možnost vypnout notifikace upozorňující na nová videa.



Obrázek 3.8: Vzhled uživatelského účtu a nastavení

3.6 Návrh TV

Aplikace pro TV se bude zásadně lišit pouze ve vzhledu. Ostatní části jako webové API a databáze budou pro obě aplikace stejné, jediný rozdíl bude v tom, že budou využity pouze ty části, které se této verze aplikace týkají. Přesněji řečeno, z API a DB budou využity pouze čísti týkající se videí a rádia.

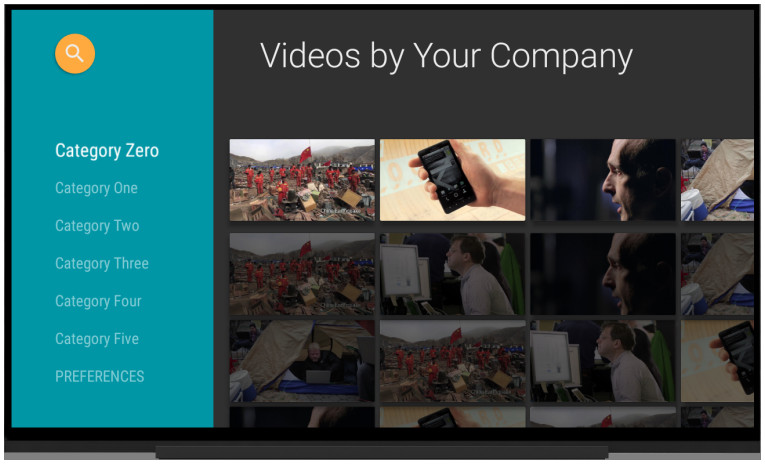
Samotný Google vydal doporučení, jak by měl vypadat přehrávač videí pro Android TV[16]. Díky tomu má většina aplikací podobný vzhled a uživatelům se v nich dobře orientuje. Proto pro tuto aplikaci nebyly vytvořeny wireframes a aplikace bude co nejvíce podobná doporučením.

Obrázek 3.9 ukazuje základní obrazovku televizní aplikace, jak by měla vypadat podle doporučení. V levé části se nachází menu, to obsahuje seznam kategorií. V této aplikaci to budou tři kategorie pro video a dvě pro rádio. Při posun mezi kategoriemi se posouvá i podsvícený řádek. Po posunu vpravo se pak uživatel rovnou nachází na řádku s videí z dané kategorie. Při posunu přímo mezi videi částečně zajede levé menu, tím je získán větší prostor pro videa.

Po vybrání některého videa se zobrazí jeho popis s možností video přehrát. Ve spodní části pak bude seznam 10-ti podobných videí nastylovaný stejně jako seznam videí na úvodní stránce. Podobná videa budou náhodně vybraná videa ze stejné kategorie. Při samotném přehrávání bude video rovnou spuštěno na celou obrazovku s plovoucím ovládáním. Toto ovládání bude obsahovat název videa, stav přehrávání, tlačítka pro pauzu/přehrávání, přechod na další a

předchozí video a možnost rychlého posouvání ve videu směrem vpřed i zpět. Pod ovladačem bude opět zobrazen seznam 10-ti podobných videí vybraných podle stejného klíče. Celý ovladač vždy po několika vteřinách zmizí a přehrávání tak nebude nic rušit. Stisknutím tlačítka pro pauzu na dálkovém ovladači se video zastaví a ovladač na obrazovce se opět zobrazí.

Sekce rádia nebude obsahovat úplný detail jako u přehrávání videa. Po kliknutí na rádio na hlavní stránce se zobrazí detail s úvodní fotografií rádia a tlačítkem pro spuštění rádia. Na obrazovce se vždy po přechodu na další písničku bude měnit text a obrázek (pokud budou tyto informace dostupné). Co se týká ovládání bude zde pouze play/pause tlačítko. Změnu hlasitosti uživateli zajistí dálkový ovladač od TV. Nic jiného není v této sekci potřeba.



Obrázek 3.9: Vzhled TV aplikace

Kapitola 4

Realizace

Aplikace pro OS android se píší v jazyce Java. Oficiálním vývojovým prostředím podporovaným tvůrci tohoto systému a navíc plně uzpůsobeným pouze pro tvorbu aplikací pro tento systém je Android Studio. Toto IDE tedy bylo použito pro tvorbu této práce.

4.1 Moduly

Při zakládání nového projektu je programátor vyzván k zadání platforem, na kterých bude možné projekt spustit. Jedná se o platformy:

- Telefon a tablet
- Wear - chytré hodinky
- TV
- Android Auto
- Glass - chytré brýle

AS pak vytvoří mimo spousty dalších souborů potřebných pro spuštění aplikace také modul pro každou vybranou platformu. Zde byly tedy hned na začátku vytvořeny dva moduly a sice jeden pro mobilní telefony a tablety a druhý pro TV. Modul je hlavní komponenta v projektu, která pokud je tak nastavena, je samostatně spustitelná.

Vzhledem k tomu, že implementace obou částí se výrazně liší jak ve vzhledu, tak ve třídách používaných pro logiku aplikace, je oddělení do samostatných modulů namístě. Největší výhodou tohoto modulárního vývoje je, že je logika oddělena a programování je daleko přehlednější. Výhoda je také v rychlosti buildování aplikace. V případě, že se programuje část pro TV je zbytečné překládat i aplikaci pro mobily. Výhody to však přináší i pro uživatele. Protože se

Tabulka 4.1: Přehled modulů v aplikaci

Název	Popis	Samostatně spustitelný
base	Základní modul obsahující společné prvky	Ne
mobile	Modul obsahující prvky pro mobilní platformu	Ano
television	Modul s prvky pro televizi	Ano

oba moduly zvlášť buildují, jsou také jako výstup dva .apk soubory (intalační soubor pro platformu android). Každý z nich je menší, než kdyby bylo vše v jednom a tím je ušetřeno místo v paměti zařízení.

Vzhledem k tomu, že některé části kódu jsou společné pro obě platformy, byl vytvořen třetí tzv. základní modul. Do něj byly vkládány všechny třídy a soubory, které jsou společné a díky tomu nedochází ke zbytečné duplikaci kódu. Tím je lépe zajištěn udržitelný rozvoj aplikace. Tento modul je vytvořen jako android knihovna, není tedy samostatně spustitelný a připojuje se k nadřazeným modulům stejně jako jakákoliv jiná knihovna. Připojování knihoven do projektu bude popsáno dále v textu.

Připojování modulů jako knihoven je další výhodou tohoto stylu vývoje. Pokud se podřazený modul vytvoří dostatečně obecný, lze ho připojovat i k jiným projektům jako knihovnu a používat ho díky tomu kdekoliv. I knihovní moduly v sobě mohou využívat další moduly a knihovny. Jedná se v podstatě o kompletní projekty, které pouze nejdou samostatně spustit. Přehled modulů v aplikaci zobrazuje tabulka 4.1.

4.2 Build gradle

Každý model obsahuje tento konfigurační soubor. Navíc je v projektu tento soubor i pro celý projekt. V hlavním konfiguračním souboru v této aplikaci je pouze informace o tom, z jakého repozitáře stahovat připojené knihovny.

Zajímavější jsou ale tyto soubory u jednotlivých modulů. V souborech je napsáno oproti jaké verzi androidu se má aplikace kompilovat a jakou minimální verzi aplikace podporuje. Dále také tzv. target verzi, což znamená, že programátor již do implementace zahrnul změny, které přišli s verzí kterou zde deklaruje, oproti verzi kterou deklaroval předtím. Tyto verze se nečíslují přímo jako verze androidu, ale jako SDK verze. Číslování začalo na verzi 1 a s každou oficiální verzí androidu se zvedá o 1. Takže aktuálně je nejnovější verze androidu 7.0 ale verze SDK je 24. Dále je zde verze aplikace a hlavně seznam všech knihoven a modulů, které modul využívá.

V base modulu se mimo společného kódu nachází právě i společné knihovny. Většina knihoven se nachází v takzvaných repozitářích. Díky tomu není nutné stahovat a připojovat k projektu .jar soubor. Na následujícím příkladu je vi-

Tabulka 4.2: Knihovny použité v aplikaci

Modul	Název	Verze	Účel
base	multidex	1.0.0	Podpora spuštění na starších verzích androidu
base	stetho-okhttp3	1.3.1	Ladění aplikace v chrome a síťová komunikace
base	gson	2.6.2	Převod JSON do Java objektů
base	glide	3.7.0	Načítání obrázků z URL
base	support-v4	24.2.1	Kompatibilita se starší verzí SDK
base	support-v7	24.2.1	Kompatibilita se starší verzí SDK
mobile	jiecaovideoplayer	4.8.3	Přehrávání videí
mobile	play-services-maps	9.6.1	Použití Google map
mobile	android-maps-utils	0.4.4	Shlukování POI do clusterů při oddálení mapy
mobile	design	24.2.1	Vysouvací panel u mapy
mobile	cardview-v7	24.2.1	Seznamy pomocí tzv. card view
television	leanback-v17	24.2.1	Aktivity a fragmenty uzpůsobené pro TV
mobile, telvision	base	-	Základní modul se společným kódem

dět připojení knihovny pro práci s Google mapami. Poslední údaj udává číslo verze, pokud tedy vyjde aktualizace knihovny, stačí zvýšit číslo verze a není potřeba znovu stahovat .jar soubor. Přehled všech připojených knihoven zobrazuje tabulka 4.2.

