FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Aplikace pro konzumaci obsahu pro Android TV
Student:	Timotej Adamec
Vedoucí:	Ing. Marek Kodr
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem bakalářské práce je vytvořit aplikaci pro operační systém Android TV, která umožní uživatelům konzumovat audio a video obsah. Aplikace se zaměří na prezentační vrstvu aplikace a naváže na již existující systémy.

Proveďte následující kroky:
1) Analyzujte konkurenční aplikace na trhu.
2) Na základě výsledků definujte požadavky na aplikaci.
3) Implementujte aplikaci pro operační systém Android TV.
4) Otestujte aplikaci a implementujte analytics.
5) Zpřístupněte aplikaci uživatelům.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 28. srpna 2023 v Praze.

Bakalářská práce

APLIKACE PRO KONZUMACI OBSAHU PRO ANDROID TV

Timotej Adamec

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Marek Kodr, MBA
13. dubna 2024

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu Markovi Kodrovi za stálou podporu, ochotu a veškerou organizaci spojenou s implementační a písemnou částí práce. Poděkovat chci také lidem z Quanti, kteří mi byli během implementace ochotni pomáhat. Děkuji Quanti a Forendors za důvěru a možnost na tomto projektu pracovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při 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. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 13. dubna 2024

Abstrakt

Tato práce se zabývá návrhem a vývojem převážně prezentační vrstvy televizní aplikace Forendors pro operační systém Android TV v programovacím jazyce Kotlin a v uživatelském rozhraní tvořeném v sadě nástrojů Compose. Návrh uživatelského rozhraní je v práci podložen analýzou existujících aplikací. Výsledkem práce je funkční Android TV aplikace, která bude sloužit uživatelům Forendors ke sledování obsahu jejich oblíbených tvůrců.

Klíčová slova TV aplikace, Android TV, Kotlin, Compose, Forendors

Abstract

This thesis deals with the design and development of mainly the presentation layer of the Forendors TV application for the Android TV operating system in the Kotlin programming language and in the user interface created in the Compose toolkit. The design of the user interface is supported by an analysis of existing applications. The result of the work is a working Android TV application that will be used by Forendors users to watch content from their favorite creators.

Keywords TV app, Android TV, Kotlin, Compose, Forendors

Seznam zkratek

D-Pad	Ovládací prvky ovladače (nahoru-dolů-levá-pravá-střed)
DTS	Digital Theatre System
PCM	Pulse-Code Modulation
HDR	High Dynamic Range

Úvod

Online platformy pro tvůrce, kteří chtějí svůj obsah všeho druhu nabízet za předplatné a kde lidé platí tvůrcům napřímo za jejich práci a není v komunikaci přítomná žádná propagace produktů či reklamy, jsou další evolucí konzumace obsahu. Dnešní internet je nasycený reklamami, ale lidé jsou ochotní za kvalitní obsah platit. S touto vizí vzniklo Forendors (dříve Pickey). Jde o jednu z domácích alternativ americké služby Patreon, která tento systém prokopala. Původně bylo Forendors pro předplatitele dostupné jen na webu, poté i na mobilních zařízeních a nyní doplní portfolio platforem i Android TV.

Mít nativní TV aplikaci je v případě jakékoliv video konzumační platformy dobrým nápadem, protože pro mnoho uživatelů může být právě televize hlavním zařízením, kde obsah sledují. TV aplikace zajistí vysoký standard uživatelského komfortu, protože lidé mají možnost sledovat svůj oblíbený obsah bez složitých procedur nebo nutnosti přepínání mezi různými zařízeními. Důsledkem může být například uživatel, který vždy sledoval nebo poslouchal obsah ze svého mobilního zařízení, ale nyní, kvůli přímému a rychlému přístupu k televizní aplikaci může bez nadbytečného úsílí obsah sledovat na velké obrazovce. To umožňuje plynulý a přirozený zážitek ze sledování na velkých obrazovkách. Nejen, že stávající uživatelé sledující na telefonech či monitorech nyní mohou sledovat svůj obsah na televizi přímo, ale s televizní aplikací se také otevřou dveře pro širší spektrum uživatelů. Televizi mnozí lidé běžně používají, což znamená, že aplikace osloví širší publikum, která sleduje obsah převážně nebo jen na televizních zařízeních. To zvyšuje dostupnost obsahu pro širší vrstvy uživatelů a umožňuje jim vychutnat si obsah bez ohledu na to, jaké zařízení mají právě k dispozici.

Televizní aplikace zvyšuje hodnotu nejen pro konzumenty, ale i tvůrce. Pro tvůrce obsahu totiž přináší TV aplikace nové možnosti prezentace své práce na větších obrazovkách, čímž poskytuje platformu pro prezentaci jejich tvorby v optimální kvalitě. Tímto způsobem zvyšuje TV aplikace schopnost tvůrců oslovit širší publikum a zároveň nabídnout divákům lepší zážitek ze sledování. Pro předplatitele jde o přidanou hodnotu, protože mají možnost sledovat své oblíbené obsahy na preferovaném zařízení bez kompromisů ve kvalitě.

Unikátnost konceptu aplikace Forendors TV spočívá v tom, že bude jedinou platformou na světě v sekci platforem pro předplacený obsah tvůrců, která podporuje nativní streamování na TV. Forendors tak může předplatitelům a tvůrcům nabízet své služby na platformách webu, mobilních zařízení a televizí, což nemá v onom poli konkurenci. TV aplikace tak posílí konkurenční pozici Forendors na jednom z dnes nejrychleji rostoucích oblastí internetu pro běžné uživatele. Tímto způsobem se Forendors Android TV aplikace stane nedílnou součástí Forendors, posílí její pozici na trhu a přinese výhody jak uživatelům, tak i tvůrcům obsahu.

Kapitola 1
Analýza

V této kapitole se nejdříve věnuji představení domény. Navazuji specifikací požadavků, jak těch unikátních pro vlastní aplikaci, tak těch, které musí splnit každá aplikace vydávaná na Android TV a specifikací případů užití. Převážnou část analýzy pak věnuji analýze existujících TV aplikací, jejichž analýzu podkládám doporučeními Android TV designu. Nakonec provádím analýzu vlastností a možností operačního systému Android TV.

1.1 Creators economy platformy

Creators economy platformy umožňují tvůrcům obsahu monetizovat jejich práci, angažovat se s podporovateli a vytvářet komunity kolem svých zájmů a expertízy. V posledních letech rostou creators economy platformy, které dávají tvůrcům možnost nabídnout svému publiku předplatné, za které odběratelé dostávají možnost sledovat obsah tvůrce. Mezi tyto platformy patří i Forendors, které je jednou z hlavních creators economy platforem tohoto nového typu působících v České republice. Způsob monetizace tvůrci pomocí předplatných umožňuje tvůrcům vytvořit komunity, kde nefigurují žádné třetí strany, které se snaží prodat svůj produkt, ale kde se tvoří přímý vztah mezi tvůrci a jejich fanoušky. Obsahem na platformě může být cokoliv, co daná platforma podporuje. V případě Forendors jsou obsahem příspěvky, které mohou obsahovat jen text nebo i přílohy, jako obrázky, video, či audio. Oproti globálním platformám jako Patreon, nabízí Forendors lokálně zaměřené služby, které jsou přizpůsobené potřebám a preferencím českých uživatelů. Tento lokální přístup může představovat výhodu v lepším porozumění potřebám českých tvůrců a uživatelů.

1.2 Specifikace požadavků

Tato kapitola specifikuje všechny funkční a nefunkční požadavky nezbytné pro zahájení procesu vývoje. Po konzultaci s vlastníky a vývojáři Forendors, analýze obecných požadavků pro Android TV aplikaci a analýzy aplikací, která probíhala souběžně s iteracemi definování požadavků, jsem sestavil následující požadavky.

1.2.1 Funkční požadavky

Funkční požadavky definují vlastnosti nebo funkce, které musí software vykonávat. Popisují chování systému. [1]

FP1 Přihlášení Každý registrovaný a ověřený uživatel musí mít možnost se přihlásit, ať už je registrován pod účtem Forendors či přes třetí stranu jako je Google nebo Facebook. Od okamžiku přihlášení již

nebude nutné se po každém spuštění aplikace přihlašovat.

FP2 Dashboard Aplikace musí uživateli zobrazovat nový odebíraný obsah. K tomu bude sloužit personalizovaný dashboard s video a audio epizodami relevantních tvůrců.

FP3 Video přehrávač Aplikace musí být schopna přehrávat video obsah. Video přehrávač musí být dobře přizpůsobený k použití s ovladačem.

FP4 Audio přehrávač Aplikace musí umožňovat přehrávat audio obsah, ať už krátkotrvající či dlouhotrvající, jako třeba podcasty. Audio přehrávač musí být dobře přizpůsobený k použití s ovladačem.

FP5 Vyhledávání tvůrce Aplikace umožní vyhledávání tvůrců.

FP6 Profil tvůrce Aplikace umožní zobrazení profilu kteréhokoliv tvůrce a jeho videí či audia tak, aby byl uživatel vždy schopný dohledat kterékoliv audio nebo video.

FP7 Odhlášení Aplikace bude podporovat odhlášení uživatele.

1.2.2 Nefunkční požadavky

NP1 TV UI/UX Efektivní (minimální počet navigačních posunů), předvídatelné a intuitivní UI přizpůsobené použití z gauče s ovladačem (tj. navigačními možnostmi nahoru, dolů, levá, pravá, střed [dále jen D-Pad] a tlačítko zpět). Aplikace bude vycházet z analýzy a doporučení Googlu pro TV design.

NP2 Přímočará dostupnost Aplikace bude jednodušše stažitelná a spustitelná. Stažení, na které jsou uživatelé zvyklý zprostředkuje Google Play Store a aplikace pak bude uživateli dostupná mezi ostatními nainstalovanými aplikacemi.

NP3 Použitelnost na všech podporovaných zařízeních Aplikace musí být použitelná na všech zařízeních, na kterých může být nainstalována. Jako indikátor splnění požadavku bude brána použitelnost aplikace na zařízení Xiaomi Mi TV Stick, které se prodává za 1 000 korun.

1.2.3 Požadavky pro publikaci na Google Play

Google vytvořil seznam požadavků pro kvalitu TV aplikace (viz [2]), který obsahuje jak UI/UX, tak funkční požadavky. Tyto požadavky je nutné splnit, jinak aplikace nebude na Google Play kvalifikovaná jako TV aplikace a nebude v obchodě Google Play na TV zařízeních dostupná.

„Uživatelé mají při sledování televize jiná očekávání než při používání telefonu nebo tabletu. Typický uživatel televizoru sedí ve vzdálenosti asi 3 metry od obrazovky, takže drobné detaily jsou méně patrné a malý text se špatně čte. Protože uživatelé sedí dál od televizoru, musí k navigaci a výběru používat zařízení s dálkovým ovládáním, nikoliv se dotýkat prvků na obrazovce. Tyto rozdíly významně ovlivňují požadavky na to, co tvoří dobrý uživatelský zážitek z používání aplikací na televizních zařízeních.“ [2]

V dokumentu jsou zmíněné také pokyny pro design na Android TV (dále jen “doporučení”), které by měly být prvním krokem pro design TV aplikace. Tato doporučení budou probrána v sekci s doporučeními Android TV designu.

Kromě UI/UX požadavků a nefunkčních požadavků existují ještě požadavky, které je nutné splnit, pokud má být aplikace publikována na Google Play. Google Play je obchod s aplikacemi vlastněný a spravovaný Googlem, který je přítomný v každém Android TV zařízení. Více o distribuci v kapitole publikace.

Vyberu a sepíšu zde požadavky, o kterých si myslím, že nejsou samozřejmé a je nutné jim věnovat pozornost a vynechám ty, které nejsou relevantní (např. požadavky na herní aplikace) nebo jsou příliš technické. Požadavky, které zde nejsou zmíněny, budou respektovány a jejich splnění bude stejně nutné, jako těch zde vypsaných.

Tabulka 1.1 Vybrané UI/UX Google požadavky

Typ	Zkratka	Popis
Spouštěč	TV-LM	Aplikace po instalaci zobrazuje ikonu spouštěče mezi ostatními aplikacemi.
Spouštěč	TV-LB	Aplikace zobrazuje jako ikony spouštěče banner o velikosti 320x180 pixelů a ikonu aplikace o velikosti alespoň 160x160 pixelů (při hustotě xhdpi).
Navigace	TV-DM	Aplikace není závislá na zařízení dálkového ovládání s tlačítkem Menu pro přístup k ovládacím prvkům uživatelského rozhraní.
Navigace	TV-DB	Stisknutí tlačítka Zpět vede zpět na domovskou obrazovku Android TV.

1.2.3.1 UI/UX Google požadavky

Vybrané požadavky pro vizuální design a interakci s uživatelem vizte v tabulce 1.1. Tato kritéria pomáhají zajistit, že aplikace dodržuje kritické vzory designu a interakce pro konzistentní, intuitivní a příjemný uživatelský zážitek na televizních zařízeních.

1.2.3.2 Funkční Google požadavky

Druhou skupinou požadavků jsou funkční požadavky (viz tabulka 1.2). Tato kritéria zajišťují, že je aplikace správně nakonfigurována a poskytuje očekávané funkční chování. Použité termíny Ambient Mode a karta Now Playing budou popsány v sekci Android TV.

1.2.3.3 Google Play požadavky

Tyto požadavky zajistí, že aplikace bude v rámci Google Play správně nakonfigurovaná (viz tabulka 1.3).

1.3 Případy užití

Na základě funkčních požadavků jsou v této části specifikovány scénáře případů užití, které popisují chování uživatele a jeho interakce s aplikací. Pokrytí funkčních požadavků případy užití je podrobně uvedeno v tabulce 1.4. Aktérem každého z případů užití je uživatel. Diagram případů užití lze vidět viz 1.1. Diagram obsahuje také testovací případy, které ukazují, jak budou uživatelské interakce, specifikované případy užití, testovány. Testující případy budou blíže probrány později.

UC1 Přihlásit Případ užití umožňuje uživateli přihlásit se do aplikace pod svým registrovaným účtem pomocí přihlašovacích údajů (e-mail + heslo) nebo pomocí třetí strany (Google či Facebook). V případě neregistrovaného uživatele je uživateli zobrazena zpráva o nutnosti registrace.

Obrazovky

- Přihlášovací obrazovka.

Pre-condition

- Uživatel je registrován.

Hlavní scénář

1. Uživatel vyplní přihlašovací údaje svého Forendors účtu.
2. Uživatel klikne na tlačítko přihlášení.
3. Aplikace provede kontrolu přihlašovacích údajů.
4. Aplikace uživatele přihlásí do aplikace.

1. alternativní scénář

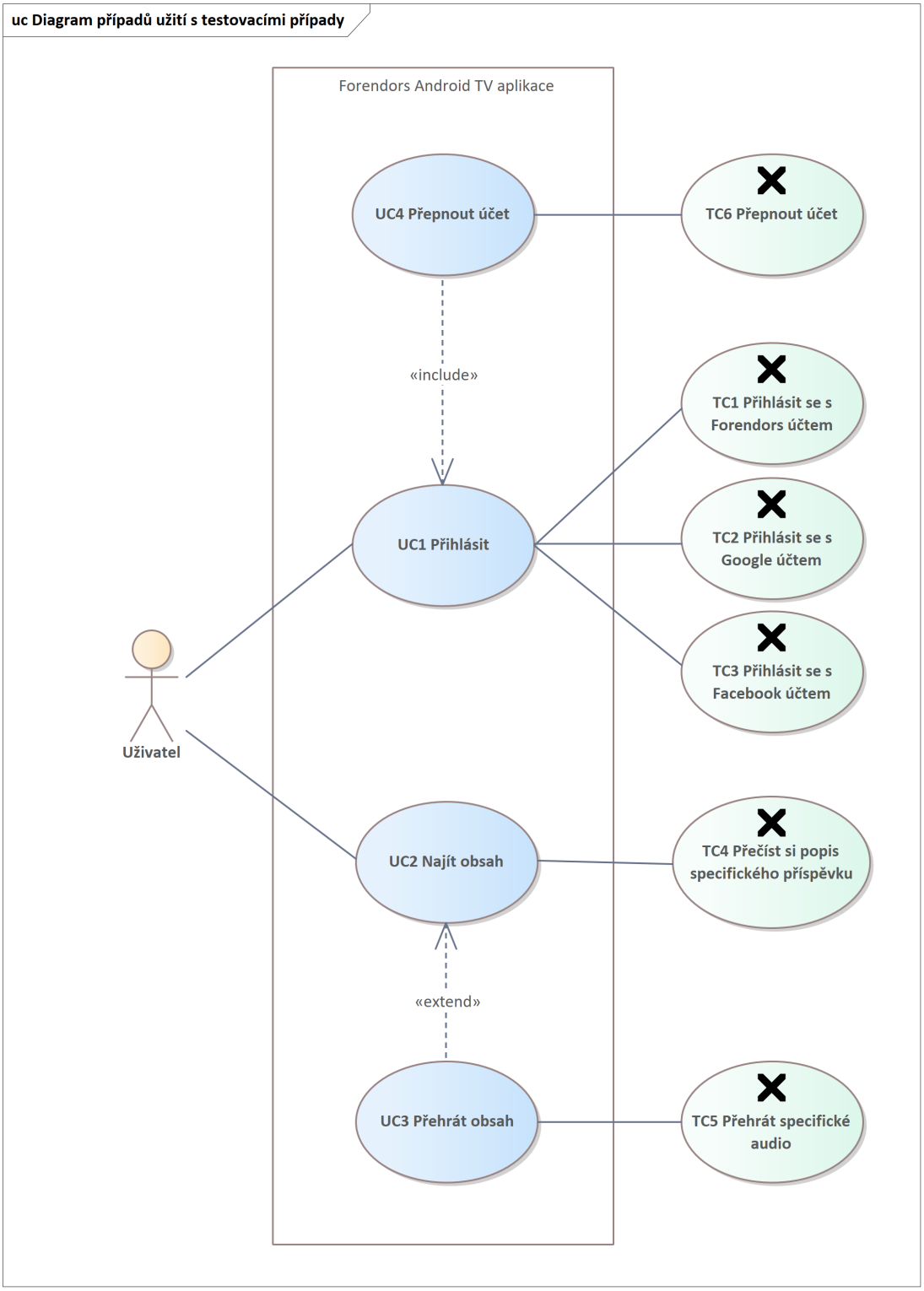
1. Uživatel klikne na tlačítko přihlášení se přes Google.

Tabulka 1.2 Vybrané funkční Google požadavky

Typ	Zkratka	Popis
Webový obsah	TV-WB	Pro webový obsah může aplikace používat pouze komponenty WebView. Aplikace se nesmí pokoušet o spuštění aplikace webového prohlížeče.
Přehrávání médií	TV-NP	Pokud aplikace pokračuje v přehrávání zvuku nebo videa i poté, co se uživatel vrátí na domovskou obrazovku nebo přepne na jinou aplikaci, aplikace na řádku doporučení domovské obrazovky zobrazí kartu Now Playing, aby se uživatelé mohli vrátit do aplikace a ovládat přehrávání.
Přehrávání médií	TV-PA	Pokud aplikace nabízí kartu Now Playing, přejde uživatel jejím výběrem na obrazovku, která mu umožní pozastavit přehrávání.
Přehrávání médií	TV-PC	Při přehrávání videa nebo zvuku se stisknutím středového tlačítka D-padu přehrávané médium pozastaví. Když je přehrávání pozastaveno, stisknutím prostředního tlačítka D-padu se přehrávání obnoví. Tlačítka D-Padu vlevo a vpravo slouží k rychlému převíjení aktuální stopy vpřed, resp. vzad.
Přehrávání médií	TV-PN	Položky jsou do kanálu Watch Next přidávány na základě Pokynů Watch Next pro vývojáře aplikací.
Ambient Mode	TV-BU	Při aktivním přehrávání videa spuštěného uživatelem, aplikace zabrání zařízení přejít do Ambient Mode.
Ambient Mode	TV-BY	Pokud není uživatelem spuštěné aktivní přehrávání videa nebo animace, aplikace nebrání zařízení v přechodu do režimu Ambient Mode.
Ambient Mode	TV-BA	Při přehrávání pouze zvuku aplikace nebrání zařízení v přechodu do režimu Ambient Mode, jestliže aplikace neimplementuje během přehrávání hudby zážitek z nestatických obrazů, jako jsou hudební videa nebo obrázky.

Tabulka 1.3 Vybrané Google Play požadavky

Typ	Zkratka	Popis
Play zásady	TV-G2	Aplikace musí splňovat požadavky Centra zásad pro vývojáře Play.
Přihlašovací údaje	TV-G5	U aplikací s placenými funkcemi je nutné zadat přihlašovací údaje do Konzole Google Play, aby bylo možné otestovat celé prostředí aplikace.



Obrázek 1.1 Diagram případů užití s testovacími případy

2. alternativní scénář

1. Uživatel klikne na tlačítko přihlášení se přes Facebook.

UC2 Najít obsah Případ užití umožňuje uživateli najít obsah, který by si chtěl zobrazit. Uživatel může obsah najít skrze doporučený obsah na domovské obrazovce nebo skrze obsah dostupný na profilu tvůrce. Na profil tvůrce lze přejít z vyhledávací obrazovky či z přehrávače, ve kterém se přehrává obsah daného tvůrce.

Obrazovky

- Domovská obrazovka.
- Vyhledávací obrazovka.
- Video přehrávač.
- Audio přehrávač.
- Profil tvůrce.

Hlavní scénář

1. Uživatel najde obsah na domovské obrazovce mezi doporučeným obsahem.

1. alternativní scénář

1. Uživatel přejde na vyhledávací obrazovku.
2. Uživatel zadá jméno hledaného tvůrce.
3. Uživatel přejde na profil tvůrce.
4. Uživatel najde obsah na profilu tvůrce.

2. alternativní scénář

1. Uživatel najde jiný obsah od hledaného tvůrce v doporučeném obsahu na domovské obrazovce.
2. Uživatel spustí nalezený obsah a přejde do přehrávače.
3. Uživatel v přehrávači přes tlačítko tvůrce přejde na profil tvůrce.
4. Uživatel na profilu tvůrce najde hledaný obsah.

UC3 Přehrát obsah Případ užití umožňuje uživateli spustit přehrávatelný obsah, který si uživatel najde, ať už se jedná o video či audio.

Obrazovky

- Video přehrávač.
- Audio přehrávač.

UC4 Přepnout účet Případ užití umožňuje se uživateli po přihlášení odhlásit a poté znovu přihlásit za jiný účet.

Obrazovky

- Přihlašovací obrazovka.
- Domovská nebo vyhledávací obrazovka (zde je zobrazena navigační lišta).

Tabulka 1.4 Mapování funkčních požadavků na případy užití

	UC1	UC2	UC3	UC4
FP1	✓			✓
FP2		✓		
FP3		✓	✓	
FP4		✓	✓	
FP5		✓		
FP6		✓		
FP7				✓

1.4 Android TV

Na poli televizí s Android TV existují televize s operačním systémem Android TV a Google TV. Google TV je nadstavba nad starší Android TV. Zatímco Android TV existuje už od roku 2014, Google TV bylo představeno v roce 2020 mělo do konce roku 2022 nahradit standardní Android TV, počínaje set-top boxy, dongly a chytrými televizemi v roce 2021 [3]. Operační systém Android TV se zatím dočkal pouze změny designu domovské obrazovky s názvem "Discover"UI, která se začala šířit v únoru 2021 [4]. Vývojem pro Android TV se tak zde myslí zároveň vývoj pro Google TV, jakožto nadstavbu rozhraní Android TV.

Hlavním rozdílem mezi Android a Google TV je, že Google TV klade důraz na doporučování a vyhledávání obsahu v různých službách a nainstalovaných aplikacích. „Google TV chce uživatelům usnadnit a zrychlit přístup k nejsledovanějšímu a doporučenému obsahu. Toho dosahuje tím, že veškerý obsah z jednotlivých streamovacích služeb smíchá dohromady a vloží do jednoho rozhraní, takže můžete vyhledávat ve všech streamovacích službách jedním kliknutím." [5]. Google TV také využívá algoritmy pro doporučování obsahu na základě historie sledování a preferencí uživatele. To znamená, že čím více uživatel sleduje, tím lépe se Google TV naučí, co se mu líbí, a může mu tak poskytnout lepší doporučení. V neposlední řadě Google TV nabízí také seznam sledování, který umožňuje snadno přidat obsah z různých aplikací do záložek a sledovat jej později. To lze provést třeba z telefonu nebo Google Search. Při pozdějším vývoji aplikace je vhodné umožnit uživatelům vyhledávání a poskytnout doporučení pomocí API Android TV. [6]

Vývoj aplikací na Android TV obnáší jak standardní vývoj Android aplikací s komponentami jako activities a content providers, tak i práci se specifickými TV funkcionalitami a přehrávacími specifikami. Funkcionality, které existují na Android TV existují i na Google TV. Jedněmi z funkcionalit je tzv. Ambient Mode, Now Playing karta a Watch Next kanál.

1.4.1 Ambient Mode

Protože naše aplikace bude sloužit ke konzumaci obsahu, bude také muset zabraňovat spuštění Ambient Mode při přehrávání video obsahu, aby bylo zabráněno přerušovanému sledování.

„Režim Ambient Mode je spořič obrazovky integrovaný do Google TV a Android TV. Jeho účelem je zabránit dlouhodobému zobrazování statických obrázků. To je důležité u zobrazovacích technologií, jako je OLED, které mohou být náchylné k vypalování obrazovky. Operační systém přepne zařízení do režimu Ambient Mode po 10 minutách nečinnosti uživatele. Po delší době nečinnosti uživatele (definované nastavením úsporného režimu zařízení) přejde operační systém do úsporného režimu a vypne displej. Aplikace pro přehrávání médií mohou zabránit přechodu zařízení do režimu Ambient Mode, přestože s ním uživatel neinteraguje, například při sledování filmu. Pokud uživatel do 30 minut od vstupu do režimu Ambient Mode se zařízením interaguje, obnoví se aplikace, která byla aktivní při vstupu do režimu Ambient Mode. Pokud uživatel se zařízením interaguje déle než 30 minut po vstupu do režimu Ambient Mode, vrátí se na domovskou obrazovku. Pokud uživatel spustí zařízení pomocí tlačítka napájení v době, kdy je v režimu úspory energie, bude přenesen na domovskou obrazovku. Případně pokud uživatel spustí zařízení v době, kdy je v úsporném režimu, pomocí tlačítek konkrétních aplikací (například YouTube),

bude přesměrován přímo do této aplikace. Pokud, jako vývojář, používáme relaci `MediaSession` pro přehrávání médií, bude aplikace při zahájení přehrávání implicitně uzamčena částečným probuzením. Pozastavení přehrávání tento zámek probuzení neuvolní. To znamená, že zařízení může automaticky přejít do režimu Ambient Mode, ale následně nepřejde do úsporného režimu. Pokud uživatel do 30 minut od vstupu do režimu Ambient Mode se zařízením interaguje, aplikace se obnoví." [7].

Z popisu Ambient Mode vychází povinnost aplikace pro manipulaci s ním. Aplikace by neměly bránit zařízení v přechodu do Ambient Mode při zastavení nebo pozastavení přehrávání. Při přehrávání zvuku by Ambient Mode být zabraňováno nemělo (pokud neimplementují vlastní spořič obrazovky s nestatickým obrazem), protože audio přehrávání bude v Ambient Mode pokračovat. Je to z důvodu implicitního držení částečného zámku probuzení u přehrávání zvuku v systému Android. To nezabrání přechodu do Ambient Mode, ale zabrání následnému přechodu do úsporného režimu. Přehrávání tedy bude pokračovat i poté, co zařízení přejde do Ambient Mode, ale zařízení bude zabráněno v přechodu do režimu spánku, aby bylo umožněno nepřerušované přehrávání [7].

1.4.2 Now Playing karta

Aplikace TV musí při přehrávání médií na pozadí zobrazovat kartu Now Playing. Tato karta umožňuje uživatelům vrátit se do aplikace, která právě přehrává média. Framework Android zobrazuje kartu Now Playing na domovské obrazovce, pokud je aktivní `MediaSession`. Karta obsahuje metadata médií, například obal alba, název a ikonu aplikace. Když uživatel kartu vybere, systém otevře aplikaci a v ní přehrávač, kde může uživatel přehrávání pozastavit. Karta je odstraněna z obrazovky launcheru, když aplikace deaktivuje relaci nebo když jiná aplikace zahájí přehrávání médií. Pokud je přehrávání zcela zastaveno a nejsou k dispozici žádná aktivní média, relace musí být okamžitě deaktivována. Pokud je přehrávání pozastaveno, relaci musí být deaktivována po určité prodlevě, obvykle od 5 do 30 minut. Kdykoli aplikace aktualizuje stav přehrávání v relaci `MediaSession`, karta Now Playing se aktualizuje a zobrazí stav aktuálního média. Podobně může aplikace aktualizovat `MediaMetadata`, aby poskytla kartě informace o aktuální relaci, například název, titulky a různé ikony [8].

1.4.3 Watch Next kanál

Kanál Watch Next je druhý řádek, který se zobrazí na domovské obrazovce za řádkem aplikací. Tento kanál vytváří a udržuje systém. Aplikace může do kanálu Watch Next přidávat pořady, které:

- uživatel označil jako zajímavé.
- přestal sledovat v polovině.
- souvisejí s obsahem, který uživatel sleduje (například další díl seriálu nebo další série pořadu).

Na domovské obrazovce má kanál Watch Next označení "Play Next". Třídy systému Android používané ke správě kanálu Watch Next se však jmenují `WatchNextProgram` a `WatchNextPrograms`. Kanál Watch Next má pár omezení: aplikace nemůže přesouvat, odstraňovat nebo skrývat řádek kanálu Watch Next [9].

1.4.4 Výstupové možnosti

U televizí je také třeba zaměřit se na výstupové možnosti, protože televize mohou mít připojených více audio výstupů s různými schopnostmi pro přehrávání audia nebo mohou být schopné zobrazovat obraz se speciálními vlastnostmi, které by aplikace měly být schopné využívat.

Google dokumentace o audio možnostech: „Zařízení Android TV mohou mít současně připojeno více zvukových výstupů: televizní reproduktory, domácí kino připojené přes HDMI, sluchátka Bluetooth atd. Tato zařízení se zvukovým výstupem mohou podporovat různé zvukové funkce, například

kódování (Dolby Digital+, DTS a PCM), vzorkovací frekvence a kanály. Například televizory připojené přes HDMI mají podporu mnoha kódování, zatímco připojená sluchátka Bluetooth obvykle podporují pouze PCM. Seznam dostupných zvukových zařízení a směrovaných zvukových zařízení se také může měnit připojením zařízení HDMI, připojením nebo odpojením sluchátek Bluetooth nebo změnou nastavení zvuku uživatelem. Protože se schopnosti zvukového výstupu mohou měnit i při přehrávání médií aplikacemi, musí se aplikace těmto změnám přizpůsobit a pokračovat v přehrávání na novém směrovaném zvukovém zařízení a jeho schopnostech. Výstup v nesprávném zvukovém formátu může vést k chybám nebo k nepřehrávání zvuku. Aplikace mají možnost výstupu stejného obsahu ve více kódováních, aby uživateli nabídly nejlepší zvukový zážitek v závislosti na možnostech zvukového zařízení. Například zvukový proud v kódování Dolby Digital je přehráván, pokud jej televizor podporuje, zatímco v případě, že není podpora Dolby Digital, je zvolen šířeji podporovaný zvukový proud PCM. V době přehrávání by tak aplikace měla vytvořit zvukovou stopu s nejlepším formátem zvuku podporovaným výstupním zvukovým zařízením.” [10].

V případě videa platí podobné možnosti pro volbu přehrávání videa podle možností výstupním zařízením. Jedná se o různá rozlišení (např. HD nebo 4K) či formáty přehrávání (např. HDR jako Dolby Vision). Narozdíl od audia se neočekává změna zařízení pro video výstup. Video výstupem je pouze TV. Aplikace se tak nemusí dynamicky přizpůsobovat různým výstupním zařízením, ale pouze dané TV.

1.5 Doporučení Android TV designu

Doporučení Android TV designu je soubor doporučení, praktik a návodů (viz [11]), jak by měla vypadat a fungovat aplikace na Android TV. Doporučení nebudu rozepisovat kvůli jejich rozsahu, který sahá od barev a typografii, až po individuální komponenty používané v TV aplikacích, ale při analýze existujících TV aplikací na ně budu odkazovat a zmíním, o čem daná doporučení jsou.

1.6 Analýza existujících TV aplikací

Cílem této části je analyzovat existující Android TV aplikace, které mají podobné funkce jako aplikace navržená v této práci – konzumace audio, video a streaming obsahu. Hlavním cílem je zjistit, jaké funkcionality nabízí, jakým stylem jsou implementovány a analyzovat UI/UX. To znamená také analyzovat, jak prvky odpovídají či neodpovídají doporučením od Googlu pro moderní design UI/UX TV aplikací a jestli neporušují UI/UX a funkční požadavky pro kvalitu aplikace. Doporučení nebo požadavky, které nebudou v analýze zmíněny jsou brány jako implicitně splňující. Následují tři aplikace, vybrané na základě jejich funkcí, které pokrývají funkční požadavky a případy užití této aplikace a na základě velké popularity – s popisem jejich funkcionalit, výhod a potenciálních problémů.

Analýza se bude zabývat:

- Přihlášením.
- Navigací.
- Video feedem.
- Audio feedem.
- Přehráváním videa.
- Přehráváním audia.
- Vyhledáváním.
- Synchronizací s ostatními zařízeními.
- Nákupem předplatného.

Sekce byly vybrány na základě analýzy funkčních požadavků a problémů, o kterých předpokládám, že mohou být relevantní a jsou tedy rovnou zahrnuté do analýzy. Na základě poznatků z této analýzy byly staré koncepty požadavků pozměněny do těch, které již byly zmíněny.

1.6.1 YouTube

Aplikace YouTube je populární platforma pro sdílení a sledování videí, která nabízí různé možnosti pro uživatele i tvůrce obsahu. Aplikace YouTube umožňuje uživatelům:

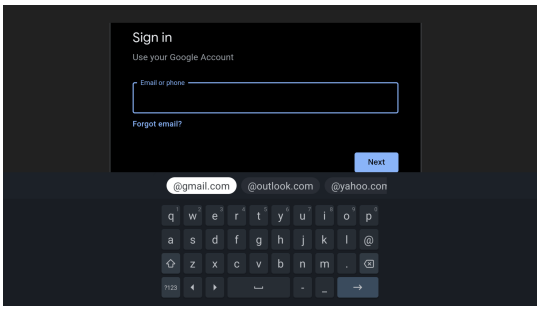
- Sledovat živá i nahraná videa.
- Poslouchat hudbu.
- Komentovat a hodnotit obsah.
- Odebírat kanály.
- Vytvářet vlastní playlisty.

YouTube má navíc více než 100 milionů stažení na TV Google Play Store [12], což ukazuje jeho velkou popularitu mezi uživateli televizních zařízení.

1.6.1.1 Přihlášení

YouTube lze používat bez účtu. Když se uživatel do aplikace chce přihlásit, využívá YouTube klasického přihlášení do Google účtu pomocí ručního vpisování emailu, kde se uživateli zobrazí systémová klávesnice a uživatel musí na klávesnici pomocí ovladače hledat znak po znaku, až zadaný mail potvrdí také tlačítkem na klávesnici (viz obrázek 1.3). Poté proces obdobně opakuje pro heslo, ale vstup uživatele se na obrazovce přeměnuje na tečkové znaky. Proces se odehrává ve `WebView` [1]. Pro použití jiného účtu může uživatel přes navigační destinaci přihlášeného účtu zvolit možnost “Změnit účet”, která využívá obrazovky volby Google TV účtu.

Obrázek 1.2 Splash obrazovka – YouTube



Obrázek 1.3 Přihlášení – YouTube

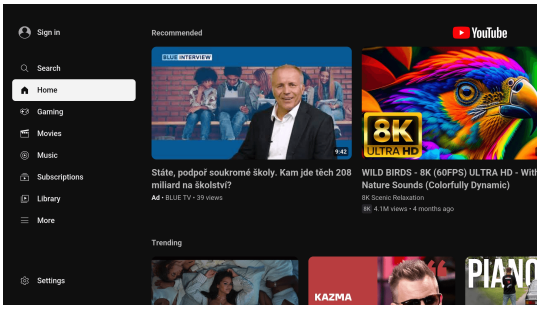
1.6.1.2 Navigace

YouTube po spuštění aplikace z TV spouštěče aplikací (dále jen “launcher”) zobrazuje tzv. “splash” obrazovku (viz obrázek 1.2), kde se většinou nachází logo aplikace a může obsahovat i prvek načítání, po které aplikace uživatele přesouvá na domovskou stránku aplikace.