Listing 4.1: Ukázka připojení knihovny

```
compile 'com.google.android.gms:play-services-maps:9.6.1'
```

Verze aplikace je zatím 1.0. Vzhledem k tomu, že android TV je až od androidu 5.0 neboli SDK 21 je minimální SDK nastaveno na tuto hodnotu. To zajišťuje kompatibilitu na všech televizorech s OS android. Oproti tomu u mobilní platformy se kvůli stáří prvních verzí těžko zajišťuje kompatibilita na úplně všech zařízeních. Proto je potřeba určit hranici, která neodřízne příliš uživatelů od použití aplikace. Zde byla vybrána SDK verze 16 neboli android 4.1.x. Jak je vidět z tabulky 4.3 je tím zaručena funkčnost na 97.3% všech zařízení. Tabulka byla vytvořena z dat z[20]. Target SDK verze je pak všude nastavena na 23.

Tabulka 4.3: Přehled verzí androidu, s využitím větším než 0.1%

Verze	Název	SDK	Procento zařízení
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.3%
4.1.x	Jelly Bean	16	4.9%
4.2.x		17	6.8%
4.3		18	2.0%
4.4	KitKat	19	25.2%
5.0	Lollipop	21	11.3%
5.1		22	22.8%
6.0	Marshmallow	23	24.0%
7.0	Nougat	24	0.3%

4.3 Manifest

Jedná se v podstatě o další konfigurační soubor, který je opět obsažen v každém modulu. Je napsán v XML.

Každá android aplikace musí obsahovat seznam oprávnění. Ty pak vidí uživatel v obchodě Play, v detailech aplikace v nastavení telefonu nebo při instalaci pokud instaluje přímo .apk soubor. Od verze 6.0 se oprávnění rozdělila do dvou kategorií - normální a nebezpečné. Nebezpečné pak nejsou vidět v obchodě Play a jsou udělovány až za běhu přímo v aplikaci. Je na každém uživateli, jestli pak oprávnění udělí nebo ne. Díky tomu může uživatel udělit jenom některá oprávnění a zabránit tak aplikaci k přístupu k tomu, k čemu jí oprávnění udělit nechce. Tato oprávnění lze také ručně zapínat a vypínat v detailu aplikace v nastavení telefonu. Každá dobře napsaná aplikace by měla vyžadovat pouze ta oprávnění, která opravdu potřebuje. Čím méně oprávnění aplikace má, tím méně uživatel přemýšlí o tom, jestli aplikaci nainstaluje.

Tato aplikace nevyžaduje žádné nebezpečné oprávnění. Zde je seznam vyžadovaných oprávnění s popisem, proč je oprávnění potřeba a v jakém je modulu:

- INTERNET - Moduly television a mobile. Je potřeba pro komunikaci s API. Díky němu je možné využívat internetové připojení.
- ACCESS NETWORK STATE - Moduly television a mobile. Oprávnění je nutné pro zjištění, jestli je zařízení online.
- WAKE LOCK - Modul mobile. Bez tohoto oprávnění by nebylo možné poslouchat rádio na pozadí a upozorňovat uživatele na nová videa. Aplikace by bez něj byla ukončena při zhasnutí displaye.

- ACCESS FINE LOCATION - Modul mobile. Kvůli zobrazení toho, kde se uživatel nachází na mapě. Umožňuje přístup k aktuální poloze zařízení.
- RECEIVE BOOT COMPLETE - Modul mobile. Kvůli notifikacím na nová videa i po restartu zařízení.

V manifestu je dále definován balíček, pod kterým se aplikace instaluje, aby bylo možné mít aplikaci pro TV i mobilní zařízení v obchodě Play v jednom záznamu, musí mít oba hlavní moduly stejný název balíčku. Název vychází z webové čísti projektu a byl zvolen:

Listing 4.2: Název balíčku aplikace

```
com.catvusa.catvusacom
```

Manifest dále obsahuje název aplikace a ikonu, kterou uživatel vidí v seznamu aplikací. Dále v něm musí být uvedeny všechny aktivity, které aplikace obsahuje. Pokud je spuštěna aktivita, která není v manifestu, aplikace spadne. Aktivita je základní komponenta v tomto OS. Vše, co se zobrazuje uživateli musí být v nějaké aktivitě. Zároveň je vždy spuštěna pouze jedna.

Manifest také určuje, která aktivita je spuštěna při startu aplikace. Dále v něm musí být seznam takzvaných content providerů, ty zprostředkovávají dotazy do databáze a vrací výsledky. V mobilní verzi této aplikace jsou dva. Jeden pro veškerou práci s databází mimo vyhledávání, druhý právě pro vyhledávání. V TV verzi je pouze jeden. Je zde také seznam tzv. service. Ty běží na pozadí mimo hlavní vlákno aplikace a mohou pracovat například i když je samotná aplikace vypnuta a uživatel jí zrovna nepoužívá. Zde se service využívá pro přehrávání rádia, aby bylo možné ho poslouchat a pracovat s jinými aplikacemi. Další service je v aplikaci kvůli notifikacím na pozadí. V menifestu jsou i další data, ta však nejsou příliš podstatná.

4.4 Struktura modulů

Mimo dvou výše popsaných konfiguračních souborů a spousty dalších souborů, které se přímo netýkají vlastní implementace, lze každý modul rozdělit na dvě základní části. Část obsahující java třídy, kde je implementována logika aplikace, a část obsahující tz. resources neboli zdroje pro první část.

4.4.1 Logika

Tato sekce obsahuje běžné java třídy s vnitřní logikou aplikace. Pro přehlednost jsou třídy roztříděny do balíčků podle tabulky 4.4.

Bez detailnějšího popisu se dá provázání popsat následujícím způsobem. Aktivita jakožto základní prvek obsahuje jeden nebo více fragmentů. Fragment již obsahuje prvky uživatelského rozhraní a pomocí něj probíhá veškerá

Tabulka 4.4: Balíčky v aplikaci

Modul	Název
activity	Obsahuje pouze aktivity, které tvoří aplikaci
adapter	Obsahuje adaptéry pro zobrazování seznamů
db	Obsahuje třídy definující DB tabulky a třídy pracující s databází již stažených dat
fragment	Zde jsou veškeré fragmenty
io	Třídy starající se o převod odpovědí ze serveru do databáze
model	Zde je datový model aplikace
service	Třídy pro běh na pozadí
util	Pomocné třídy s obecnými metodami
ws	WebService třídy pro veškerou práci s API

interakce. Pokud má fragment zobrazovat seznam, potřebuje adapter, který se stará o skrolování a naplňování seznamu. Aby bylo co zobrazit uživateli, je potřeba naplnit databázi. O data do DB se postarají třídy z *WebService*. Data jsou po získání z API pomocí tříd v *io* převedeny na objekty z balíčku model a uloženy do databáze.

Balíček service je tvořen třídami, které dědí od android třídy *Service*. Jsou to speciální třídy pro běh na pozadí. Můžou běžet například i když uživatel uspal zařízení nebo je aktuálně v jiné aplikaci. Stejně tak se mohou spouštět na pozadí v naplánovaném čase, nebo pokud se v rámci zařízení stala nějaké akce jako například změna nastavení.

V balíčku util se pak nachází třídy zejména plné statických metod. Obsahují metody, které se v aplikaci používají na více místech a týkají se tříd napříč různými balíčky. Jde například o práci se *SharedPreferences*, zjištění jestli je dané zařízení tablet, sdílení a další.

4.4.2 Zdroje

Tabulka 4.5 obsahuje pouze základní strukturu zdrojů a vše lze jednoduše rozšířit o různé konfigurace. Například vytvořením složky layout-land lze dosáhnout různého vzhledu při otočení zařízení na šířku. Do složky je potřeba umístit layout se stejným názvem jako má při normálním zobrazení, ale i jiným vzhledem. Díky tomu je automaticky načten layout podle aktuální orientace. Pokud layout není v této složce, je zobrazen ten pro zobrazení na výšku. Lze takto rozlišit různé layouty pro tablety, jednoduše přeložit aplikaci do různých jazyků, používat různě kvalitní obrázky pro různá rozlišení atd.

Jako grafiku lze použít buďto klasické png, jpg nebo gif obrázky. Nejlepší z daných tří je png. Přesto se ale jedná se o nejhorší variantu použití grafiky. Soubory jsou zbytečně velké a musí se přidávat pro různá rozlišení. Další možností je použití tzv. nine-patch obrázků. V podstatě se jedná o png, které

Tabulka 4.5: Základní rozdělení zdrojů

Modul	Název
drawable	Složka s grafikou využitou v aplikaci
layout	Obsahuje všechny layouty uživatelského rozhraní
menu	Zde jsou definice menu obsažených v aplikaci
mipmap	Složka pro uložení ikony aplikace
values	Konstanty využívané v aplikaci
xml	Definice vyhledávání

se ale přizpůsobí rozlišení zařízení. Poslední využívanou možností je grafiku definovat pomocí xml souborů. Tyto soubory jsou velmi malé vůči klasickým grafickým formátům, zároveň je lze ale využít jenom pro jednoduchou grafiku jako například ikony. V této aplikaci je co nejvíce použit poslední styl. Například všechny ikonky v menu jsou vytvořené pomocí xml.

Složka mipmap obsahuje pouze ikonu aplikace. Ikona je ve speciální složce z toho důvodu, že je vhodné jí mít v jiných rozlišeních než ostatní grafiku.

Layouty jsou definovány také pomocí xml, stejně jako vše ostatní mimo grafiky. Přímo v layoutech se definuje většina vlastností prvků, jako barvy, velikost nebo texty. V kódu pak lze s prvky pracovat a měnit jejich vlastnosti.

Menu jsou zde využita pouze v mobilní verzi a je zde definice hlavního vysouvacího menu ze strany, menu na detailu videa pro sdílení a uložení do oblíbených a také menu pro vyhledávání.

Vzhledem k tomu, že cílovou skupinou pro aplikaci jsou anglicky hovořící uživatelé, byla jako jazyk aplikace vybrána angličtina. Všechny texty, které nepřichází z API, jsou ve složce *values* v souboru *strings.xml*. Tato složka také obsahuje definice barev, stylů uživatelských prvků, atd..

Poslední složka s názvem xml je zde pouze kvůli vyhledávání. Definují se zde texty a akce, které se spustí po vyhledávání. Stejně tak je zde zapnuto hlasové vyhledávání.

4.5 Komunikace se serverem

Vzhledem k obsahu a účelu aplikace se jedná o jednu z nejdůležitějších částí implementace. O zobrazení dat se stará fragment. Ten tedy také kontroluje, jestli jdou data uložena v databázi. Předpokládejme že nejsou, je tedy potřeba databázi naplnit.

Každý fragment dědí od třídy *BaseFragment*. Tato třída mimo jiné obsahuje metodu *startTask* pro spuštění *AsyncTask*, která data stáhne mimo hlavní vlákno. Metoda *startTask* obsahuje dva parametry, jako první tzv. bundle. Do bundle lze uložit veškeré primitivní datový typy, ale i složitější objekty pokud implementují tzv. parcelable(jsou rozložitelné na primitivní datový typy a obsahují metody na toho rozkládání a skládání). Do bundle je vždy po-

třeba vložit identifikátor API metody, kterou právě voláme. K tomu slouží třída *MsgConstants* obsahující konstanty pro všechny druhy zpráv a jejich URL adresy. Pokud se mají za adresu přidat parametry nebo by byla zpráva typu POST, je potřeba tyto informace také vložit do tohoto bundle. Druhý parametr je tzv. *WSTaskFragment*, ten zajistí, abychom se s výsledkem z API vrátili do fragmentu a také startuje samotnou operaci na pozadí, je zde tedy uveden fragment který potřebuje data.

WSTaskFragment je ale potřeba vytvořit, proto je nutné před *startTask* zavolat metodu *createWSTaskFragment* starající se o jeho vytvoření. Tato metoda je v *BaseFragmentu*, díky tomu jí mají všechny fragmenty.

BaseFragment je ale pro každý modul jiný, název base má pouze proto, že od něj dědí ostatní fragmenty. Aby ale moduly mohli používat co nejvíce společného kódu pro komunikaci, přidá tento fragment do *WSTaskFragmentu* ještě fasádu[21]. Ta zajistí vytvoření zprávy, její odeslání a zpracování odpovědi. Vytvoření a zpracování je tvořeno v každém modulu jinak, protože některé API metody nejsou potřeba v obou modulech. Fasáda je nutná z toho důvodu, že base modul nevidí do modulů nad sebou. Zároveň díky ní dochází v úspoře kódu a znovupoužitelnosti celé této komunikace.

WSTaskFragment tedy spustí *AsyncTask*, která provede komunikaci na pozadí. V metodě *doInBackground* je potřeba zmíněná fasáda. Ta totiž obsahuje metodu *sendMessage*, díky ní je odeslána zpráva na server a zachycena odpověď.

Fasáda vytvoří zprávu pomocí třídy *WSMsgCreator* a její metody *createMsg*. Tato metoda přidává parametry do POST metody, zatím v této práci žádná POST API metoda není. Pokud se ale podaří zprovoznit registraci uživatelů, tak tomu tak bude, a metoda je zde kvůli tomu připravená. Dále je potřeba vytvořit URL. K tomu slouží třída *WSUrlCreator* a její metoda *CreateUrl*. Ta podle zadaného typu zprávy vrátí URL, která k této zprávě patří.

Listing 4.3: Tělo metody sendMsg ve fasádě

```
BaseNetworkCaller caller = new BaseNetworkCaller();
String msg = new WSMsgCreator()
        .createMsg(msgType, params);
mAnswer = caller.sendRequest(msgType, new WSUrlCreator()
        .createUrl(msgType, params), msg, params);
if (!task.isCancelled()) {
        new WSMobileParser().parse(msgType, mAnswer,
                params, BaseApplication.getContext());
}
```

Následně je vytvořen objekt třídy *BaseNetworkCaller*. Jak říká sám název, metoda se stará o samotné zavolání a zpracování odpovědi. Tato třída je společná pro oba moduly. Při volání metody *sendRequest* je potřeba jako parametry přidat druh zprávy a bundle poslaný již na začátku, navíc ale výše

vytvořenou URL a POST zprávu. V metodě *sendRequest* je využita knihovna OkHttp3. Ta umožňuje jednoduše ze zadané URL vytvořit reguest a odeslat ho. Pokud se jedná o API metodu pro přihlášení, je pomocí stejné knihovny přidána do hlavičky autentizace popsaná v návrhu ze jména a hesla. Z odpovědi ze serveru se vezme tělo, vytvoří se z něj string a ten je vrácen do fasády.

Listing 4.4: Vytvoření a přijetí odpovědi ze serveru včetně autentizace

```
Request request = builder.build();
String credential = Credentials.basic(
        bundle.getString(BaseMsgConstants.EMAIL),
        bundle.getString(BaseMsgConstants.PASSWORD));
request.newBuilder()
        .header("Authorization", credential).build();
Response response = client.newCall(request).execute();
String json = response.body().string();
```

Nyní máme textový řetězec obsahující tělo odpovědi ve formátu JSON. Poslední akce fasády je vytvoření objektu *WSParser* a zavolání metody *parse*. *WSParser* je opět pro každý modul zvlášť, protože tato metoda zpracovává pouze odpovědi, které do modulu patří. Jako většina ostatních objektů ale kvůli úspoře kódu dědí od své base třídy z modulu base. Metoda *parse* se stará o naplnění databáze, o tom až dále.

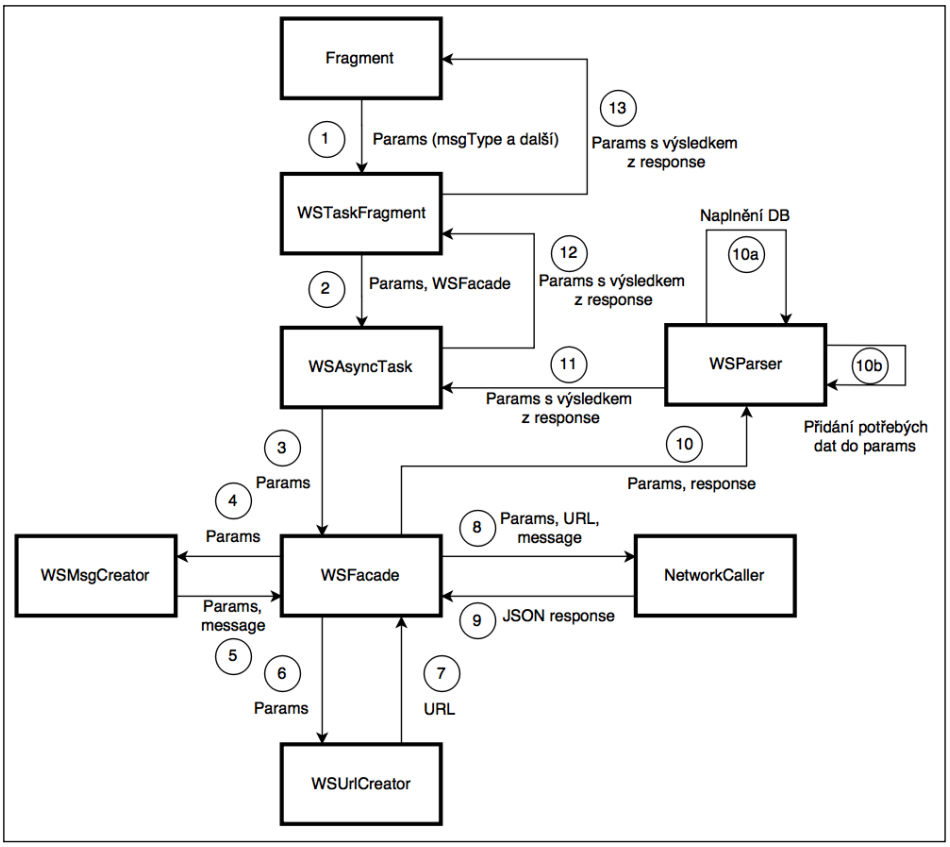
Po naplnění databáze, které stále proběhlo díky *AsyncTask* na vedlejším vlákně, toho vlákno skončí a automaticky se zavolá metoda *onPostExecuxe* stejné třídy. V té se zavolá metoda *onPostExecute* původního *WSTaskFragmentu*. Tuto metodu je potřeba přepsat v původním fragmentu. Jako parametry obsahuje ID API metody, chybový kód a bundle s případnými hodnotami. Tím v původním fragmentu zjistíme, které zavolané API metody tímto fragmentem již doběhli a můžeme na to reagovat.