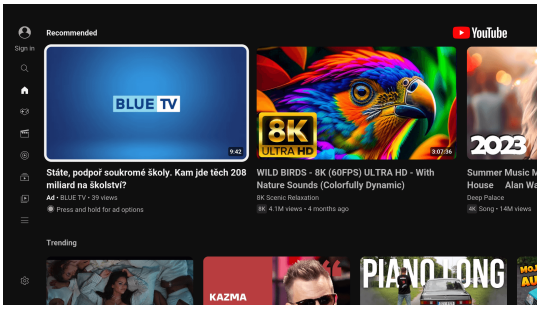
Aplikace používá pro hlavní navigaci postranní panel (viz obrázek 1.4), který je vždy přístupný a když není zrovna používaný a je minimalizovaný, zobrazuje ikony destinací a zvýrazňuje ikonu aktuální destinace (viz obrázek 1.5). Panel obsahuje kromě dalších segmenty dělení obsahu jako filmy, hraní a hudba.

[1] `WebView` zobrazuje webové stránky

Kamkoliv se uživatel v aplikaci potřebuje dostat, postačí mu vertikální postranní panel, což znamená, že uživatel se v aplikaci neztratí. Z navigačního panelu lze vystoupit jednoduchým stisknutím tlačítka “doprava” a do panelu se dá vstoupit z obsahu buď to po přechodu na začátek obsahového seznamu a poté stisknutím tlačítka “doleva” nebo stisknutím tlačítka “zpět” a tak se uživatel pomocí jednoho stlačení tlačítka (popř. více, pokud je vnořený v jedné z destinací), dostane k navigačním prvkům.



Obrázek 1.4 Postranní panel – YouTube



Obrázek 1.5 Minimalizovaný panel – YouTube

Z pohledu doporučení ohledně navigační architektury umožňuje YouTube uživatelům procházet uživatelské rozhraním s jasnou orientací. Umisťuje ovládací prvky, například akci vyhledávání, na místa, která se nepřekrývají s jinými klikatelnými prvky v souladu s doporučeními Googlu ohledně jasné cesty ke všem zaměřitelným prvkům [13]. Zároveň má rozložení takové, že využívá horizontální a vertikální osy. Každému směru přiřazuje specifickou funkci, aby bylo možné rychle procházet rozsáhlými hierarchiemi, což je také v souladu s doporučeními ohledně os [14].

Podle doporučení by první obrazovka, kterou uživatel vidí po spuštění aplikace z launcheru, měla být zároveň poslední obrazovkou, kterou vidí po návratu do launcheru po posledním stisknutí tlačítka “Zpět”, což YouTube splňuje. Za první obrazovku lze považovat “splash” obrazovku, ta má ale v daném doporučení vyjímku a při posledním stlačením tlačítka “Zpět” se uživatel navrací zpátky do launcheru a už nenavštěvuje “splash” obrazovku [15].

Co se týče doporučení pro navigační panel, splňuje YouTube všechna doporučení až na počet hlavních destinací, kterých by mělo být pět až šest. V případě YouTubu je to ale osm primárních destinací [16].

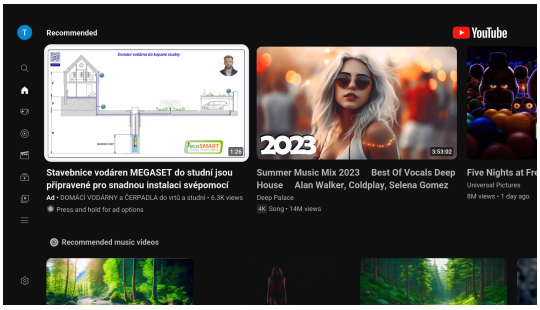
1.6.1.3 Video feed

Video feed je vertikálně rozdělen do sekcí podle kategorie videa a videa uvnitř sekce jsou řazena do horizontálního seznamu a uživatel se mezi nimi může pohybovat odpovídajícími směrovými tlačítky. Momentálně zaměřené video je ohraničeno bílým pruhem a po chvíli začne přehrávat několik sekund začátku videa, po kterém se na vybraném videu znovu zobrazí náhled.

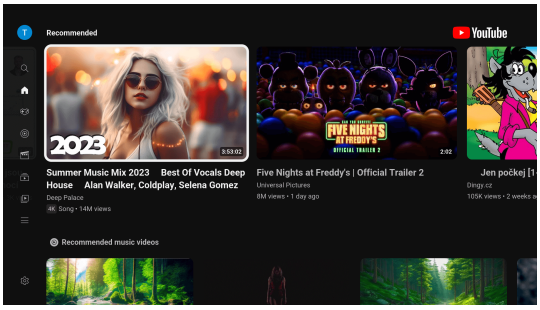
Video, které je živě streamované a je pro uživatele relevantní, mu je zobrazeno na domovské stránce mezi ostatními nestreamovanými videy. Liší se ikonou “LIVE” na náhledu a také tím, že pro něj není přehrávaný kus videa místo statického náhledu. Pokud důvod není čistě technický, může jim být, že pokud by se přehrávala nejčerstvější část videa, tak by se uživateli mohl přehrát kus videa, který by odhalil výsledek průběhu videa, např. při streamu nějakého zápasu. Pokud by se měl přehrávat začátek videa, tak ten bývá často doprovázený odpočtem nebo statickým obrazem před oficiálním zahájení streamu. To by mohlo uživatele zmást a mohli si myslet, že stream ještě oficiálně nezačal.

Uživatel při pohybu v sekci v seznamu videí mění zaměření zvoleného videa, ale momentální místo, kde se nachází zvýrazněné video se nemění, nýbrž se jen posouvá seznam a zvýrazněné video je vždy drženo na levé straně (viz obrázky 1.6 a 1.7). Přechod je doprovázen rychlou animací, která jasně indikuje posun seznamu. Zároveň je při posunu doprava posledně zaměřené video nerušivým způsobem stále zobrazené v pozadí pod navigačním panelem – to znamená, že navigační panel je mírně průhledný a vykreslovaný nad hlavním obsahem. Pokud je ale selekce videa na jiném než na prvním videu v seznamu a uživatel zmáčkne tlačítko “Zpět”, dostane so rovnou do navigačního panelu, což je ale v rozporu s

doporučeními Googlu ohledně použití předvídatelného chování tlačítka “Zpět” [17], kde by se správně mělo zaměřit první video v seznamu a až po dalším stlačení tlačítka “Zpět” by se měl zobrazit navigační panel.



Obrázek 1.6 Feed – YouTube



Obrázek 1.7 Feed (posunutý) – YouTube

1.6.1.4 Audio feed

Audio feed (viz obrázek 1.8)je na hlavní stránce podobný tomu videovému. Audio prvek je vizuálně menšího formátu, zvýraznění prvku je stejné, pohyb v seznamu dává vlevo od zvoleného prvku prostor jedné skladbě navíc a ukázka přehrává audio, přičemž vizuálně je přehrávání rozdílné a indikuje, že se jedná pouze o audio.

Dalším typem audio feedu je zobrazení alba (viz obrázek 1.9). Zde je seznam skladeb držený ve vertikálním seznamu, zřejmě protože skladby obsahují pouze textové elementy (někdy i dlouhé názvy) a nejde tak o žádný speciální prvek a nedává smysl tyto seznamové prvky skládat vedle sebe. V aplikaci existuje ještě jiný typ audio kolekce a to playlist (viz obrázek 1.10), který je velmi podobný tomu albovému, ale místo textových jednořádkových elementů používá pro skladby karty.

Obě výše zmíněné obrazovky při procházení seznamu drží vždy jednu až dvě skladby nad momentálně zvolenou skladbou, podobně, jako u horizontálních seznamů. Zde je jediným důvodem přehlednost navigace mezi skladbami. Po chvíli, co je nějaká skladba zaměrená, se začne skladba přehrávat, indikováno ikonou u dané skladby. To je doprovázeno zobrazením přehrávací obrazovky, buď to statického obrázku pro obsah pouze s audiem (viz obrázek 1.10) nebo videa na pozadí (viz obrázek 1.12). Na rozdíl od video feedu se po chvíli přehrávání video nezastaví, ale aplikace automaticky plně přejde na obrazovku přehrávání (viz obrázky 1.11 a 1.13).

Obrazovky odpovídají doporučením ohledně obrazovky kolekce, protože obrazovky zobrazují podrobnosti o kolekci na levé straně obrazovky, a její skladby, na pravém panelu [18]. Seznam jasně zvýrazňuje momentální selekci a nepoužívá tlačítkové kontejnery a nejsou zde žádné problémy, co se týče doporučení ohledně seznamů [19].

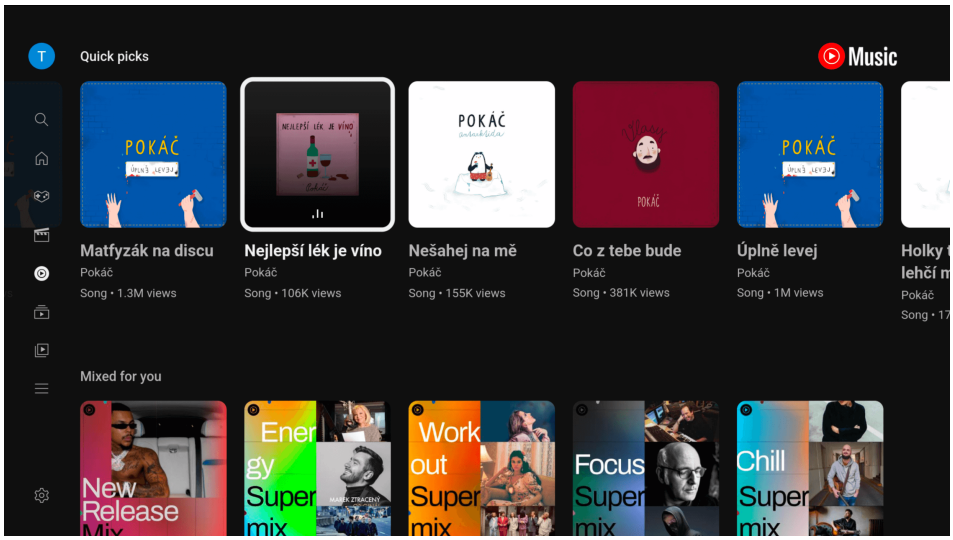
1.6.1.5 Přehrávání videa

Na přehrávací obrazovce (viz obrázek 1.14) jsou zobrazeny detaily přehrávaného videa, přepínání mezi předchozím či následujícím videem (pokud se videa nachází v kolekci), likování, dislikování, přidání do vlastního seznamu a nastavení přehrávání videa.

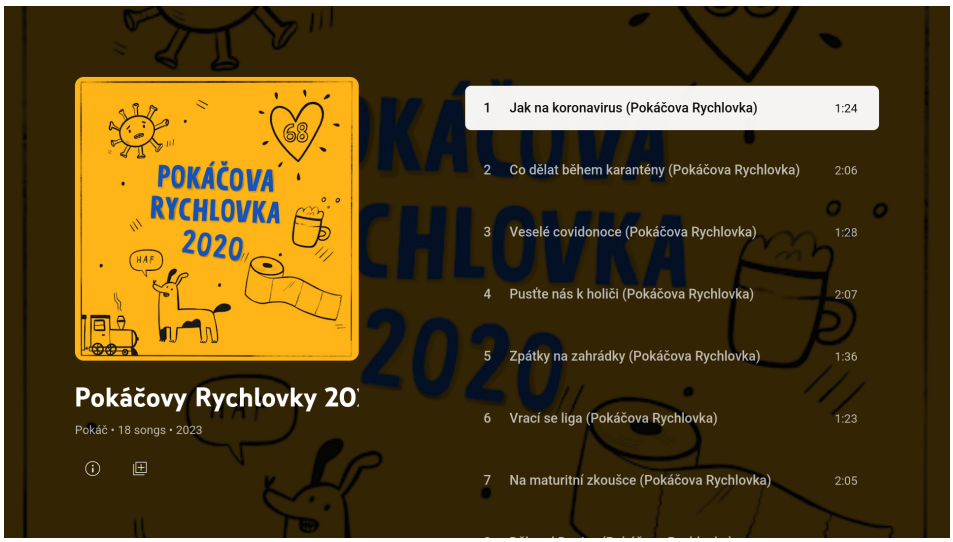
Při přesouvání se na časové ose videa je uživateli zobrazován náhled časového úseku pro lepší orientaci a světlý indikátor přehrávanosti dané části (viz obrázek 1.15).

Pokud jde o hudební video, lze přepnout mezi videem a statickým obrazem. Uživatel kromě informací a ovládání videa může přejít do nižší sekce, kde jsou uživateli zobrazovány videa nejen související s právě přehrávaným videem (viz obrázek 1.16).

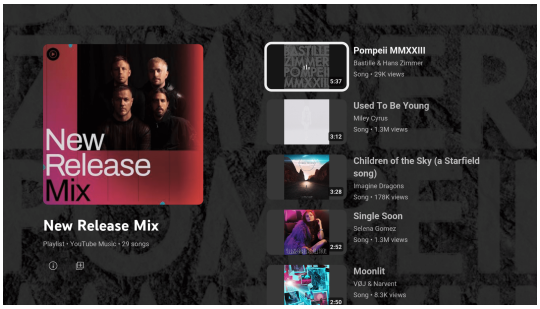
Pokud jde o živě streamované video, časová osa videa je doplněna o indikátor barevné tečky a textu “Live” (viz obrázek 1.17). Když je právě přehrávaná sekce živě, indikátor je červený, stejně jako osa.



Obrázek 1.8 Audio feed – YouTube

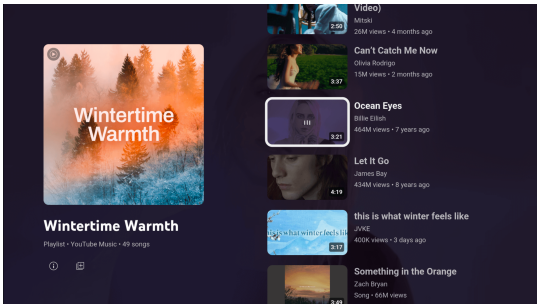


Obrázek 1.9 Album – YouTube



Obrázek 1.10 Playlist (obrázek v pozadí) – YouTube

Obrázek 1.11 Přehrávání audia (obrázek) – YouTube



Obrázek 1.12 Playlist (video v pozadí) – YouTube

Obrázek 1.13 Přehrávání audia (video) – YouTube

Když uživatel přehrává záznamovou část, indikátor je šedý. Časová osa při pohybu na ní nezobrazuje čas od začátku videa, nýbrž čas, jak je přehrávaná sekce posunutá od živého vysílání, která je doprovázena prefixem mínus, která indikuje čas v minulosti. Pokud se tedy uživatel přesune na čas 0:00, tak "Live" indikátor zčervená a uživatel se dívá živě.

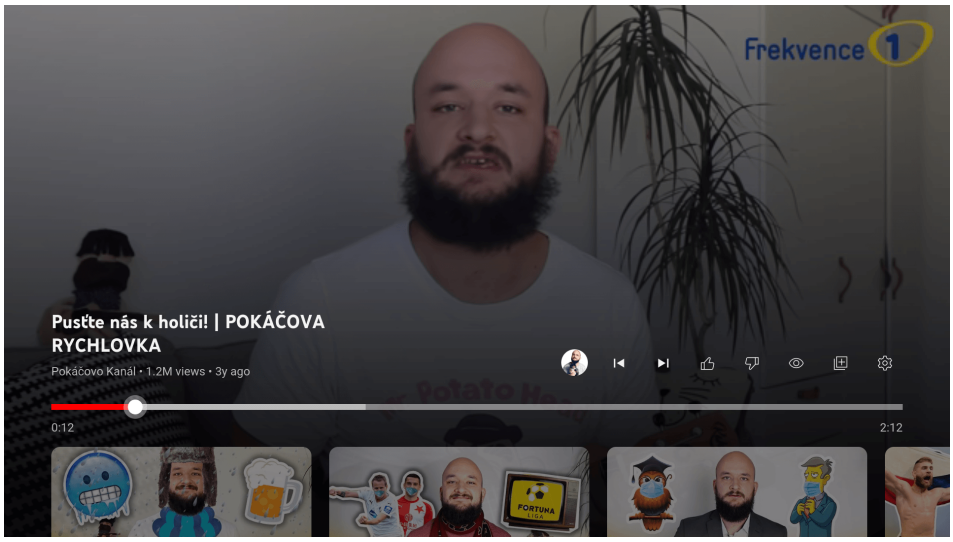
Obsah je v přehrávači členěn podobně jako na domovské stránce – do vertikálních sekcí, které lze horizontálně procházet. Informace o videu, tedy název, popis, sekce videa a podobné věci spolu s komentáři mají náhled v buňce nad časovou osou. Tato buňka může být zaměřena (viz obrázek 1.18) a lze na ni kliknout.

Po kliknutí se zobrazí na části obrazovky rozšířené informace se sekcemi (viz obrázek 1.19), na které lze dále klinout. Sekcemi jsou autor videa, popis a komentáře. Kliknutí na autora videa nabídne možnosti odběru a přechodu na kanál autora, kliknutí na popis zobrazí celý popis s možností vertikálního rolování (dále jen "scrollování") a kliknutí na komentáře zobrazí všechny komentáře.

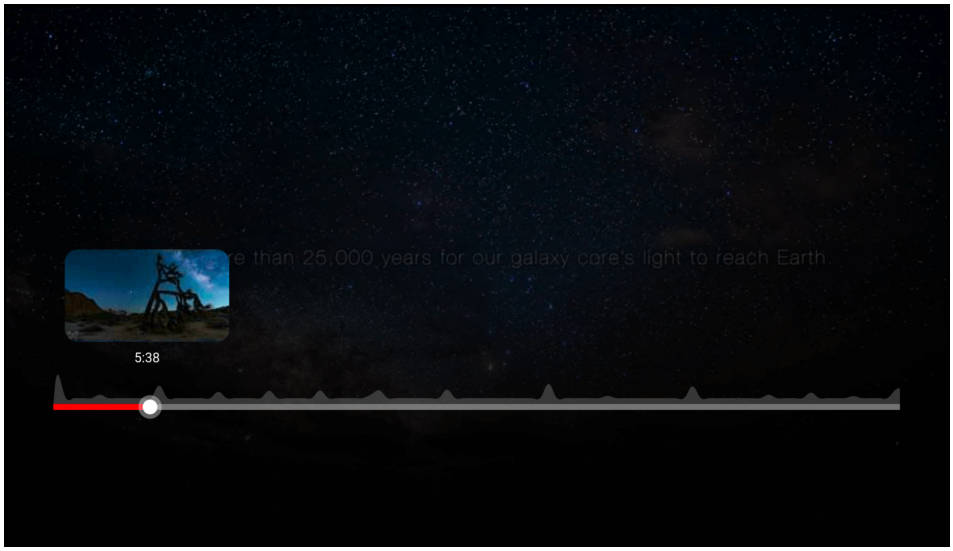
Komentáře lze likovat, dislikovat nebo nahlásit, ale uživatel samotný vlastní komentáře přidávat nemůže. Místo toho je zde karta "Pro přidání komentáře nebo odpovědi otevřete mobilní aplikaci YouTube.", kterou lze rozkliknout (viz obrázek 1.20). Zde nám ale aplikace pouze řekne, že si máme mobilní aplikaci otevřít a přihlásit se pod správným účtem, což není úplně užitečná informace. Užitečnější by bylo například zobrazit QR kód pomocí kterého by se uživateli na mobilním zařízení otevřela aplikace YouTube přímo v sekci, kde by uživatel chtěl přidat komentář, případně i hned zobrazit klávesnici. Je ale možné, že by tuto funkcionalitu nepoužilo dost lidí, aby to pro vývojáře bylo prioritou.

Informace v buňce s informacemi k videu nejsou pro uživatele vitální a tak nevadí je mít schované a dostupné až po několika kliknutích. Při přechodu na domovskou stránku se TV video přestane přehrávat, takže aplikace nepodléhá skupině požadavků spojenými se stálým přehráváním i mimo aplikaci.

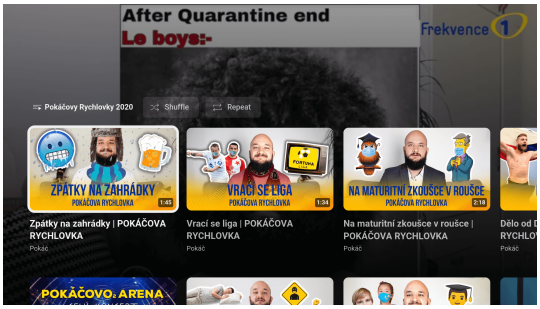
Jestliže se jedná o klipy a podobné krátkotrvající videa a ne o filmy nebo seriály, nepodléhá aplikace požadavkům ohledně kanálu Watch Next [20]. Při sledování videa o délce jedné hodiny a přerušení jeho přehrávání po desti minutách se video na domovské stránce TV přidalo do kanálu Watch Next (viz obrázek 1.21) a nepřidalo se, pokud byla délka sledování menší než deset minut.



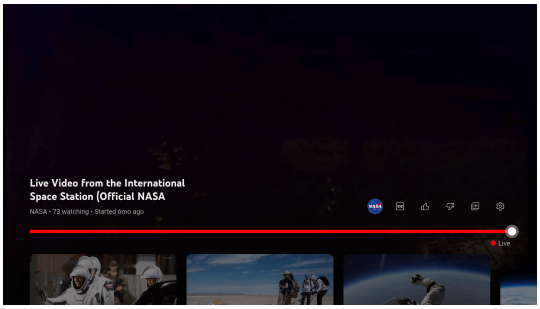
Obrázek 1.14 Video přehrávač – YouTube



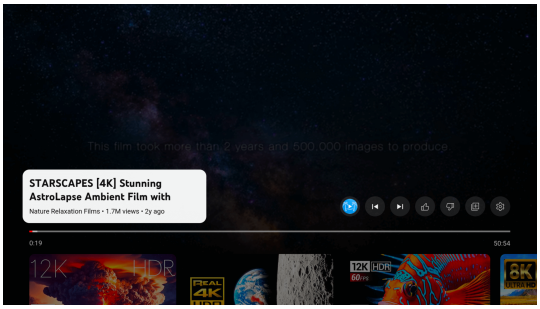
Obrázek 1.15 Přesouvání se na časové ose – YouTube



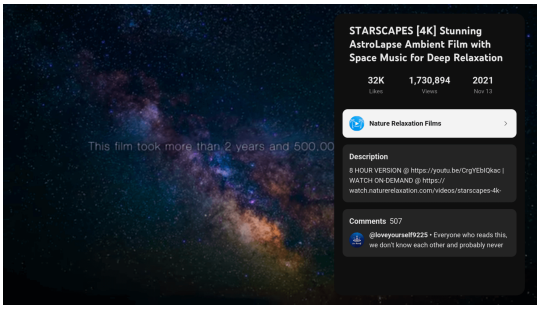
Obrázek 1.16 Přehrávačová sekce pod videem – YouTube



Obrázek 1.17 Živý přenos – YouTube



Obrázek 1.18 Informace o videu – YouTube



Obrázek 1.19 Rozšířené informace o videu – YouTube

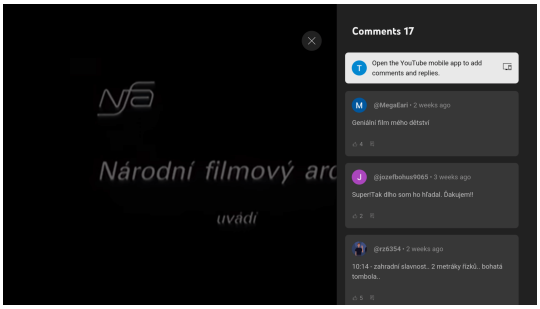
Aplikace podléhá požadavku TV-PC spojeným s ovládáním videa a ten kupodivu porušuje. “Při přehrávání videa nebo zvuku se stisknutím středového tlačítka D-padu přehrávané médium pozastaví.” [2]. Po stlačení středového tlačítka se zobrazí overlay s informacemi a ovládacími prvky, ale nezastaví se přehrávání videa. Důvod mi není jasný a myslím si, že je to nedostatek aplikace. Jedná se o požadavek a kvůli nesplnění požadavku by tak aplikace správně neměla být dostupná v TV obchodu Google Play.

Co se týče Ambient Mode u přehrávání videa, tak aplikace dané požadavky splňuje: TV-BU – zabraňuje přechodu do Ambient Mode; TV-BY – pokud je video pozastavené nebo není otevřený přehrávač, Ambient Mode aplikace dále neblokuje [2].

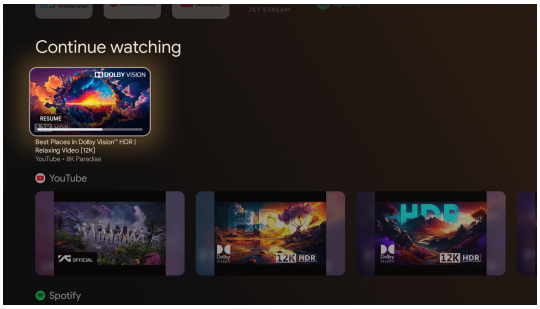
1.6.1.6 Přehrávání audia

Obrazovka je zde stejná jako u přehrávání videa, jediný rozdíl je, že se zde místo videa zobrazuje statický obrázek (viz obrázek 1.22). Ovládání přehrávání není úplně intuitivní, jak bych si ho představoval. Vzhledem k tomu, že aplikace poskytuje skladby jak s videem, tak bez, dodává toto rozhraní konzistenci v rámci aplikace, ale v obecném pojetí nikoliv. Ovládání je uzpůsobeno přehrávání videa a tak tu například chybí ovládací prvky jako přehrávání v náhodném pořadí, když se přehrává jakákoliv kolekce, nebo opakování přehrávání skladby či kolekce. Tyto ovládací prvky se v nějaké formě vyskytují na webu či v mobilních aplikacích YouTubu. Místo toho tu je třeba tlačítko nastavení, kde se nachází nastavení zobrazení titulek, které ale u obsahu pouze s audiem není dostupné. Je zde tak preferována konzistence UI v rámci aplikace a znovu-použití komponent, oproti UI na míru audio přehrávače.

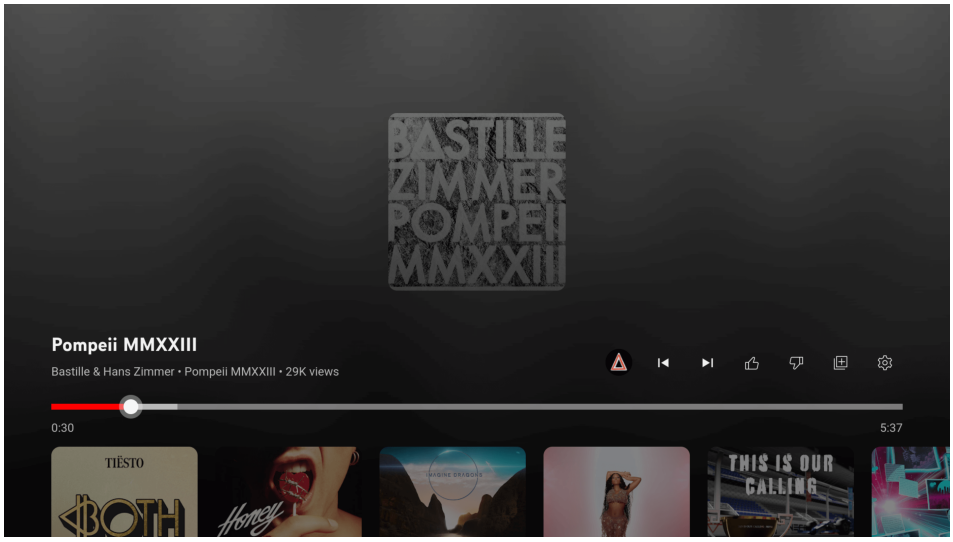
Diskutabilní je zde prevence přechodu do Ambient Mode. Podle TV-BA by přechodu do Ambient Mode v případě přehrávání audia se statickým vizuálním obsahem nemělo být zabraňováno [2], což se ale v případě YouTubu děje. Může to být způsobeno tím, že žádná ze skladeb v kolekci nebyla delší než 10 minut a tak se Ambient Mode neaktivoval. Zároveň zabránění Ambient Mode mohlo být způsobeno přehráváním reklam mezi skladbami.



Obrázek 1.20 Komentáře v přehrávači – YouTube



Obrázek 1.21 Přidání videa do Watch Next kanálu – YouTube



Obrázek 1.22 Audio přehrávač – YouTube

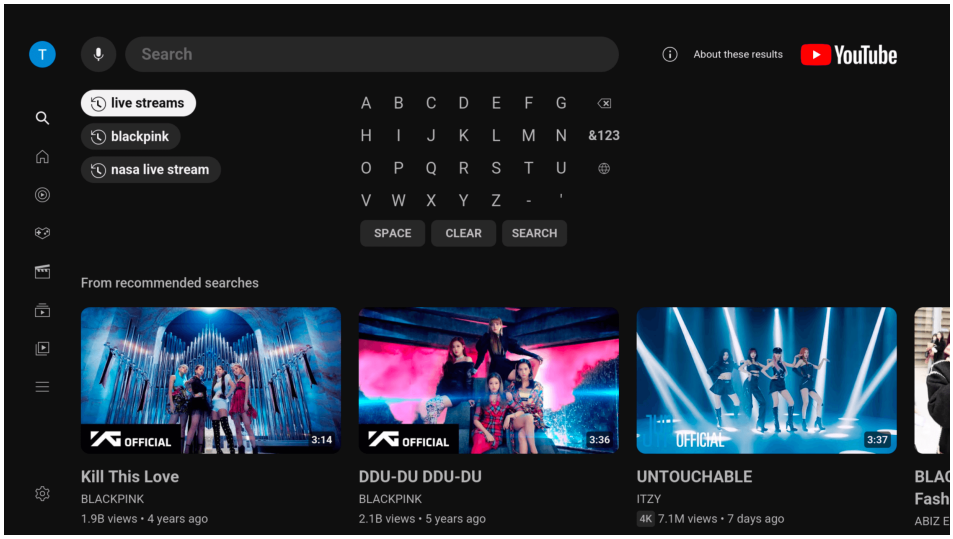
1.6.1.7 Vyhledávání

Pro vyhledávání je v aplikaci vyčleněna navigační destinace zařazená mezi ostatními destinacemi v navigačním panelu. Vyhledávací obrazovka (viz obrázek 1.23) obsahuje naposledy hledané fráze (mají ikonku hodin), vlastní klávesnici, tlačítko pro hlasový vstup a ve spodní části obrazovky doporučený obsah.

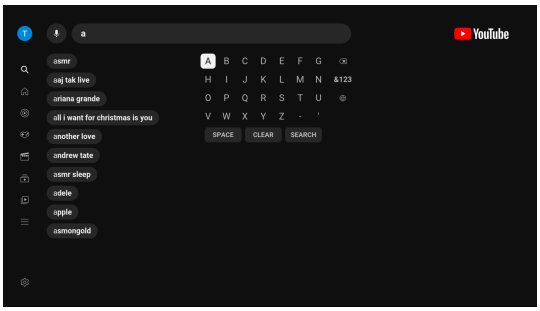
Při zadávání hledané fráze se na levé straně obrazovky, místo naposledy hledaných frází, průběžně zobrazují návrhy různých dokončených frází (viz obrázek 1.24), které může uživatel zvolit pro dokončení zadávání hledané fráze. Po dokončení vstupu hledání se zaměření přesune na první vyhledané video. Zobrazí se sekce, které jsou děleny podobně jako na domovské obrazovce aplikace, tedy do různých sekcí videí, které v tomto případě souvisí s vyhledávanou frází (viz obrázek 1.25). Vždy je zde sekce s "Výsledky vyhledávání pro …", někdy pak sekce pro hudbu, YouTube "Shorts", "Lidé také sledovali", "Pro tebe" nebo "Nedávno nahrané".

1.6.1.8 Synchronizace s ostatními zařízeními

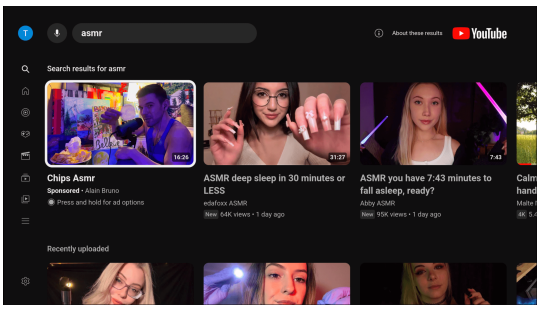
V YouTube aplikaci na žádné platformě nelze plynule přejít na jiné zařízení a pokračovat ve sledování. Neexistuje zde žádná sekce pro právě přehrávané video. YouTube může rozdívané video nabídnout v domovské sekci, jistotou to ale není. YouTube si u každého videa pamatuje, kde uživatel skončil a historie sledování je pro zařízení pod stejným účtem společná. Tyto informace se tedy, včetně TV apli-



Obrázek 1.23 Vyhledávání – YouTube



Obrázek 1.24 Vyhledávání (se vstupem) – YouTube



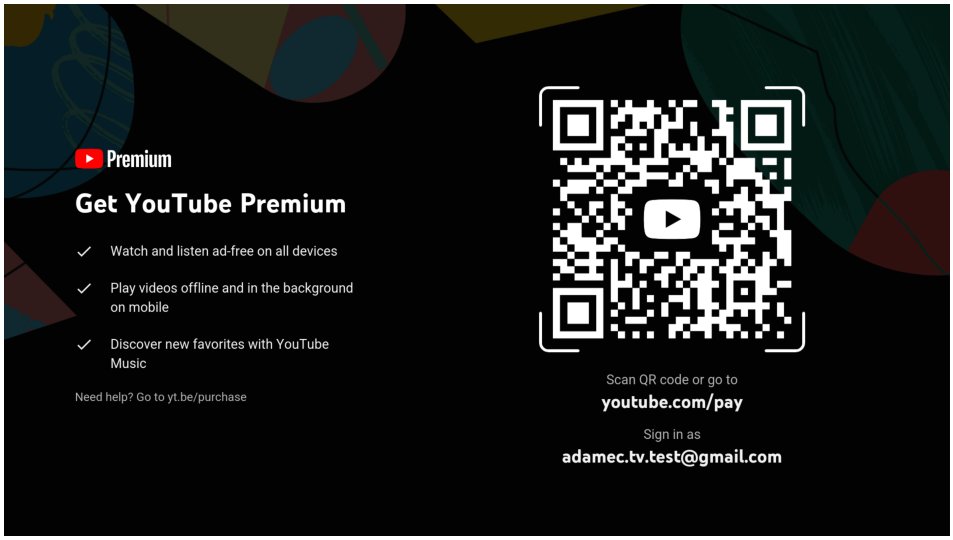
Obrázek 1.25 Výsledky vyhledávání – YouTube

kace, synchronizují, takže lze přes historii zvolit poslední video (či audio) a pokračovat v jeho sledování. Aktuální informace zobrazuje TV aplikace s několikasekundovým zpožděním oproti webové aplikaci, která synchronizované informace zobrazuje okamžitě. Na ostatních zařízeních se informace objeví až po manuálním znovunačtení historie. Na YouTube lze sledovat pod jedním účtem z více zařízení více videí najednou.

1.6.1.9 Nákup předplatného

V případě YouTube se dá zakoupit předplatné YouTube Premium. Pro zakoupení předplatného využívá TV aplikace delegace na jiné zařízení pro pohodlnější vyplňování údajů apod. Pro správné fungování zakoupení předplatného je nutné být na druhém zařízení nejprve přihlášený a až poté lze pokračovat. V opačném případě se na tomto zařízení zobrazí chyba.

Proces začíná zobrazením QR kódu a webové adresy youtube.com/pay jako alternativního zdroje pro zakoupení předplatného (viz obrázek 1.26). Při naskenování QR kódu telefonem se otevře YouTube aplikace, která nákup automaticky dále deleguje do webového rozhraní. Zde už lze vybrat platební metodu, kterou má už uživatel na telefonu uloženou nebo přidat novou a zakoupit předplatné. Při zadání webové adresy pro zaplacení se v prohlížeči dostáváme do stejné destinace, jako po naskenování QR kódu.



Obrázek 1.26 Nákup Premium – YouTube

1.6.2 Netflix

Netflix je známá a široce používaná služba pro sledování filmů a seriálů. Netflix se od YouTubu liší výhradně obsahem dlouhotrvajícího video obsahu – filmy. Netflix má, stejně jako YouTube, přes 100 milionů stažení na Google Play Store pro TV [21] a nabízí podobné funkcionality jako je hodnocení obsahu nebo vlastní seznamy. Struktura TV aplikací je mezi aplikacemi často velmi podobná a tak se při analýze Netflix aplikace budu soustředit na to, co dělá Netflix jinak.

1.6.2.1 Přihlášení

Netflix na rozdíl od YouTubu nelze používat, pokud uživatel není přihlášený. Proto je uživateli jako první ukázaná obrazovka s možnostmi “Přihlásit” a ”Začínáme” (viz obrázek 1.27), která provádí uživatele registrací. Naše aplikace ale možnost registrace potřebovat nebude a tak se podíváme jen na přihlášení.