Chybový kód je využit například pokud není připojení k internetu. *BaseFragment* pokud se vrátí tento kód, zobrazí dialog informující uživatele, že má zajistit připojení k internetu a akci opakovat. Diagram celého postupu volání tříd je zobrazen na obrázku 4.1.

Díky předchozímu postupu, pokud chceme přidat další API metodu, stačí přidat konstanty pro její identifikátor a URL. Případně pro POST zprávy vytvořit obsah. O všechno ostatní se postarají třídy z předchozího postupu, které již není potřeba při přidávání nové API metody o nic rozšiřovat.

4.6 Naplnění databáze

Nejprve je potřeba zadefinovat DB tabulky. Každá tabulka je definovaná jako samostatná třída, která dědí od *BaseTable*. Tato třída je abstraktní, definuje metody pro vytvoření tabulky, zvýšení nebo snížení verze databáze a smazání



Obrázek 4.1: Diagram průběhu načtení informací ze serveru

a znovuvytvoření tabulky. Dále má dvě abstraktní metody, ty musí přepsat každá tabulka. Jedná se o metodu pro získání jména tabulky a definici tabulky. Definice tabulky je textový řetězec obsahující minimálně název sloupců, datové typy, zda musí být sloupec vyplněn nebo ne, primární klíč, případně může obsahovat další omezení.

Názvy sloupců jsou jako konstanty definované ve třídě *Contract*, ta obsahuje interface pro každou tabulku obsahující tyto konstanty. Třída ale hlavně obsahuje statické třídy pro všechny tabulky, obsahující mimo zmíněného interface název tabulky, a URI pod kterou budeme s daty pracovat.

Pro samotné vytvoření databáze slouží třída *DatabaseHelper*. V té jsou v příslušných metodách volány nad každou tabulkou metody z třídy *BaseTable*. Třída se tedy stará o vytvoření, upgrade, downgrade nebo smazání DB. Je zde také definován název databáze a její verze. Pokud je pak před první prací s databází zjištěno, že DB neexistuje nebo se změnila její verze, je pomocí této třídy DB vytvořena nebo změněna.

Pro samotnou práci s databází slouží tzv. *ContentProvider*. Jedná se o android třídu, od které dědí například třída *MobileContentProvider*, starající

se o práci s databází v modulu pro mobilní zařízení. Třída obsahuje hlavně metody pro dotazy, vkládání, mazání a změny dat v databázi. Jedním z parametrů těchto metod jsou výše zmíněné URI. Například při vkládání určíme podle URI z jaké tabulky data vybrat a jak. Pokud víme, že pro získání dat je potřeba spojení více tabulek, je pod danou URI spojení více tabulek pomocí JOIN. Při dotazu do databáze jsou další parametry projekce, selekce, řazení. Dotaz do databáze např. ve fragmentu je pak pomocí těchto metod s příslušnými parametry.

Pokud se jedná o operaci manipulující s daty, *ContentProvider* notifikuje, že data pod danou URI byla změněna. Díky tomu se pak při použití dat např. fragment ihned dozví, že se data, se kterými pracuje změnila a může na to reagovat.

Máme tedy vytvořenou databázi včetně tabulek, stejně tak operace, které nad tabulkami chceme provádět a je potřeba DB naplnit. Podle kapitoly 4.5 dostaneme do třídy *WSParser* řetězec ve formátu JSON obsahující odpověď ze serveru. Data z této odpovědi je potřeba uložit do databáze.

Pro uložení do databáze je použit seznam *ContentProvider* operací. Do tohoto seznamu přidáme všechno, co chceme s databází provést, následně celý tento list jedním příkazem na databázi aplikujeme. Pro vytvoření těchto operací slouží pro každou API odpověď třída *Handler* obsahující metodu *parse*. Například zavolání třídy *VideoHandler* a její metody *parse* s JSON v parametru vrátí seznam DB operací, ten je pak na DB aplikován.

Třída handler nejprve za pomoci knihovny Gson[22] převede odpověď na objekt. Objekty musí obsahovat vše co chceme z API využít včetně odpovídajících datových typů. Například při parsování seznamu videí, knihovna převede JSON na seznam objektů typu *Video* z balíčku model. Tento objekt obsahuje textové řetězce a čísla, ale také pole kapitol ke každému videu. Je tedy vytvořen i objekt *Chapter* reprezentující kapitolu. Aby knihovna mohla převést JSON na objekty, musí se proměnné v objektech jmenovat naprosto stejně jako se jmenují v API.

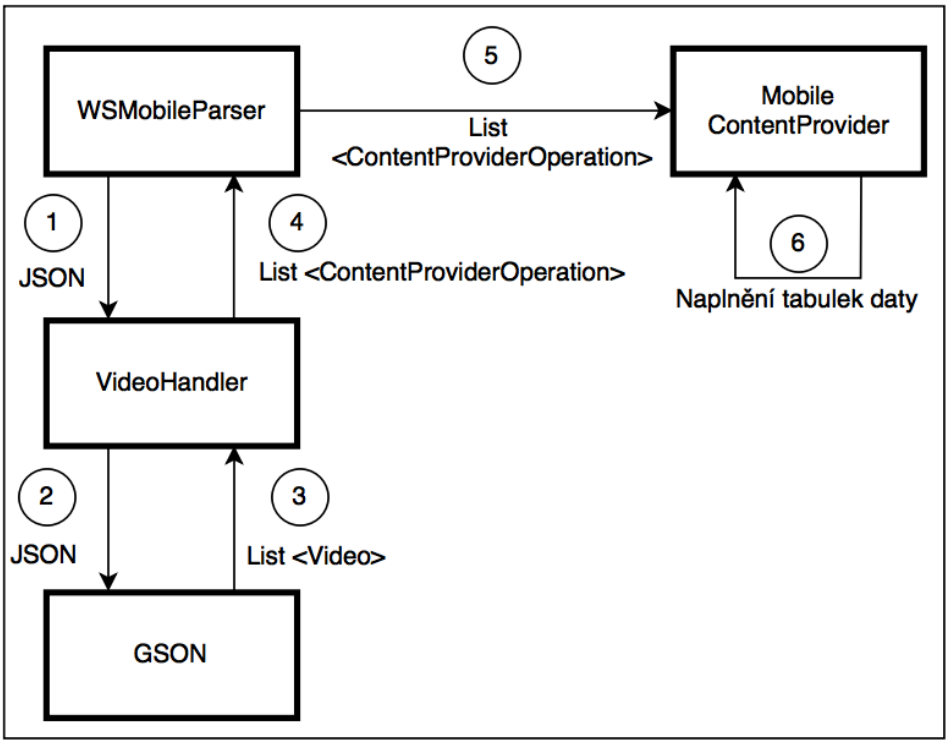
Listing 4.5: Převod JSON na seznam videí pomocí Gson

```
List<Video> videos = Arrays.asList(new Gson()
                    .fromJson(json, Video[].class));
```

Následně se celý seznam videí projde a pro každé video se vytvoří DB operace. Kapitoly u videí jsou v jiné tabulce. Proto se tvoří zvlášť operace do DB i pro každou kapitolu. Seznam operací je nakonec vrácen a operace jsou aplikovány na DB. Celý průběh procesu je znázorněn na obrázku 4.2.

Listing 4.6: URI pro konkrétní video

```
content://com.catvusa.catvusacom/video/2037
```



Obrázek 4.2: Diagram naplnění databáze

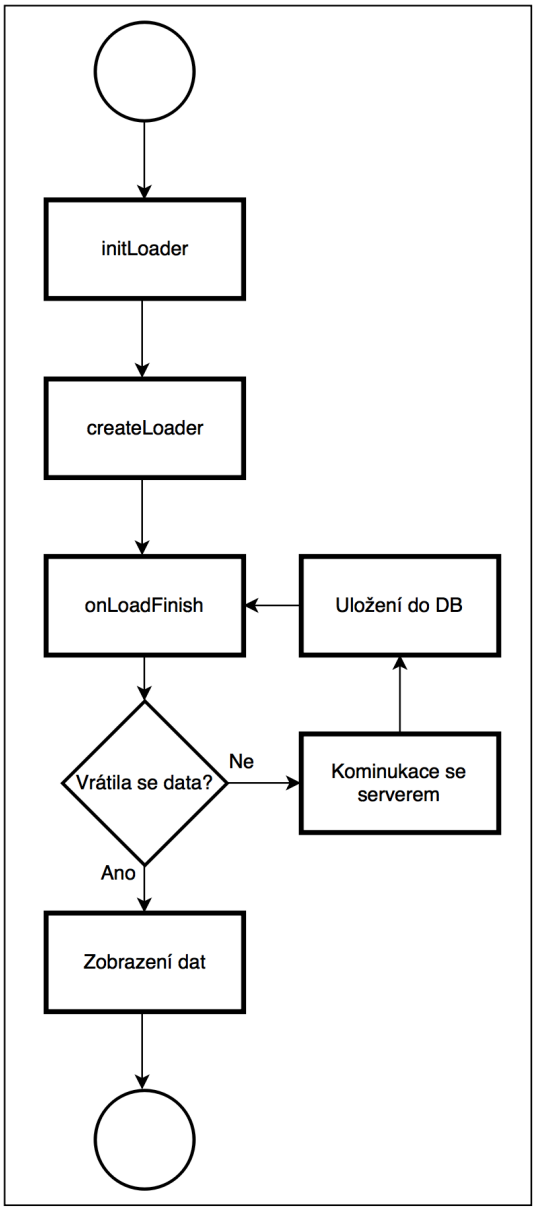
4.7 Fragment využívající API

Většina fragmentů v aplikaci je závislá na datech ze serveru. Tyto fragmenty implementují interface *LoaderManager.LoaderCallbacks<Cursor>*. Ve fragmentu se nejprve inicializuje tzv. loader s identifikátorem. Loader se postará o získání dat na vedlejším vlákně. Automaticky je zavolána metoda pro vytvoření loaderu, která je součástí zmíněného interface. V té se vytvoří loader. Zadá se mu URI, odkud data chceme, projekce, případně selekce nebo řazení. V metodě *onLoadFinish* jsou vrácena data. Data se vrátí jako cursor, což je v podstatě seznam řádků, které prošly selekcí a obsahující sloupce z projekce.

Pokud se vrátí seznam prázdný ale víme, že by neměl, tak je DB prázdná. Pokud se toto stane, je nastartována AsyncTask a provedena inicializace z kapitol 4.5 a 4.6. Na diagramu 4.3 je vidět průběh načtení dat do fragmentu, využívající předchozích kapitol.

4.8 Cache

Jako první v každé aktivitě pro sekci, které obsahuje cache, je po jejím spuštění zjištěno, jestli se má smazat DB. K tomu slouží metody v třídě *SharePrefs*



Obrázek 4.3: Vývojový diagram načtení dat do fragmentu

dědící od *SharedPreferences*. Ze *SharePrefs* je zjištěn čas posledního update. Jestliže uplynul čas, pro opětovné načtení této sekce, je uložen do preferences aktuální čas. Je o tom také informována aktiva, ta smaže příslušné tabulky. Fragment si pak opět data stáhne.

4.9 Implementace jednotlivých částí modulu mobile

4.9.1 Menu

Menu je tvořeno v *BaseMobileActivity* od níž dědí všechny aktivity tohoto modulu. Aktivity pak mohou určit, zda se v nich má menu inicializovat. Bez menu jsou aktivity, které nejsou přístupné ihned z menu, ty pak místo něj obsahují šipku pro návrat zpět. Aktivity musí stejně jako fragmenty mít vlastní layout. Do tohoto layoutu je tedy přidán prvek *NavigationView* pro vytvoření menu. Pro zobrazení položek využívá definici menu ze souboru *mainmenu.xml*, zde je pro každý prvek definován identifikátor, ikona a text. Podle identifikátoru je následně po kliknutí na položku spuštěna aktivita odpovídající dané sekci.

Sekce pro sdílení na sociálních sítích neobsahují žádné aktivity, ale otevírají odkazy na stránky organizace na sociálních sítích. Aplikace vyšle tzv. intent s URL. Pokud je v systému nainstalované aplikace, která umí tento intent zpracovat, je uživateli nabídnuto otevření v této aplikaci nebo v prohlížeči. Například u Google+ pokud má uživatel aplikaci Google+, je přesměrován do ní. Pokud ne, stránka organizace se otevře v prohlížeči.

Pokud je uživatel přihlášen, je při každém vytvoření menu zjištěno jeho jméno a email a tyto informace jsou zobrazeny v jeho horní části. Stejně tak se podle přihlášení mění text v této položce menu.

4.9.2 Video

Aktivita pro zobrazení seznamu videí se nazývá *VideoListActivity*. Tato sekce obsahuje tzv. *PagerAdapter*, ten rozděluje aktivitu do sekcí, mezi kterými lze přepínat pomocí horního menu nebo přejetím do strany. Protože *PagerAdapter* je ve více aktivitách, je metoda pro jeho inicializaci v *BaseMobileActivity*. Tuto metodu je potřeba volat s instancí *PagerAdapteru* příslušného pro danou sekci. *PagerAdapter* vždy obsahuje názvy sekcí pro horní menu a také jaký fragment se má v jaké sekci zobrazit.

V této sekci všechny sekce obsahují *VideoListFragment*. Do fragmentu je pouze jako argument vložena informace o tom, jakou sekci reprezentuje, aby získával z DB správná data. *VideoListFragment* pro zobrazení seznamu využívá tzv. *GridLayout*. Podle druhu a orientace zařízení má pak seznam různý počet sloupců pro lepší zobrazení obsahu.

Pro zobrazení položek v seznamu je vytvořen *BaseCursorAdapter*, od něj dědí všechny adaptery v jednotlivých sekcích. Adaptery se starají o zobrazení cursoru z DB jako seznamu. Fungují tak, že stále dokola recyklují view pro zobrazení položek. Uživatel tedy vidí na obrazovce např. 5 prvků, načtený je navíc pouze jeden pod a jeden nad. Při posouvání seznamu se pak horní přesouvá dolu stále dokola. Díky tomu je seznam úsporný na paměť. Stejně

tak jsou obrázky z internetu stahovány až když jsou potřeba, nikoliv všechny ihned.

Prvky adaptéru jsou v této sekci reprezentovány pomocí *CardView*. Jedná se o prvek, který přišel s Material Design. Jednotlivé prvky jsou narozdíl od staršího *ListView* vetší, jsou odděleny od pozadí, obsahují více informací, atd. Při implementaci *CardView* byl využit tutoriál dostupný z [17].

Pro načtení obrázků do seznamu se používá knihovna Glide. Té stačí dát URL na obrázek a prvek, do kterého má načtený obrázek zobrazit. Knihovna pak na pozadí začne obrázek stahovat.

O detail videa se stará aktivita *VideoDetailActivity* a v ní inicializovaný fragment *VideoDetailFragment*. Fragment inicializuje loader pro načtení videa včetně kapitol. Kapitoly jsou pak zobrazeny jako *ListView*, s načítáním obrázků pomocí Glide. Obtékání textu okolo obrázků však není standardní vlastnost androidu, proto je tato vlastnost vytvořena podle návodu z [18]. Další loader je pro oblíbené, je zjištěno zda je video v oblíbených, pokud ano, ikona pro oblíbené je obarvena. Poslední loader je kvůli kvízům, po jeho výsledku je zjištěno, jestli popis videa obsahuje název některého českého kraje. Pokud ano, je zobrazen *SnackBar* s informací o tom, že kvíz bude spuštěn po skončení videa. *SnackBar* je lišta která vyjede ze spodní části obrazovky a po daném čase opět zajede.

Listing 4.7: Nalezení názvu kraje v popisu videa

```
private void findQuiz(Cursor data) {
    data.moveToFirst();
    for (int i = 0; i < data.getCount(); i++) {
        String name = data.getString(data
                .getColumnIndex(Contract.Quiz.NAME));
        name = name.replaceAll("Region", "")
                .replaceAll("of", "")
                .replaceAll("␣␣", "");
        if (name.startsWith("␣")) {
            name = name.substring(1);
        }
        if (mExcerpt.contains(name)) {
            mQuiz = data.getInt(data.
                getColumnIndex(Contract.Quiz.INT_ID));
            Snackbar snackbar = Snackbar.make(
                mCoordinatorLayout,
                R.string.text_video_quiz,
                QUIZ_SNACK_DURATION);
            View snackbarView = snackbar.getView();
            TextView textView = (TextView) snackbarView
                .findViewById(android.support.design.R
                                .id.snackbar_text);
```

```
                textView.setMaxLines(5);
                snackbar.show();
                return;
            }
            data.moveToNext();
            }
}
```

Knihovně pro přehrávání videa stačí předat URL na video, aby vše fungovalo. Dále je do ní nastaven obrázek, aby nebyl přehrávač černý dokud na něj uživatel neklikne. Pro automatickou detekci videa jsou k přehrávači připojeny *UserActions*. Jedná se o třídu odchytávající všechny akce spojené s přehrávačem. Jednou z akcí je i konec videa. Tato akce slouží ke spuštění kvízu.