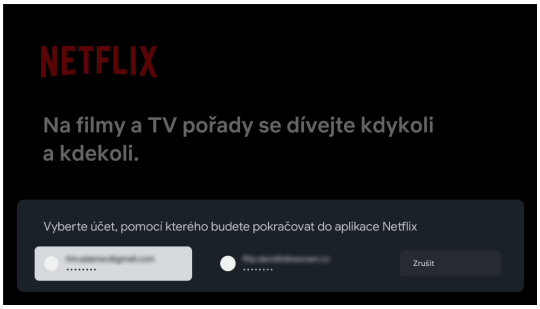
Zajímavostí v procesu přihlášení bylo, že po otevření aplikace se mi po chvíli zobrazila možnost pokračování pomocí jednoho z mých Netflix účtů (viz obrázek 1.28). Tato možnost přihlášení přišla ze systému a přihlašovací údaje převzala z Google účtu, kterým jsem byl přihlášený na TV. Po zvolení přihlašovacích údajů k mému Netflix účtu mě aplikace přenesla do volby sledovacího profilu a mohl jsem přejít na domovskou stránku. Tato možnost uživatelům, kteří se mohou přihlásit pomocí pouhého stlačení tlačítka, umožňuje zcela přeskočit proces přihlašování. Umožnění přihlášení pomocí této funkcionality v aplikaci Forendors bude zváženo na základě časových prostředků a priorit. Minimálně se ale může jednat o jedno z budoucích možných vylepšení.

Nyní ke klasickému přístupu přihlašování skrze aplikaci. Netflix nabízí přihlášení pomocí telefonu (viz obrázek 1.29) nebo pomocí dálkového ovládání (viz obrázek 1.30). Začneme přihlášením pomocí dálkového ovládání. Netflix nevyužívá přihlášení přes `WebView`, nýbrž využívá nativní obrazovky s vlastní klávesnicí k vyplnění e-mailové adresy a následně i hesla. Uživateli je zobrazena i možnost “Zapomněli jste heslo?” přes kterou uživatel zadá e-mail, na který mu přijde link k resetování hesla.

Při přihlášení pomocí telefonu lze naskenovat QR kód, pomocí kterého se na telefonu otevře aplikace Netflix, ve které se zobrazí kód (viz obrázek 1.31), který je zároveň zobrazený i na TV a uživatel provede potvrzení. Pokud uživatel aplikaci nemá, otevře se uživateli potvrzení v prohlížeči na webových stránkách Netflixu. Pokud je zde přihlášený, stačí pouze potvrdit předvyplněný kód (viz obrázek 1.32). Pokud přihlášený není, potvrdí kód a následně se pomocí svých přihlašovacích údajů v prohlížeči přihlásí.

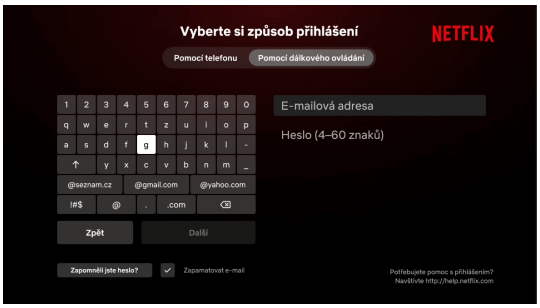
Možností je také ruční zadání webové adresy, kde musí uživatel ručně zadat kód. Poté přihlášení fun-

Obrázek 1.27 Uvítací obrazovka – Netflix



Obrázek 1.28 Automatické přihlášení – Netflix

Obrázek 1.29 Přihlášení pomocí telefonu – Netflix



Obrázek 1.30 Přihlášení pomocí dálkového ovládání – Netflix

guje jako při naskenování QR kódu. To znamená, že QR kód nese informace o číselném kódu zobrazeném na TV a jak mobilní aplikace, tak webová aplikace umí na takový vstup reagovat. Webová aplikace umí potvrzení delegovat na mobilní aplikaci, pokud je na telefonu nainstalovaná. Jakmile uživatel kód kdekoliv potvrdí, TV přijme signál o potvrzení přihlášení, přihlásí uživatele a automaticky přechází k volbě profilu pro sledování obsahu (viz obrázek 1.33).

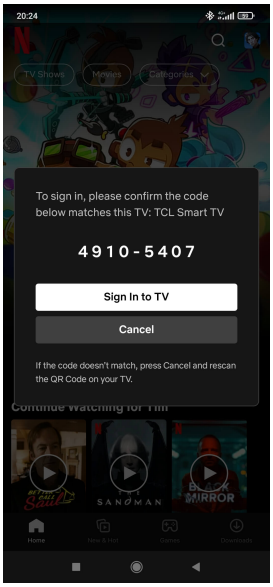
Při zobrazení postupu pro přihlášení pomocí telefonu jsem byl zprvu zmatený, protože jsem si dva kroky (1. na levé straně obrazovky a 2. na pravé) vyložil jako dva způsoby přihlášení a to také proto, protože titulek obrazovky zní “Vyberte si způsob přihlášení”. Druhý krok jsem si vyložil tak, že mi automaticky (tj. bez skenování QR kódu) přijde oznámení s kódem na telefon, který jen potvrdím. To samozřejmě nedává smysl, protože TV aplikace neví, na která zařízení má TV požadavek pro potvrzení odeslat, protože jsem na TV nezadal žádné přihlašovací údaje. Je to nicméně subjektivně matoucí a preferoval bych slovní značení jako “Nejprve” a “Poté” či jiné grafické znázornění, nežli “1” a “2”.

1.6.2.2 Navigace

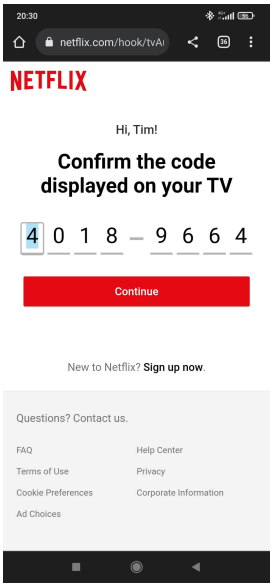
V případě, že má uživatel předplatné, které zahrnuje více jak jednoho člověka, nechává aplikace uživatele nejdříve vybrat, se kterým profilem chce do aplikace pokračovat. Toho by šlo využít i pokud je více klasicky přihlášených uživatelů, jako je tomu např. v YouTube aplikaci.

V aplikaci YouTube, pokud jste jiný uživatel, než který je právě přihlášen a chcete se dívat pod svým účtem, musíte přejít do sekce s účtem, kde kliknete na “Změnit účet” a aplikace pak chvíli pracuje na přepnutí účtu. Pokud by uživatelé YouTube nechtěli sledovací profil měnit, nový profil nepřidávají. Pokud ho ovšem přidají, chtějí pravděpodobně svoje sledovací aktivity separovat k vlastním profilům a pro tento účel je optimalizován jen Netflix.

Dalo by se tvrdit, že YouTube nezvažuje doporučení Googlu ohledně toho, že “televizor je společné zařízení” – „Televizor je obvykle společným zařízením v domácnosti. To znamená, že při navrhování aplikací pro televizi je důležité brát ohled na soukromí. Například aplikace, které zobrazují osobní in-



Obrázek 1.31 Přihlášení přes telefon (aplikace) – Netflix



Obrázek 1.32 Přihlášení přes telefon (prohlížeč) – Netflix

Obrázek 1.33 Volba profilu – Netflix

formace, by měly mít nastavení ochrany soukromí umožňující příslušné úpravy.” [22]. Jelikož YouTube přímo zobrazuje doporučená videa a podobné informace na první stránce, kterou vidí i ostatní uživatelé, kteří se chtějí dívat pod vlastním profilem, tak toto doporučení není z mého pohledu plně respektováno. Je to tak také otázka pohodlí uživatelů, kteří TV používají, aby měli možnost rovnou konzumovat obsah pod vlastním profilem.

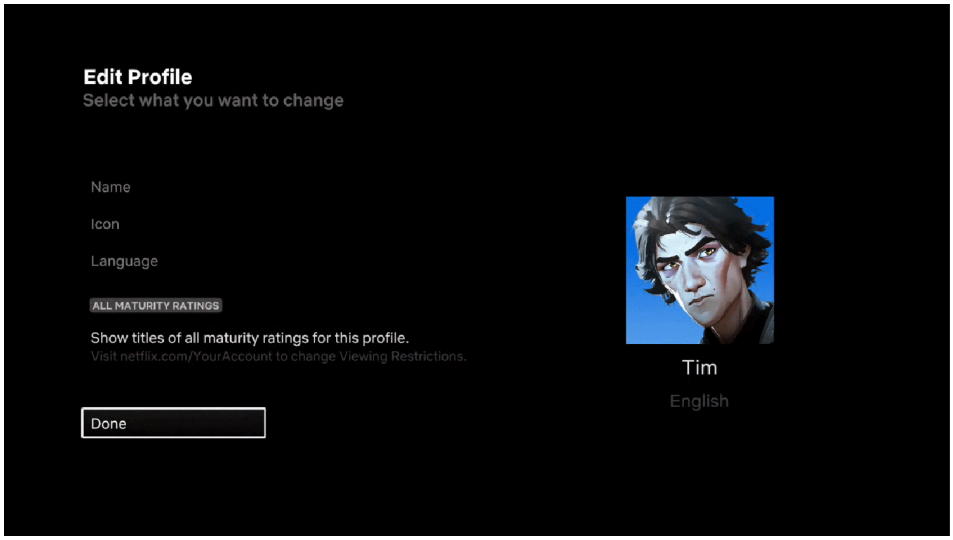
Netflix dále na obrazovce volby sledovacího profilu umožňuje u každého profilu přejít do úpravy daného profilu, kde uživatel může pro daný profil změnit jméno, ikonu a jazyk profilu (vit obrázek 1.34). Nachází se zde informace, že další úpravy jsou možné pouze na webové aplikaci.

Netflix na rozdíl od YouTube používá modální navigační panel (při otevření se vykresluje přes obsah) místo klasického (při otevření odsune obsah) (viz obrázek1.35). Hlavních destinací je v navigačním panelu, stejně jako v YouTube aplikaci, více než 6. Hlavním cílem omezení počtu destinací ze strany Googlu je dle mého snaha o vyhnutí se scrollování, tedy aby se navigační panel nechoval jako seznam, s čímž Netflix ani YouTubu problém nemají. Doporučení ohledně počtu primárních destinací nicméně obě aplikace porušují.

Rozložení prvků v navigačním panelu je velmi podobné YouTube rozvržení. Netflix ale využívá na spodu obrazovky i horizontálního místa pro tlačítka “Pomoc” a “Odejít z Netflixu”. Jsou to řídce používaná tlačítka a tak dává smysl nezahrnovat je do vizuálního pole uživatele. Dost možná je to také proto, protože Netflix obsahuje 7 hlavních destinací a přidání ostatních destinací by uživatele zahltilo a separace hlavních navigačních destinací od ostatních by pravděpodobně nebyla tak zřejmá.

Netflix porušuje doporučení, co se týče používání tlačítka “Zpět”. V doporučeních se píše: „Abyste vytvořili předvídatelný navigační zážitek, přeneste uživatele po stisknutí tlačítka ”Zpět”na dálkovém ovladači do předchozího cíle. Nakonec by měl uživatel přistát na domovské obrazovce nebo spouštěči Google TV, pokud bude stále mačkat tlačítko “Zpět”. Zajistěte, aby akci při stlačení tlačítka “Zpět” nebylo bráněno potvrzovacími obrazovkami nebo nekonečnou smyčkou.” [17].

V případě Netflixu se uživatel opakovaným stisknutím tlačítka “Zpět” nedostane z aplikace pryč a zároveň je součástí nekonečné smyčky mezi otevíráním a zavíráním navigačního panelu. Porušuje tím požadavek TV-DB – “Opakované stisknutí tlačítka ”Zpět”vede zpět na domovskou obrazovku Android TV.” [2]. V průvodci Googlu pro tvorbu TV aplikací je psáno: „Tlačítko “Zpět” nesmí nikdy fungovat jako přepínač. Nepoužívejte jej například zároveň k otevření a zavření nabídky.” [23]. Je zajímavé, že



Obrázek 1.34 Úprava profilu – Netflix

takové chování v aplikaci stále přetrvává.

1.6.2.3 Video feed

Uživateli je jako první přes většinu obrazovky ukázano jedno doporučení od Netflixu, které lze ihned spustit (viz obrázek 1.36). Pod tímto doporučením už se nachází sekce se seznamy.

Místo klasických seznamů využívá Netflix "pohlcujících seznamů", které uživateli u právě zvoleného seriálu nebo filmu na části obrazovky zobrazují statický náhled či náhledové video, logo filmu, technické informace k filmu, jako jestli je dostupný ve 4K, krátký popisek a zajímavosti k filmu (viz obrázek 1.37). Zde Netflix splňuje všechny doporučení týkající se pohlcujících seznamů [24].

Navigace tlačítkem "Zpět" ze seznamu do navigačního panelu zde funguje stejně jako v YouTube aplikaci, tedy že při stlačení tlačítka "Zpět" se uživatel přesouvá rovnou do navigačního panelu místo na první film v seznamu. To je v rozporu s doporučeními a je otázka, jestli se řídit doporučeními nebo momentální implementací hlavních streamovacích služeb, na které jsou uživatelé zvyklí.

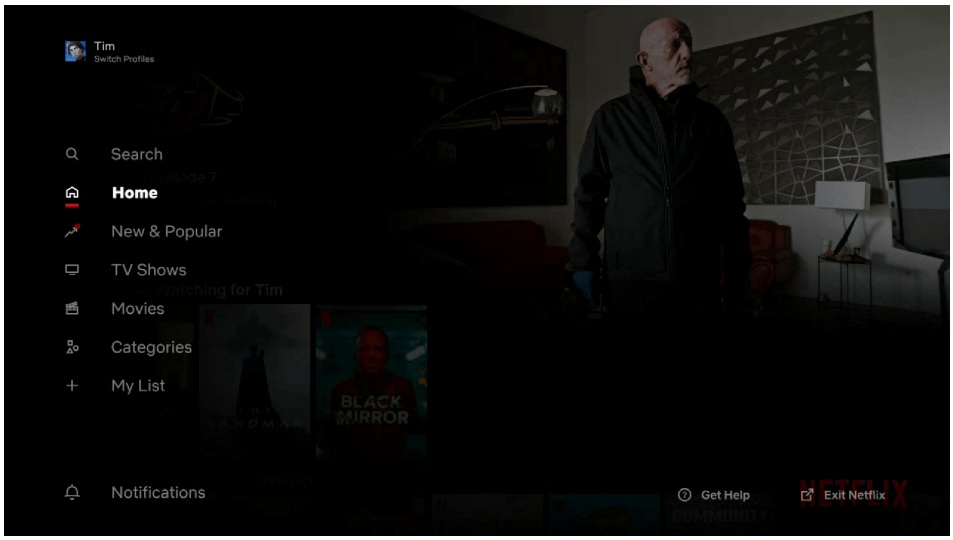
První sekcí je vždy sekce s názvem "Pokračovat ve sledování", kde se zobrazují jak rozkoukané filmy a seriálové epizody (viz obrázek 1.38), tak navazující epizody, když uživatel viděl celou předešlou epizodu. Každý film a seriál lze rozkliknout. Uživatel se pak dostává do obrazovky s rozšířenými informacemi a možnostmi jako hodnocení obsahu, výběr epizody, přidání do seznamu apod. (viz obrázek 1.39).

Pod ním se, podobně jako při přehrávání videa v aplikaci YouTube, zobrazuje obsah relevantní k tomu právě zvolenému. Zobrazuje se ale pouze, když je obsah z nějaké kolekce. V tom případě se zde ukáže zbylý obsah z dané kolekce. Pokud není obsah z žádné kolekce, sekce se nezobrazí.

Místo doporučení relevantního obsahu zobrazovaného ve spodní části obrazovky se u některých filmů zobrazuje tlačítko "Více jako toto" (viz obrázek 1.40). Ten přenese uživatele do obrazovky připomínající kolekci (viz obrázek 1.41). Obrazovka je rozdělena na dvě poloviny. Na první polovině je daný film a neaktivní tlačítko "Více jako toto". Na druhé polovině je pak vertikální seznam s kartami filmů, na jejichž detail lze přejít kliknutím.

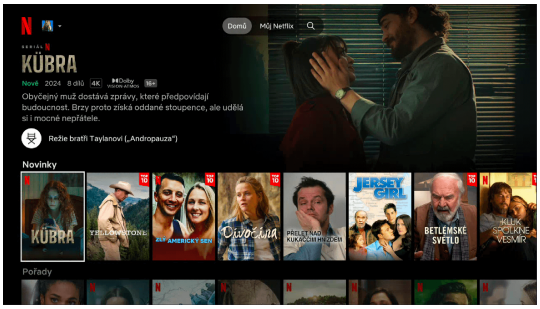
1.6.2.4 Přehrávání videa

Obrazovka přehrávání videa na spodní půlce obrazovky obsahuje zkratky pro volbu jazyka a titulků, které uživatel používá (viz obrázek 1.42). Je tu zde i tlačítko pozastavit/spustit, které na YouTube chybí. Je zde



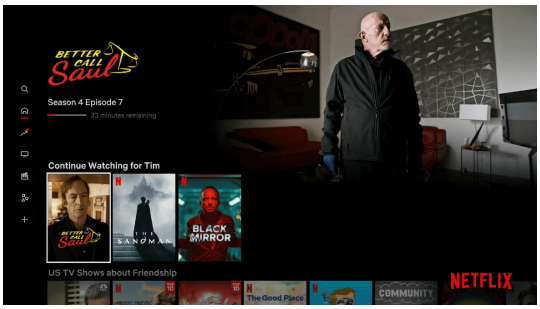
Obrázek 1.35 Navigační panel – Netflix

Obrázek 1.36 Hlavní doporučený obsah – Netflix

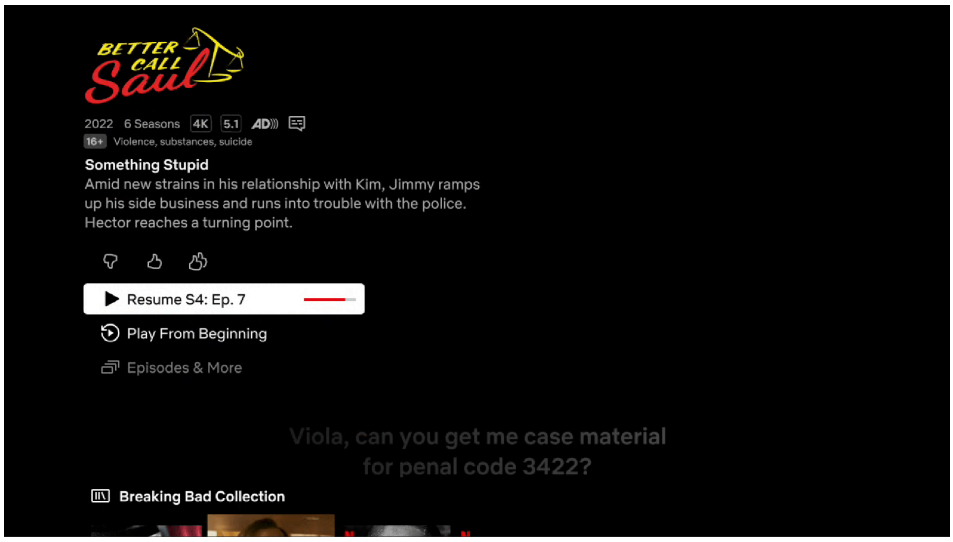


pozn.: Netflix prošel aktualizací a tak rozložení navigace není tím, které bylo analyzované.

Obrázek 1.37 Feed – Netflix



Obrázek 1.38 Feed (rozdívaný seriál) – Netflix

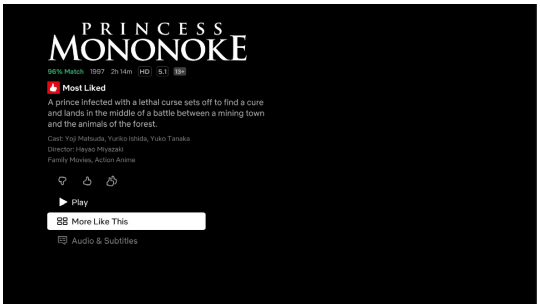


Obrázek 1.39 Detail obsahu – Netflix

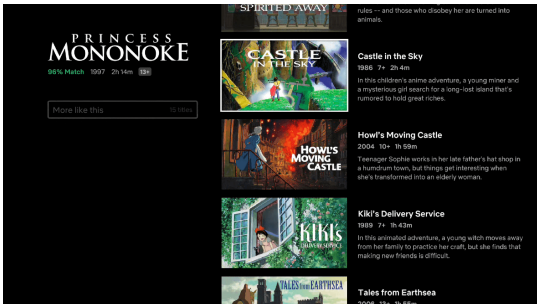
využito i horního prostoru obrazovky, kde vlevo se nachází navigační prvky nazvané “Možnosti”. Jsou to tlačítka “Zpět”, “Přehrát od začátku” a “Další epizoda”. Tlačítko “Zpět”by ale, podle doporučení, nikdy na obrazovce nemělo být zobrazováno, protože uživatel má tlačítko “Zpět” na ovladači [25].” Na pravé straně je pak název seriálu, číslo série, číslo epizody a název epizody.

Netflixové rozhraní pro přehrávání videa na první dojem působí jako lépe optimalizované pro použití na TV než to YouTubové. Nejspíše je to faktem, že se prvky jako hodnocení obsahu, přidat do seznamu a podobný obsah objevují na mezi-obrazovce s detaily o obsahu a ne v přehrávači. Subjektivně je to správný krok pro design UI na TV, protože uživatel nesmí být zahlcen možnostmi a informacemi. Tento přístup ale na druhou stranu přidává další nutný krok, když si chce uživatel nějaký obsah spustit. V případě Netflixu je ke každé položce dost možností a informací pro obhájení existence mezi-obrazovky. Dalším subjektivním důvodem je, že tlačítka v YouTube přehrávači jsou malá a splývají dohromady a všechny prvky včetně těchto tlačítek jsou na spodu obrazovky. Netflix přehrávač tak působí více přehledně, nežli ten YouTubový.

Co se týče požadavků, tak Netflix požadavkům pro Ambient Mode vyhovuje. Zajímavostí Netflixu pro TV je jeho vlastní kvazi-Ambient Mode. Ten se sice nespouští, pokud je pozastavené video, v takovém případě se spouští klasický Ambient Mode, ale v jakékoliv jiné části aplikace, pokud je uživatel nějakou dobu nečinný, spustí Netflix vlastní Ambient Mode, kde prezentuje slideshow filmů a seriálů. Tento vlastní Ambient Mode zabraňuje spuštění klasického Ambient Mode a vlastní Ambient Mode přetrvává dokud uživatel nevykoná nějakou akci. Toto chování s vlastním Ambient Mode není vyslo-



Obrázek 1.40 "Více jako toto"– Netflix



Obrázek 1.41 Kolekce podobného obsahu – Netflix

Obrázek 1.42 Přehrávání videa – Netflix

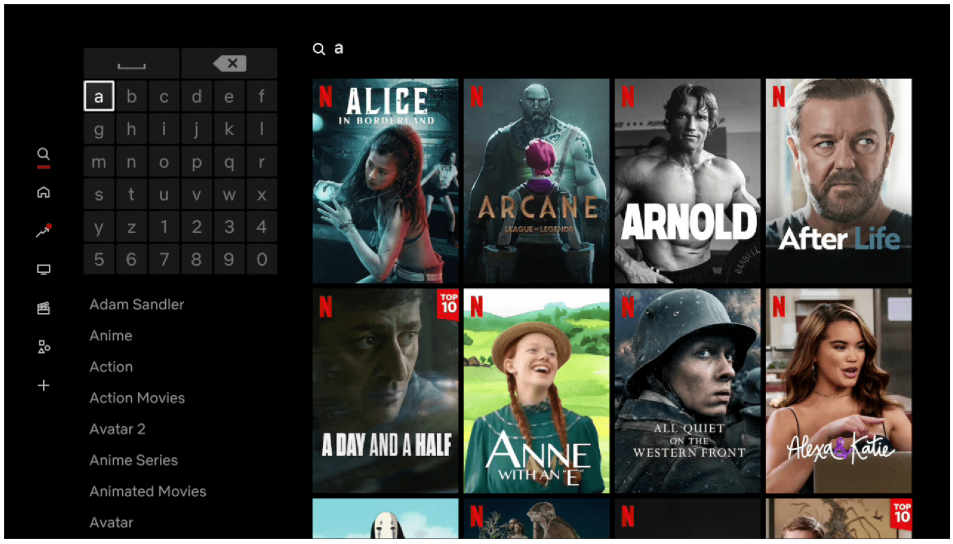
veně zakázáno, ani povoleno. Zmínka o vlastním spořiči obrazovky je v sekci s přehráváním audia: „V případě přehrávání zvuku by aplikace neměly bránit režimu Ambient Mode během přehrávání, pokud neimplementují vlastní spořič obrazovky s nestatickým obrazem." [7].

1.6.2.5 Vyhledávání

Sekce vyhledávání je umístěná stejně, jako v případě YouTubu, tedy jako první hlavní navigační sekce v navigačním panelu. Obrazovka se dělí, stejně jako YouTube, na tři sekce: aplikační navigace, návrhy dokončených frází a výsledky hledání (viz obrázek 1.43). Chybí zde naposledy hledané fráze, což ale není nutně nedostatkem. Rozpoložení je rozdílné. Netflix využívá celé šířky obrazovky.

Na levé straně je klávesnice s návrhy frází a na pravé výsledky hledání. Při zadávání se automaticky obnovují navrhované dokončené fráze a zároveň se průběžně obnovují i výsledky hledání, umožňující uživateli rovnou zvolit film/seriál dřív, než dokončí vpis hledané fráze. U návrhů frází, kde většinou nejsou návrhy názvů filmů či seriálů (protože ty se zobrazují ve výsledcích), se zobrazuje dokončení frází kategorických, jako hledání obsahu s daným hercem, typ filmů/seriálů jako komedie apod. nebo technologie, které obsah podporuje, jako např. 4K.

I když se u vyhledávání oproti YouTubu využívá celé plochy obrazovky, nepůsobí obsah obrazovky o nic více rušivě nebo nepřehledně. Výsledky hledání jsou uspořádané do mřížky a při zaměření jednoho z výsledků se nezobrazuje dodatečné info. Při zvolení výsledku se uživatel přesouvá na stejnou obrazovku, jako při zvolení obsahu z domovské stránky.



Obrázek 1.43 Vyhledávání – Netflix

1.6.2.6 Synchronizace s ostatními zařízeními

Synchronizace probíhá s velkým zpožděním a to jak odesílání informací, tak jejich stahování. Nelze tak využít plynulého přechodu mezi zařízeními. Pouze s časovým odstupem. Rozkoukané filmy a seriály jsou na domovské stránce aplikace zobrazovány ve vlastní sekci. Rozpoložení prvků je na všech platformách velmi podobné a tak je přístup k filmům a seriálům uživatele jednoduchý a rychlý.

1.6.2.7 Nákup přeplatného

Nákup předplatného je zde řešen v rámci registrace, kde uživatel na TV zadá e-mail, na který mu přijde zpráva. Registraci uživatel dokončí na zařízení, kde zprávu otevřel. V rámci dokončení registrace musí uživatel zvolit i předplatné.

1.6.3 Spotify

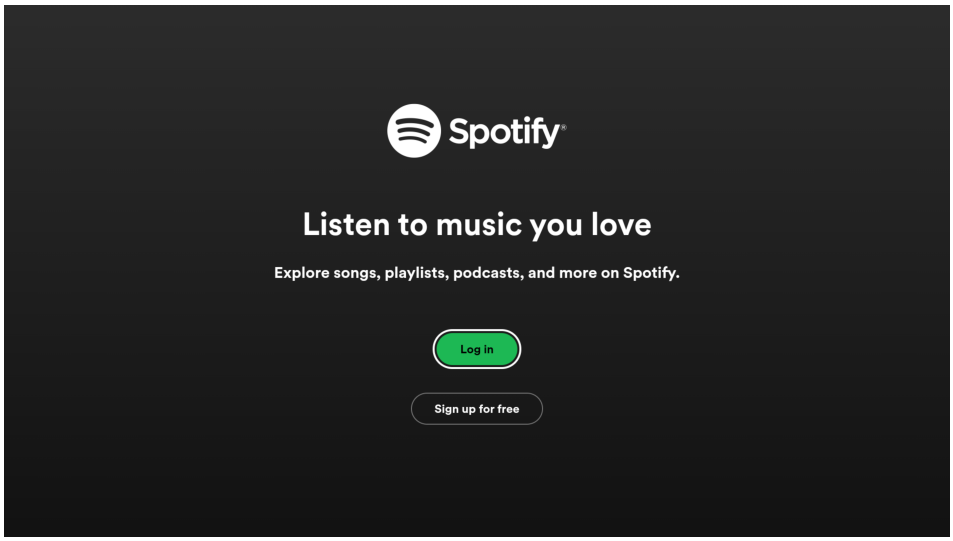
Spotify je asi nejpopulárnější službou pro poslech hudby a podcastů. Spotify se soustředí na audio obsah i v podobě podcastů a může tak nabídnout jiná řešení. I Spotify je na TV velmi stahovaná s 50+ miliony stažení na Google Play Store pro TV [26]. Zde se také budeme soustředit na odlišnosti od dvou předešle analyzovaných aplikací.

1.6.3.1 Přihlášení

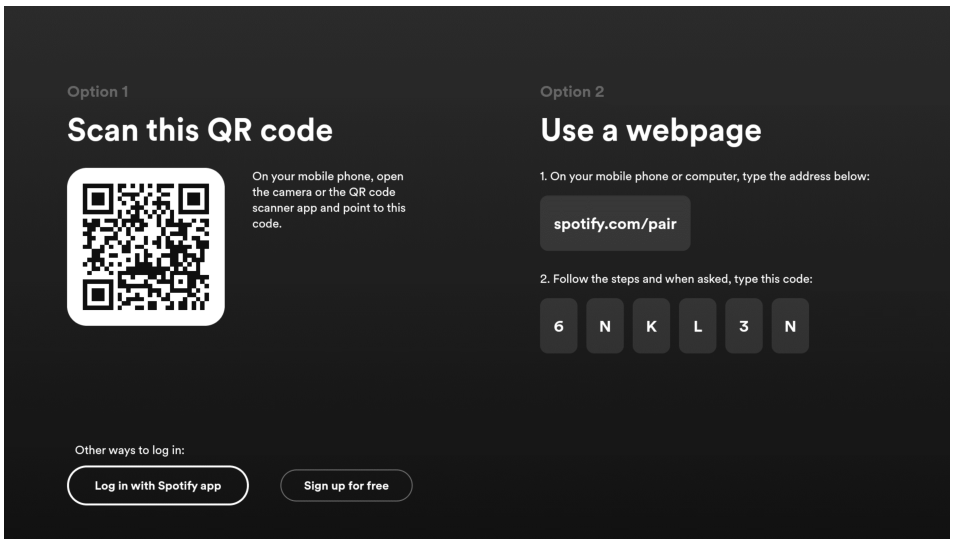
Spotify lze používat bez předplatného, je zde tak možnost registrace zdarma a možnost přihlášení (viz obrázek 1.44). Při registraci není uživatel přesměrován na jinou platformu, ani `WebView`, ale vyplňuje údaje spojené s registrací pomocí ovladače.

Pokud uživatel již účet má, může se přihlásit přes kód tak, jak to dělá Netflix. Spotify narozdíl od Netflixu poměrně zřetelně odděluje způsob skenováním QR kódu od ručního zadání URL a kódu do prohlížeče. Při přihlášení přes QR kód neukazuje číselný kód, protože to není potřeba (viz obrázek 1.45). Uživatel v případě skenování QR kódu nemusí o žádném kódu nic vědět, jen to pro uživatele komplikuje přihlašovací proces.

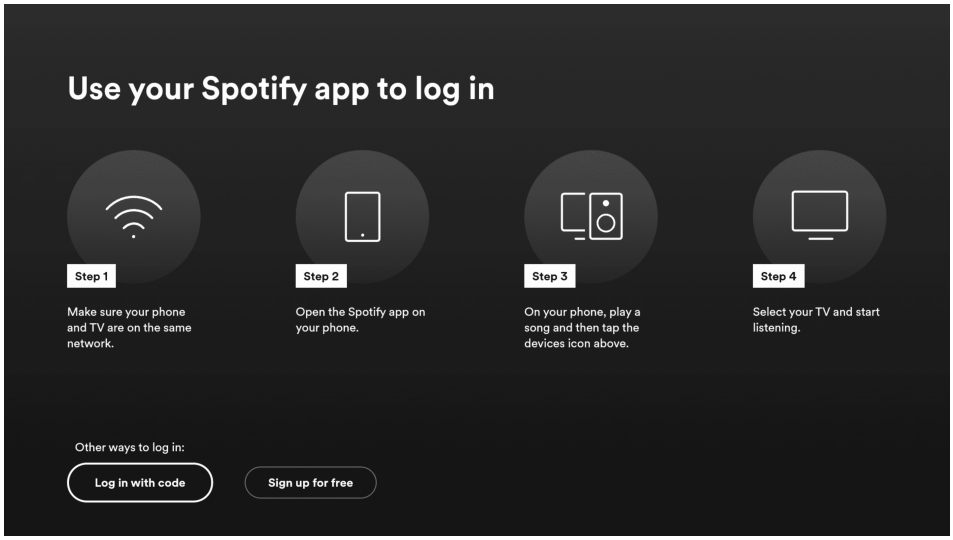
Také je vhodné zmínit, že proces v mobilní aplikaci po naskenování QR kódu neprobíhá v nativním rozhraní, jako je tomu u Netflixu, ale uvnitř aplikace se zobrazí `WebView`, kde se postupuje dále. To není



Obrázek 1.44 Uvítací obrazovka – Spotify



Obrázek 1.45 Přihlášení přes kód – Spotify



Obrázek 1.46 Přihlášení přes mobilní aplikaci – Spotify

optimální řešení, zato se ale obrazovka musí vytvořit jen jednou a to pro web. V mobilní aplikaci stačí jen zobrazit tuto webovou obrazovku.

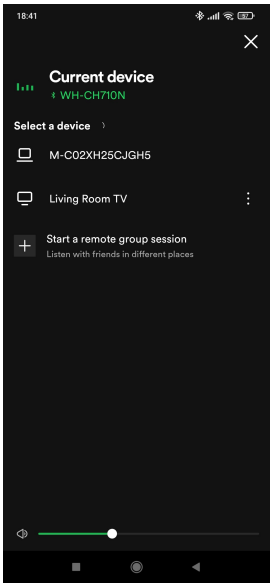
Zajímavější je zde primární možnost přihlášení přes aplikaci, kde aplikace využívá toho, že jsou zařízení na stejné síti a tak mobilní aplikace dokáže TV zařízení najít (viz o brázek 1.46). Při přehrávání audia je v aplikacích na všech platformách zobrazeno, na kterém zařízení se audio zrovna přehrává. Pokud si ikonu zařízení uživatel rozklikne, jsou mu ukázána zařízení, která jsou dostupná k přehrávání. Jsou to zařízení, kde je uživatel přihlášený a které jsou online. V tomto seznamu při přihlášení přibývá i právě TV Spotify aplikace, pomocí které, když na ni uživatel klikne, se v TV aplikaci přihlásí (viz obrázek 1.47). Pokud se na telefonu přehrává skladba či podcast, TV po výběru zařízení začne přehrávat stejnou skladbu (nebo podcast) tam, kde uživatel poslouchal naposledy (viz obrázky 1.47 a 1.48).

1.6.3.2 Navigace

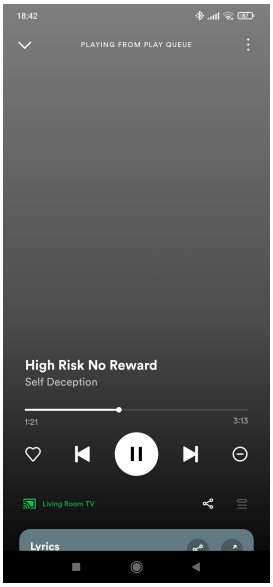
Spotify narozdíl od YouTubu a Netflixu nemá postranní navigační panel, ale využívá horní navigační lišty s taby (viz obrázek 1.50). Při testování navigace pomocí tlačítka ”Zpět”po příchodu do aplikace se uživatel dostane do navigační lišty a po dalším stlačení tlačítka “Zpět” by se správně měl uživatel dostat na domovskou obrazovku TV. Místo toho ale Spotify zobrazuje potvrzovací dialogové okno pro potvrzení odchodu z aplikace. Aplikace tak podobně jako Netflix zabraňuje uživateli odejít z aplikace a přímo tak porušuje doporučení spojené s používáním tlačítka “Zpět” [17].