Kvůli automatickému přehrávání na celou obrazovku při otočení zařízení je zde implementován *SensorManager* ze systému android. Do něj je zaregistrován listener z knihovny, starající se o zvětšení na celou obrazovku.

Pro sdílení se podobně jako u sociálních sítích používá intent. Do něj se nastaví popis a URL videa. Tento intent odchytí systém android a zobrazí všechny možnosti, jak lze obsah sdílet. Po kliknutí na oblíbené je do DB tabulky pro oblíbené přidán/odebrán záznam o videu.

4.9.3 Radio

O radio se stará *RadioActivity*, ta obsahuje *RadioPagerAdapter*, pro různé kategorie rádia. Spodní ovládací panel rádia je navíc zobrazen přímo v aktivitě. Díky tomu je ovládací panel napevno a *PagerAdapter* přepíná pouze mezi stanicemi.

Aby bylo možné poslouchat na pozadí a rádio se nezastavilo po přepnutí na jinou aktivitu, stará se o přehrávání *MediaPlayerService*. Tento service, ale i aktivita a *RadioFragment* se zaregistrují k odchytávání broadcastu. Všichni si zaregistrují akce, na které budou reagovat. Fragment reaguje na změnu obrázku a textu. Aktivita reaguje na čas přehrávání. Service reaguje na pauzu/-přehrávání a změnu hlasitosti. O řízení rádia se tedy stará servie, ten informuje aktivitu a fragment, že mají změnit UI. Aktivita informuje service, že uživatel klikl např. na tlačítko pro pauzu. Service na to reaguje zastavením přehrávání. Service také zobrazuje notifikace s ovládacími prvky, které se také mění podle komunikace mezi komponentami.

O samotné přehrávání rádia se stará třída Radio. Ta obsahuje seznam písniček vytvořený z DB, která náhodně automaticky po skončení písničky vybere další. Navíc u rádia folk musí před každou písničkou přehrát její intro. U rádia classic je přehráno jedno intro na začátku. Také se zde ovlivňuje hlasitost přehrávání, která je nastavována ze service. Při vybrání další písničky je u ní v DB nastaven příznak, že již byla přehrána, aby se písničky neopakovaly. Pokud písničky dojdou, tak třída Radio aktualizuje celou tabulku tím, že

všem písničkám nastaví příznak, že nebyly přehrány. Tato třída také obsahuje interface pro změnu IU. Pomocí něj informuje service např. o změně textu, ten pak informuje fragment a notifikaci, které text změní. Stejná třída dostane při inicializaci od service informaci o tom, jak dlouho se již přehrává bez přihlášení. Pokud se rádio přehrává, pak se tato hodnota zvyšuje společně se zvyšováním času písničky. Vždy při pauze uloží service aktuální čas přehrávání bez přihlášení do *SharedPreferences*. Pokud je celkový čas větší než 10 minut, rádio je zastaveno, je o tom informována *RadiaActivity* a ta zobrazí dialog, že je potřeba se přihlásit. Po kliknutí na přihlásit je uživatel přesměrován na *LoginFragment*.

O notifikaci s ovládacími prvky se starají android komponenty *MediaSessionCompat* a *MediaControllerCompat*. Controlleru se nastaví, jaké akce má obecně obsahovat. Díky tomu je pak zavolána service, která reaguje na konkrétní tlačítko danou akcí a informuje aktivitu o změně. V notifikaci se podle aktuálního stavu rádia zobrazí různá tlačítka, obrázek a text. Po změně stavu se vždy kompletně tvoří nová notifikace.

4.9.4 POI

Sehnaná data byla potřeba převést z excel do JSON. K tomu byl využit online nástroj dostupný na [19]. Při přidávání míst do mapy jsou potřeba souřadnice místa. V datech byly ale pouze adresy. Existují metody na získání souřadnic z adresy, zjišťují to ale pomocí internetu a provolání všech více jak 600 míst trvalo několik minut. Proto byl napsán skript, který data převedl a přidal do vstupního JSON. Výsledný JSON byl pak nahrán na server společnosti. U některých míst souřadnice přesto nebyly rozpoznány. Souřadnice jsou u nich nastaveny na (-1,-1). Jsou zobrazeny v seznamu ale ne na mapě.

U map je potřeba zjištění aktuální polohy, jediné nebezpečné oprávnění, které aplikace má. Proto na verzi androidu vetší než 6 je zobrazen dialog vyžadující toho oprávnění.

Stejně jako u videa jsou fragmenty v *PagerAdapteru*, ten ale pro mapu vrací *MapFragmnet* a pro seznamy *PoiListFragment*. *MapFragment* využívá pro zobrazení míst Google mapy přesněji *SupportMapFragment*. Pro zobrazení podrobností o místě je použito tzv. *NestedScrollView*. To umožňuje zmenšování a zvětšování podrobností. Lze také reagovat na různé stavy vysunutí, toho je využito při zobrazení navigačního tlačítka.

Pro zobrazení fotografie se využívá Street View pro dané místo. Google má veřejnou API, kde do URL stačí přidat požadované rozlišení fotografie a souřadnice místa. Pokud k místu neexistuje Street View, vrátí se obrázek, který o tom informuje.

Tlačítko pro aktuální polohu zajišťují Google mapy. Pro rychlejší a přehlednější zobrazení jsou místa shlukována do clusterů. Každému bodu v Google mapách lze přiřadit souřadnice, název a popis. Toho je pak využito při kliknutí

na místo v podrobnostech. O clustery se kompletně starají Google mapy stačí přidat všechna místa do *ClusterManageru*.

Sdílení a ukládání do oblíbených je implementováno stejně jako u videí. Tlačítko pro navigaci v seznamu i na mapě opět vyšle intent, ten zpracuje aplikace Google mapy a rovnou zobrazí trasu z aktuální polohy na dané místo.

Vzhledem k tomu, že fragmenty spolu nemůžou navzájem přímo komunikovat, probíhá zobrazení místa na mapě přes *MapActivitu* starající se o všechno ohledně map, které obsahuje metodu pro zobrazení místa. Tu zavolá *PoiListFragment* a metoda s pomocí *PagerAdapteru* spustí *MapFragment* a přesune v něm kameru na dané místo. Při otevření *MapFragmentu* je defaultně vycentrováno na geografický střed České republiky.

4.9.5 Oblíbené

Tato sekce pomocí *FavoriteActivity* a *FavoritePagerAdapter* pouze zobrazuje fragmenty již vytvořené v předchozích sekcích.

4.9.6 Kvíz

K zobrazení seznamu kvízů se používá *QuizListActivity* a v ní spuštěný *QuizListFragment*. Ten pro zobrazení prvků používá *ListView* s názvy kvízů. Vzhledem k tomu, že z API se vrací jako název kvízu dlouhý název vytažený z webu, je již před uložením do DB v handleru název oříznut a uložen pouze název kraje. Po kliknutí na kraj se spustí *QuizActivity* a *QuizFragment*. Tento fragment má tři stavy, úvod, otázka, konec. Podle stavu zobrazuje a skrývá některé prvky z layoutu. Při rotaci obrazovky se uloží stav, skóre a ID aktuální otázky do bundle a při opětovném vytvoření fragmentu po rotaci jsou data načtena, aby byl zachován stav kvízu. View je přebarvováno a měněno také vždy při kliknutí na odpověď a přechodu na další otázku, aby pro každou otázku nemusel být startován další fragment. Při režimu na šířku se všechny informace nevešly na display a uživatel musel skrolovat k tlačítku, proto tato sekce obsahuje druhý layout pro zobrazení na šířku. Ten zobrazuje zadání a otázky vedle sebe a je tak potřeba méně místa pro zobrazení obsahu.

4.9.7 Uživatelský účet

Informace o tom zda je uživatel přihlášen je uložena v *SharedPreferences*. *AccountActivity* zkontroluje přihlášení. Pokud je uživatel přihlášen zobrazí *AccountFragment*. Ten podle *SharedPreferences* zobrazí jméno a email uživatele. Tlačítko pro odhlášení smaže *SharedPreferences* a otevře úvodní aktivitu aplikace.

Pokud uživatel není přihlášen, je zobrazen *LoginFragment*. Po kliknutí na přihlášení je ihned zkontrolováno zda jsou obě hodnoty vyplněné, pokud ne, je uživatel informován pomocí *SnackBaru*. Mimo ID zprávy se musí do

AsyncTask poslat ještě jméno a heslo. Odpověď ze serveru není ukládána do DB. Místo do seznamu DB operací je tedy vrácené jméno uloženo do bundle, který se dostane až do *LoginFragmentu*. Tam se jméno i heslo uloží do *SharedPreferences* a opět se spustí úvodní aktivita. Pokud uživatel vyplnil některý údaj špatně, je opět informován o chybě. Fragment to pozná díky chybějícímu jménu v bundle, který se do něj vrátí.

Bohužel bylo při realizaci zjištěno, že propojit registraci v aplikaci s registrací na webu není díky WorsPressu možné. Proto k alespoň částečnému vytvoření této funkcionality tlačítko pro registraci v *LoginFragmentu* otevře webový prohlížeč s webovou stránkou obsahující registrační formulář, po registraci na webu se uživatel může do aplikace vrátit a přihlásit se v ní. Stejně tak tlačítko pro zapomenuté heslo otevírá příslušnou URL v prohlížeči.

4.9.8 Notifikace

Je potřeba, aby kontrola běžela na pozadí, i když samotná aplikace neběží. K tomu slouží třída *NotificationService*, třída má nastaveno, aby byla systémem spuštěna i po restartu zařízení. Po startu service je z API stažena informace o naposledy přidaném videu. Video je porovnáno s videi v DB. Pokud se toho video ještě nenachází v DB, je uživatel informován formou notifikace, že je v aplikaci dostupné nové video.

V service se také nastaví opakované spouštění této třídy. K tomu slouží třída *NotificationAlarm*. Čas opětovného spouštění je nastaven na jeden den. Tato třída dědí od android *BroadcatsReceiver*. Po dni systém spustí tuto třídu, pokud je v *SharedPreferences* uloženo, že uživatel chce notifikace, je spuštěna service pro případné notifikování. Jinak se service nespustí a tím pádem se ani nenastaví nový alarm.

4.9.9 Nastavení

SettingActivity a *SettingsFrafment* slouží pouze k uložení informace o tom, jestli uživatel chce být notifikován do *SharedPreferences*. Hodnota je pak používá při spouštění *NotificationService* kontrolující nová videa.

4.9.10 Vyhledávání

Tlačítko se do vyhledávání přidává v *BaseMobileActivity* v metodě *onCreateOptionsMenu*. Vyhledávací tlačítko je díky tomu v každé aktivitě. Pokud jako např. na detailu videa má být jiné menu, musí fragment pomocí podobné metody ale pro fragmenty vyhledávací menu přepsat.

O celý řádek s vyhledáváním včetně všech tlačítek i našeptávání se stará android komponenta *SearchView*. Prvku je potřeba nastavit aktivitu, které se spustí pokud uživatel klikne na tlačítko vyhledat. K tomu slouží *SearchResultActivity*.

Třída *MobileSuggestinProvider* slouží k zobrazení našeptávaných položek. Ten projde DB tabulky pro video a POI. Vrátí *SearchView* cursor obsahující ID, název, kategorii(video nebo POI) a ID v konkrétního prvku s přidaným prefixem, jestli jde o video nebo POI. Vyhledávání probíhá z textu, který již uživatel zadal. Ve videu se prohledává název, v POI adresa. S tím, že stačí když název obsahuje daný řetězec, nemusí jím začínat. Limit pro výsledky je 10 z každé tabulky.

Po kliknutí na prvek nebo stisknutí vyhledávacího tlačítka je spuštěna *SearchResultActivity*. Ta pozná jestli šlo o kliknutí na prvek nebo na tlačítko. Pokud na prvek, podle prefixu pozná, jestli šlo o video nebo POI a spouští konkrétní detail. Pokud na tlačítko, je zobrazen *SearchResultFragment*, kterému je předán vyhledávaný výraz.

Fragment si pak udělá vlastní dotaz do DB již bez limitu na počet výsledků, ale prohledává data stejným principem. Pomocí komponenty *Spinner* může uživatel přepínat mezi výsledky vyhledávání ve videích a v POI. Výsledky jsou zobrazovány jako seznamy totožné se seznamy přímo v sekci video a POI.

4.10 Implementace jednotlivých částí modulu tv

4.10.1 Menu

Menu je podobně jako u mobilní verze v levé části aplikace. Je však na začátku vysunuté a mezi ním a obsahem se přepíná šipkou doleva a doprava na dálkovém ovladači. Menu je zde vidět pouze na hlavní stránce aplikace, proto je součástí *VideosFragmentu*, ten je ve *VideosActivity*, což je hlavní aktivita televizní aplikace. Tento fragment z DB inicializuje postupně všechny kategorie videí a pro každou přidá položku do menu. Po této inicializaci je přidána řádka pro každé rádio a jedna pro přihlášení.

Jednotlivá videa a rádia jsou reprezentována pomocí presenneru, respektive od něj dědícími třídami *VideoCardPresenter* a *RadioCardPresenter*. Pole těchto objektů naplníme např videi. Po vybrání konkrétního prvku je zjištěno o který objekt jaké instance jde a podle toho se pokračuje do detailu videí, rádia nebo na přihlášení.

4.10.2 Video

Po vybrání konkrétního videa z menu je zobrazena *DetailActivity* a *VideoDetailFragment*. Po načtení dat z DB jsou zobrazeny v hlavní části informace o videu pomocí *DetailVideoDescriptionPresenter*. Ve spodní části jsou podobná videa zobrazena pomocí stejného presenteru jako v hlavní části. Podobná videa jsou vybrána jako náhodná ze stejné kategorie.

Po kliknutí na přehrání videa se spustí přehrávač, ten je v *PlayerActivity* a *PlayerFragment*. Aktivita se stará o samotný přehrávač, ten je zde reprezentován prvkem *VideoView*. Aktivita reaguje na tlačítka a tím se stará i o

zobrazení ovládání videa. To je obsaženo v *PlayerFragment*. Tento fragment pak do aktivity posílá zprávy o tom, co se má dít s přehrávačem po vybrání tlačítka z ovládání. U tlačítek pro následující a předchozí video je kontrolováno, zda uživatel není na posledním/prvním videu ze seznamu videí dané kategorie. Při rychlém přehrávání fragment každých 100ms vyšle do aktivity informaci, že má video o vteřinu posunout daným směrem. Ovladač videa je vytvořen pomocí android komponenty *PlaybackControlsRow*, té je potřeba říci jaké prvky zobrazit a napsat akce spuštěné po kliknutí na prvek. Podobná videa jsou zde reprezentovaná stejně jako na detailu.

4.10.3 Radio

Pomocí *RadioActivity* a *RadioFragmentu* je zobrazen detail rádia ve stejném stylu, jako detail videa, pouze bez podobných položek a s jinými informacemi. K zobrazení se používá *DetailRadioPresenter*. Rádio zde neběží na pozadí, proto se o přehrávání stará přímo *RadioFragment*. Ten v sobě má instanci třídy Radio, pomocí ní ovládá rádio, *RadioFragmentu* vrací informace o změnách a fragment podle toho mění zmíněný presenter. Logika je ale stejná jako pokud byla tato instance rádia v service v mobilní verzi. I zde je stejným principem jako v mobilní verzi kontrolována délka přehrávání a v případě překročení 10-ti minut za 24 hodin je uživatel vyzván k přihlášení pomocí dialogu.

4.10.4 Uživatelský účet

Pro jednoduchost je zde pouze možnost přihlášení a odhlášení. Na hlavní obrazovce je podle aktuálního stavu různý text. Po vybrání možnosti přihlásit, je uživateli zobrazen dialog obsahující prvky pro vyplnění emailu a hesla a tlačítko pro spuštění přihlášení. Ověření probíhá stejně jako v mobilní verzi. Při špatném vyplnění je uživatel informován chybovou hláškou. Odhlášení opět odstraní uložené *SharedPreferences*.

Kapitola 5

Testování

Před publikováním aplikace je potřeba provést testování. Tím se odhalí chyby aplikace dříve, než se aplikace dostane veřejně mezi uživatele. Díky tomu mohou být problémy aplikace opraveny před jejím zveřejněním a předejde se tak negativnímu hodnocení způsobenému prvotními problémy, které provázejí většinu projektů.

5.1 Alfa testování

Toto testování se provádí jako první a má za úkol odhalit zejména pády aplikace. Rozdíly mimo výkonu jsou na android zařízeních zejména ve verzi systému. V jednotlivých verzích systému může aplikace pracovat jinak než na jiných, proto v této fázi bylo využito třech zařízení se systémy 4.x, 5.x a 6.x.

Uživatelům vlastnících tato zařízení byla aplikace nainstalována z APK souboru a byli vyzvání k proklikání celé aplikace. Jediným zaznamenaným pádem byl pád rádia na zařízení s androidem 4.x. Dalším závažným problémem bylo náhodné překlápění obrazovky při přehrávání videa a přeskakování mezi normálním přehráváním a přehráváním na celou obrazovku. Tento problém se vyskytoval na všech verzích systému. Mimo pád rádia se tato sekce na všech zařízeních velmi dlouho načítala. Žádný další zásadní problém nebyl objeven.

Pád rádia byl ihned po testu opraven. Problém dlouhého načítání souvisí s API, v této sekci je velké množství písniček a vzhledem ke způsobu jejich uložení na serveru a omezeným možnostem pluginu, nelze v této sekci aplikovat limit na vrácení menšího počtu výsledků. Z tohoto důvodu je potřeba informace o všech písničkách uložit najednou a problém byl vyřešen upozorněním uživatele, že se tato sekce může opravdu dlouho načítat.