Navíc, při stlačení tlačítka “Zpět”, když je zobrazené potvrzovací okno, se uživatel vrací na domovskou obrazovku aplikace místo domovské obrazovky TV, což, jak bylo zmíněno u Netflixu, je samozřejmě stejně špatně. Porušuje se tím požadavek TV-DB [2].

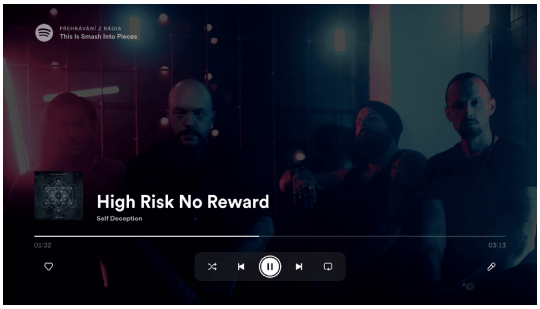
Horní lišta zobrazuje zkratku pro právě přehrávaný obsah, kam lze přejít, klasické navigační destinace, na pravé straně tlačítko pro upgrade na Spotify Premium, nastavení a profil. Možnost přepnutí profilu se nachází v desitnaci s profilem. Obrazovka je zde podobná YouTube přihlašovací obrazovce (viz obrázek 1.51). Na screenshotu lze vidět, že pokud je uživatel přihlášený pod svým účtem, který je asociovaný s nějakým jazykem, aplikace se automaticky lokalizuje. Dá se tu vytvořit nový Spotify účet, přidat již existující nebo přidaný účet odebrat. Obrazovky přihlášení a registrace jsou stejné, jako při prvotním spuštění aplikace. Při dvou a více přidaných účtech se aplikace při spuštění aplikace neptá, kdo má být momentálně přihlášený, ale rovnou vstupuje na domovskou obrazovku pod naposledy přihlášeným účtem. Jak jsem již zmiňoval, není to z mého pohledu vhodný přístup ke sdílenému zařízení.



Obrázek 1.47 Dostupná zařízení pro přehrávání – Spotify



Obrázek 1.48 Přehrávání po přihlášení zvolením TV v mobilní aplikaci – Spotify



Obrázek 1.49 Přehrávání po přihlášení zvolením TV v mobilní aplikaci – Spotify

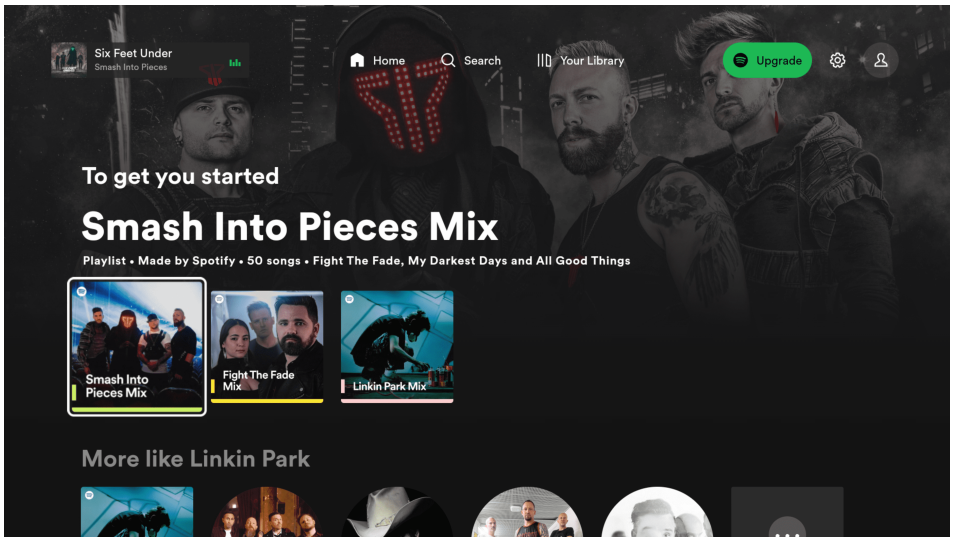
1.6.3.3 Audio feed

Na domovské obrazovce se zobrazují tři typy prvků: umělci – kulaté prvky, playlisty – hranaté prvky a podcasty – také hranaté. Rozdílným chováním u feedu na domovské obrazovce aplikace jsou seznamy o fixním počtu prvků, kde všechny jsou viditelné a poslední prvek je s textem "Více" (viz obrázek 1.52). Procházení horizontálních seznamů se tak nechová jako v případě YouTubu a Netflixu, kde se drží pivot pro aktuálně zaměřený prvek, ale seznam se prochází bez pivota a seznam není pohyblivý.

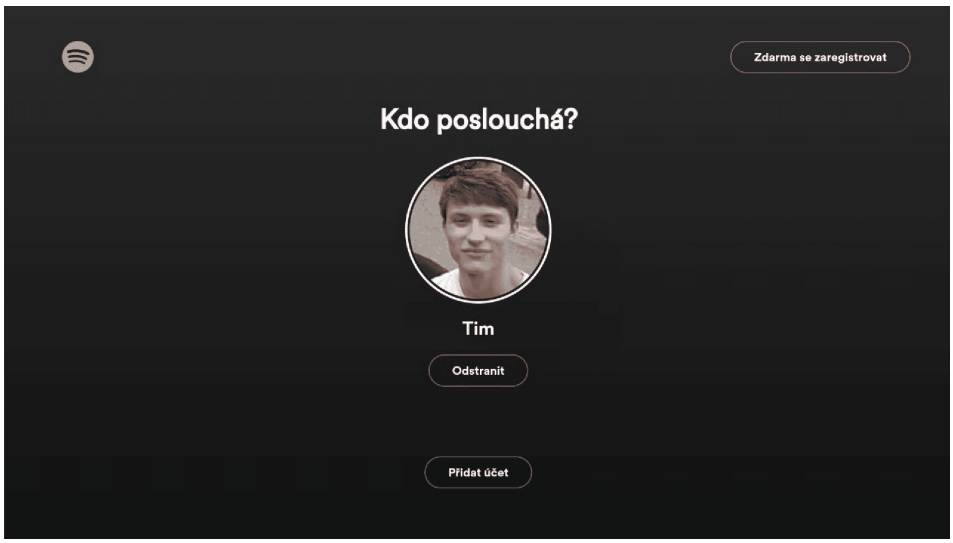
Při rozkliknutí posledního prvku "Více" se místo jednoho řádku s obsahem zobrazí další řádky s obsahem, tedy mřížka (viz obrázek 1.53). Mřížka ale stále funguje na principu "pohlcujícího seznamu", který využívá i Netflix. Při zaměření prvku se tedy stále na horní polovině zobrazují informace a statický obrázek spojený se zvoleným prvkem. Pokud k prvku není dostupný obrázek na pozadí, pak je na pozadí šedá barva. Tato mřížka už funguje jako vlastní obrazovka a nedá se tak vertikálně přecházet mezi sekcemi. K návratu může uživatel použít tlačítko "Zpět".

Na domovské obrazovce lze pozorovat nové chování tlačítka "Zpět". Když uživatel zmáčkne dvakrát nebo vícekrát toto tlačítko, přesune se uživatel pouze o jednu sekci výše, ale hned se dále přesune do navigační lišty. Může to být užitečná zkratka, doporučení se ohledně takového chování nijak nevyjadřují a subjektivní pohled na to mám neutrální. Co ale doporučení zmiňují je chování s pomocí tlačítka zpět a to, stejně jako Netflix a YouTube, Spotify porušuje. Když se uživatel v seznamu nachází na jiném než prvním prvku a zmáčkne tlačítko "Zpět", vrací se rovnou do navigační lišty místo na první prvek v seznamu [17]. Toto nevyhovující chování tak mají všechny tři z analyzovaných aplikací.

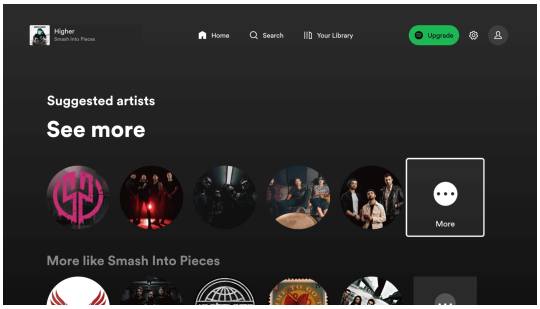
Obrazovkami, kde se ještě nachází audio obsah jsou kolekce se skladbami (viz obrázek 1.54), podcastové kolekce (viz obrázek 1.55) a obrazovky umělců (viz obrázek 1.56). Obrazovky kolekcí jsou rozděleny, stejně jako na YouTubu a Neflixu, do dvou vertikálních sekcí, kde na levé straně jsou informace o kolekci a na pravé straně seznam obsahu. Dole na levé straně se navíc zobrazuje právě přehrávaný obsah, do kterého může uživatel rychle přejít. Mezi ovládací prvky patří tlačítka relevantní ke zvolené skladbě či epizodě. Obrazovka kolekce skladeb se od obrazovky kolekce podcastů liší. Jsou mírně přizpůsobené typu obsahu. Stejně je tomu v přehrávací obrazovce. Vystupujícím prvkem je u podcastů zvýrazněná rozposlouchaná epizoda, pokud uživatel nějakou má, která je jako první v seznamu a při příchodu na



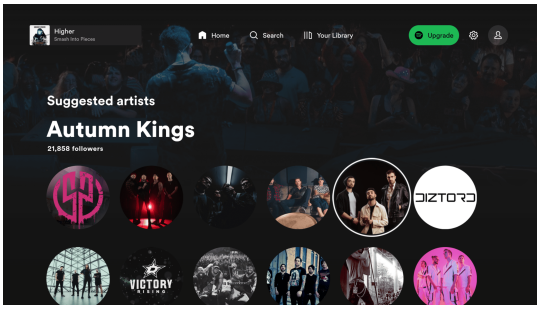
Obrázek 1.50 Domovská obrazovka – Spotify



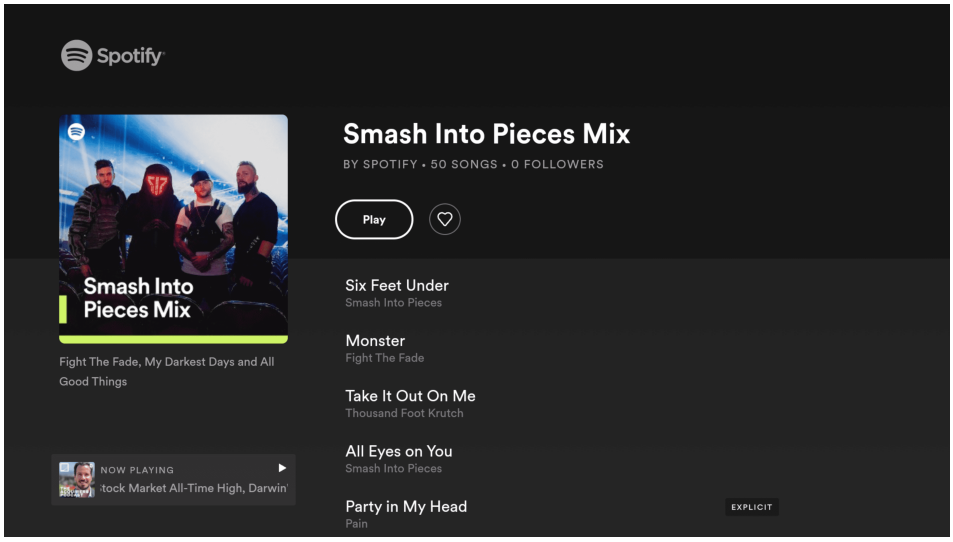
Obrázek 1.51 Profil – Spotify



Obrázek 1.52 ”Více”prvek ve feedu – Spotify



Obrázek 1.53 Mřížka po rozkliknutí ”Více”– Spotify



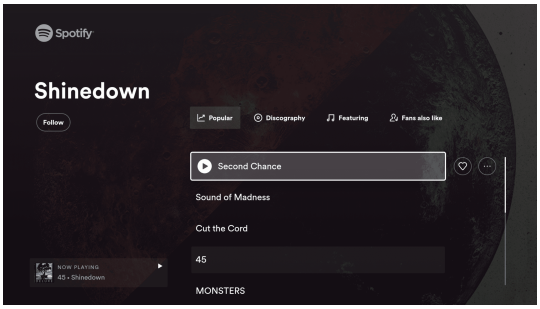
Obrázek 1.54 Kolekce se skladbami – Spotify

obrazovku je zaměřená, takže uživatel může jen kliknout a pokračovat v poslechu.

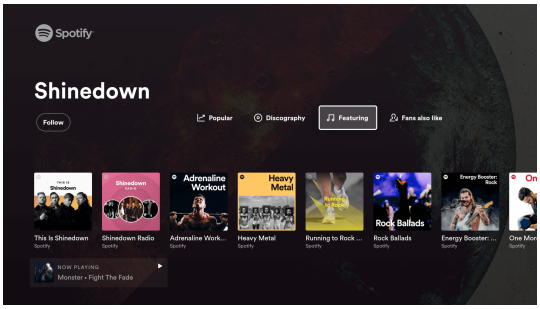
Poslední je obrazovka umělce. Rozpoložení obrazovky vypadá na první pohled velmi podobně. V sekci se seznamem skladeb je navíc členění do sekcí. Když zde uživatel zvolí jakoukoliv sekci mimo sekce “Populární”, obsah se začlení horizontálně (viz obrázek 1.57). Stejně, jako je tomu v sekcích domovské stránky aplikace. Prvky jsou zarovnány horizontálně, protože obsahové prvky nezabírají horizontální prostor a jsou to klasické karty. Můj subjektivní názor je, že by přepínání sekcí mohlo být posunuto výše a obsah by pak měl více prostoru. Prostor pro obsah totiž působí zbytečně limitovaně a využilo by se celého prostoru obrazovky. Jednak pro vertikální seznam skladeb by se skladeb na obrazovku vešlo více a jednak by horizontální seznam karet mohl být přeorganizován v mřížku, kam by se karet vešlo více.

Při analyzování jsem v seznamu narazil na problém se zvýrazňováním skladby/epizody, jejíž přehrávání bylo pozastaveno. Takový prvek je v seznamu, ve kterém se nachází, zvýrazněn barvou indikující, že je momentálním obsahem přehrávače (viz obrázek 1.55 a 1.54). Nicméně uživatel může být zprvu zmatený, co toto zvýraznění znamená. Když je skladba přehrávána, je doprovázena ikonou zvuku. Při pozastaveném přehrávání tato ikona mizí a zůstává jen barevné zobrazení. Toto chování je dokonce doporučeními Google zakázáno: „Pomocí indikátoru výběru s ikonou můžete jasně zobrazit vybranou položku v seznamu. To pomůže uživatelům snadno určit, kterou položku vybrali, a zlepší celkový uživatelský komfort. Při označování výběru v seznamu se nespoléhejte pouze na barvu pozadí.” [27]. Doprovodná ikona by pravděpodobně k plnému pochopení stačila.

Obrázek 1.55 Podcastová kolekce – Spotify



Obrázek 1.56 Obrazovka umělce – Spotify



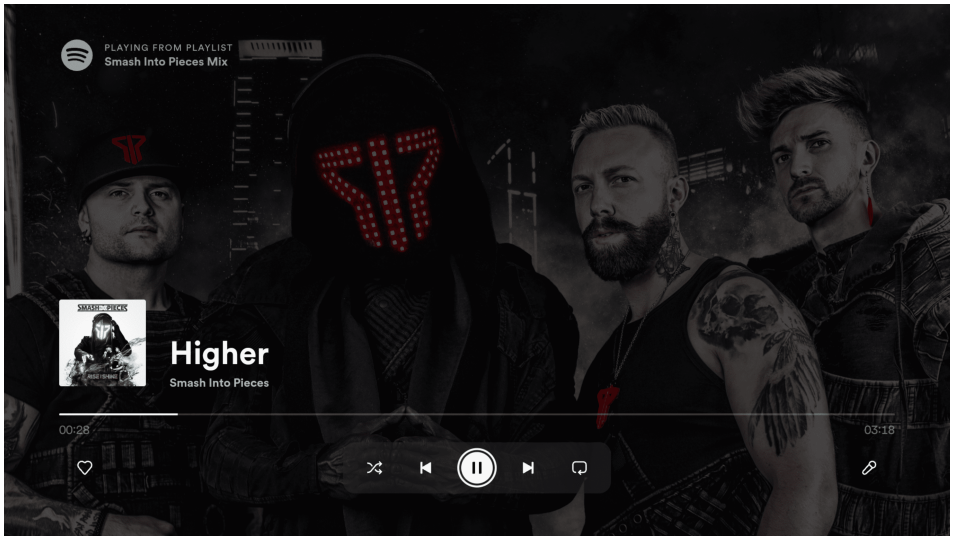
Obrázek 1.57 Obrazovka umělce (sekce s kolekcemi) – Spotify

1.6.3.4 Přehrávání audia

Obrazovka přehrávání je podobná pro hudbu (viz obrázek 1.58) i podcasty (viz obrázek 1.59). Prvky, které jsou na přehrávací obrazovce stejné jsou: text “Přehráváno z ”, obrázkový náhled obsahu, název, autor, časová osa přehrávání, tlačítko pro pozastavení/spuštění přehrávání a navigační tlačítka pro přechod na další či předchozí skladbu/epizodu.

Pokud je k obsahu dostupný obrázek na pozadí (ve formátu 16:9), zobrazí se na pozadí přehrávače přes celou obrazovku a je na něm navíc tmavá vrstva, aby byl čitelný text a ovládací prvky v přehrávači. Pokud obrázek na pozadí dostupný není, obrázkový náhled obsahu ve formátu 1:1 je zvětšen a zabírá poté na výšku zhruba třetinu obrazovky. Když není na pozadí obrázek, ale na obrazovce je jen zvětšený náhled, pak na pozadí je šedá barva, která je ale z mého pohledu příliš světlá a způsobuje tak nízký kontrast mezi textem a pozadím.

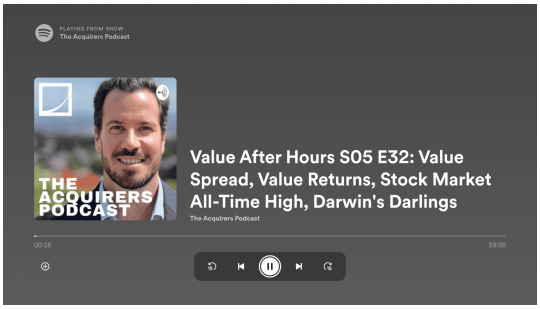
Aplikace YouTube má také nízký kontrast na “splash” obrazovce (ta, která zobrazuje logo a animaci načítání při spuštění aplikace). Z doporučení Google na téma kontrast: „Kontrast je jedním z nejdůležitějších aspektů kvality obrazu, zejména u moderních displejů HDR. Je to rozdíl mezi nejtmavší černou a nejsvětlejší bílou barvou, kterou televizor dokáže vytvořit. Vyšší kontrastní poměr obvykle znamená hlubší černou, což má velký vliv na celkovou kvalitu obrazu.”. Jako příklad nízkého kontrastu ukazuje Google kontrast šedého pozadí a bílého textu (kontrastní poměr 562:1). Jako příklad perfektního kontrastu bílou na černém (kontrastní poměr ∞:1) [28].



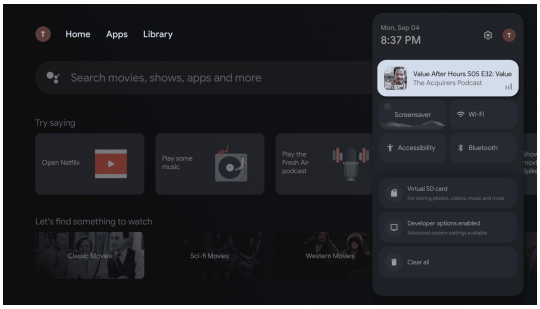
Obrázek 1.58 Přehrávač skladeb – Spotify

Na spodní straně obrazovky se nachází prvky, které se liší při přehrávání hudby a podcastu. Prvním takovým prvkem ja samostatné tlačítko pro přidání do seznamu. U hudby je to specifický seznam "Oblíbené písničky", proto je zde jiná ikona. Funkcionalita je ale stejná. V případě podcastu se epizoda přidá do seznamu "Moje epizody". Dále jsou ve středu po stranách tlačítka pro zapnutí náhodného přehrávání a na druhé straně zapnutí opakovaného přehrávání. Ta jsou přítomna u přehrávání skladeb. Pro podcasty jsou zde tlačítka pro posun na časové ose přehrávání o patnáct sekund, buď to vzad nebo vpřed. Přehrávač hudby má ještě navíc na pravé straně možnost přepnutí do "lyrics módu", ve kterém se na obrazovce zobrazuje zpívaný text.

Zvláštní je v případě Spotify zabraňování spuštění Ambient Mode i když je přehrávání přerušeno. Spotify nic zajímavého, co by aplikaci ospravedlňovalo zabraňovat přechodu do Ambient Mode, nezobrazuje a porušuje tak požadavek TV-BA. Aplikace přehrává audio i na pozadí a správně se tak v TV UI zobrazuje Now Playing karta [8] (viz obrázek 1.60). Kliknutí na tuto kartu přenese uživatele do přehrávače v aplikace, kde může přehrávání zastavit a Spotify tak splňuje požadavky TV-NP a TV-PA spojené s přehráváním na pozadí a kartou Now Playing [2].



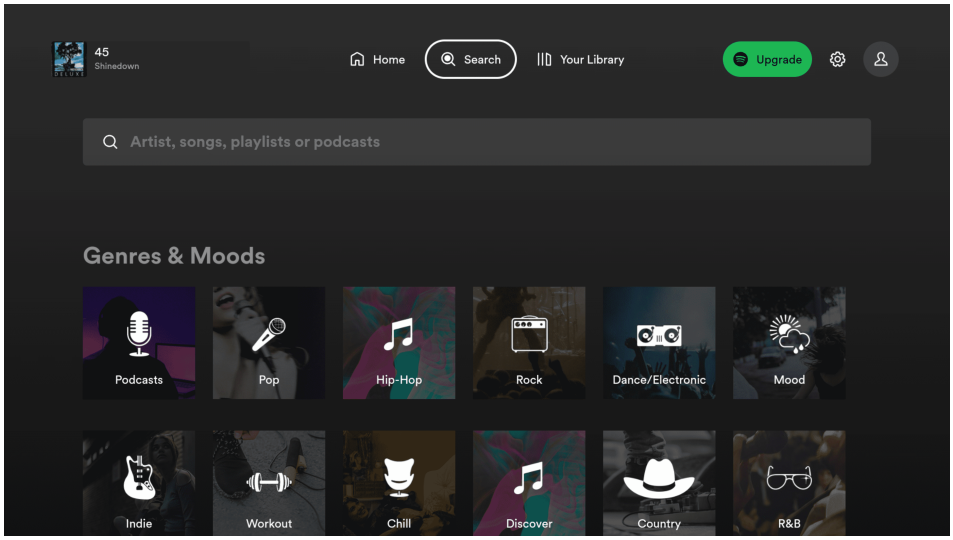
Obrázek 1.59 Přehrávač podcastů – Spotify



Obrázek 1.60 Now Playing karta – Spotify

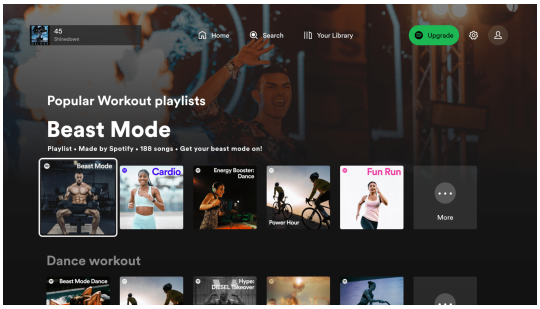
1.6.3.5 Vyhledávání

Vyhledávání ve Spotify není jen vyhledávačem, ale je zde i rozsáhlá mřížka obsahující kategorie obsahu (viz obrázek 1.61). Při rozkliknutí nějaké z nich se obsah člení do horizontálních sekcí s vertikálními

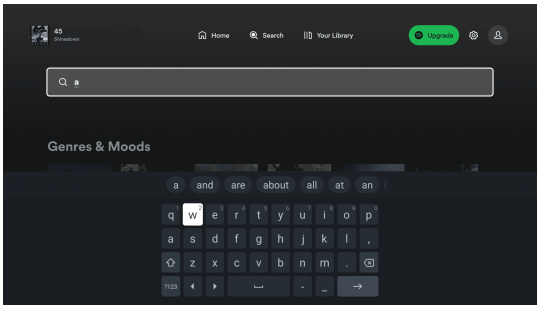


Obrázek 1.61 Vyhledávání – Spotify

seznamy (viz obrázek 1.62), tak, jak je tomu na domovských stránkách analyzovaných aplikací. Vyhledávání využívá na rozdíl od Netflixu a Youtubu systémovou klávesnici, která se zobrazuje přes celou šířku na spodní straně obrazovky (viz obrázek 1.63). Zabírá tak výrazně více místa, než by potřebovala. Vývojáři mají možnost zobrazovat klávesnici jen na části obrazovky, která je nutná pro její zobrazení [29] – tj. bez prázdných míst po stranách klávesnice. Zde klávesnice přes celou šířku nevadí, protože aplikace nezobrazuje žádné průběžné výsledky či návrhy dokončených frází, což je ale spíše důsledkem...



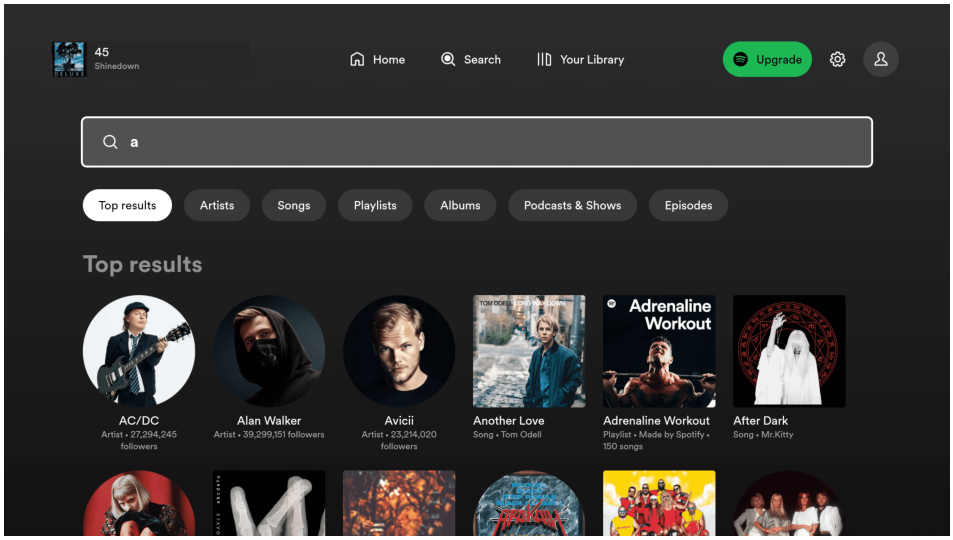
Obrázek 1.62 Kategorie obsahu – Spotify



Obrázek 1.63 Vyhledávání (s klávesnicí) – Spotify

Aplikaci by nicméně rozhodně šlo doplnit o prvky průběžné doplňované při zadávání vyhledávacího vstupu. Momentálně jsou navrhovány jen slova v rámci klávesnice, která ale nemají kontext aplikace a jsou to obecná slova. Po zadání vyhledávacího dotazu se zobrazí výsledky v mřížce (viz obrázek 1.64). Mezi vyhledáváním a mřížkou výsledků jsou v tabech sekce pro filtrování výsledků. Tyto možnosti filtrace jasně indikují svou funkcionalitu uživateli a jsou užitečným prvkem, který není přítomný na Netflixu ani na YouTubu. Filtrace umožňuje uživatelům zadat klíčová slova a filtr, namísto celého názvu, který by mohlo být zdlouhavé pomocí ovladače zadávat.

Takové filtrace by z mého pohledu těžila i aplikace YouTube, kdyby ji implementovala. Spotify tak sice nenabízí průběžné výsledky a našeptávání, ale ve výsledcích hledání se uživatel lépe orientuje. Kombinace těchto funkcionalit by, dle mého názoru, byla optimální implementací funkcionality hledání v obsahu, který je členěný do sekcí a lze ho filtrovat.



Obrázek 1.64 Výsledky vyhledávání – Spotify

1.6.3.6 Synchronizace s ostatními zařízeními

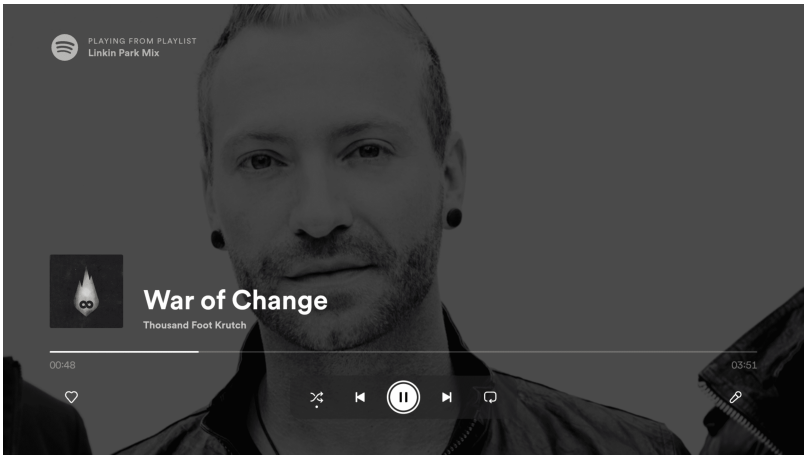
Synchronizace je v případě Spotify vyřešena asi nejlépe z analyzovaných aplikací. Vychází to také z potřeby uživatelů pro zachování aktuální relace při přecházení mezi zařízeními (tím je myšleno mezi platformami, kde má uživatel Spotify spuštěné). Uživatel například poslouchá hudbu/podcast při cestě do práce a v práci pracuje na notebooku, odkud chce v poslechu pokračovat. Nebo přepne na sluchátkách zdroj z telefonu na notebook a chce pokračovat v poslechu. Pro tyto situace je synchronizace mezi zařízeními plynulá. Při přehrávání obsahu na jednom zařízení, se na ostatních zařízeních, kde je uživatel přihlášen, také zobrazuje právě přehrávaný obsah (viz obrázky 1.65 a 1.66). Na ostatních zařízeních (web, stolní PC, mobil) je indikováno, na kterém zařízení se obsah přehrává a dá se z nich zařízení, kde chce uživatel poslouchat, změnit.

V případě Spotify TV aplikace je ale tato funkcionalita omezená. Změnu aktuálního přehrávacího zařízení na tu televizní provedenou z jiného zařízení zvládá TV aplikace v pořádku a začne přehrávat. Naopak to ale nelze – tj. z TV aplikace zvolit jiné zařízení pro přehrávání. Spotify na TV takovou funkcionalitu vůbec neposkytuje. Na TV aplikaci Spotify lze vidět, co se právě přehrává na jiném zařízení (skrze prvek aktuální relace na domovské obrazovce nahoře na levé straně). Při zobrazení přehrávače na TV se automaticky přepíná přehrávání z jiného zařízení na TV, což je nekonzistentní chování v rámci Spotify aplikací na jiných platformách, kde uživatel přehrávanou relaci může zobrazit bez ovlivnění přehrávání.

Tato funkcionalita existuje také z nutnosti existence jediné aktivní relace na účet, tj. že nelze poslouchat současně na více zařízeních, natož s různým obsahem. Funkcionalita tak existuje ne jen pro účely uživatele, ale také jako mechanismus pro omezení počtu aktivních relací na jednu. V rámci aktuální relace TV aplikace neřeší synchronizaci spojenou s historií přehrávání, ani frontou přehrávání. Nelze tak například na TV přidávat skladby do fronty, protože fronta je sdílená mezi zařízeními a TV aplikace umí pouze přijímat a odesílat informace o aktuálně přehrávaném obsahu.

1.6.3.7 Nákup předplatného

Pro nákup předplatného uživatel musí kliknout na tlačítko “Upgrade” v navigační liště. TV aplikace poté zobrazí statickou neinteraktivní stránku s odkazem na spotify.com/upgrade, kde uživatel může upgrade provést (viz obrázek 1.67). Otázku nákupu tak Spotify TV aplikace vůbec neřeší.



Obrázek 1.65 Synchronizace aktivní relace (1) – Spotify

Obrázek 1.66 Synchronizace aktivní relace (2) – Spotify

1.6.4 Shrnutí

Návrh a implementace aplikace se bude inspirovat analyzovanými aplikacemi, při nichž ale bude brán ohled na nedostatky aplikací, co se týče porušování doporučení, nesplňování požadavků, ale i vlastních poznatků.

1.6.4.1 Google doporučení

Tabulka 1.5 shrnuje plnění doporučení analyzovanými aplikacemi, které nebyly plně splněny alespoň jednou z aplikací. Na odlišnosti od doporučení lze nahlédnout pro inspiraci na místě, kde by odchylka od doporučení dávala smysl. Pokud při implementaci aplikace narazím na problém pro splnění nějakého z doporučení zmíněných v tabulce, lze také nahlédnout na implementaci analyzovanými aplikacemi.

Z tabulky 1.5 lze vidět, že žádná z aplikací plně nesplňuje doporučení daná Googlem a je tedy dobré

Tabulka 1.5 Plnění doporučení analyzovanými aplikacemi

Doporučení	YouTube	Netflix	Spotify
Nevytváří smyčku při navigaci pomocí tlačítka “Zpět” [17]	Ano	Ne	Ano
Použití tlačítka “Zpět” v pohlcujícím seznamu na domovské obrazovce [17]	Ne	Ne	Ne
První obrazovka je i poslední [15]	Ano	–	Ano
Nezobrazování potvrzovacího okna při odchodu z aplikace [17]	Ano	–	Ne
Jasné a rychlé procházení hierarchiemi obsahu [13, 14]	Ano	Ano	Ano
Max. počet primárních destinací v navigačním panelu [16]	Ne	Ne	Ano
Práce s kontrastem [28]	Spíše ano	Ne	Ano
Zvýrazňování selekce v seznamu [27]	Ano	Ano	Ne vždy
Nezobrazování tlačítka “Zpět” [25]	Ano	Ne vždy	Ano
Ambient mode [7]	Spíše ano	Spíše ano	Ne

Obrázek 1.67 Nákup předplatného Premium – Spotify

Tabulka 1.6 Plnění požadavků analyzovanými aplikacemi

Požadavky	YouTube	Netflix	Spotify
Ovládání videa [2]	Ne	Ano	Ano
"Zpět" vede na domovskou obrazovku TV [2, 23]	Ano	Ne	Ne

se při návrhu v některých oblastech vydat vlastní cestou a neinspirovat se analyzovanými aplikacemi.

1.6.4.2 Požadavky Googlu

Požadavky Google slouží k tomu, aby aplikace, které splňují tato kritéria kvality, byly v Google Play klasifikovány jako aplikace pro Android TV. Splnění těchto požadavků je tedy nezbytné. Při implementaci lze jako referenci použít analyzované aplikace, které daná kritéria splňují. V tabulce 1.6 jsou požadavky, které nebyly plně splněny alespoň jednou z analyzovaných aplikací.

Jak lze vidět z tabulky 1.6, ani jedna z analyzovaných aplikací nesplňuje všechny požadavky, aby byly kvalifikované na Google Play jako TV aplikace. Moje aplikace se chybám, které dělají analyzované aplikace, vyhne, a bude tak jedinou aplikací mezi čtveřicí aplikací, která bude splňovat všechna doporučení a požadavky.

Kapitola 2
Návrh

V této kapitole se budu věnovat návrhu rozhraní a fungování aplikace podloženého provedenou analýzou.

2.1 Přihlášení

Pro splnění prvního funkčního požadavku bude použita přihlašovací obrazovka. Nepřihlášený uživatel nebude mít možnost se dostat k obsahu. Přihlašovací obrazovka tak bude první zobrazovanou obrazovkou (kromě splash obrazovky). Aby mu byl zpřístupněn zbytek aplikace, uživatel se zde musí přihlásit.

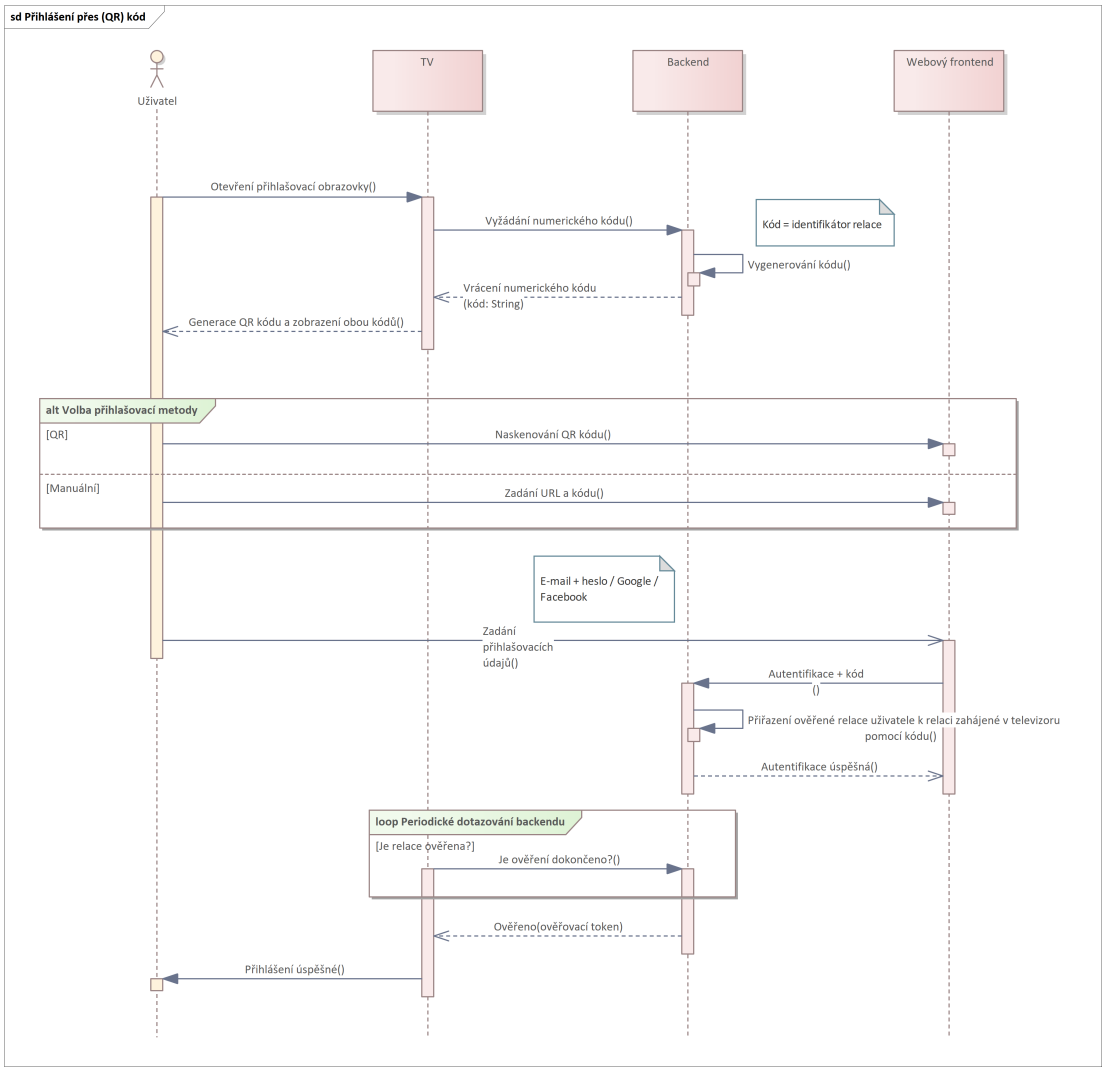
Forendors uživatelé mají možnost se na ostatních platformách přihlásit jak e-mailem a heslem, tak i přes zprostředkovatele jako Google nebo Facebook. Pro pokrytí přihlášení všech uživatelů je potřeba buď to poskytnou možnost přihlášení všemi zmíněnými způsoby přímo na TV nebo pomocí kódu a odkazem či jen QR kódem. Odkaz či QR kód bude vyžadovat implementaci funkcionality minimálně na backendu a webovém frontendu, ideálně i na mobilním. Proces přihlášení přes QR kód či zadání URL adresy a numerického kódu by vypadal následovně: viz diagram 2.1.

Poslední a nejoptimálnější možností přihlášení, které však není aplikovatelná na všechny uživatele, je, ve vyskakovacím okně na spodu obrazovky, nabídnutí přihlášení přes uložená hesla v Google účtu, které by bylo implementováno pomocí Credentials Manager knihovny. Protože ale není aplikovatelné pro všechny uživatele, ať už protože nemají heslo uložené na svém Google účtu nebo se ke službě bude přihlašovat jiný uživatel, než který je přihlášený na televizi, tak tato metoda slouží hlavně jako zpříjemnění procesu, nežli nutnost. Proto má tento způsob nejnižší prioritu, co se týče implementace přihlášení.

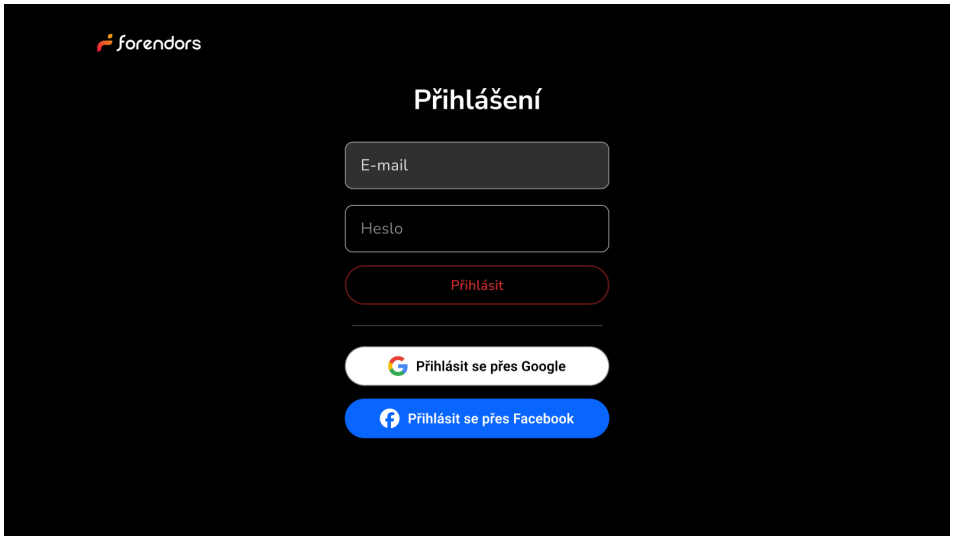
Kvůli nutnosti zapojování dalších osob spojených s implementací na webovém frontendu a backendu pro možnost přihlášení přes (QR) kód má tento způsob přihlášení nižší prioritu implementace, než první způsob ručního zadávání údajů. Pro ruční zadávání údajů na TV stačí jen backendová podpora, která již existuje a snižuje to tak časové nároky spojené se splněním funkčního požadavku FP1 – Přihlášení.

Možnost přihlášení přes (QR) kód přesahuje časový rozsah, který lze strávit na funkcionalitě přihlášení. Bylo by nutné zapojit další osoby spojené s implementací webového frontendu a backendu. Oproti tomu pro ruční zadávání údajů na TV stačí jen již existující backendová podpora a to snižuje časové nároky spojené se splněním funkčního požadavku FP1 Přihlášení.

Pro splnění funkčního požadavku FP1 Přihlášení musíme zajistit možnost přihlášení uživatele registrovaného kterýmkoliv z možných způsobů: e-mail + heslo, Google a Facebook přihlášení. Proto implementuji manuální způsob přihlášení pro dané způsoby, tedy viz návrh 2.2. Způsob přihlášením s e-mailem a heslem je přímočarý. Uživatel zadá informace a stiskne tlačítko přihlášení, aplikace provede ověření a po ověření přenese uživatele na Dashboard. Protože pracujeme s Android/Google TV, je v aplikacích dobře integrovaný způsob přihlášení přes Google. Aplikace používající přihlášení přes Google využívají totiž stejného přihlašovacího prostředí (se stejnými již přihlášenými účty na TV), jako systém. Přihlášení přes Google je tak delegováno na systém. Takovou možnost pro Facebook přihlášení nemáme.



Obrázek 2.1 Přihlášení přes (QR) kód – sekvenční diagram



Obrázek 2.2 Návrh obrazovky přihlášení e-mailem nebo Google/Facebook účtem

U přihlášení přes Facebook lze postupovat dvěma způsoby. Buď to využijeme WebView a necháme uživatele vyplňovat údaje v něm nebo se uživatel přihlásí pomocí QR kódu či zadání kódu na webové stránce. Tento způsob je zaprvé uživatelsky nepřívětivý, ale je také implementační neznámou. Při analýze zmiňovaných aplikací žádná aplikace nenabízela přihlášení přes Facebook. Stejně tak i další aplikace nezmiňované v analýze nenabízejí možnost přihlášení přes Facebook, ale nabízejí přihlášení přes Google. A to i přesto, že některé platformy mají možnost registrace přes Facebook. V aplikaci pak nedávají těmto uživatelům, registrovaným přes Facebook, možnost přihlásit se do aplikace. Při zkoušení implementování Facebook přihlášení přes WebView jsem si potvrdil, že obavy byly oprávněné. Neexistuje jednoduchý způsob přihlášení přes WebView.

Proto jsem zvolil druhou variantu přihlášení - pomocí jiného zařízení a kódu. Tato varianta nabízí více uživatelsky přívětivý způsob, protože deleguje práci na více způsobilé zařízení jako telefon nebo jiné zařízení. Implementace vyžaduje komunikaci s Facebook backendem, jejíž průběh je až na detaily prakticky stejný, jako popisuje sekvenční diagram přihlášení přes QR kód 2.1. V tomto případě ale nebudeme komunikovat s Forendors backendem, nýbrž s Facebook backendem, který je tento způsob přihlašování plně podporuje. Implementace funkce bude tedy zahrnovat i implementaci datové a doménové vrstvy. Obrazovka musí obsahovat QR kód nebo webovou stránku pro přihlášení a kód nebo oboje. Některé aplikace poskytují pouze QR kód, jiné oboje. Rozhodl jsem se pro implementaci obou variant, protože zobrazení webové stránky a kódu vyžaduje minimální úsilí. Informace jsou totiž totožné, jako pro vygenerovaný QR kód. Pro zachování přehlednosti jsem se rozhodl pro separátní stránku, kde budou informace zobrazené. Návrh obrazovky je viz 2.3.

Se zmíněnými zjištěnými a rozhodnutími lze dodefinovat scénáře a obrazovky pro případ užití UC1 Přihlásit. Finální případ užití UC1 Přihlásit vizte níže, kde jsou **tučným písmem** zvýrazněné nově dodefinované obrazovky a scénářové detaily.

UC1 Přihlásit Případ užití umožňuje uživateli přihlásit se do aplikace pod svým registrovaným účtem pomocí přihlašovacích údajů (e-mail + heslo) nebo pomocí třetí strany (Google či Facebook). V případě neregistrovaného uživatele je uživateli zobrazena zpráva o nutnosti registrace.

Obrazovky

- Přihlášovací obrazovka.
- **Systémová Google přihlašovací obrazovka.**
- **Facebook přihlašovací obrazovka.**

Obrázek 2.3 Návrh obrazovky přihlášení Facebook účtem

Pre-condition

- Uživatel je registrován.

Hlavní scénář

1. Uživatel vyplní přihlašovací údaje svého Forendors účtu.
2. Uživatel klikne na tlačítko přihlášení.
3. Aplikace provede kontrolu přihlašovacích údajů.
4. Aplikace uživatele přihlásí do aplikace.

1. alternativní scénář

1. Uživatel klikne na tlačítko přihlášení se přes Google.
2. **Aplikace zobrazí systémovou obrazovku pro přihlášení se Google účtem.**
3. **Uživatel zvolí existující účet nebo přidá nový.**
4. **Aplikace uživatele přihlásí do aplikace.**

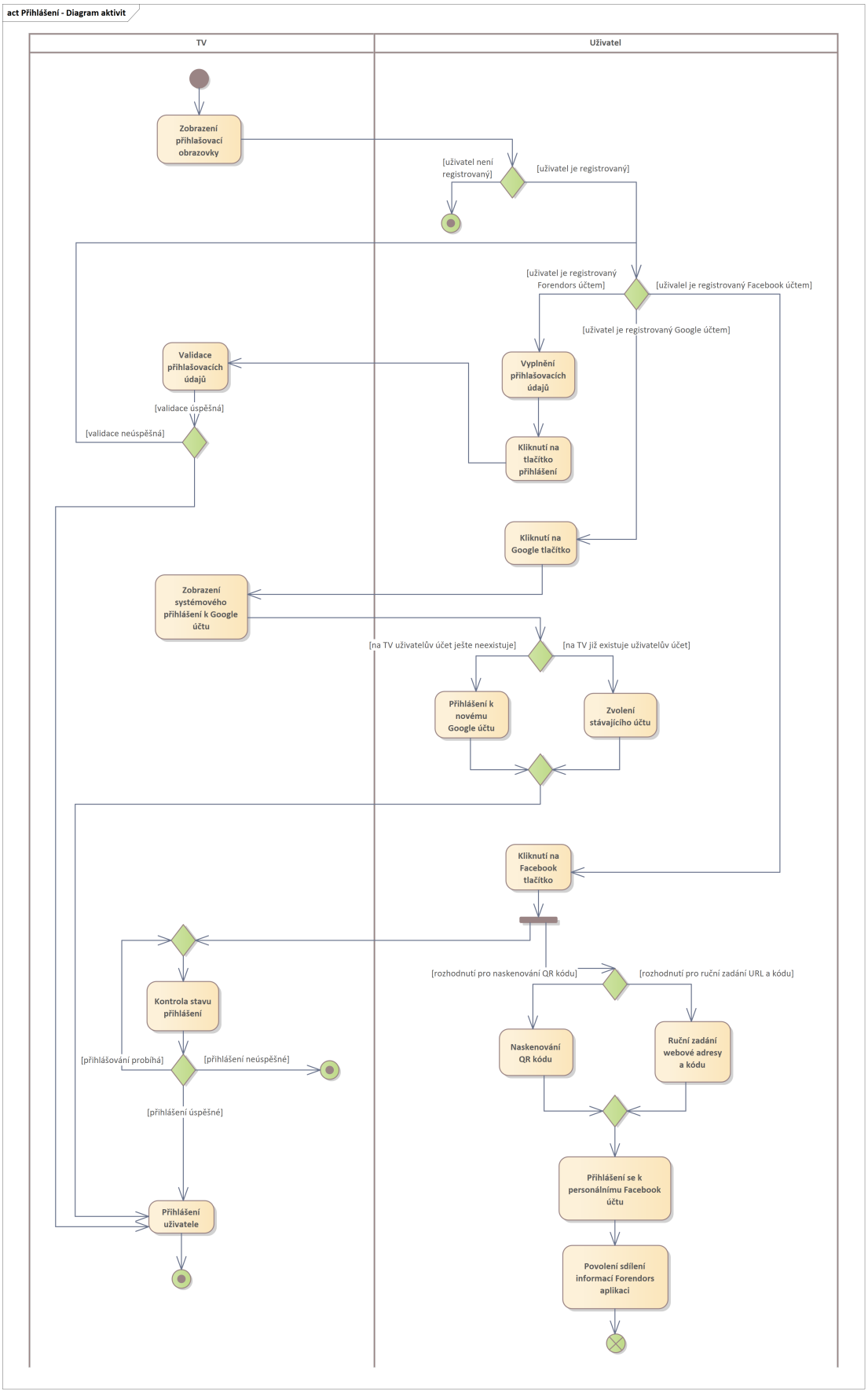
2. alternativní scénář

1. Uživatel klikne na tlačítko přihlášení se přes Facebook.
2. **Aplikace zobrazí obrazovku přihlášení přes Facebook.**
3. **Uživatel naskenuje QR kód nebo ručně otevře webovou Facebook přihlašovací stránku a vyplní kód.**
4. **Uživatel se mimo aplikaci přihlásí do svého Facebook účtu a potvrdí nutná povolení pro přístup k informacím Forendors TV aplikací.**
5. **Aplikace uživatele přihlásí do aplikace.**

Pro vizualizaci daných aktivit vizte diagram aktivit přihlášení 2.4.

2.2 Dashboard

Druhým funkčním požadavkem je dashboard. Popis požadavku je jasný, uživateli se musí zobrazovat video a audio epizody jeho tvůrců. Na hlavní obrazovce bude využito ”pohlcujících seznamů”, jak je

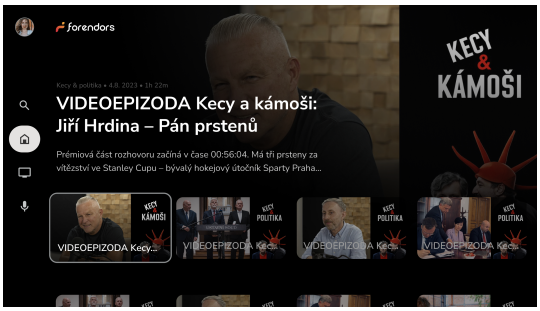


Obrázek 2.4 Diagram aktivit přihlášení

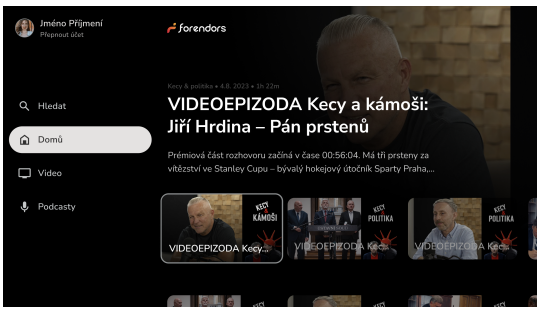
tomu v případě Netflixu. Bude zde tak možnost zobrazení informací k dané epizodě a uživatel bude vědět o čem zhruba epizoda je, aby se mohl rovnou pustit do přehrávání.

První návrh počítal s rozdělením obrazovek do navigačních destinací Domů, Video a Audio (viz návrh 2.5 a 2.6). Dané rozdělení bylo intuitivním krokem, jelikož mobilní aplikace využívá stejného rozdělení obsahu (viz obrázek 2.7). V případě TV jsem ale později došel k rozhodnutí se navigačních destinací Video a Audio prozatím zbavit. Rozhodnul jsem se tak, protože daných destinací by nebylo optimálně využito. Prvním důvodem je, že epizody, např. v navigační sekci Video, by byly buď to organizované do mřížky nebo stejně, jako na domovské obrazovce.

V případě mřížky bude k dispozici jeden, možná dva řádky obsahu navíc, protože stále musíme zobrazovat detaily o videu, které mohou být pro uživatele důležitější než pouhý náhled. Tyto informace jsou v případě platformy Forendors důležitější, než na Netflixu nebo YouTube, protože na těchto platformách lze jeden prvek od druhého dobře odlišit a pochopit, o co se jedná. V případě Forendors budou ale epizody vypadat velmi podobně a bude důležité např. datum vydání epizody. Zároveň uživatelé na Forendors funguje na principu odběru a uživatelé často vůbec nemusí epizodu hledat, protože čerstvé epizody budou mezi prvními výsledky. Tím pádem se i snižuje nutnost zobrazování a prohledávání většího množství obsahu. V případě, že uživatel sleduje více tvůrců a bude potřebovat nějakou starší epizodu dohledat, bude mít tu možnost v profilu tvůrce, kde se už budou zobrazovat epizody daného tvůrce. To samé platí pro sekci s audio epizodami. Nový návrh má následující podobu: viz návrh 2.8. Audio a video sekce jsou zde přesunuty do vlastních pohlcujících seznamů.



Obrázek 2.5 Původní návrh domovské obrazovky



Obrázek 2.6 Původní návrh domovské obrazovky (navigační panel)

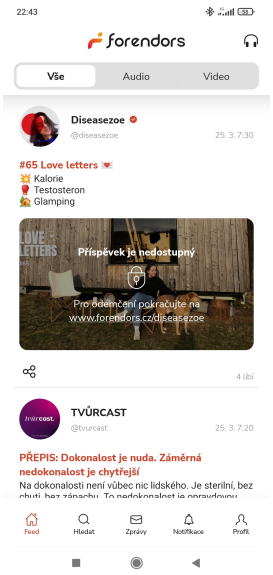
2.3 Video přehrávač

Připomenutí požadavku TV-PC: „Při přehrávání videa nebo zvuku se stisknutím středového tlačítka D-Padu přehrávané médium pozastaví. Když je přehrávání pozastaveno, stisknutím prostředního tlačítka D-Padu se přehrávání obnoví. Tlačítka D-Padu vlevo a vpravo slouží k rychlému převíjení aktuální stopy vpřed, resp. vzad."

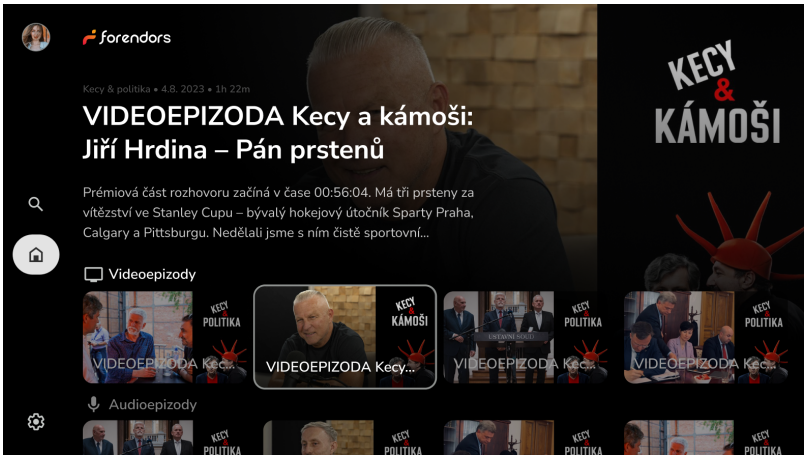
Při návrhu přehrávače je pár věcí, na které je potřeba si dát pozor:

- Přizpůsobení k použití s ovladačem.
- Informace o přehrávaném obsahu.

Přizpůsobení k použití s ovladačem Všechny ovládací prvky by měly být jednodušše a intuitivně dostupné. Není možné si prvky po obrazovce rozmístit, jak je libo, protože uživatel nemá dostupnou myš, ani dotykový displej. Není tak například možné umisťovat více ovládacích prvků vedle od časové osy přehrávání, protože tlačítka "Vlevo"a "Vpravo"slouží pro pohyb po časové ose (to vychází z nutnosti splnění požadavku TV-PC). Stejně tak se uživatel špatně dostane do horního pravého rohu, protože je



Obrázek 2.7 Mobilní aplikace Forendors



Obrázek 2.8 Nový návrh domovské obrazovky

vždy zaměřen na tlačítko pozastavení/spuštění. To jsou věci, co nedělat. Optimální přístup je z mého pohledu mít ovládací prvky co nejvíce pospolu (pokud to nebude na úkor vzhledu) a ještě více tím usnadnit navigaci. Protože je tlačítko pozastavení/spuštění vždy zaměřeno jako první (viz TV-PC).

Informace o přehrávaném obsahu Součástí obsahu jsou informace, které se uživateli musí zobrazovat. Kromě názvu, jména autora a časových údajů je s videem spojený popis epizody. Tato informace by mohla být obsažena na obrazovce, která by se zobrazovala po zvolení epizody a před přechodem do přehrávače. Tu jsem se ale rozhodl neimplementovat, protože není informace nebo akce, která by nemohla být zobrazena v přehrávači. V budoucnu, pokud přibydou informace či akce, které by mohl uživatel provést, se může taková obrazovka implementovat a popis převést sem. Popis může být různých délek, takže se musí počítat i s delším popisem, který by si uživatel mohl chtít přečíst.

Možností zobrazení, kterými se lze inspirovat u analyzovných aplikací jsou následující. Spotify informace o epizodě podcastu neukazuje nikde. Netflix krátký popis zobrazuje na obrazovce zobrazovanou před přechodem do přehrávače. YouTube je nám potenciální implementací nejblíže, jelikož popis videa obsahuje v přehrávači. Rozdílem YouTube aplikace oproti Forendors aplikaci je fakt, že popis je pro uživatele Forendors důležitý. Dodává jim kontext k dané epizodě. To v případě YouTube tolik důležité není. Uživatelé YouTube jsou zvyklí na rozložení obsahu v klasických, nepohlcujících seznamech. Forendors obsah je publikován jako příspěvek s videem, nežli video s popisem. Z toho důvodu jsem se rozhodl popis neschovat, ale zobrazit přímo v přehrávači. Místo pro něj v přehrávači je. Přistup k popisu bude možný po pozastavení přehrávání videa a zobrazení překrytí s ovládacími prvky a ostatními informacemi.

Nakonec bude obrazovka obsahovat, kromě názvu a data zveřejnění, i tlačítko s avatarem tvůrce, který při stlačení přenese uživatele na profil tvůrce. Finální návrh obrazovky je následujicí: viz návrh 2.9.

Co se týče přehrávače, který by byl schopný zvládnout možnosti přehrávání zmíněné v analytické části, jsem se rozhodl pro použití přehrávače ExoPlayer z knihovny Media3 [30], která je oficiální a doporučovanou knihovnou pro mediální obsah z kolekce Jetpack. Jetpack zahrnuje kolekci knihoven pro Android, které začleňují osvědčené postupy a zajišťují zpětnou kompatibilitu v aplikacích pro Android.

ExoPlayer poskytuje rozsáhlé možnosti pro manipulaci a správu audio a video streamů. To je zásadní, vzhledem k požadavkům na podporu různých audio formátů a kódování (jako Dolby Digital+, DTS, PCM) a rozlišení videa (HD, 4K, HDR). ExoPlayer umožňuje snadnou integraci s různými zvukovými

Obrázek 2.9 Návrh video přehrávače

a video formáty a zajišťuje, že aplikace může nabízet nejlepší možnou kvalitu obsahu v závislosti na schopnostech připojeného zařízení. Navíc s podporou daných formátů je možné začít obsah backendem streamovat v lepších formátech a aplikaci nebude nutné příliš upravovat a nový typ streamovaného obsahu může být zaveden téměř okamžitě. Mezi podporovanými streamovacími formáty je i M3U8, který Forendors momentálně využívá.

Ačkoliv se v případě videa neočekává dynamická změna výstupního zařízení, flexibilita ExoPlayeru v dynamickém přizpůsobování je výhodná pro audio. ExoPlayer může automaticky detekovat změny v audio výstupních zařízeních a přizpůsobit se jim, což je v souladu s požadavky na podporu různých audio výstupů a jejich možných změn během přehrávání. ExoPlayer navíc nabízí možnosti pro customizaci a rozšíření, což umožňuje přizpůsobit přehrávač specifickým potřebám aplikace, které se mohou v budoucnu měnit, ale i připravená řešení problémů, jako např. adaptivní streamování, což je streamovací technika, která mění kvalitu streamu v závislosti na dostupné šířce pásma sítě.

2.4 Audio přehrávač

Pro audio přehrávač budeme mít buď to dostupný náhled anebo pouze audio. V obou případech se ale bude jednat o statické pozadí. To dává prostor ostatním ostatním informacím, které se mohou zobrazovat na celé obrazovce, jelikož jsou primárním vizuálním obsahem. Když se podíváme na implementaci přehrávače podcastů ve Spotify (viz obrázek 1.59), můžeme si všimnout velkého náhledu podcastu. Spotify se snaží využívat celé obrazovky a k naplnění obrazovky využívá zvětšeného náhledu s názvem. V případě Forendors aplikace se ale naskýtá idealní možnost pro naplnění prostoru zobrazením popisu epizody. Ke zdůraznění tvůrce obsahu a dodání obrazovce rozmanitosti jsem oproti video přehrávači zvětšil i tlačítko s avatarem tvůrce. Výsledná obrazovka vypadá následovně: viz návrh 2.10. Ovládací prvky a informace nebudou fungovat jako překryv, který se po chvíli přehrávání skryje, ale vše bude přitomné stále.

2.5 Vyhledávání tvůrce

Vyhledávání nebude umět našeptávání, ani zobrazovat průběžné výsledky, ani filtraci, jako má Spotify. Vyhledávání v bude vyhledávat jen tvůrce, nikoliv obsah. Protože obrazovka bude schopná hledat pouze tvůrce, tak se na obrazovce nemusí zobrazovat tolik výsledků, jelikož nebude nutné hledat např. ještě

Obrázek 2.10 Návrh audio přehrávače

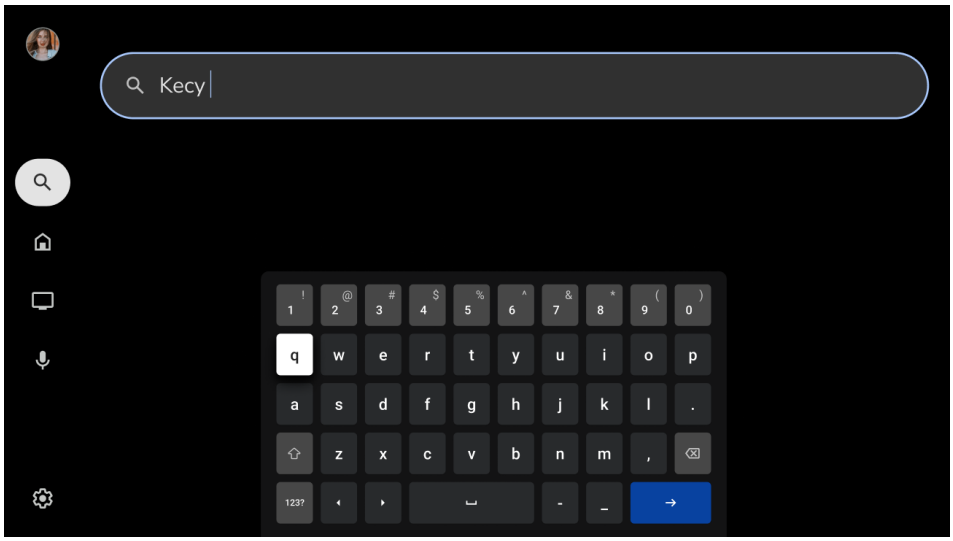
navíc mezi příspěvky. Počtem výsledků myslím počet viditelných výsledků, seznam bude scrollovatelný. To dává prostor obrazovce zobrazovat i jiné informace než jen jméno hledaného tvůrce a náhled, ale i popis kanálu apod. a dodat vhodnou prezentaci tvůrců. Pro případ užití Forendors aplikace se bude více hodit vertikální seznam, nežli mřížka. To je odlišný přístup od analyzovaných aplikací, ale z mého pohledu, ze zmíněných důvodů, ospravedlněný. Návrh obrazovky vyhledávání je následující: viz návrh 2.11 a 2.12.

2.6 Profil tvůrce

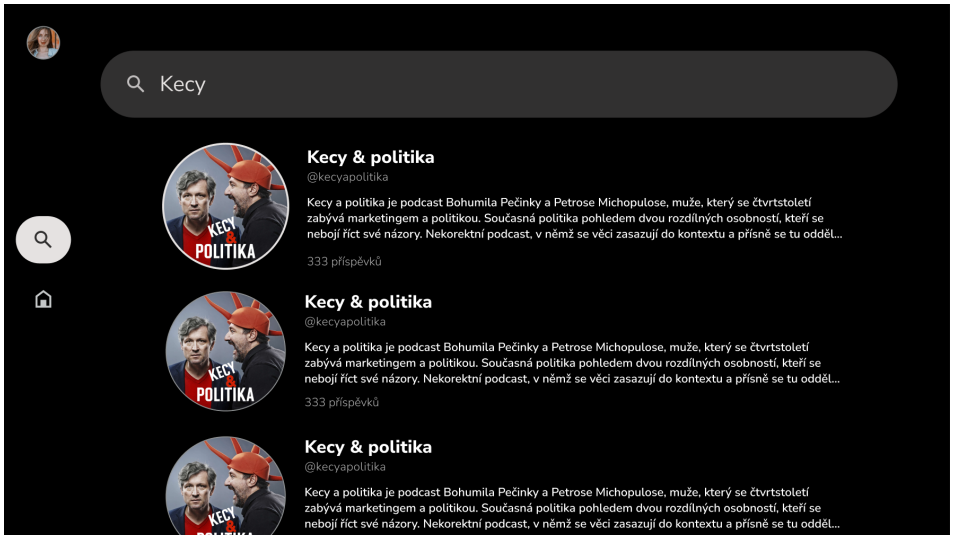
Bude se jednat o obrazovku kolekce, tedy detaily o tvůrci na levé straně a epizodami na pravé straně. Při návrhu obrazovky jsem se inspiroval obrazovkou podcastové kolekce Spotify (viz 1.55), protože obsahuje jak prezentaci tvůrce, tak seznam epizod. V případě Forendors aplikace jsou ale dva významné rozdíly. První je, že Forendors nabízí tvůrcům členění do vlastních kategorií. Tyto kategorie můžou být pro snadné nalezení obsahu vitální. Druhým rozdílem je, že uživatel může obsah vybírat podle perexu, takže tento perex by měl být, vedle názvu a náhledu, primárním zobrazovaným prvek epizody. Návrh má následující podobu: viz návrh 2.13. Kvůli zmíněným důvodům je celá pravá strana věnována pouze obsahu. To je rozdíl oproti Spotify kolekci (viz obrázek 1.55) –název je přenesen do levé sekce a je nahrazen kategoriemi obsahu. A namísto plného zobrazení několika epizod je zobrazena jen jedna s co největším možným prostorem pro popis, ale s možností scrollovat, indikováno částečným zobrazením další epizody.

2.7 Přepnutí účtu

Přepnutí účtu lze dosáhnout jednoduchým odhlášením a přihlášením za jiný účet. Proto implementuji pouze odhlášení. Uživatel bude mít možnost odhlásit se v postranním navigačním panelu viz návrh 2.14. Po odhlášení se půjde ihned přihlásit.



Obrázek 2.11 Návrh obrazovky vyhledávání (aktivní)



Obrázek 2.12 Návrh obrazovky vyhledávání (výsledky)

Obrázek 2.13 Návrh obrazovky profilu tvůrce

Obrázek 2.14 Návrh umístění odhlášení

Kapitola 3
Implementace

V této kapitole se zaměřím na architekturu aplikace, kde budou představeny základní principy, které by měly být základem každého Android projektu, a poté představím doporučenou architekturu, která pomáhá těmto principům dostát. Zejména důležitá je prezentační neboli UI vrstva, které se budu věnovat rozsáhleji. Po uvedení architektury se budu zabývat nástroji a knihovnami použitými při implementaci aplikace.

3.1 Architektura

Dobrá architektura přináší mnoho výhod:

Zlepšuje udržovatelnost, kvalitu a robustnost celé aplikace Implementace čisté architektury, jako je Model-View-ViewModel (MVVM), umožňuje oddělit business logiku od uživatelského rozhraní. To usnadňuje údržbu kódu a aktualizace, protože změny v jedné části systému nemají přímý dopad na ostatní. Použitím principů SOLID a designových vzorů může být kód více modularizovaný, což znamená, že jednotlivé části aplikace jsou navrženy s ohledem na jednu zodpovědnost a vyhýbají se závislosti na modulech, které nejsou pro modul logicky relevantní. V případě této aplikace se balíčky snaží takovoutu modulovou separaci imitovat.

Umožňuje aplikaci škálovat Více lidí a více týmů může přispívat do stejné kódové základny s minimálními konflikty v kódu. S růstem projektu a přidáním nových funkcí, dobrá architektura umožňuje rozšiřování aplikace bez nutnosti kompletního přepisu kódu. Modularita a znovu-použitelnost dělá přidávání nových funkcionalit snadné.

Pomáhá při zorientování se v projektu Architektura přináší do projektu konzistenci, noví členové týmu se tak mohou rychleji zapracovat a být efektivnější v kratším čase.

Snadněji se testuje Dobrá architektura podporuje jednodušší typy, které se obecně lépe testují. Oddělení logiky od uživatelského rozhraní a dalších částí systému znamená, že jednotlivé komponenty lze testovat izolovaně. Použití dependency injection (DI) dále zvyšuje testovatelnost tím, že umožňuje snadné vkládání mock nebo fake objektů při testování.

Chyby lze zkoumat metodicky díky dobře definovaným procesům Díky modulárnímu návrhu a jasné separaci zodpovědností lze při vývoji aplikace snadněji identifikovat a řešit chyby, protože vrstvy a jejich části mají jasně definované zodpovědnosti. Logování a monitoring, integrované do architektury, ještě umožňují sledování chování aplikace a rychlou diagnostiku problémů.

[31]

3.1.1 Základní principy

Moderní Android aplikace se řídí několika základními principy. Tyto principy vznikly po létech kolektivního vývoje Android aplikací a jsou základními kameny realizace i této aplikace. S rostoucí velikostí aplikace je důležité definovat architekturu, která umožní rozšiřitelnost aplikace, zvýší její robustnost a usnadní testování.

3.1.1.1 Oddělení zodpovědností

Nejdůležitějším principem, kterého se držet, je oddělení zodpovědností nebo také oddělení zájmů (angl. "Separation of concerns"). Běžnou chybou je psaní veškerého kódu ve třídách systémových komponent aplikace. Tyto třídy založené na vstupním/výstupním rozhraní by měly obsahovat pouze logiku, která řídí dané rozhraní a interakce s operačním systémem. Udržením těchto tříd co nejstručnějších se můžeme vyhnout mnoha problémům souvisejícím s životním cyklem komponent a zlepšit testovatelnost těchto tříd. Třídy komponent nevlastníme (tj. používáme je jako API), spíše jsou to pouze spojovací třídy, které představují smlouvu mezi operačním systémem Android a aplikací. Operační systém je může kdykoli zničit na základě uživatelských interakcí nebo kvůli systémovým podmínkám, jako je nízká paměť. Pro poskytnutí uspokojivého uživatelského zážitku a zároveň snazší údržby aplikace je nejlepší minimalizovat závislost na těchto třídách. [31]

V případě prezentační vrstvy umožňuje použití architektonických vzorů lepší oddělení a organizaci kódu. Jedním z těchto vzorů je Model-View-ViewModel (MVVM). Vzor MVVM odděluje data a logiku aplikace (Model) od uživatelského rozhraní (View) s použitím třídy `ViewModel`, která slouží jako most mezi Modelem a View. Tento vzor usnadňuje údržbu kódu a poskytuje větší flexibilitu při změně v kterékoliv části kódu, což také vede ke snadnějšímu testování. Dalším aspektem oddělení starostí je správné řízení zdrojů, jako jsou databázové připojení a síťové požadavky. Tato oddělení zdrojů do samostatných služeb nebo tříd umožňují, aby každá část aplikace byla více odpovědná a tím se snížillo riziko chyb.

Dodržování principu oddělení zodpovědností tedy ve vývoji Android aplikací:

- Vede ke zvýšení kvality kódu.
- Usnadňuje rozšiřitelnost a správu.
- Usnadňuje údržbu.

Tento přístup umožňuje lépe reagovat na změny v požadavcích a technologiích, čímž zajišťuje, že aplikace zůstává aktuální a bezproblémová. Navíc při tvorbě aplikace umožňuje odložená implementační rozhodnutí, jako zvolení UI technologie, databázové technologie či třeba technologie pro komunikaci se vzdálenými zdroji a tak podobně.

3.1.1.2 Řízení UI z datových modelů

Dalším důležitým principem je řízení uživatelského rozhraní (UI) aplikace prostřednictvím datových modelů, ideálně persistentních modelů. Datové modely (může zahrnovat např. i modely týkající se Modelu a ViewModelu v prezentační vrstvě) představují data aplikace a jsou nezávislé na prvcích uživatelského rozhraní a ostatních komponentách aplikace. To znamená, že nejsou vázány na životní cyklus uživatelského rozhraní a komponent aplikace, ale budou zničeny, když se operační systém rozhodne odstranit proces aplikace z paměti.

Persistentní modely jsou ideální z následujících důvodů:

- Zajišťují, že data nebudou ztracena, pokud operační systém Android zničí aplikaci, aby uvolnil zdroje.
- Umožňují, aby aplikace pokračovala v práci i v případě nestabilního nebo nedostupného síťového připojení.

- Umožňují svižnější načítání dat.
- Umožňují snížení počtu požadavků na vzdálené servery.

Důležitým faktem v řízení UI z datových modelů je, že změny v datových modelech způsobují dynamickou aktualizaci uživatelského rozhraní. Základ architektury aplikace na třídách datových modelů zvyšuje testovatelnost a robustnost aplikace. [31]

3.1.1.3 Single Source of Truth (SSOT)

Definice nového datového typu v aplikaci vyžaduje přiřazení jednotného zdroje pravdy (angl. "Single Source of Truth"– SSOT). SSOT se stává vlastníkem těchto dat a je jediný, kdo může data měnit nebo poskytovat. Pro tento účel SSOT poskytuje data pomocí neměnného (angl. "immutable") typu (v Kotlinu třída, která má neměnné vlastnosti, značené `val`) a pro jejich modifikaci vystavuje funkce nebo přijímá události, které mohou být volány jinými typy.

Tento vzor nabízí několik výhod:

- Soustřeďuje všechny změny určitého typu dat na jednom místě.
- Chrání data před manipulací dat z jiných zdrojů.
- Umožňuje lepší sledování změn v datech, což usnadňuje odhalování chyb.

Zdrojem pravdy může být ViewModel nebo i uživatelské rozhraní. V jiných případech, kde je aplikace typu offline-first bývá zdrojem pravdy pro data obvykle databáze. [31]

Jednotný zdroj pravdy zajišťuje soudržnost a konzistenci dat v celé aplikaci. Jakmile je zdroj dat definován, všechny komponenty aplikace by měly získávat data z tohoto jednoho místa, což eliminuje duplicitu dat a potenciální nekonzistence.

3.1.1.4 Unidirectional Data Flow (UDF)

Princip jediného zdroje pravdy (SSOT) je často používán ve vývoji aplikací pro Android ve spojení s vzorem jednosměrného toku dat (angl. "Unidirectional Data Flow"– UDF). V UDF proudí stav pouze jedním směrem. Události, které modifikují data, proudí opačným směrem.

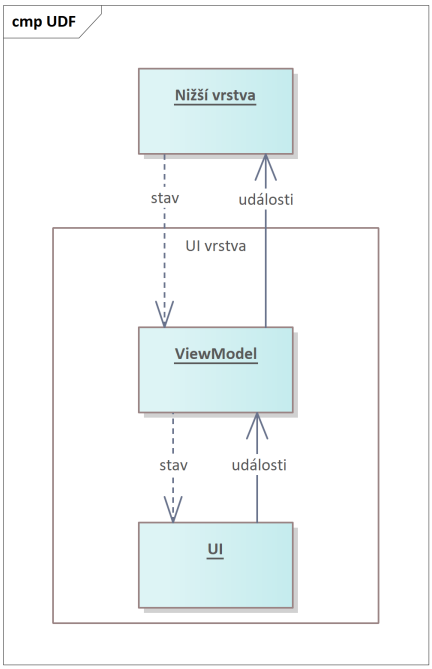
V systému Android stav nebo data obvykle proudí z nižších vrstev hierarchie (zdroje dat) do vyšších (které data konzumují). Události jsou obvykle spouštěny z vyšších vrstev, dokud nedosáhnou SSOT pro příslušný datový typ. Například data aplikace obvykle proudí ze zdrojů dat do uživatelského rozhraní. Uživatelské události, například stisknutí tlačítka, proudí z uživatelského rozhraní do SSOT, kde jsou aplikační data modifikována a vystavena v neměnném typu. Toto chování je vizualizováno diagramem 3.1.

Tento vzor lépe zaručuje konzistenci dat, je méně náchylný k chybám, snadněji se ladí a přináší všechny výhody vzoru SSOT. [31]

Důležitým aspektem tohoto přístupu je, že se jedná o transparentní a předvídatelný způsob správy dat. Jednosměrný tok dat zajišťuje, že veškeré změny v datech jsou centralizovány a kontrolovány jedním zdrojem pravdy, čímž se eliminuje možnost nekonzistencí dat, které mohou nastat při použití více zdrojů pro správu stavu aplikace. Kromě toho, tento vzor usnadňuje testování a ladění aplikací, protože můžeme snadno sledovat a predikovat tok dat a událostí v rámci celé aplikace. V neposlední řadě, jednosměrný tok dat usnadňuje integraci s architektonickými vzory, jako MVVM a technologiemi, jako jsou reaktivní programování a frameworky pro prezentační vrstvu, jako deklarativní Compose.

3.1.2 Doporučená architektura

Doporučenou architekturou od Googlu je architektura UI vrstvy a datové vrstvy s volitelnou doménovou vrstvou (viz diagram 3.2) pro zjednodušení a opakované použití interakcí mezi UI a datovou vrstvou. V



Obrázek 3.1 UDF –diagram

případě dvouvrstvé aplikace závisí UI vrstva na datové vrstvě a datová vrstva je nezávislá. V případě třívrstvé architektury závisí UI vrstva na doménové vrstvě a ta závisí na datové, která je vždy nezávislá. Touto architekturou se zabývám, protože architektura definovaná v projektu, až na malé odchylky, tuto architekturu používá.

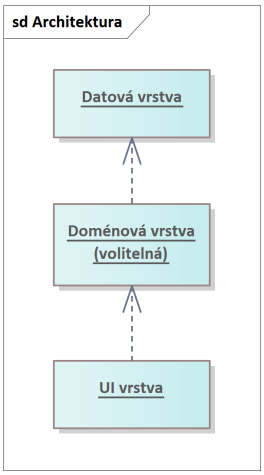
Google doporučuje využití těchto technik:

- Reaktivní a vrstvená architektura.
- Jednosměrný tok dat (UDF) ve všech vrstvách aplikace.
- Vrstva uživatelského rozhraní s držiteli stavů pro správu složitosti UI.
- Couroutines a flows.
- Dependency injection.

[31]

3.1.2.1 Datová vrstva

Datová vrstva aplikace obsahuje business logiku. Business logika tvoří pravidla, která určují, jak aplikace vytváří, ukládá a mění data [31]. V datové vrstvě se také nachází dvě podvrstvy. Spodní, nezávislá vrstva, se nazývá Zdroj Dat (angl. "Data Source"). Vrstva na ní závislá se nazývá Repozitář (angl. "Repository"). Pouze Repozitáře jsou viditelné pro doménové a UI vrstvy, Zdroje Dat nikoliv. Každý Repozitář se může skládat z více Zdrojů Dat, stejně tak jako z dalších Repozitářů. Zdroje Dat manipulují s daty a poskytují je repozitářům. Každý Zdroj Dat by měla mít na starosti práci pouze s jedním zdrojem dat, kterým může být soubor, síťový zdroj nebo místní databáze. Zdroje Dat jsou mostem mezi aplikací a systémem pro operace s daty. Repozitáře poskytují ucelený zdroj dat a rozhoduje, jak je s nimi nakládáno. Rozhodují např. jestli pro požadavek poskytnutí dat použít vzdálený API zdroj nebo lokální úložiště a jak s těmito různými



Obrázek 3.2 Architektura –diagram

zdroji nakládat. Typickou implementací v Android aplikacích je repozitář, který obsahuje vzdálený zdroj dat a lokální databázi.

Třídy úložiště jsou zodpovědné za následující úkoly:

- Vystavení dat zbytku aplikace.
- Centralizace změn dat.
- Řešení konfliktů mezi více zdroji dat.
- Abstrahování zdrojů dat od zbytku aplikace.
- Obsahují business logiku.

[31]

3.1.2.2 Doménová vrstva

Doménová je volitelná vrstva mezi datovou a UI vrstvou. Je užitečná pro zapouzdření složitých operací a znovu používaných operací, nezávislých na systémových API. V menších aplikacích může tato vrstva vytvářet nadbytečnou režii. Třídy v této vrstvě se často nazývají Případy Užití (angl. "Use Cases") a mají jen jednu zodpovědnost. Poskytují užitečnou abstrakci, jako např. `LoginUseCase`.

3.1.2.3 UI vrstva

Úkolem vrstvy uživatelského rozhraní (neboli prezentační vrstvy) je zobrazovat data aplikace na obrazovce. Kdykoli se data změní, ať už v důsledku interakce uživatele (např. stisknutí tlačítka) nebo vnějšího vstupu (např. odezvy sítě), uživatelské rozhraní by se mělo aktualizovat, aby tyto změny odráželo [31].

Uvnitř UI vrstvy se nachází dvě podvrstvy. Jednou jsou prvky UI, které se vykreslují na obrazovce a druhou držitele stavů, které uchovávají data, vystavují je uživatelskému rozhraní a zpracovávají logiku. Při tvorbě UI existují ve světě Android dvě možnosti, jak UI tvořit a renderovat. Starší Views (imperativní) způsob a novější Compose (deklarativní). Při tvorbě nových aplikací už se používá výhradně jen Compose, kde je to možné.

Existující mobilní aplikace Forendors má vytvořenou datovou i doménovou vrstvu. Proto bude UI vrstva majoritním obsahem realizace aplikace a tudíž se do této vrstvy ponoříme hlouběji.

Výpis kódu 3.1 Příklad zapouzdření stavu

```
data class PlayerUiState(
    val isPlaying: Boolean,
    val currentPosition: Long,
    val duration: Long,
)
```

Data získávaná z datové vrstvy aplikace obvykle nesplňují přímé požadavky na formát informací určených k prezentaci v uživatelském rozhraní. Tato situace vyžaduje, že část dat může být pro účely vizualizace redundantní, nebo naopak, může se ukázat jako nezbytné integrovat více rozdílných zdrojů dat s cílem vytvořit soubor informací, který je přizpůsoben zobrazovacím potřebám uživatelského rozhraní. Aby vrstva uživatelského rozhraní mohla efektivně plnit svou roli ve vztahu k datům z nižší vrstvy, je nezbytné, aby prováděla sérii specifických kroků, které umožňují transformaci a prezentaci dat v pro uživatele přívětivé formě. Těmito kroky jsou:

1. Zpracování dat aplikace s cílem jejich transformace do formátu, který umožňuje snadné vykreslení v uživatelském rozhraní. Tento proces zahrnuje selekci, filtraci a možné modifikace dat tak, aby byla zajištěna jejich relevace a přehlednost pro koncového uživatele.
2. Přeměnu těchto zpracovaných dat na konkrétní elementy uživatelského rozhraní, které jsou následně prezentovány uživateli. Tento krok vyžaduje detailní porozumění technickým možnostem a omezením použitých technologií.
3. Zpracování událostí vstupu od uživatele prostřednictvím těchto sestavených elementů uživatelského rozhraní a adekvátní reakce na tyto události v souladu s daty prezentovanými v uživatelském rozhraní.
4. Iteraci předchozích kroků tak dlouho, jak je to potřebné, s cílem udržet uživatelské rozhraní aktuální a odpovídající potřebám uživatele. Tento cyklický proces je základem pro udržení dynamického a interaktivního uživatelského prostředí, které se adaptuje na měnící se požadavky, očekávání uživatelů a zobrazování aktuálních informací.

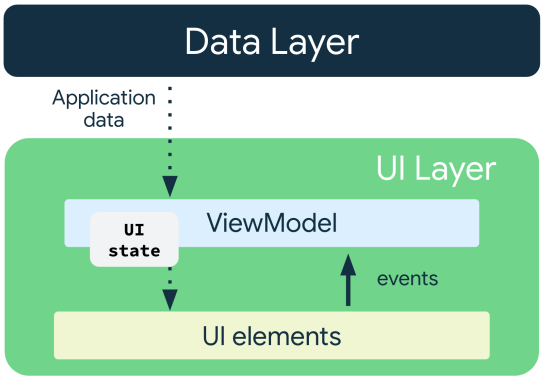
Elementy uživatelského rozhraní (UI)[1] fungují nezávisle na specifických aplikačních programovacích rozhraních (API), která mohou být pro tyto účely využívána. Tato nezávislost poskytuje flexibilitu a adaptabilitu v designu a implementaci uživatelského rozhraní. [32]

3.1.2.3.1 UI stav Krátce, pokud je uživatelské rozhraní to, co uživatel vidí, stav uživatelského rozhraní je to, co by podle aplikace měl vidět. Uživatelské rozhraní vizuální reprezentací stavu uživatelského rozhraní. Jakékoli změny stavu uživatelského rozhraní se okamžitě projeví v uživatelském rozhraní. Definice stavu uživatelského rozhraní se většinou deklaruje jako datová třídá (Kotlin), jejíž hodnoty jsou neměnné (viz kód 3.1).

Výhodou je, že neměnné objekty poskytují záruky ohledně stavu aplikace v daném časovém okamžiku (tj. že nemůže nastat nekonzistentní stav, jako výsledek přímého nastavování měnného objektu). Tím se uživatelské rozhraní může soustředit na jedinou úlohu: číst stav a podle něj aktualizovat prvky uživatelského rozhraní. V důsledku toho by se nikdy neměl měnit stav uživatelského rozhraní přímo v uživatelském rozhraní, pokud není samo uživatelské rozhraní jediným zdrojem svých dat. Porušení této zásady má za následek více zdrojů pravdy pro stejnou informaci, což vede k nekonzistenci dat a chybám.

Například pokud by byla nějaká z proměnných ze stavu uživatelského rozhraní aktualizována ve třídě Activity, tento příznak by konkuroval datové vrstvě jako zdroj stavu. Neměnné datové třídy jsou velmi užitečné pro prevenci tohoto druhu anti-vzoru.

[1] Termín ”uživatelské rozhraní”(UI) obecně odkazuje na soubor elementů, jako jsou aktivity a fragmenty, sloužící k vizualizaci dat.



Obrázek 3.3 UI vrstva – diagramová vizualizace. Získáno z webu Android Developers [32]

3.1.2.3.2 ViewModel UI by mělo benefitovat z faktu, že nemusí vytvářet a manipulovat s daty, to zařizuje datová nebo doménová vrstva. Jedinou její odpovědností je zobrazovat stav uživatelského rozhraní a konzumovat události spojené s uživatelským rozhraním, ať už vznikají ve ViewModelu nebo uživatelském rozhraní samotném. ViewModely existují, aby odstínily logika spojenou se správou stavu a UI mohlo pouze tento stav zobrazovat. Uživatelský vstup UI deleguje také správu uživatelského vstupu do ViewModelu, jestliže vyžaduje business logiku. Bez přítomnosti ViewModelu by UI nebylo pouze uživatelské rozhraní, ale i držitel stavu, jeho správce, transformátor, apod. Mimo to toto dělení zjednodušuje testování, protože můžeme ViewModel a UI testovat separátně a detailněji, s větší kontrolou. Je to dané i tím, že UI v praxi nemá žádnou závislost na doménové či datové vrstvě a je tak velmi odlehčené. Transformaci dat z nižších vrstev do UI stavu obstarává ViewModel. ViewModel tak přijímá jak aplikační data, která se mohou měnit, tak události z UI a má plný kontext pro definování logiky, která má být aplikovaná při každé události. Vztah ViewModelu a UI lze vizualizovat následujícím diagramem: 3.3.

Aplikovaný je v tomto případě princip jednosměrného datového toku (UDF). Z diagramu je vidět, že data tečou pouze jedním směrem, tedy z datové (popř. doménové) vrstvy. Opačným směrem pouze události. Na stejné bázi, jako funguje UI s ViewModelem, funguje i komunikace mezi vrstvami, kde UI vrstva přijímá data a odesílá události či požadavky.

3.1.2.3.3 Předání stavu do UI Při dodržování UDF znamená vystavení UI stav v immutable stavu a případné změny jsou aplikované zevnitř ViewModelu. S vystavením stavu vystavuje ViewModel i funkce, které lze z UI volat při různých událostech a tím docílit aktualizace stavu. ViewModel také představuje prostředí s vlastním CoroutineScope, které existuje tak dlouho, jako daný ViewModel. Je tak možné spouštět různé asynchronní či dlouhoběžící procesy, jejichž stav by při exekuci v UI nemusel být definovaný z důvodu likvidace UI elementu, který by mohl být držitelem dané operace. Po dokončení kteréhokoliv procesu, který doběhl ve scopu ViewModelu může být aktualizován stav a UI se podle toho uzpůsobí. Pro vystavení stavu se hodí struktura, která bude tento stav držet a udřovat kanál mezi ViewModelem a UI, kde je efektivní vytvářet pozměněné verze stavu a kde na tuto změnu může UI okamžitě reagovat. Okamžité reakce lze dosáhnout primitvními cykly, ale existují sofistikovanější způsoby, jako např. Kotlin Flows, kde koncový odběratelé přímo konzumují nová data, ale dělají to efektivně použitím Flows, která nijak neblokují vlákna. Může se dít, že ve ViewModelu dojde k emitaci nového stavu, který je identický tomu předchozímu. UI stav na to okamžitě zareaguje a aktualizuje UI. To je ale zbytečné a lze se tomu vyhnout variantou Flows, jako je StateFlow. StateFlow při změne drženého stavu tento stav emituje pouze pouze v případě, že je tento stav odlišný od předchozího. Pokud není, stav neemituje a UI je ušetřeno nadbytečné aktualizace UI prvků. StateFlow má navíc výhodu, že odběratel může jeho stav číst kdykoliv a ne jen odebírat nové emise, protože StateFlow si vždy drží nejnovější stav. To v praktickém využití znamená, že UI se může aktualizovat, kdykoliv potřebuje, ne jen na základě změny UI. Tedy např. při změně konfigurace, která způsobí znovuvykreslení UI. StateFlow tedy jako způsob vystavení stavu

Výpis kódu 3.2 Příklad finálního transformovaného UI stavu

```
sealed class PlayerUiState {
    data object InitialLoading : PlayerUiState()
    data object Error : PlayerUiState()
    data class Ready(
        val player: Player,
        val isSeeking: Boolean,
        val playerState: MediaPlayerUiState,
        val episodeDetailsState: EpisodeDetailsUiState,
    ) : PlayerUiState()
}
```

Výpis kódu 3.3 Příklad vystavení UI stavu

```
private val viewModelState = MutableStateFlow(LoginViewModelState())
internal val uiState: StateFlow<LoginUiState> = viewModelState
    .combine(getLoggedStatusFlow()) { s, loggedInStatus ->
        s.toUiState(loggedInStatus)
    }.stateIn(
        scope = viewModelScope,
        started = SharingStarted.WhileSubscribed(5_000),
        initialValue = LoginUiState(),
    )
```

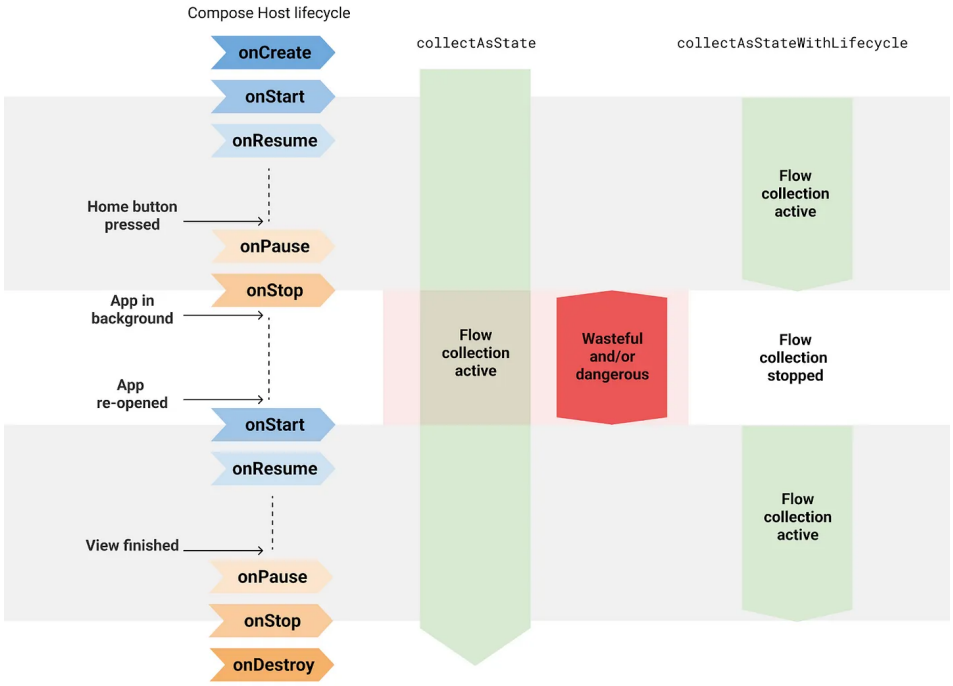
představuje UDF-kompatibilní, reaktivní přístup ke stavu, kde UI nemusí implementovat téměř žádnou logiku spojenou se správou stavu.

Flows jsou obecně v Android architektuře využívající UDF velmi používáné. Jsou používané nejen jako kanál mezi ViewModelem a UI v UI vrstvě, ale i mezi vrstvami. Ve ViewModelu tak lze využívat tohoto faktu, že z nižších vrstev můžeme informace přijímat stejným způsobem, jako ho vystavujeme do UI. Lze tedy například kombinovat Flows z nižších vrstev s vnitřním Flow ViewModel stavu a vystavit jen finální UI stavový Flow kombinující všechny vnitřní Flows. Pro reprezentaci stavu UI jako uceleného celku, odstínění komplexity a čtení hodnot, které mohou být prázdné lze UI stav v tomto finálním Flow transformovat do různých typů stavů, jako stav načítání nebo chybného stavu, čehož lze velmi dobře docílit využím Sealed třídy (viz kód 3.2).

Vystavení stavu lze uzpůsobit tak, aby výsledný StateFlow přestal pracovat a odebírat z Flows z nižších vrstev v případě, když neexistuje nikdo, kdo by daný stavový Flow odebíral. StateFlow totiž v základu pracuje stále, bez ohledu na počet odběratelů. Použitím funkce stateIn lze docílit toho, že StateFlow přestane vykonávat práci, když neexistuje žádný odběratel. Při praktickém použití se definuje i timeout, který umožňuje zrušení až několik sekund po odhlášení posledního odběratele. Lze se tím vyhnout zbytečné práci na pozadí, při rychlé změně stavu odebírání odběratelem, což se děje např. při změně konfigurace, což na mobilních zařízeních znamená rotace obrazovky a tedy následné znovu-sestavení UI a znovu-odebirání stavu. Jednou z forem finálního vystavení stavu může vypadat jako ukázka kódu 3.3.

3.1.2.3.4 Odběr stavu v UI V UI je pak odběr stavu z ViewModelu jednoduchý, stačí referencovat property ViewModelu se stavem a za využití vhodných terminálních operátorů či jiných implementací ho konzumovat. V případě použití klasického Kotlinu bez Compose by se hodilo použít metodu `collect` či její variantu.

V Compose existuje separátní pojem stavu (”State”), který je specifický ke Compose. Zkrácně, při změně Compose stavu, se aktualizuje ta část UI, která na daném stavu závisí. Stavů v Compose může být více, např. stavy týkající se pouze UI. Proto existuje v Compose funkce pro odběr stavu ze StateFlow, jmenující se `collectAsState`. Tato funkce odebírá a transformuje data na Compose stav. Použitím této jediné funkce tak dokážeme odebírat stav v Compose a nechat UI aktualizovat vždy, když přijde



Obrázek 3.4 Diagram porovnání přístupů odběru UI stavu. Získáno z článku Manuela Viva dostupného na portálu Medium [33]

nový stav. Toto se ale děje permanentně, dokud existuje životní cyklus pro dané UI.

UI se ale bohužel nevyhne logice se správou odběru UI stavu, minimálně pokud chceme brát v potaz životní cyklus UI. UI by nemělo odebírat stav, když není zobrazováno uživateli, aby zbytečně neplýtvalo zdroji aplikace. Zde přichází vhod funkce `collectAsStateWithLifecycle`. V základu `collectAsStateWithLifecycle` používá `Lifecycle.State.STARTED` pro nastartování a zastavení odběru stavu.

Rozdíl mezi klasickým `collectAsState` a `collectAsStateWithLifecycle` lze vidět v diagramu 3.4. S použitím dané funkce je komplexita odběru stavu s ohledem na životní cyklus nakonec zapouzdřena do funkce a odběr je tak snadný. Příklad, jak použít `collectAsStateWithLifecycle` k odběru stavu `uiState` typu `StateFlow`, které vystavil ViewModel, je v ukázce kódu 3.4.

Výpis kódu 3.4 Příklad odběru stavu

```
@Composable
fun VideoPlayerScreen(
    onCreatorClick: (creatorHandle: String) -> Unit,
    onBack: () -> Unit,
    modifier: Modifier = Modifier,
    viewModel: VideoPlayerViewModel = koinViewModel(),
) {
    val uiState by viewModel.uiState.collectAsStateWithLifecycle()
    ...
}
```

3.1.2.4 Správa závislostí

Pro získání závislostí pro třídu lze prostřednictvím dvou přístupů.

- Dependency Injection (DI): Umožňuje definovat závislosti, které v runtime dodá jiná třída.
- Service Locator: Umožňuje získaní instancí závislostí skrze registr.

[31]

Hlavním rozdílem je, že v případě použití Service Locatoru se vytváření instancí musí manuálně nadefinovat, zatímco v DI se závislosti konstruují automaticky.

Google doporučuje DI framework Hilt, který automaticky konstruuje objekty procházením stromu závislostí, poskytuje záruky závislostí při kompilaci a vytváří kontejnery závislostí pro třídy frameworku Android. Hiltu ale bývá často vytýkána komplexita. Dalším frameworkem pro správu závislostí, který není tak komplexní, ale zato je to Service Locator, je Koin. Koin si zakládá na pragmatičnosti a jednoduchosti. V mobilní aplikaci Forendors je používaný Koin a tak byl Koin zvolen i pro Android TV implementaci.

3.2 Technologie

Nyní se budu zabývat nástroji a knihovnami použitými při implementaci aplikace. Zvláštní pozornost bude věnována knihovně pro tvorbu UI – Compose – a jejím aplikacím pro vývoj aplikací na TV.

3.2.1 Gradle

Gradle je základem automatizace sestavení při vývoji systému Android a využívá doménově specifický jazyk (DSL) Kotlin či Groovy pro definování konfigurací projektu. Android poskytuje vlastní rozšíření Gradlu, kde lze definovat verze SDK, podpisy nutné pro produkční variantu aplikace, kompilační možnosti a jiné. Proces sestavení Gradle v systému Android je posloupnost úloh prováděných za účelem kompilace zdrojového kódu, správy závislostí, spuštění testů a zabalení binárních souborů. Tento proces lze konfigurovat prostřednictvím souborů `build.gradle` přítomných na úrovni projektu a modulu aplikace, což umožňuje přizpůsobit sestavení požadavkům projektu či individuálních modulů.

Fázemi sestavení jsou:

1. Inicializace – Gradle určí, které projekty jsou součástí sestavení, a vytvoří pro každý z nich instanci projektu.
2. Konfigurace – Spustí se skripty sestavení. Během této fáze se nakonfigurují závislosti a další nastavení úloh.
3. Provedení – V této fázi se sestavení sestaví z několika částí: Gradle určí podmnožinu úloh, které se mají spustit, na základě voleb příkazového řádku a grafu závislostí úloh. Díky podpoře inkrementálního sestavování v Gradle se spustí pouze nezbytné úlohy, které je třeba provést.

Jednou z klíčových předností systému Gradle je jeho efektivní systém správy závislostí, který zjednodušuje deklarování externích knihoven a správu jejich životního cyklu. Závislosti se deklarují v projektovém souboru `libs.version.toml` pomocí stručné syntaxe, která uvádí ID skupiny knihovny, ID artefaktu a číslo verze. Tento soubor funguje jako centrální soubor závislostí pro celý projekt. V `build.gradle` na úrovni aplikačního modulu se pak tyto závislosti referencují. Gradle závislosti vyřeší z úložišť, jako je JCenter nebo Maven Central, a zajistí, že projekt bude mít v době sestavení všechny potřebné komponenty.

Gradle zavádí koncept variant sestavení, což jsou kombinace typů sestavení (debug, release) a variant produktu (přizpůsobené verze aplikace). To umožňuje definovat více verzí aplikace (např. vývojovou, staging, produkční) s různými konfiguracemi (koncové body API, klíče) beze změn ve zdrojovém kódu.

Výpis kódu 3.5 Inicializace Koinu

```
private fun loadKoinModules() {
    val configuration = BuildConfigurationProvider.getConfiguration()
    MultiplatformKoinFactory.koinDI.start(configuration) {
        listOf(
            module {
                single { this@ForendorsTVApp } binds arrayOf(
                    Context::class,
                    Application::class,
                )
            },
            appModule,
        )
    }
}
```

Výpis kódu 3.6 Koin aplikační modul

```
val appModule = module {
    includes(
    mainModule,
        sharedModule,
        loginModule,
        playersModule,
        searchModule,
        creatorProfileModule,
        webViewModule,
    )
}
```

V případě TV aplikace Forendors této možnosti využívám a testuji proti testovacímu i produkčnímu backendu. Rozšiřitelnost systému Gradle umožňuje psát vlastní úlohy a zásuvné moduly a nabízí možnost automatizovat prakticky jakýkoli aspekt procesu sestavení nebo nasazení. Tato vlastní logika sestavení může zahrnovat generování kódu, statickou analýzu kódu nebo jakékoli další kroky před a po kompilaci, které jsou pro projekt nezbytné. [34]

3.2.2 Koin

Koin je lehký, pragmatický framework pro Dependency Injection (DI) navržený speciálně pro vývojáře Kotlinu. S využitím funkcí jazyka Kotlin nabízí Koin jednoduchý a intuitivní způsob správy závislostí v aplikacích pro Android. Koin funguje na několika základních principech, které jej odlišují od ostatních frameworků pro vstřikování závislostí. Je od základu vytvořen pro jazyk Kotlin a plně využívá jeho vlastností, jako jsou rozšiřující funkce, lambdy a možnosti jazyka DSL. Na rozdíl od některých jiných DI frameworků využívá Koin k řešení závislostí funkční schopnosti jazyka Kotlin a vyhýbá se reflexi. Tento přístup přispívá k lepšímu výkonu a předvídatelnosti. V neposlední řadě klade Koin důraz na minimální konfiguraci, což umožňuje začít bez rozsáhlého boilerplate nastavování. Celé nastavení Koinu v aplikaci je zavolání funkce 3.5 na úrovni aplikace. `appModule` v dané funkci drží všechny ostatní moduly aplikace (viz 3.6). Individuální moduly pak už definují, jak vytvořit jednotlivé instance (viz 3.7).

Za běhu poskytuje Koin kontejner, který obsahuje všechny definice uvedené v modulech. Tento kontejner je zodpovědný za plnění požadavků na závislosti. Lze si všimout, že Koin nabízí specializované funkce pro vývoj v systému Android, včetně snadné integrace s komponentami systému Android, jako jsou aktivity, fragmenty a ViewModels, což usnadňuje bezproblémový vývoj. Koin samozřejmě, jako

Výpis kódu 3.7 Koin modul s definicemi tvorby instancí

```
val creatorProfileModule = module {
    viewModelOf(::CreatorProfileViewModel)
    viewModel { (creatorHandle: String) ->
        CreatorAllFeedViewModel(
            creatorHandle = creatorHandle,
            getCreatorFeed = get(),
            stringProvider = get(),
        )
    }
}
```

Výpis kódu 3.8 Příklad zobrazení obrázku s knihovnou Coil

```
AsyncImage(
    modifier = Modifier
        .size(32.dp)
        .clip(CircleShape)
        .align(Alignment.CenterVertically),
    model = userAvatarUrl ?: R.drawable.ic_placeholder_avatar,
    contentDescription = "User avatar",
)
```

ostatní DI frameworky, umožňuje deklarovat různé životní cykly instancí [35].

3.2.3 Coil

Coil (Coroutine Image Loader) je výkonná a efektivní knihovna pro načítání obrázků v Android aplikacích, navržená tak, aby zjednodušila získávání, zobrazování obrázků a jejich ukládání do cache. Postavená na Kotlin coroutines a využívající moderní technologie, je Coil optimalizová pro vývoj v Kotlinu a Androidu, a nabízí intuitivní API, které se snadno integruje do Android aplikací, ať už s UI postaveným ve Views nebo Compose. Coil je podle slov vývojáčů dané knihovny:

- Rychlý – Coil provádí řadu optimalizací, včetně ukládání do mezipaměti a na disk, převzorkování obrázků v paměti, automatického pozastavení/zrušení požadavků a dalších.
- Lehký – Coil přidává do APK 2000 metod (u aplikací, které již používají OkHttp a Coroutines), což je srovnatelné s Picassem a výrazně méně než Glide a Fresco, které jsou dalšími možnostmi pro práci s obrázky.
- Snadno použitelný – API Coil využívá vlastnosti jazyka Kotlin pro jednoduchost a minimální množství boilerplatu.
- Moderní – Coil je Kotlin-first a používá moderní knihovny včetně Coroutines, OkHttp, Okio a AndroidX Lifecycles.

[36]

V srdci designu Coil jsou Kotlin coroutines, které umožňují provádět operace načítání obrázků asynchronně, efektivně spravovat úlohy na pozadí a snížit složitost systémů založených na callbacích. Tento přístup založený na coroutines usnadňuje zvládání více úloh načítání obrázků současně bez blokování hlavního vlákna. Praktické využití Coilu pro zobrazení obázku v Compose vypadá viz kód 3.8. `AsyncImage` je v tomto případě Coil funkce.

Výpis kódu 3.9 ExoPlayer v Compose

```
AndroidView(
    modifier = modifier,
    factory = {
        PlayerView(context).apply {
            player = uiState.player
            layoutParams = FrameLayout.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT,
            )
            useController = false
        }.also {
            onPlayerViewCreated()
        }
    },
    update = { it.player = uiState.player },
    onRelease = { onPlayerViewRelease() },
)
```

Výpis kódu 3.10 Ukázka ExoPlayer callbacku

```
player.addListener(
    object : Player.Listener {
        override fun onIsPlayingChanged(isPlaying: Boolean) {
            super.onIsPlayingChanged(isPlaying)
            emitStateImmediately()
        }
    }
)
```

3.2.4 ExoPlayer

ExoPlayer jsem již zmínil v kapitole návrhu. ExoPlayer narozdíl od Coilu nemá nativní podporu pro Compose. To není velký problém, protože Compose je s Views kompatibilní a lze do UI složené z Compose prvků vkládat i Android Views prvky. ExoPlayer lze do Compose zakomponovat pomocí Compose funkce `AndroidView`, jako je vidět v kódu 3.9. `AndroidView` spolu s `AndroidViewBinding` a `ComposeView` jsou totiž totiž API umožňující interoperabilitu mezi systémy uživatelského rozhraní Views a Compose. Konkrétně `AndroidView` přijímá lambdu, která vrací `View`. K rekompozici dochází, kdykoliv se změní stav, který je čten uvnitř callbacku `update` argumentu. Tento `update` argument je volaný, když je dané `View` poprvé zobrazené. [37]

Problém přichází se správou stavu. Protože si přehrávač drží stav vnitřně a nemůžeme jeho stav vlastnit, ani ho odebírat, protože ExoPlayer takové API nenabízí, je nutné implementovat callbacky a kontinuální sledování stavu přehrávače, aby ViewModel držel aktuální stav přehrávače a UI prvky, jako překryv s ovládacími prvky, mohl na stav reagovat. Např. pozastavení přehrávání neznamená pouze změnu jedné hodnoty ve stavu, ale zavolání metody na přehrávači, který přehrávání pozastaví. K zachování jednotného zdroje pravdy není dobré ihned měnit stav, ale poslouchat na přehrávači na změnu jeho vnitřního stavu. Jinými slovy implementujeme callback, který bude na změnu tohoto vnitřního stavu reagovat změnou UI stavu, viz ukázka kódu 3.10.

Naneštěstí ExoPlayer neposkytuje žádné API, co se týče aktuální délky trvání přehrávání. To je v našem případě nutné pro zobrazení aktuální pozice přehrávání na časové ose v UI. Řešením v mém případě bylo vytvořit nekonečný cyklus, který bude s prodlevami emitovat stav (viz kód 3.11). To umožní

Výpis kódu 3.11 Způsob monitorování stavu přehrávače

```
private fun monitorPlayerState() {
    viewModelScope.launch {
        while (true) {
            delay(MONITOR_PLAYER_MS_DELAY)
            emitState()
        }
    }
}
```

Výpis kódu 3.12 Mapování stavu přehrávače do UI stavu

```
fun Player.toUiState(isPlaybackEnded: Boolean): MediaPlayerUiState {
    val duration = if (duration == C.TIME_UNSET) 0 else this.duration
    return MediaPlayerUiState(
        isPlaying = this.playWhenReady,
        currentPosition = this.currentPosition,
        totalDuration = duration,
        isEnded = isPlaybackEnded,
    )
}
```

mít téměř aktuální informace s maximální časovou deltou danou délkou prodlevy. S coroutines je nekonečný cyklus, který používá suspendovací funkci, bezpečný, co se týče blokování vláken.

Abychom se vyvarovali držení mutable stavu, který by držel identické informace, co drží přehrávač a přímým změnám těchto stavů pocházejících odjinud než přímo z přehrávače, existují ve ViewModelu přehrávače dva stavy – mutable ViewModel stav a immutable UI stav. Informace o stavu přehrávače se nachází pouze v UI stavu. UI stav přehrávače má jediné místo v kódu, kde je aktualizován a to při mapování z ViewModel stavu na UI stav. Jedině tak lze dosáhnout nemožnosti upravovat UI stav přehrávače jinak, než mapováním z vnitřního stavu přehrávače (viz kód 3.12) a dosáhnutí konzistence stavu. Znamená to ale, že uvnitř ViewModelu musíme emitovat stav, který je na pohled stejný, protože tento sta drží pouze instanci přehrávače, ale ta se nemění. Není proto pro stav ViewModelu vhodné použít StateFlow, protože neemituje stavy, jejichž hodnoty se nezměnily. StateFlow je implementací SharedFlow. SharedFlow je konfigurovatelný a lze např. emitovat stavy jejichž hodnoty jsou identické. SharedFlow je proto ideální pro využití emise stavu s přehrávačem a následné transformace do UI stavu při každé emisi tohoto ViewModelového stavu. Co se týče vystaveného Flow s UI stavem, tak ten je stále StateFlow, takže pokud přijde ViewModel stav s přehrávačem, který má hodnoty vnitřního stavu stejné, jako ty předchozí držené v UI stavu, tak nedojde k emisi a UI tak nemusí být aktualizováno. Posledně, protože by takovéto časté emise stavu a následné transforamce do UI stavu mohly působit výkonostní problémy, je stav emitován neblokujícím způsobem za použití suspend funkce. Dohromady tyto implementační části umožňují centralizovanou a v rámci možnosti reaktivní extrakci stavu z přehrávače.

3.2.5 Compose

Compose je moderní toolkit od Google pro tvorbu nativních UI v Android aplikacích. Postavený na deklarativním přístupu k programování, Compose umožňuje snadno a rychle vytvářet interaktivní uživatelská rozhraní s ménším množství kódu a vysokou úrovní opětovné použitelnosti komponent. Oproti systému Views se tím lzevyhnout celým třídám bugů. V systému Views bylo například potřeba pro implementaci seznamu implementovat třídy pro správu daného seznamu. V Compose lze pouze použít Composable funkci listu a definovat, jak mají prvky seznamu vypadat. Od svého představení začal být Compose graduálně adoptován, až se stal preferovaným způsobem vývoje UI pro Android, nabízející intuitivní API

Výpis kódu 3.13 Příklad Composable funkce

```
@Composable
fun EmailField(
    email: String,
    onEmailChange: (email: String) -> Unit,
    modifier: Modifier = Modifier,
) {
    ForendorsTextField(
        modifier = modifier,
        value = email,
        onValueChange = onEmailChange,
        ...
    )
}
```

a integraci s ostatními Jetpack knihovnami a nástroji. Compose využívá deklarativní paradigma pro definici UI, což umožňuje vývojářům popsat, jak by UI mělo vypadat a jak by se mělo chovat, zatímco systém se stará o aktualizace a rendering. Compose je navržen s využitím Kotlinu, což přináší do vývoje UI výhody Kotlinu, jako jsou coroutines pro asynchronní operace, extension funkce, a jeho celková čistota a expresivita. Compose umožňuje vytvářet opakovaně použitelné, modifikovatelné a bezstavové UI komponenty, které lze snadno sdílet a znovu použít napříč aplikací, což zvyšuje modularitu a snižuje duplikaci kódu. V Compose lze každý prvek naprogramovat tak, aby byl bezstavový, tj. nedržel žádný vlastní stav, a stav byl explicitně předáván formou funkčních parametrů. Tímto způsobem dodržíme princip SSOT, modifikovatelnost a zonvupoužitelnost komponenty. Zároveň je Compose navržen tak, aby dobře spolupracoval s ostatními knihovnami v ekosystému Jetpack, včetně Flows, ViewModelů a Navigation, což umožňuje snadnou integraci funkcí a architektonických vzorů, jako třeba UDF – příjímání stavu a vracení událostí.

3.2.5.1 Struktura

Vývoj UI s Compose začíná definováním Compose funkcí, které reprezentují části UI. Tyto funkce deklarují UI strom pomocí kombinace vestavěných a vlastních Compose funkcí. Příkladem nechť bude Composable funkce `EmailField` 3.13.

Tato Compose funkce:

- Je anotována `@Composable`. Tato anotace informuje Compose překladač, že tato funkce je určena k převodu dat na uživatelské rozhraní.
- Přijímá data. Funkce mohou přijímat parametry, které umožňují logice aplikace popsat uživatelské rozhraní. V tomto případě náš widget přijímá řetězec String, aby mohl uživateli zobrazit text v textovém poli.
- Zobrazuje UI prvky. Ať už voláním jiných Compose funkcí nebo samotným vytvořením UI elementu. Zde se volá Compose funkce se zobecněným textovým polem.
- Nic nevrací. Compose funkce, které emitují uživatelské rozhraní, nemusí nic vracet, protože místo konstrukce widgetů uživatelského rozhraní popisují požadovaný stav obrazovky.
- Je rychlá, idempotentní a bez vedlejších účinků. Chová se stejně i při vícenásobném volání se stejným argumentem a nepoužívá jiné hodnoty, například globální proměnné nebo volání funkce `random()`. Zároveň popisuje UI bez jakýchkoli vedlejších účinků, jako je změna vlastností nebo globálních proměnných. Obecně platí, že všechny Compose funkce by měly být psány s těmito vlastnostmi.

Výpis kódu 3.14 Struktura deklarace stavové proměnné

```
var value by remember { mutableStateOf(initialValue) }
```

[38]

Protože se Compose funkce snaží být bezstavové, nemění přímo žádné vlastnosti či stav, pokud není inheretní k danému UI elementu. Compose funkce má stav, když udržuje svůj vlastní vnitřní stav, který se může v průběhu času měnit. Chování Compose funkcí lze ovlivnit pouze dodáním argumentů, protože se jedná o funkci a ne o třídu, která by vystavovala gettery a settery. To platí i pro chování UI elementů. Protože jsou Compose funkce psané v Kotlinu a plně ho podporují, je možné deklarovat jako parametr lambda funkce, které může Compose funkce volat při různých událostech. Už v tom lze vidět princip UDF, který je uplatňován i v architektuře Compose. Data jsou Compose funkcemi přijímány a zobrazovány. K jejich modifikaci, lze využít lambda parametru, která bude definována v nejvrchnějším UI elementu a delegována do ViewModelu. ViewModel na tuto událost zareaguje změnou stavu. Tento stav je přijímán Compose funkcemi. Takto cyklicky probíhá aktualizace stavu. Stav ale nemusí být vždy nutně držen ViewModelem. Compose funkce mohou držet své stavy, které jsou čistě spojené s daným UI prvkem.

3.2.5.2 Stav

Stav v Compose je interface `State<T>`, což je pozorovatelný typ integrovaný s Compose runtime. Kdykoliv se změní stav instance tohoto typu, dojde k rekompozici. Compose stavové instance sleduje a spouští všechny Composable funkce, které čtou daný `State<T>`, a všechny jimi volané Composable funkce, které nelze přeskočit. Popis rozhraní při spuštění Composable funkcí se nazývá kompozice. Rekompozice je tedy znovuspuštění Composable funkcí za cílem aktualizace kompozice. když se změní stav. Stav je obvykle deklarován pomocí funkce `mutableStateOf`, která vytvoří `MutableState<T>`, která dědí z `State<T>`. Stav ale nelze pouze deklarovat pomocí `mutableStateOf`, protože proměnné deklarované v Composable funkce jsou vždy při kompozici znovu inicializované. Hodnota stavové proměnné by se tak vždy resetovala na hodnotu přiřazenou při kompozici. To může být vyhovující pro nestavové proměnné nevyžadující persistenci mezi rekompozicemi. Pro persistenci hodnoty mezi rekompozicemi je dostupné `remember` API, které dokáže hodnotu proměnné uchovat v paměti. Hodnota vypočtená pomocí funkce `remember` je při počáteční kompozici uložena do kompozice a uložená hodnota je vrácena při rekompozici. `remember` lze použít k ukládání mutable i immutable objektů. Deklarace v kódu typu 3.14 by tak měla být jasnější. `by` delegátová syntaxe zpřístupňuje getter a setter pro proměnnou drženou stavem. Při referenci stavu tak při využití `by` nereferencujeme `MutableState<T>`, ale drženou hodnotu typu `T`.

Pro vytvoření Compose stavu z ViewModel Compose-agnostického `StateFlow` stavu existuje již zmíněná platformově agnostická funkce `collectAsState` a Android-specifická `collectAsStateWithLifecycle`.

Použití `remember`, ale dělá Composable stateful. Pro použití, kde je volajícímu stav jedno, toto není problém. Bezstavovost je potřeba tehdy, když volající potřebuje daný stav změnit nebo ho číst. Ze stejného důvodu je bezstavovost lepší pro testování. Transformace z stateful na stateless Composable je ale jednoduchá. Nahrazením stavové proměnné funkčními parametry. Při přesunutí stavové proměnné do parametrové hodnoty a lambdy, která bude volána při změně hodnoty, se vracíme k původní premise – UDF [39].

3.2.6 Compose na TV

Pro vývoj s Compose na TV neexsituje mnoho materiálů. Nativní funkcionalita Compose by byla dostatečná, kvůli podpoře zaměřování (angl. "focus"). Podpora pro vývoj UI pro televize existuje, co se týče Views. Existují knihovny, které dnes používají pravděpodobně téměr všechny Android TV aplikace. Pro Compose naštestí vzniká také knihovna podporující vytváření uživatelského rozhraní Android TV. Compose for TV je Googlem doporučený přístup k tvorbě Android TV UI. Není proto důvod nejít touto cestou a využít všech výhod přicházejících s moderním Compose. Snad jen fakt, že Compose for TV knihovna je v alpha fázi vydání první verze této knihovny. Znamená to, že knihovnu lze použít, ale vše je experimentální a může se v příští verzi knihovny změnit. Znamená to také, že komponenty nemusí být úplně doladěné a komponenty se budou muset dolaďovat. Při vývoji UI na televizi je potřeba zohlednit několik věcí. Nejdůležitější je navigace pomocí D-Padu. Primárním způsobem interakce s Android TV je použití D-Padu na dálkovém ovladači. Aplikace musí zajistit, aby prvky uživatelského rozhraní byly snadno ovladatelné pomocí vstupů z D-Padu, včetně správy zaostření a vizuální zpětné vazby pro zaměřené položky. Druhou je větší prostor na obrazovce, protože televizní obrazovky jsou podstatně větší než obrazovky mobilních zařízení nebo tabletů, což umožňuje umístit více obsahu, ale také vyžaduje věnovat pozornost rozvržení, typografii a velikosti obrázků, aby byla zajištěna čitelnost a zapojení uživatele. Poslední věcí, na kterou se zejména hodí Compose oproti Views, je výkon. Obsah ve vysokém rozlišení a animace jsou základními prvky televizního zážitku. Aplikace Compose musí udržovat plynulý výkon, minimalizovat zpoždění a zajistit rychlé načítání. Zároveň uživatelé využívají Android TV systémů s suboptimálními hardwarovými kapacitami, ať už to jsou televize samotné, které často na výkon nedají nebo externí systémy jako Android TV stick.

3.2.6.1 Focus

V aplikaci lze jakoukoli Composable komponentu upravit na zaměřitelnou použitím modifikátoru `focusable`. Díky tomu může komponenta přijímat zaměření. Stav zaměření složky může nabývat jedné z několika hodnot, včetně Active, což znamená, že komponenta je aktuálně zaměřena; Inactive, což znamená, že může získat zaměření, ale aktuálně ho nemá; a Captured, kdy má komponenta zaměření, které nelze běžnými prostředky odebrat. Compose má výchozí způsob zpracování zaměření, který je v případě navigace na stejné úrovni, tedy mezi prvky pod stejným rodičem, správný. V ostatních případech je ale často nutné toto výchozí chování upravit. Přestože Compose má nativní podporu pro zaměřování, tato podpora obsahuje většinou pouze základní navigaci. Pro komplexnější způsoby navigace je nutné použít experimentální API (v kódu označené `@OptIn(ExperimentalComposeUiApi::class)`) nebo si logiku naprogramovat sám.

Primárním prvkem pro práci se zaměřováním je třída `FocusRequester`, která slouží k programovému vyžádání zaostření pro komponentu. Lze jej použít k nasměrování zaměření na konkrétní komponentu, často v reakci na událost.

K zaostření komponenty:

1. Deklarujeme mezi rekompozicemi persistentní instanci `FocusRequester`:
   `val focusRequester = remember  FocusRequester()` .
2. Zaregistrujeme pomocí modifikátoru `focusRequester()` u komponenty, na kterou chceme převést zaměření:
   `Button(modifier = Modifier.focusRequester(focusRequester), ...)` .
3. Na základě události zavoláme `focusRequester.requestFocus()` .

Například u overlaye v přehrávači chceme mít při jeho zobrazení vždy jako první prvek zaměřené tlačítko spuštění/pozastavení. Pro zaměření po zobrazení komponenty používám modifikátorovou funkci `focusOnAppearance` 3.15, která pomocí `LaunchedEffect` po prvotní kompozici daný prvek zaměří.

Výpis kódu 3.15 Zaměření komponenty po zobrazení

```
@Composable
fun Modifier.focusOnAppearance(): Modifier {
    val focusRequester = remember { FocusRequester() }

    LaunchedEffect(key1 = true) {
        focusRequester.requestFocus()
    }

    return focusRequester(focusRequester)
}
```

Implementace funkcionality v komponentě je pak přímočará: `PlayPauseButton(modifier = Modifier.focusOnAppearance(), ...)`.

V kódu se vedle `focusRequester` a `focusable` vyskytují další funkce spojené se zaměřováním: `focusRestorer`, který si ukládá poslední zaměřený prvek a při příchodu události zaměření zaměří tento uložený prvek, třída `FocusManager`, která umí přemisťovat zaměření nebo ho zrušit či modifikátor `onFocusChanged`, který dokáže reagovat na změnu zaměření. Často je ale třeba implementovat vlastní logiku zaměřování či jiného chování při zmáčknutí nebo držení tlačítek na ovladači. Pro zpracování takýchto interakcí ještě v Compose knihovně pro TV neexistuje žádný modifikátor a proto v kódu často používám vlastní modifikátor `handleDPadKeyEvents` 3.16. Tento modifikátor umí reagovat na definované tlačítkové vstupy v případě, že je argument akce definovaná a není `null`. V základu tato funkce používá `onKeyEvent`. Když `onKeyEvent` vrátí `false`, znamená to, že vstupová událost se nezpracovala. `onKeyEvent` v takovém případě propaguje událost výše ve stromu Composable komponent, dokud nějaká z těchto komponent nezareaguje v její implementaci deklarovaným `onKeyEvent`, který vrátí `true`. Lze tak mít v hierarchii UI komponent více modifikátorů s `handleDPadKeyEvents` a řešit na různých úrovních události relevantní pouze pro danou úroveň.

Výpis kódu 3.16 Zpracování uživatelského vstupu

```
private val DPadEventsKeyCodes = listOf(
    KeyEvent.KEYCODE_DPAD_LEFT,
    KeyEvent.KEYCODE_SYSTEM_NAVIGATION_LEFT,
    KeyEvent.KEYCODE_DPAD_RIGHT,
    KeyEvent.KEYCODE_SYSTEM_NAVIGATION_RIGHT,
)

private val defaultShouldHandleKeyEventAction = { action: Int ->
    action == KeyEvent.ACTION_DOWN
}

fun Modifier.handleDPadKeyEvents(
    onLeft: (() -> Any?)? = null,
    shouldHandleLeftAction: ((keyEventAction: Int) -> Boolean) =
        defaultShouldHandleKeyEventAction,
    onRight: (() -> Any?)? = null,
    shouldHandleRightAction: ((keyEventAction: Int) -> Boolean) =
        defaultShouldHandleKeyEventAction,
    ...
) = onKeyEvent { keyEvent ->
    val keyCode = keyEvent.nativeKeyEvent.keyCode
    val action = keyEvent.nativeKeyEvent.action

    if (!DPadEventsKeyCodes.contains(keyCode)) {
        return@onKeyEvent false
    }

    when (keyCode) {
        KeyEvent.KEYCODE_DPAD_LEFT,
        KeyEvent.KEYCODE_SYSTEM_NAVIGATION_LEFT -> {
            if (shouldHandleLeftAction(action) && onLeft != null) {
                return@onKeyEvent onLeft()?.let { true } ?: false
            }
        }

        KeyEvent.KEYCODE_DPAD_RIGHT,
        KeyEvent.KEYCODE_SYSTEM_NAVIGATION_RIGHT -> {
            if (shouldHandleRightAction(action) && onRight != null) {
                return@onKeyEvent onRight()?.let { true } ?: false
            }
        }
        ...
    }
    false
}
```

Kapitola 4
Publikace

V této sekci se budeme zabývat procesem před oficiálním spuštěním aplikace. To znamená zpřístupněním aplikace testerům, testováním aplikace k odhalení dříve nepozorovaných defektů, pochopením chování uživatelů v aplikaci a nakonec použitím nástrojů pro monitoring aplikace.

4.1 Distribuce

Pro splnění nefunkčního požadavku NP2 Přímočará dostupnost, bude nutné aplikaci dostat na obchod Google Play, pro dosažení maximálního šířky publika uživatelů Android TV a aby každý uživatel měl možnost si přímo aplikaci stáhnout. Google Play je de facto jediným oficiálním zdrojem, kde mohou uživatelé Android TV vyhledávat, stahovat a instalovat aplikace přímo na svá zařízení. Tento fakt podtrhuje význam Google Play jako primární platformy pro distribuci aplikací pro Android TV. Proces zpřístupnění aplikace na Google Play je relativně přímý. Postup se skládá z registrace na Google Play Console, nahrání APK souboru aplikace, nastavení detailů produktové stránky a splnění všech požadavků a doporučení Google Play pro uveřejnění aplikací. Jednou z vlastností, které Google Play Console nabízí, je možnost vytvoření testovacího kanálu. Tato funkce umožňuje zpřístupnit aplikaci selektivně skupině pozvaných testerů předtím, než je uvolníme širší veřejnosti. Tento přístup nabízí cennou příležitost získat zpětnou vazbu a identifikovat případné problémy, které je potřeba řešit, což zvyšuje kvalitu aplikace před jejím oficiálním spuštěním a prioritizovat implementační nedostatky.

Testovací kanál na Google Play Console je navržen tak, aby poskytoval flexibilní kontrolu nad tím, kdo může aplikaci stahovat a testovat. Můžeme snadno pozvat testery prostřednictvím e-mailu a spravovat jejich přístup k aplikaci. Tato metodika usnadňuje iterativní vývoj a testování aplikací pro Android TV. Zpřístupnění aplikace na Google Play a využití testovacího kanálu jsou kroky k zajištění otestování a distribuci aplikace na platformě Android TV. Poskytují nejen nezbytné prostředky pro dosažení cílových uživatelů, ale také zpřístupňují nástroje pro zajištění kvality a stability aplikace, což jsou nezbytné komponenty pro vytvoření úspěšné aplikace.

4.2 Monitorování

Monitorování aplikace je procesem, který umožňuje vývojářům a vedení projektu sledovat a zajišťovat bezproblémový chod a vysokou dostupnost aplikací na základě kvantitativních statistik jak pro pordukční, tak i testovací prostředí. Tento proces není omezen pouze na sledování výkonu a dostupnosti; zahrnuje také detekci a řešení problémů, sledování bezpečnostních hrozeb a optimalizaci uživatelské zkušenosti. Efektivní monitorování poskytuje hluboký vhled do chování aplikace v reálném čase, což umožňuje rychlé identifikování a řešení jakýchkoli problémů, ještě než ovlivní koncové uživatele. Důležitou součástí

monitorování je také sledování logů, které mohou poskytnout cenné informace o chování aplikace a pomoci při diagnostice problémů. Logy mohou odhalit chyby v kódu, problémy s konfigurací a další otázky, které by mohly způsobit problémy pro uživatele.

V dnešní době existuje mnoho nástrojů a služeb určených k monitorování aplikací, které nabízejí různé funkce, včetně automatického sledování, vizualizace dat a upozornění v reálném čase. Tyto nástroje mohou pomoci automatizovat a zjednodušit proces monitorování, umožňující soustředit se na analýzu dat a optimalizaci aplikací. Monitoring pomáhá zajistit, že aplikace jsou spolehlivé, výkonné a schopné poskytovat kvalitní uživatelskou zkušenost. Monitorování také umožňuje lépe porozumět, jak je aplikace používána, což může informovat o budoucím vývoji a inovacích, zajišťující, že aplikace zůstanou relevantní a hodnotné pro jejich uživatele. Osvědčenou a bezplatnou službou pro monitoring obecných informací o používání aplikace a informací o pádech aplikace je Firebase a jejich služby Analytics a Crashlytics.

4.2.1 Firebase Analytics

Firebase Analytics je nástroj integrovaný do Firebase, který umožňuje shromažďovat užitečná data o tom, jak uživatelé interagují s naší aplikací a také poskytovat agregovaná data o uživatelích, resp. jejich zařízeních a zacházení s aplikací, třeba jako statistiky o instalacích a odinstalacích. Díky tomuto nástroji můžeme získat hlubší porozumění chování uživatelů, což je důležité pro optimalizaci uživatelské zkušenosti a zvyšování uživatelské angažovanosti. Firebase Analytics automaticky sbírá určité události, jako jsou spuštění aplikace, aktualizace, chyby a mnoho dalších, ale vývojáři mohou také definovat a sbírat vlastní události, které jsou specifické pro jejich aplikaci. Data shromážděná pomocí Firebase Analytics mohou být využita k řadě účelů. Například analýza chování uživatelů může pomoci identifikovat funkce, které jsou nejpoužívanější, nebo naopak ty, které uživatelé ignorují. Tato poznatky mohou vést k lepšímu rozhodování při prioritizaci vývojových úsilí. Kromě toho, segmentace uživatelů podle jejich chování nebo demografických charakteristik umožňuje optimalizovat aplikaci pro potřeby dané audience, což může zvýšit angažovanost a udržitelnost uživatelů.

Další využití dat zahrnuje optimalizaci uživatelského rozhraní a zlepšování uživatelské zkušenosti na základě pochopení, jak uživatelé navigují v aplikaci a jaké obtíže mohou při používání aplikace zažívat. Díky analytickým datům můžeme také identifikovat a řešit technické problémy, jako jsou chyby aplikace nebo problémy s výkonem, což vede k vyšší spolehlivosti a plynulosti aplikace. Firebase Analytics poskytuje cenné informace, které mohou pomoci aplikacím vyčnívat tím, že nabídnou lepší uživatelskou zkušenost, vyšší personalizaci a efektivnější dosah k cílovému publiku. Využitím těchto dat mohou vývojáři nejen zlepšit současnou výkonnost aplikace, ale také strategicky plánovat budoucí rozvoj a inovace. Firebase Analytics pomáhá pochopit chování uživatelů, jako které obrazovky navštěvují a kolik času na nich stráví nebo jaká zařízení používají. Lze také sledovat data spojená s instalcemi a odinstalcemi, jako které verze se v produkci používají či retenci uživatelů. Všechna tato i další data může používat vlastník produktu k nastavení priorit pro vývoj či rozhodování o technologické stránce produktu, jako např. nastavení vyšší minimální verze Android SDK, pokud pro nižší verze Android SDK neexistuje audience.

4.2.2 Firebase Crashlytics

Firebase Crashlytics je specializovaný nástroj od Firebase, navržený specificky pro sledování a analýzu pádů aplikací. Jeho vlastností je schopnost poskytnout vývojářům detailní přehledy o příčinách pádů, což umožňuje nejen rychlou diagnostiku a opravu chyb, ale také pomáhá v předcházení budoucím problémům. Díky své integrovanosti do Firebase platformy nabízí Crashlytics plynulý způsob, jak monitorovat zdraví aplikace v reálném čase.

Co Crashlytics odlišuje od ostatních nástrojů pro sledování chyb, je jeho zaměření na minimalizaci práce potřebné k identifikaci a opravě kritických chyb. Crashlytics automaticky třídí problémy na základě závažnosti a frekvence, umožňuje vývojářům prioritizovat opravy podle skutečného dopadu na uživatele. Tento nástroj také poskytuje kontextové informace, včetně zařízení, operačního systému a verze aplikace,

kde došlo k pádu, což výrazně usnadňuje lokalizaci a řešení problémů. Využitím Firebase Crashlytics můžeme nejen snížit frekvenci a dopad pádů aplikací, ale také zlepšit celkovou kvalitu a stabilitu aplikace. S podrobnými reporty o chybách a intuitivním uživatelským rozhraním, Crashlytics zastává podstatnou roli v procesu zveřejnění a následného vývoje aplikace, poskytující potřebné nástroje pro vytváření robustnějších a spolehlivějších aplikací, což v konečném důsledku vede k lepší uživatelské spokojenosti.

4.3 Testování

V této kapitole bdue probráno testování před zpřístupněním aplikace veřejnosti.

4.3.1 Vývojářské testování

Pro splnění nefunkčního požadavku NP3 Použitelnost na všech podporovaných zařízeních, byl koupen Xiaomi Mi TV Stick s operačním systémem Android TV za méně než 1 000,-. Plynulá funkčnost na tomto zařízení zaručí splnění daného nefunkčního požadavku. Dále bylo při testování využito televizních emulátorů a reálných televizorů s operačním systémem Google a Android TV. Televizní emulátory, dostupné prostřednictvím Android Studio, umožňují rychle a efektivně testovat pokroky v implementaci. Přestože emulátory poskytují ryhclý způsob, jak testovat základní funkčnost a uživatelské rozhraní, nemohou plně nahradit testování na reálných zařízeních kvůli odlišnostem v hardware a chování zařízení. Testování na reálných zařízeních s operačním systémem Google a Android TV je tedy nezbytné pro zajištění nejvyšší kvality aplikace. Reálná zařízení poskytují nejautentičtější zkušenost z hlediska výkonu aplikace, ovladatelnosti a interakce s uživatelským rozhraním, jelikož je při jejichž použití dostupné dálkové ovládání. Celkově, kombinace testování na populárních streamingových zařízeních jako Xiaomi Mi TV Stick, využívání televizních emulátorů pro rychlé iterace a testování na reálných zařízeních s Android TV pokryje naše potřeby pro testování na zařízeních.

4.3.2 Xiaomi Mi TV Stick

Xiaomi Mi TV Stick, jakožto kompaktní a cenově dostupné zařízení pro streamování, je jedním z hlavních zařízení, na kterých by měla být aplikace otestována. Díky jeho popularitě mezi uživateli a schopnosti transformovat jakoukoliv televizi na chytrou televizi s přístupem k Android TV aplikacím, poskytuje Mi TV Stick užitečný benchmark pro testování. Zahrnutí tohoto zařízení do testování pomůže ověřit, že aplikace je optimalizována pro zařízení s nižším výkonem a menší pamětí. Toto zařízení bylo také použito na různých televizorech pro zjištění jiného chování na různých zařízeních. A vskutku se během testování objevili odlišnosti v zobrazení. Některé televizory měli problém s jevem zvaným *overscan*. Ten se projevuje tak, že se obsah na obrazovce po okrajích zobrazuje mimo obrazovku, obsah je tedy ořízlý. Většina moderních televizí tento problém už nemají, přesto je nutné s ním počítat, jak bylo odhaleno při testování. Obsah aplikace byl po zjištění upraven a důležitému obsahu byly dodány 5% marže od okrajů. Seznamům a podobným prvkům tato marže přidána nebyla, aby bylo zajištěno optimální UX, které nebude kompromitované na úkor zařízení mějící problémy s overscanem.

4.3.3 Uživatelské testování

Pro získání zpětné vazby od uživatelů bude provedené uživatelské testování, jak podle předem definovaných testovacích scénářů, tak podle volného používání a sběru poznatků o chování testerů a jejich poznámek. Na konci testování budou testeři dotázáni na hodnocení různých aspektů aplikace. Před průchodem testovacími případy si uživatel projde mobilní aplikaci, aby se seznámil s obsahem platformy Forendors. Navíc získáme ještě zpětnou vazbu z Forendors, pro které je aplikace vyvíjena. Pro uživatelské testování bude využito zařízení s operačními systémy Android či Google TV různých verzí. Mezi testovacími zarízeními bude i Xiaomi Mi TV Stick. Poznatky z testování budou přetvořeny v možná rozšíření a vylepšení, které budou popsané v následující kapitole.

4.3.3.1 Testovací případy

Testovací případy a jejich vztah k případům vizte na 1.1. Testovací případy pokrývají hlavní i alternativní scénáře přihlášení, abychom dostali zpětnou vazbu na každý způsob přihlášení a dávají uživateli volnost při volbě průchodu UC2 Najít obsah. Průchod uživatele pro dosažení cílů specifikovanými testovacími případy nebude uživateli předepsán. Jediná informace poskytnutá uživateli bude název testovacího případu a případná nutná dospecifikace. Z toho důvodu popsané testovací případy neobsahují scénáře. Očekávaný průběh u testovacích případů je úzce spojen se scénáři případů užití. Cílem je ověřit, jestli je splnění testovacího případů intuitivní a snadné. Proto také nebude uživateli poskytována žádná podpora, pouze připomenutí specifikace testovaného případu.

TC1 Přihlásit se s Forendors účtem Testovací případ testuje uživatelovo přihlášení do aplikace pod testovacím účtem pomocí přihlašovacích údajů (e-mail + heslo).

Obrazovky

- Přihlášovací obrazovka.

TC2 Přihlásit se s Google účtem Testovací případ testuje uživatelovo přihlášení do aplikace pomocí Google účtu. Uživatel se pro splnění pre-condition registruje svým Google účtem v mobilní aplikaci.

Obrazovky

- Přihlášovací obrazovka.
- Systémová Google přihlašovací obrazovka.

Pre-condition

- Uživatel je registrován.

TC3 Přihlásit se s Facebook účtem Testovací případ testuje uživatelovo přihlášení se do aplikace pomocí Facebook účtu. Uživatel se pro splnění pre-condition registruje svým Facebook účtem v mobilní aplikaci.

Obrazovky

- Přihlášovací obrazovka.
- Facebook přihlašovací obrazovka.

Pre-condition

- Uživatel je registrován.

TC4 Přečíst si popis specifického příspěvku Testovací případ testuje uživatelův postup při hledání popisu specifického příspěvku. Příspěvek, pod kterým má uživatel hledat popis, bude před provedením testu uživateli specifikován. Specifický příšěvek bude dohledatelný pod profilem tvůrce.

Obrazovky

- Domovská obrazovka.
- Vyhledávací obrazovka.
- Video přehrávač.
- Audio přehrávač.
- Profil tvůrce.

TC5 Přehrát specifické audio Testovací případ testuje uživatelův postup při hledání, spuštění a přehrání audio příspěvku. Příspěvek, který si má uživatel přehrát, bude před provedením testu uživateli specifikován. Specifický příspěvek bude dohledatelný mezi příspěvky na domovské stránce i pod profilem tvůrce.

Obrazovky

- Domovská obrazovka.
- Vyhledávací obrazovka.
- Video přehrávač.
- Audio přehrávač.
- Profil tvůrce.
- Audio přehrávač.

TC6 Přepnout účet Testovací případ testuje uživatelův postup při přepnutí účtu. Přepnutí účtu bude testováno přechodem mezi testovacími případy TC1, TC2 a TC3.

Obrazovky

- Přihlašovací obrazovka.
- Systémová Google přihlašovací obrazovka.
- Facebook přihlašovací obrazovka.
- Domovská nebo vyhledávací obrazovka (zde je zobrazena navigační lišta).

4.3.3.2 Testeři

Pro uživatelské testování byla vybrána malá skupina osob, kde osoby měli různou úroveň technické znalosti. Nielsenova heuristika říká, že malý počet dobře vybraných testerů může být velmi efektivní pro odhalení většiny problémů s použitelností systému. To je podloženo principem Jakoba Nielsena, že testování s pouhými čtyřmi uživateli může odhalit více než tři čtvrtiny problémů použitelnosti. [40]

Šířka úrovní technické znalosti testerů je pro testování důležitá, protože potenciální audience Android TV Forendors aplikace bude mít také širokou úroveň technické znalosti. Testování bude probíhat vždy fyzicky a odděleně od ostatních testerů. Níže je přiblížena charakteristika testerů:

- Tester A je osoba středního věku s nízkou úrovní technické znalosti běžně používající streamovací aplikace na operačním systému Android TV. Tester A bude aplikaci testovat na moderní televizi s operařním systémem Google TV.
- Tester B je osoba studující technický obor na vysoké škole a jehož technická znalost je na vysoké úrovni. Tato osoba běžně aplikace na systému Android TV nepoužívá. Tester B bude aplikaci testovat s Xiaomi Mi TV Stick.
- Tester C je osoba studující netechnický obor na vysoké škole a je středně technicky zdatný. Tester C má málo zkušeností s TV aplikacemi. Tester C bude aplikaci testovat s Xiaomi Mi TV Stick.
- Tester D je osoba studující netechnický obor na vysoké škole a je středně technicky zdatný. Tester D má málo zkušeností s TV aplikacemi. Tester D bude aplikaci testovat s Xiaomi Mi TV Stick.

4.3.3.3 Výsledky testování

4.3.3.3.1 TC1 Přihlásit se s Forendors účtem

Tester A Uživatel měl problémy s odchodem z klávesnice, aby mohl vyplnit heslo. Po stlačení tlačítka enter na klávesnici pokračoval na vyplňování hesla.

Tester B Uživatel na telefonu hledal možnost přihlášení z telefonu, ještě než začal cokoliv vyplňovat na TV. Konkrétně hledal přihlášení přes síť nebo QR kód. Nepředpokládal, že tato funkce bude existovat a tak následně na TV údaje vyplnil bez problémů.

Tester C Uživatel snadno vyplnil email a poté heslo. Poprvé se mu zobrazila zpráva, že se něco nepovedlo. Uživatel si neměl jak zkontrolovat, že zadal správně heslo, protože aplikace neumožňovala zobrazení hesla. Chtěl znovu zadat heslo, protože si myslel, že udělal chybu. Nejdříve ale zkusil znovu tlačítko přihlásit, protože internet u testera není stabilní. Druhý pokus přihlášení byl úspěšný, problém byl způsobený internetovým připojením.

Tester D Uživatel neměl se zadáváním přihlašovacích údajů a následným přihlášením žádné problémy, dokonce při vložení emailu použil návrh klávesnice a to mu doplnilo celý e-mail.

4.3.3.3.2 TC2 Přihlásit se s Google účtem

Tester A Následující chování je dané operačním systémem, který spravuje účty Google. Uživatel byl automaticky přihlášen Google účtem, který je přihlášen na televizi a již má účet Forendors a nebylo mu nabídnuta možnost přihlášní s vlastním účtem. Uživatel nevěděl, jak by se přihlásil s jiným účtem, aplikace mu takovou možnost nenabídla. Při opětovném pokusu už bylo možné se přihlásit s jiným účtem. Když uživatel začal přidávat svůj Google účet, zkusil skenování QR kódu kamerou pro přihlášení. To na zařízení pomocí aplikace fotoaparátu nefungovalo a uživatel přihlašování vzdal. Uživatel měl možnost přihlásit se ještě ručním vyplněním přihlašovacích údajů ke svému Google účtu, avšak tato možnost nebyla jasně viditelná a uživatel ji přehlédl. Uživateli byly dány rady, aby byl schopný proces přihlašování dokončit.

Po přidání Google účtu a návratu do přihlašovací obrazovky aplikace byl uživatel dočasně zmaten, že nebyl po přidání účtu přihlášen. Aplikace totiž nijak neindikovala stav ještě probíhajícího přihlašovacího procesu.

Tester B Uživatel kliknul na přihlášení přes Google a v TV systému přidal svůj Google účet. Uživatel po přidání účtu a návratu na přihlašovací obrazovku poměrně dlouho čekal, ale věděl, že přihlašovací proces probíhá a tak nebyl zmatený.

Poznámka nesouvisející s testovacím případem: Při volném používání aplikace uživatel očekával, že pokud by nebyl registrován, automaticky ho aplikace přesměruje na registraci.

Tester C Uživatel vykázal známky zmatení při návratu na přihlašovací obrazovku, když na pozadí probíhal proces přihlašování. Po chvíli byl uživatel přenešen na domovskou obrazovku.

Tester D Uživatel neměl se zadáváním přihlašovacích údajů a následným přihlášením žádné problémy, dokonce při vložení emailu použil návrh klávesnice a to mu doplnilo celý e-mail.

4.3.3.3.3 TC3 Přihlásit se s Facebook účtem

Tester A Uživatel se pokusil přihlásit QR kódem přes kameru, ale to uživatelův telefon nepodporuje a tak uživatel zvolil variantu přihlášení přes web a ruční zadání kódu. Touto možností, na rozdíl od systémového přihlášení do Google účtu, prošel uživatel bez problémů.

Tester B Uživatel rovnou začal skenovat QR kód, ani se nepodíval na možnost ručního zadání URL a kódu. Uživatel prošel přihlášením bez problémů.

Tester C Testovací scénář probíhal na nestabilní síti a obrazovka přihlášení Facebook účtem se dlouho načítala, načež se zobrazila zpráva, že se něco nepovedlo. Uživatel přešel zpět a zkusil spustit přihlášení pomocí Facebook účtu znovu. Proces se znovu nepovedl a následovalo řešení problémů s internetovým připojením.

Problém přetrvával a uživatel se zkusil přihlásit účtem Google. Přihlášení účtem Google znovu působilo zmateně, protože systémová obrazovka byla průsvitná a na pozadí se zobrazovala přihalšovací obrazovka Forendors, která neinidikovala žádný probíhající proces.

Poté, při vhodných síťových podmínkách, se při úspěšném přechodu na přihlašovací obrazovku Facebook nějakou dobu načítal QR kód. Uživatel ho naskenoval fotoaparátem a v aplikaci Facebook vyplnil kód, protože nebyl předvyplněný. Po vyplnění a potvrzení kódu na telefonu se na TV dokončil proces přihlašování a uživatel byl přenesen na domovskou obrazovku.

Tester D Uživatel naskenoval QR kód pomocí fotoaparátu, na telefonu pokračoval zadáním kódu a po krátké chvíli byl přenesen na domovskou obrazovku.

4.3.3.3.4 TC4 Přečíst si popis specifického příspěvku

Tester A Uživatel přešel téměř okamžitě do vyhledávání a začal hledat tvůrce. Uživatel vpisoval celé jméno, protože se mu průběžně nenačetly výsledky a začal zadávat i klíčová slova příspěvku, což mu ve výsledcích nic nezobrazilo. Začal tedy odznovu a napsal jen celé jméno. Uživatel úspěšně přešel na profil tvůrce a začal hledat příspěvek.

Poznámka nesouvisející s testovacím případem: Uživatel si chtěl příspěvek zobrazit nebo přehrát, protože popis naznačoval další obsah. Uživatel nechápal, k čemu by se zobrazovaly příspěvky bez obsahu.

Tester B Uživatel rovnou přešel do navigační destinace vyhledávání a začal vpisovat název příspěvku. Obrazovka umí ale hledat pouze tvůrce, takže uživatel pouze čekal a nevěděl, jestli aplikace vyhledává nebo proč vůbec se nezobrazuje žádný výsledek. Aplikace totiž přestala zobrazovat text načítání a nezobrazovala nic. Po delší době, začal vyhledávat tvůrce. Uživatel napsal celé jméno tvůrce. Po přechodu do profilu tvůrce začal uživatel snadno hledat mezi příspěvky a příspěvek našel.

Tester C Uživatel přešel do vyhledávací obrazovky a začal vpisovat název příspěvku. Uživateli se nezobrazil žádný výsledek a tak uživatel smazal z vyhledávání název příspěvku a zadal jméno tvůrce. Tvůrce se zobrazil mezi výsledky a uživatel přešel na jeho profil. Na profilu se začal probírat příspěvky a příspěvek s perexem nalezl.

Tester D Uživatel přešel do obrazovky vyhledávání a začal vpisovat jméno tvůrce. Po vepsání celého jména mu byl zobrazen výsledek a uživatel přešel do profilu tvůrce. Na profilu tvůrce začal hledat daný příspěvek podle názvu. Uživatel příspěvek našel. Uživatel nechápal, že mu daný příspěvek nejde zobrazit nebo spustit.

4.3.3.3.5 TC5 Přehrát specifické audio

Tester A Uživatel rovnou přešel do obrazovky vyhledávače a začal zadávat název příspěvku. Uživatel napsal několik slov a pak stiskl na klávesnici enter. To mu žádné výsledky neukázalo, protože obrazovka zobrazuje jen tvůrce. Uvědomil si, že obrazovka je schopná vyhledávat jen tvůrce. Uživatel vymazal vyhledávací vstup a začal hledat mezi doporučenými personalizovanými výsledky, které se zobrazují, když je vyhledávací vstup prázdný a zde našel tvůrce.

Uživatel explicitně zmínil, že rozeznává mezi audio a video příspěvky mezi kartami příspěvků. Uživatel byl na druhou stranu nespokojen, že neměl na profilu tvůrce možnost hledat podle vstupu, podobně, jako na předešlé obrazovce vyhledávání.

Při přehrávání neměl uživatel s ovládacími prvky přehrávání problém a jednodušše přešel zpět z přehrávání pomocí tlačítka zpět.

Tester B Uživatel začal hned vyhledávat na obrazovce vyhledávání. Do vyhledávacího pole zadal pouze část jména tvůrce, načež přešel na profil tvůrce a spustil epizodu.

Navigaci v přehrávaní uživatel rozuměl. Uživatel se z přehrávače dostal snadno, pomocí tlačítka zpět.

Poznámka nesouvisející s testovacím případem: Ve video přehrávači se mu velmi nelíbil textový popis epizody. Líbilo by se mu umístění perexu v sekci pod přehrávačem, která by byla indikována ikonou

dolů. Uživatel by během či po přehrání chtěl možnost přejít na další či předešlou epizodu. Po přehrání by chtěl doporučené epizody, co přehrávat dál.

Uživatel by na domovské obrazovce chtěl vidět rozdívané epizody. U epizod by chtěl vidět, jestli epizodu viděl, jestli je rozdívaná nebo jestli ji už viděl.

Tester C Uživatel si přehrál první audio daného tvůrce a z přehrávače pomocí tlačítka profilu tvůrce přešel do profilu tvůrce. Zde našel daný příspěvek s audiem. Uživatel viděl rozdíl mezi příspěvky s audio a video obsahem.

Co se týče audio přehrávače, neměl s ním uživatel žádný problém.

Poznámka nesouvisející s testovacím případem: Uživatel neměl žádný problém ani s video přehrávačem.

Tester D Uživatel přešel do obrazovky vyhledávání a vepsal část jména tvůrce. To mu tvůrce vyhledalo a uživatel přešel na profil tvůrce. Na profilu tvůrce našel uživatel epizodu snadno.

4.3.3.3.6 TC6 Přepnout účet

Tester A Uživatel přepnul účet snadno.

Tester B Uživatel našel možnost odhlášení okamžitě. Odhlášení a přihlášení proběhlo bez problémů.

Tester C Uživatel snadno našel možnost odhlášení a přihlásil se jiným účtem.

Tester D Uživatel našel možnost odhlášení okamžitě a začal se přihlašovat jiným účtem.

4.3.3.3.7 Shrnutí

Pro vizualizaci jsem výsledky testů zjednodušil na tři možné výstupy:

✓ Uživatel splnil zadání bez váhání či obtíží.

○ Uživatel splnil zadání, ač v průběhu váhal či se potýkal s menšími obtížemi.

× Uživatel při plnění zadání nebyl schopný se v aplikaci zorientovat nebo měl větší potíže.

Vizualizaci výslekdů lze vidět v tabulce 4.1.

Tabulka 4.1 Shrnutí průchodů testovacích případů

	Tester A	Tester B	Tester C	Tester D
TC1	○	✓	○	✓
TC2	×	✓	○	○
TC3	✓	✓	✓	✓
TC4	○	○	○	○
TC5	○	✓	✓	✓
TC6	✓	✓	✓	✓

4.3.3.4 Hodnocení aplikace testery

Testeři po průchodech testovacími případy a volném používání aplikace byli požádáni o udělení hodnocení na stupnici od 1 (nejhorší) do 10 (nejlepší) v následujících kritériích:

1. Uživatelské rozhraní a požitek (UI/UX)
2. Funkčnost a vlastnosti

3. Rychlost a stabilita

Hodnocení individuálních testerů lze vidět níže. Hodnocení uživatelů je shrnuto v tabulce 4.2. Uživatelé velmi kladně hodnotili UI a UX aplikace, za které jsem plně zodpovědný já, jako autor aplikace. O něco hůře, ale stále velmi kladně byla hodnocena rychlost a stabilita aplikace. Veškeré negativní hodnocení bylo nicméně směřováno na trvání načítání dat v aplikaci. Optimalizovat tento proces lze, co se týče vyhledávání a načítání obsahu přehrávačů, hlavně na straně backendu. Na straně aplikace lze optimalizovat zavedením cache a uživatelsky zpříjemnit načítání pomocí animací či jiných prvků, namísto statického nápisu. Nejnižší hodnocení obdržela funkčnost a vlastnosti aplikace. Opět se jedná o nutnost vynechání rozšířené implementace vyhledávání, kde by do vyhledávání byly zahrnuty i příspěvky. Nedostatek aplikace, který mohu jako autor ovlivnit, je fakt, že funckionalit je málo. Toto je ale dáno časovou náročností definovaných požadavků a rozsahem, kterou je bakalářská práce.

Celkově hodnocení vnímám velmi kladně. Pokud do hodnocení nepočítám nedostatky spojené s backendem a časovým rozsahem daným bakalářskou prací a počítám výsledky vlastní práce, tak je hodnocení perfektní.

Tabulka 4.2 Hodnocení aplikace testery

Tester	UI/UX	Funkčnost a vlastnosti	Rychlost a stabilita
Tester A	10	5	9
Tester B	9	6	6
Tester C	8	6	10
Tester D	10	10	9
Průměr	9,25	6,75	8,5

4.3.3.4.1 Tester A

1. UI/UX: 10 – Uživatel přirovnal aplikaci k Netflixu.
2. Funkčnost a vlastnosti: 5 – Uživateli vadilo vyhledávání jak na obrazovce vyhledávání, tak jeho absence na profilu tvůrce.
3. Rychlost a stabilita: 9 – Plný počet bodů nedal uživatel kvůli délce trvání načítání obsahu přehrávače.

4.3.3.4.2 Tester B

1. UI/UX: 9 – Plný počet bodů nedal uživatel kvůli perexu ve video přehrávači.
2. Funkčnost a vlastnosti: 6 – Uživatel byl s funkcemi a vlastnostmi spokojený, ale je jich na uživatelův vkus málo.
3. Rychlost a stabilita: 6 – Negativně uživatel hodnotil vyhledávání a to konkrétně, že výsledky se načítaly příliš dlouho.

4.3.3.4.3 Tester C

1. UI/UX: 8 – Uživatel explicitně zmínil, že se mu líbily přehrávače.
2. Funkčnost a vlastnosti: 6 – Uživateli vadila absence funkcionality hledání epizody.
3. Rychlost a stabilita: 10

4.3.3.4.4 Tester D

1. UI/UX: 10
2. Funkčnost a vlastnosti: 10
3. Rychlost a stabilita: 9 – Uživatel nedal plný počet bodů kvůli častému zobrazování načítání, které se zobrazovalo téměř všude.

4.3.3.5 Zpětná vazba z Forendors

Nezbytné je získání zpětné vazby také od Forendors jako koncového zákazníka. Finální podoba aplikace totiž musí vyhovovat hlavně Forendors, aby mohla být aplikace spuštěna a zařazena do portfolia aplikací Forendors. Tato zpětná vazba se nebude organizovat jako předešlé testy podle testovacích případů, ale půjde o volné procházení aplikace s představením obrazovek a funkcionalit. Cílem je sesbírat akční body, které bude nutné podstoupit, než se bude moci aplikace vydat.

4.3.3.5.1 Výsledky Vedle menších grafických a technických nedostatků, které lze opravit během jednoho dne nebo nejsou definitvním požadavkem, protože jsou také subjektivní a ne nutností z pohledu platformy. Co vzešlo z jednání jsou body, které se musí dále prodiskutovat a popřípadě implementovat:

1. Když je příspěvek, či příspěvky tvůrce, nedostupný, protože ho uživatel nemá předplacený, může si ho jiném zařízení jít předplatit. Po předplacení se ale obrazovka TV aplikace tomuto faktu nepřizpůsobí a stále zobrazuje příspěvek jako nedostupný, přestože uživatel si ho právě předplatil. Z technického hlediska lze obnovy stavu příspěvku docílit smazáním instance `ViewModelu` z paměti. ViewModel existuje v paměti dokud nezmizí `ViewModelStoreOwner`, ke kterému je přiřazen. V této aplikaci, používající Compose a Navigation komponent, dojde ke smazání instance `ViewModelStoreOwner`, na který je daný `ViewModel` navázán, když je položka navigačního záznamu odstraněna ze zásobníku záznamů a to proto, protože tyto záznamy přímo dědí z `ViewModelStoreOwner`. V praxi to tedy momentálně znamená, že uživatel musí přejít na jinou obrazovku, aby došlo k odebrání navigačního záznamu ze zásobníku a při přechodu zpět na danou obrazovku `ViewModel` obstará nová data. Optimální stav by tedy byl takový, kde aplikace průběžně, v nějakém intervalu, stahuje nová data.
2. Na domovské obrazovce při přechodu z řádku videoepizod na řádek audioepizod není řádek videoepizod viditelný a není jasné, že tlačítkem nahoru lze přejít zpět na řádek s videoepizodami. Toto chování je u ostatních streamovacích aplikací běžné, avšak v této aplikace existují jen dva řádky, zatímco u ostatních aplikací bývá řádků mnoho. Proto by tento potenciální nedostatek šlo vyřešit zobrazováním právě nezaměřeného řádku pod řádkem zaměřeným a prakticky tak vytvořit tak nekonečný seznam, kde se řádky rotují. Pak budou oba řádky vždy viditelné.
3. Posledním bodem z námětů bylo daní uživatelovi možnost zpětné vazby či zobrazení častých dotazů (FAQ). Návrhů padlo několik, jako zobrazení kontaku nebo QR kódu pro vyplnění zpětné vazby na telefonu. Umístění této možnosti v aplikaci, např. jako upravenou navigační destinaci v postranním panelu, bylo také diskutováno.

Kapitola 5
Budoucnost aplikace

Z testování a konzultace vzešlo několik možných rozšíření a vylepšení. Některá z těchto rozšíření a vylepšení možná budou muset být součástí aplikace před oficiálním vydáním aplikace. Jejich zvolení bude výsledkem dalších diskusí s Forendors. Poté bude následovat oficální spuštění pro všechny uživatele.

5.1 Možná rozšíření a vylepšení

V testování byla obdržena zpětná vazba, která se vázala na funkcionality aplikace, které nejsou podporovány nebo jsou omezovány backendem. Jednalo se primárně o tyto nedostatky:

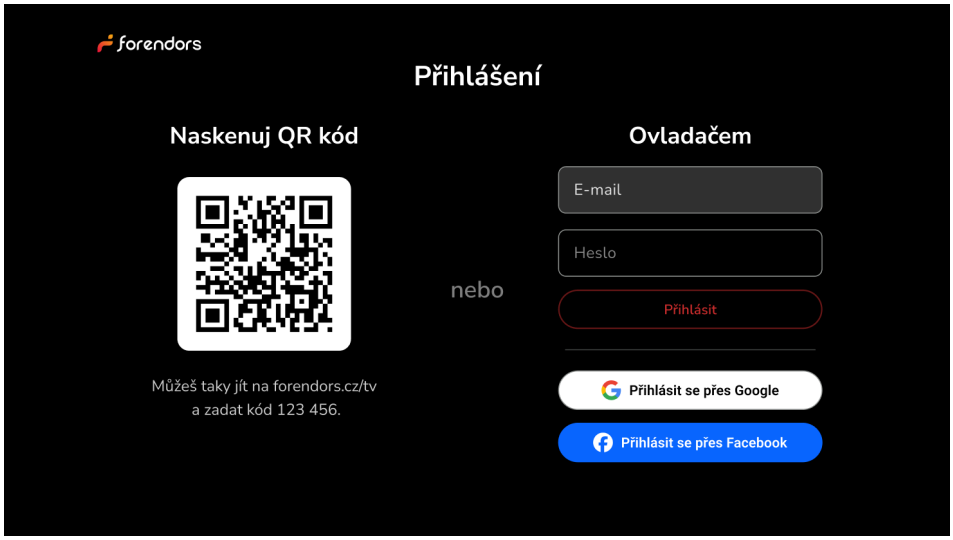
- Funkcionalita vyhledávání příspěvku.
- Doba načítání nejen výsledků vyhledávání a obsahu přehrávačů.

Neoptimální implementační rozhodnutí, způsobeno nutností práce na backendu, bylo uděláno i v přihlašovací části aplikace, kde uživatelé měly problémy s průchodem se spojenými testovacími případy nebo by preferovaly jinou cestu. Zjištěny byly ale i problémy spojené čistě s implementací uživatelského rozhraní. Popisy problémů a řešení následují v podsekcích.

5.1.1 Přihlášení

Z testování vzešlo, že přihlašovací obrazovka Forendors, by při přihlašování účtem Google, měla nějakým způsobem indikovat průběh přihlašování. Ikdyž se většina nebo celý proces odehrává na systémové obrazovce, často se stává, že systémová obrazovka je např. průsvitná a na pozadí se zobrazuje přihlašovací obrazovka Forendors nebo systémová obrazovka zmizí úplně, ale proces stále probíhá na pozadí. Řešením může být nezobrazování původních elementů přihlašovací obrazovky a jednoduché zobrazení načítání přes celou obrazovku či konkrétnější indikace spojená s tlačítkem Google.

Dalším nedostatkem se v procesu přihlašování projevilo nedostatečné řešení chybových situací. V případě problému obrazovka nezobrazuje, co je důvodem chybového stavu, ale pouze zobrazuje zprávu, že se něco nepovedlo. V případě, že uživatel zadá chybné přihlašovací údaje a jsou ideální podmínky, tzn. aplikace má přístup k internetu, server je funkční, atd., aplikace zobrazuje konkrétní chybové zprávy, podle kterých uživatel zjistí, co je špatně. V opačném případě, kdy nejsou ideální podmínky, aplikace zobrazuje pouze jednu obecnou chybovou zprávu. Aplikace by správně měla zobrazovat konkrétní chybové zprávy i v případech, které se mohou stát a nejsou způsobeny uživatelovým vstupem, ale externími podmínkami. Tento stav není ale příliš častý i proto, protože televize jsou umístěny staticky a bývají připojeny na stabilní internetový zdroj. Priorita řešení tohoto nedostatku musí být dořešena.



Obrázek 5.1 Návrh obrazovky plnohodnotného přihlášení

Když aplikace zobrazuje generickou chybovou zprávu a uživatel si chce překontrolovat heslo, nemá způsob, jak by toho docílil, jelikož aplikace uživatelův vstup na UI transformuje a není čitelný. Možnost deaktivovat transformaci a zobrazit si heslo momentálně uživatel nemá. Pokud budou implementovány specifické chybové zprávy, uživatel bude muset heslo kontrolovat jen za situace, kdy zadá chybné přihlašovací údaje a ne jindy. Proto má tato funkcionalita z mého pohledu nižší prioritu, než specifické chybové zprávy.

Plnohodnotné UI přihlašovací obrazovky by mohlo vypadat následujícím způsobem: viz návrh 5.1. Pokud navíc existuje záznam s heslem pro Forendors účet v Google účtu uživatele, zobrazí se zde zmiňované vyskakovací okno, pomocí kterého se může uživatel ihned přihlásit. Takový by byl optimální stav přihlášení.

Jednou z dalších možností je přihlášení pomocí mobilní aplikace Forendors. Mobilní aplikace mohla existovat funkce přihlášení do TV. Tato funkce by fungovala pomocí jedné nebo oběma dále popsaných implementací. První možnou implementací by bylo skenování QR kódu, kde by uživatel nemusel provádět žádné další kroky a automaticky by byl přihlášen na TV. Druhou možností by bylo přihlášení přes sít, jako existuje v aplikacích Spotify.

Navíc by přidání této funkce do mobilní aplikace fungovala jako propagace televizní aplikace. Tento způsob propagace produktu se nazývá cross-selling a je použit i na webových stránkách Forendors, kde je propagována mobilní aplikace. Co se týče cross-sellingu by mobilní aplikace nemusela sloužit jen pro účely přihlašování, ale také jako čistě akviziční kanál ve formě jednorázové obrazovky promující Android TV Forendors aplikaci.

5.1.2 Přehrávače a domovská obrazovka

Zmíněným sdíleným nedostatkem přehrávačů bylo dlouhé trvání načítání. Podle mého testování je doba načítání variabilní a načítání je opravdu ne vždy krátké. Příčinu bude nutné dále prozkoumat.

Jako vylepšení pak byla zmíněna možnost v přehrávači přejít na další či předešlou epizodu. To je vhodné vylepšení pro pozdější verzi aplikace spolu s jinými novými vylepšeními, které nejsou pro uživatele vitální.

Ve video přehrávači bylo jedním testerem kritizováno umístění perexu. Zatímco v audio přehrávači nebyl perex intruzivní, ve video přehrávači působil perex na uživatele tak, že když byl překryv videa viditelný, video nebylo po danou dobu, dle testera B, dívatelné. Obrazovka, kde by důležité informace o obsahu a příspěvku byly separovány od samostatného přehrávatelného obsahu, byla v procesu návrhu

zvážena a toto je možná důvod znovu navštívit myšlenku implementace této obrazovky. Přehrávače ale byly ostatními testery chváleny, takže tento potenciální nedostatek bude znovu zvýžen na základě zpětné vazby z produkčního prostředí.

5.1.3 Domovská obrazovka

Na domovské obrazovce byla navrhnutozobrazovat rozdívané epizody. Z pohledu implementace by se tedy pravděpodobně jednalo o nový řádek s epizodami. Tato řádka by mohla obsahovat i nové epizody navazující na ty, které uživatel viděl naposledy. Co se týče epizod na domovské obrazovce bylo také zmíněno u epizod indikovat, jestli je daná epizoda rozdívaná nebo jestli už byla shlédnuta. Dané funkcionality nejsou, stejně jako funkcionalita s přechodem na další či předešlou epizodu v přehrávači, pro koncového uživatele zásadní a mohou být implementovány až v některé z pozdějších verzí aplikace.

5.1.4 Vyhledávání a profil tvůrce

Uživately byla na obrazovce vyhledávání vytýkána jak rychlost vyhledávání, tak jeho funkcionalita, že neexistuje možnost hledat podle klíčových slov, týkajících se ne pouze tvůrce, ale i příspěvků. Tato funkcionalita jim chyběla i na profilu tvůrce, kde neměly možnost konkrétní příspěvky vyhledávat podle textového vstupu.

Tyto problémy jsou však spojené s momentální implementací a funckionalitou poskytovanou backend serverem. Backend neposkytuje data vyhledávání v optimálním čase a také neposkytuje možnost hledat příspěvky. Proto tyto nedostatky momentálně nemá smysl řešit a budou řešeny, jakmile dojde k posunu na straně backendu.

Potenciální problémy, které lze řešit kdykoliv, jsou:

1. Zobrazování příspěvků na profilu tvůrce, které nelze zobrazit či přehrát.

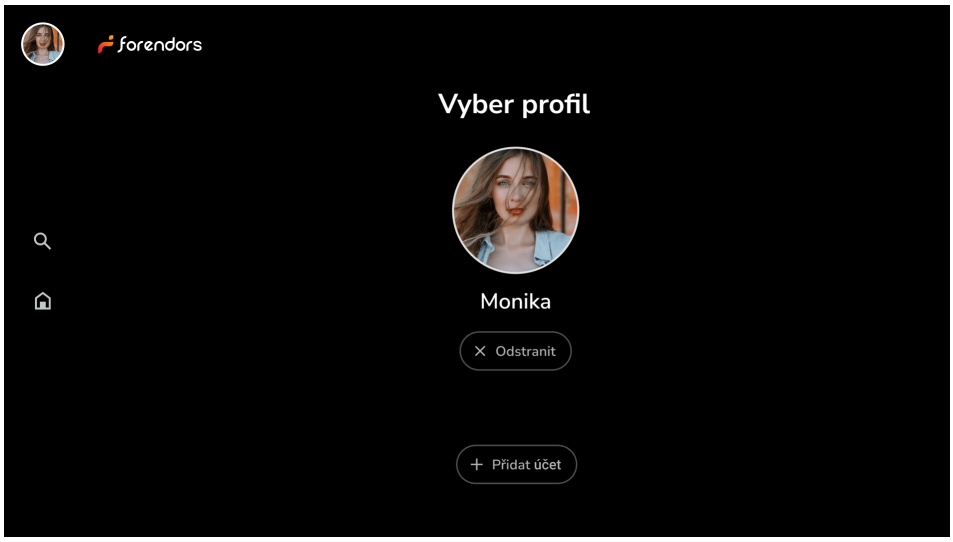
   Někteří z testerů měli s těmito příspěvky problém, ale jde z mého pohledu o nekritický nedostatek a uživatelé po zjištění faktu, že příspěvky bez audia a videa nemají náhled a nelze s nimi dále zacházet, nebudou nadále pociťovat tento potenciální nedostatek dále řešit a nebude jim to narušovat zážitek. Z tohoto subjektivního důvodu neberu tuto záležitost jako prioritu pro další vývoj aplikace.

2. Neindikování žádných výsledků na vyhledávací obrazovce.

   Aplikace by měla zobrazit text, že nebyly nalezeny žádné výsledky. Toto je nedostatek uživatelského rozhraní a zobrazování textu bude doimplementováno.

5.1.5 Přepnutí účtu

Optimání cestou pro přepnutí účtu by byla obrazovka, která by uměla přepnout, přidat a odebrat účet. Případ užití by v tomto případě byl identický s tím v aplikaci Spotify. Rozložení prvků ve Spotify mi přijde intuitivní a čisté. Navrhnutá obrazovka tak vypadá téměř totožně. Výsledná obrazovka vypadá viz návrh 5.2. Tato obrazovka by se zobrazovala v hlavní destinaci uživatelského účtu, ale bude se také by se zobrazovala při spuštění aplikace v případě, že je počet přihlášených účtů více nebo rovno dvěma.



Obrázek 5.2 Návrh obrazovky přepnutí uživatele

Závěr

Cílem práce bylo vytvořit Android TV aplikaci, kterou mohou uživatelé televizí s operačním systémem Android používat ke konzumaci obsahu na platformě Forendors.

Pro správný návrh aplikace byly definovány požadavky a případy užití. Byly analyzovány funckionality Android TV a doporučení pro design Android TV aplikací. Nakonec byla provedena analýza existujících Android TV aplikací poskytující možnosti konzumace obsahu.

Na základě návrhu proběhla implementace podle architektury dané základními principy a následováním architektury stavěné na doporučené architektu aplikací. Aplikace byla vyvinuta nativním Android vývojem pomocí jazyka Kotlin. Vývoj navázal na již existující datovou a doménovou vrstvu.

Při vývoji bylo využito standarních technologií a pro tvorbu uživatelského rozhraní bylo využito technologie Compose, která byla dosud adoptována převážně ve světě mobilního Android vývoje. Bylo využito také rané knihovny Compose TV v experimentální fázi vývoje, umožňující snazší implementaci běžných TV UI komponent. Vedlejším efektem práce je i fakt, že Compose TV knihovnu již lze využít pro vývoj Android TV aplikací a vývoj Android TV aplikací lze použitím této knihovny akcelerovat.

Pro distribuci aplikace bylo využito systému Google Play a pro sledování aplikačních metrik byly implementovány monitorovací služby Firebase. Aplikace byla otestována několika testery a byla sesbírána zpětná vazba od Forendors. Na základě zjištění a navazujích poznatků vznikla možná rozšíření a vylepšení.

Aplikace bude brzy dostupná pro uživatele platformy Forendors, kteří si budou moci užít své oblíbené tvůrce na velkých obrazovkách u sebe doma a poskytne Forendors s Quanti konkurenční výhodu na trhu novodobých creators economy platforem.

Bibliografie

1. ALTEXSOFT. *Functional and Nonfunctional Requirements: Specification and Types* [online]. [B.r.]. Dostupné také z: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types.
2. ANDROID DEVELOPERS. *TV app quality* [online]. [B.r.]. Dostupné také z: https://developer.android.com/docs/quality-guidelines/tv-app-quality.
3. BEN SCHOON. *Android TV will be replaced by Google TV* [online]. [B.r.]. Dostupné také z: https://9to5google.com/2020/09/30/android-tv-replaced-google-tv-update.
4. BEN SCHOON. *Android TV homescreen gets revamped with some Google TV goodness* [online]. [B.r.]. Dostupné také z: https://9to5google.com/2021/02/11/android-tv-discover-tab-homescreen-update.
5. RACHAEL PHILLIPS. *Google TV vs Android TV: which is better?* [Online]. [B.r.]. Dostupné také z: https://www.techradar.com/news/google-tv-vs-android-tv.
6. GOOGLE. *Google TV* [online]. [B.r.]. Dostupné také z: https://tv.google/.
7. ANDROID DEVELOPERS. *Ambient Mode* [online]. [B.r.]. Dostupné také z: https://developer.android.com/training/tv/playback/ambient-mode.
8. ANDROID DEVELOPERS. *Display a Now Playing card* [online]. [B.r.]. Dostupné také z: https://developer.android.com/training/tv/playback/now-playing.
9. ANDROID DEVELOPERS. *Add programs to the Watch Next channel* [online]. [B.r.]. Dostupné také z: https://developer.android.com/training/tv/discovery/watch-next-add-programs.
10. ANDROID DEVELOPERS. *Audio capabilities* [online]. [B.r.]. Dostupné také z: https://developer.android.com/training/tv/playback/audio-capabilities.
11. ANDROID DEVELOPERS. *TV UI Design Guides* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/design-for-tv.
12. GOOGLE PLAY. *YouTube for Android TV on Google Play Store* [online]. [B.r.]. Dostupné také z: https://play.google.com/store/apps/details?id=com.google.android.youtube.tv&hl=en_US.
13. ANDROID DEVELOPERS. *Navigation on TV: Clear path to all focusable elements* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/navigation-on-tv#clear-path-to-all-focusable-elements.
14. ANDROID DEVELOPERS. *Navigation on TV: Axis* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/navigation-on-tv#axis.

15. ANDROID DEVELOPERS. *Navigation on TV: Fixed start destination* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/navigation-on-tv#fixed-start-destination.
16. ANDROID DEVELOPERS. *Components: Navigation drawer* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/components/navigation-drawer.
17. ANDROID DEVELOPERS. *Navigation on TV: Use predictable back button behavior* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/navigation-on-tv#use-predictable-back-button-behavior.
18. ANDROID DEVELOPERS. *Styles: Layout templates* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/styles/layouts#layout-templates.
19. ANDROID DEVELOPERS. *Components: Lists* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/components/lists.
20. ANDROID DEVELOPERS. *Types of content to include in the Watch Next channel* [online]. [B.r.]. Dostupné také z: https://developer.android.com/training/tv/discovery/guidelines-app-developers#types-of-content.
21. GOOGLE PLAY. *Netflix on Google Play Store* [online]. [B.r.]. Dostupné také z: https://play.google.com/store/apps/details?id=com.netflix.ninja&pcampaignid=web_share.
22. ANDROID DEVELOPERS. *Design for TV: Communical device* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/design-for-tv#communical-device.
23. ANDROID DEVELOPERS. *Controllers: Back button* [online]. [B.r.]. Dostupné také z: https://developer.android.com/training/tv/start/controllers#back-button.
24. ANDROID DEVELOPERS. *Components: Immersive lists* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/components/immersive-list.
25. ANDROID DEVELOPERS. *Navigation on TV: Don't display a back button* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/navigation-on-tv#dont-display-a-back-button.
26. GOOGLE PLAY. *Spotify - Music and Podcasts on Google Play Store* [online]. [B.r.]. Dostupné také z: https://play.google.com/store/apps/details?id=com.spotify.tv.android.
27. ANDROID DEVELOPERS. *Components: Lists - selection controls* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/components/lists#selection-controls.
28. ANDROID DEVELOPERS. *Color on TV: Contrast* [online]. [B.r.]. Dostupné také z: https://developer.android.com/design/ui/tv/guides/foundations/color-on-tv#contrast.
29. ANDROID DEVELOPERS. *On-screen keyboard: Placement* [online]. [B.r.]. Dostupné také z: https://developer.android.com/training/tv/start/onscreen-keyboard#placement.
30. ANDROID DEVELOPERS. *Media3 ExoPlayer* [online]. [B.r.]. Dostupné také z: https://developer.android.com/media/media3/exoplayer.
31. ANDROID DEVELOPERS. *Architecture* [online]. [B.r.]. Dostupné také z: https://developer.android.com/topic/architecture.
32. ANDROID DEVELOPERS. *UI layer* [online]. [B.r.]. Dostupné také z: https://developer.android.com/topic/architecture/ui-layer.
33. MANUEL VIVO. *Consuming flows safely in Jetpack Compose* [online]. [B.r.]. Dostupné také z: https://medium.com/androiddevelopers/consuming-flows-safely-in-jetpack-compose-cde014d0d5a3.

34. GRADLE INC. *Gradle User Manual* [online]. [B.r.]. Dostupné také z: https://docs.gradle.org/current/userguide/userguide.html.
35. KOIN & KOTZILLA. *Koin Documentation* [online]. [B.r.]. Dostupné také z: https://insert-koin.io/docs/.
36. COIL CONTRIBUTORS. *Coil Overview* [online]. [B.r.]. Dostupné také z: https://coil-kt.github.io/coil/.
37. ANDROID DEVELOPERS. *Using Views in Compose* [online]. [B.r.]. Dostupné také z: https://developer.android.com/develop/ui/compose/migrate/interoperability-apis/views-in-compose.
38. ANDROID DEVELOPERS. *Thinking in Compose* [online]. [B.r.]. Dostupné také z: https://developer.android.com/jetpack/compose/mental-model.
39. ANDROID DEVELOPERS. *State and Jetpack Compose* [online]. [B.r.]. Dostupné také z: https://developer.android.com/jetpack/compose/state.
40. RAY SLATER BERRY. *The magic number for user testing: How many users do you need?* [Online]. [B.r.]. Dostupné také z: https://maze.co/blog/user-testing-how-many-users/.

Obsah příloh