Problém s přehrávačem souvisel s knihovnou využitou pro přehrávání videa, proto byl o tomto problému informován tvůrce knihovny, který slíbil nápravu chyby s další verzí knihovny. Tvůrce chybu opravil, ale v nové verzi knihovny se objevil další problém, a sice ten, že po skončení videa není automaticky spuštěn kvíz. Tvůrce byl opět informován a přislíbil opravu v další

verzi knihovny. V další verzi knihovny bylo vše v pořádku a v aplikaci se již s přehrávačem videí nevyskytoval žádný problém. Proto byla aktuálně nejnovější verze použita.

Aplikace pro TV byla v této fázi testována pouze na emulátoru a jediným problémem bylo dlouhé načítání rádia.

5.2 Beta testování

Beta testování se provádí až po alfa testování, využívá vetší skupinu uživatelů a zaměřuje se zejména na získání zpětné vazby o aplikaci, díky tomu lze aplikaci vylepšit ještě před vydáním. Před beta testováním byl pro jednoduché sdílení aplikace vytvořen záznam v obchodě Play. Byl vytvořen kompletní záznam pro zveřejnění aplikace, aby se ale nedostali uživatelé k aplikaci veřejně, byl záznam nastaven pouze na beta testování.

Uživatelé podílející se na beta testování obdrží odkaz na URL, na té potvrdí, že se chtějí stát testery aplikace. Poté již mohou z dalšího odkazu zobrazit záznam v obchodě Play nebo aplikaci v obchodě vyhledat.

Odkaz byl odeslán všem spolupracovníkům organizace společně s pokyny jak postupovat. K odeslání zpětné vazby mohli buďto odpovědět zpět na email nebo vyplnit připravený formulář, který mimo položek pro okomentování jednotlivých sekcí a aplikace celkově, obsahoval povinné položky pro název zařízení a verzi systému. To proto, kdyby aplikace někomu spadla, aby bylo možné lépe zjistit proč.

Zpětná vazba byla získána od 9-ti lidí, kteří dohromady sepsali cca 20 různých připomínek. Stejně jako v alfa testování nefungovalo zatím automatické spuštění kvízu. Pár uživatelům se objevily problémy s ovládáním rádia poté, co již hrálo, ale přešli do jiné sekce. Jinak se všechny ostatní připomínky týkaly zejména designu a podobných drobností. Žádný další zásadní problém nebyl v tomto testování zaznamenán.

Všechny připomínky byly konzultovány s vedoucím TV a následně v aplikaci opraveny. Jednou ze změn bylo automatické vysunutí menu hned po spuštění aplikace. V menu bylo také označeno, pokud sekce videí obsahuje nové video od posledního spuštění aplikace. Dále přeskládání menu, odstranění oblíbených z podsekcí(byla zachována pouze hlavní sekce oblíbené dostupná z menu). Byl změněn design sekce class se videích, aby nebylo stále dokola zobrazeno logo projektu. A zpracovány všechny další připomínky.

Byl nalezen pouze jeden uživatel vlastnící android TV. Ten tudíž aplikaci otestoval, stejně jako v první fázi byla pak i zde otestována alespoň na emulátoru. V této verzi aplikace nebyly nalezeny žádné problémy, které by bylo potřeba opravovat.

Závěr

Nejprve byla provedena analýza aplikací zabývajících se podobnou tématikou. Analyzované aplikace sloužily jako vzor pro to, jak by se aplikace měla chovat a vypadat, nebo naopak jak by vypadat neměla. Poté byly analyzovány možnosti přehrávání videa, zde byly uvedeny výhody a nevýhody jednotlivých řešení a bylo vybráno nejlepší řešení přehrávače. Následně byl proveden návrh API a databáze, poté bylo navrhnuto chování aplikací a vzhled reprezentovaný wireframes. Následně byly obě aplikace implementovány. Na závěr byly aplikace otestovány a připomínky z testování do aplikací zahrnuty.

Podařilo se splnit původní zadání práce. Aplikace zprostředkovávají nejdůležitější obsah webové části projektu. Aplikace jsou plně funkční, otestované a připravené k nasazení mezi uživatele. Registrace uživatele sice není zatím možná, tento požadavek však nebyl v úplně původním zadání, navíc nebyl splněn z důvodu technických problémů redakčního systému WordPress a jeho API.

Po celou dobu probíhala aktivní spolupráce s producentem TV, díky tomu se jednotlivé požadavky postupně vyvíjely. Odevzdáním diplomové práce rozhodně nekončí vývoj aplikací. Neustále se rozvijí webová část projektu a spolu s ní se budou nadále vyvíjet i android aplikace.

Stejně tak je na aplikacích dále co zlepšovat. Největší slabinou aplikací je nyní API, zejména problém s dlouhým načítání rádia, proto se budou první budoucí práce ubírat tímto směrem. Dále je důležité do aplikací zprostředkovat i další sekce webu, které v ní zatím nejsou dostupné, aby uživatelé nemuseli využívat web i aplikaci současně. Dalším cílem do budoucna je implementace technologie Chromecast[24] pro zprostředkování obsahu do televizorů obsahujících i jiný operační systém než android.

Literatura

[1] Czech Republic Land of stories [online]. [cit. 2016-6-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.czechtourism.land_of_stories

[2] Google trips [online]. [cit. 2016-6-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.google.android.apps.travel.onthego

[3] TouristEye - Travel Guide [online]. [cit. 2016-6-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.touristeye

[4] Guides by Lonely Planer [online]. [cit. 2016-6-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.lonelyplanet.guides

[5] TVA: Travel Video Collections [online]. [cit. 2016-6-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.topvideos.travel

[6] Android PlayerAPI [online]. [cit. 2016-6-20]. Dostupné z: https://developers.google.com/youtube/android/player/

[7] Android MediaPlayer [online]. [cit. 2016-6-28]. Dostupné z: https://developer.android.com/guide/topics/media/exoplayer.html

[8] JieCao Video Player [online]. [cit. 2016-6-29]. Dostupné z: https://github.com/lipangit/JieCaoVideoPlayer

[9] API [online]. [cit. 2016-7-17]. Dostupné z: http://www.itbiz.cz/slovnik/informacni-technologie-it/api

[10] REST: architektura pro webové API [online]. [cit. 2016-7-17]. Dostupné z: |https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/

[11] WordPress REST API (Version 2) [online]. [cit. 2016-7-18]. Dostupné z: https://cs.wordpress.org/plugins/rest-api/

[12] Basic access authentication [online]. [cit. 2016-7-25]. Dostupné z: http://httpwg.org/specs/rfc7617.html

[13] Wireframing [online]. [cit. 2016-7-26]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/wireframing.html

[14] Balsamiq Mockups 3 [online]. [cit. 2016-7-26]. Dostupné z: https://balsamiq.com/download/

[15] Up and running with material design [online]. [cit. 2016-7-26]. Dostupné z: https://developer.android.com/design/index.html

[16] Designing for Android TV [online]. [cit. 2016-7-28]. Dostupné z: https://developer.android.com/design/tv/index.html

[17] Getting Started With RecyclerView and CardView on Android [online]. [cit. 2016-10-2]. Dostupné z: https://code.tutsplus.com/tutorials/getting-started-with-recyclerview-and-cardview-on-android--cms-23465

[18] GJava Code Examples for android.text.style.LeadingMarginSpan [online]. [cit. 2016-10-6]. Dostupné z: http://www.programcreek.com/java-api-examples/index.php?api=android.text.style.LeadingMarginSpan

[19] CSV to JSON [online]. [cit. 2016-10-15]. Dostupné z: http://www.csvjson.com/csv2json

[20] Dashboards [online]. [cit. 2016-10-15]. Dostupné z: https://developer.android.com/about/dashboards/index.html

[21] Facade Pattern Tutorial with Java Examples [online]. [cit. 2016-08-10]. Dostupné z: https://dzone.com/articles/design-patterns-uncovered-1

[22] Gson [online]. [cit. 2016-08-10]. Dostupné z: https://github.com/google/gson

[23] JW Player [online]. [cit. 2016-6-28]. Dostupné z: https://www.jwplayer.com/

[24] Chromecast [online]. [cit. 2016-12-15]. Dostupné z: https://www.google.com/chromecast/built-in/

Příloha A

Seznam použitých zkratek

CATVUSA Czech-American TV in USA

POI Point of interest

API Application Programming Interface

UI User Interface

HTTP Hypertext Transfer Protocol

SDK Software development kit

RDM Digital Rights Management

DASH Dynamic adaptive streaming over HTTP

REST Representational state transfer

PHP Hypertext Preprocessor

JSON JavaScript Object Notation

XML Extensible markup language

SQL Structured Query Language

DB Databáze

SP Shared Preferences

SW Software

OS Operační systém

IDE Integrated Development Environment

AS Android Studio

WS WebService

URL Uniform Resource Locator

URI Iniform Resource Identifier

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `apk`
  - `mobile.apk` ..... APK soubor pro instalaci na mobilní zařízení
  - `tv.apk` ..... APK soubor pro instalaci na televizory
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF