České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Marek Petr

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: Video on demand aplikace pro Smart TV

Pokyny pro vypracování:

1. Analyzujte platformy Smart TV: Firefox OS TV, Tizen TV a webOS TV s ohledem na možnosti využití společného zdrojového kódu, možnostech implementace DRM, způsobu ovládání aplikací, apod.
2. Navrhněte aplikaci pro video on demand určenou pro zmíněné platformy.
3. Implementujte danou aplikaci pro zmíněné platformy.
4. Navrhněte a proveďte SW i uzivatelske testování aplikace.

Seznam odborné literatury:

1) GAMMA, Erich. Návrh programů pomocí vzorů: stavební kameny objektově orientovaných programů. Praha: Grada, 2003. Moderní programování. ISBN 80-247-0302-5
2) BROOKS, Frederick P. The mythical man-month: essays on software engineering. Reading, Mass.: Addison-Wesley Publishing Company, c1995. ISBN 0-201-83595-9.
3) RAUSCHMAYER, Axel. Speaking JavaScript: [an in-depth guide for programmers]. ISBN 1449365035.
4) http://developer.vieraconnect.com/dev-guide/html5-v2.0
5) http://webostv.developer.lge.com/

Vedoucí: Ing. Michal Havryluk

Platnost zadání do konce letního semestru 2017/2018

L.S.

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 31.1.2017

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Video on demand aplikace pro Smart TV

Petr Marek

Vedoucí práce: Ing. Michal Havryluk

22. května 2017

Poděkování

Děkuji rodině, přátelům a hlavně Soně za jejich trpělivost a soustavnou podporu. Děkuji vedoucímu práce, Ing. Michalu Havrylukovi, za cenné rady a poskytnutí potřebného zázemí pro realizaci práce.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 22. května 2017 .....

České vysoké učení technické v Praze
Fakulta elektrotechnická

Odkaz na tuto práci

Marek, Petr. *Video on demand aplikace pro Smart TV.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2017.

Abstrakt

Cílem této práce je návrh, implementace a testování video on demand aplikace pro Smart TV platformy Firefox OS TV, Tizen TV a WebOS TV. Důraz je přitom kladen na využití jednotného zdrojového kódu. Práce se zabývá i chráněním video obsahu pomocí DRM a různými způsoby ovládání Smart TV aplikací. Diplomová práce rovněž zahrnuje testování, včetně SW a uživatelských testů.

Klíčová slova Smart TV, video on demand, HTML video, Angular

Abstract

The goal of this thesis is the design, implementation and testing of a video on demand application for Smart TV platform TV Firefox OS, Tizen TV and WebOS TV. Emphasis is put on the use of a single source code. This thesis also deals with protecting video content using DRM and different ways to control Smart TV applications. The thesis also includes testing, including software and user tests.

Keywords Smart TV, video on demand, HTML video, Angular

Obsah

Seznam obrázků

Úvod

Posledních několik let pozorujeme nárůst podílu Smart TV na trhu s televizemi a zdá se, že nastolený trend bude v příštích letech nadále pokračovat, alespoň podle statistiky [1]. Stejně jako v jiných silně konkurenčních odvětvích, tak i na poli výrobců TV, je vytvářen na výrobce tlak, aby inovovali a neztráceli tak podíly prodejů. Výsledkem těchto inovací jsou v současnosti právě Smart TV jako nejrozšířenější podoba TV. Zaznamenáváme také stále vyšší integraci tzv. chytrých zařízení (smart devices) do našich životu. Ať už se jedná o chytré hodinky, auta nebo chytré televize o telefonech ani nemluvě. Největší výhodu těchto zařízení lze vidět v jejich snadném propojení se zbytkem chytré domácnosti a tím pak usnadněním našich každodenních úkonů.

Smart TV si lze zjednodušeně představit jako klasickou TV, která má navíc přístup k Internetu a možnost doinstalovat aplikace prostřednictvím obchodu s aplikacemi. Každá platforma takový obchod poskytuje, což zajišťuje výrobci kontrolu trhu, podobně jako v případě chytrých telefonů. Výrobce si tak muže snadno hlídat kvalitu aplikací, distribuci jen v některých teritoriích, ale především finanční stránku věci. Smart TV mají samozřejmě všechny atributy klasických TV, přístup k Internetu nabízí navíc sledování internetové TV, on demand obsahu či přístup k domácí multimediální knihovně.

Smart TV platformy jako takové spojují HW televize a příslušenství s upraveným OS. Výrobci Smart TV zpravidla staví svůj operační systém na některé linuxové distribuci. OS muže být dodáván integrovaný v TV nebo jej může výrobce dodávat externě v podobě malého HW zařízení. Další možností, jak se z obyčejného zobrazovacího zařízení stane Smart TV, je připojením jakéhokoli jiného zařízení s dříve jmenovanými vlastnostmi. Takovými zařízeními jsou například herní konzole (Microsoft Xbox, Sony Playstation) či multimediální centra (Google Chromecast, Apple TV).

S možnostmi Smart TV se do jisté míry změnila i forma domácí zábavy spjatá se sledováním TV. Ve světě i u nás je patrný rozmach video on demand jako formy získávání obsahu. Celosvětově nejrozšířenější službou je v současnosti beze sporu Netflix, dostupný je i u nás. Z českých služeb nelze opomenout

Stream.cz od Seznamu, Prima Play nebo iVysílání České televize.

Video on demand (VOD) systém nabízí uživateli možnost volby sledování multimediálního obsahu, tzn. že není odkázán na vysílání a omezen vysílacími časy. Multimediální obsah je distribuován z pohledu uživatele buďto jako živý stream nebo ve formě knihovny. S distribucí multimediálního obsahu souvisí jeho ochrana pomocí Digital Rights Management (DRM), která se děje na pozadí a platící uživatel se o ní v ideálním případě ani nedozví. VOD je placená služba, platba a přístup k obsahu se realizuje různě, může to být časově omezeným předplatným, kreditem a jeho ponižování nebo jiným způsobem.

VOD funkce patří mezi hlavní rysy výsledné aplikace, která kombinuje jednoduchou video knihovnu a video přehrávač se standardními funkcemi jako je kontrola videa a změna znění či titulků.

V zadání diplomové práce jsou záměrně zvoleny Smart TV platformy Tizen TV (Samsung), WebOS TV (LG) a FirefoxOS TV (Panasonic). Tyto platformy ještě spolu s Android TV (Sony) mají v době psaní této práce největší a mezi sebou zhruba rovnoměrně rozdělený podíl na trhu Smart TV, např. podle průzkumu [2]. Tato práce se nezabývá Android TV, jelikož ta se od zbývajících třech velice liší a společné prvky pro využití jednotného zdrojového kódu by se hledali velice těžko.

Každá Smart TV platforma má svá specifika, možnosti, limitace a úskalí realizace. U aplikací cílených pro použití v obýváku je nutné mít na paměti UI a UX s jejich specifiky – ovládání dálkovým ovladačem, větší pozorovací vzdálenost. Každá platforma má navíc sadu směrnic, tzv. guidelines, které se mírně liší, tudíž je potřeba volit společné jmenovatele a zároveň udržet konzistenci v rámci dané platformy. Nedílnou součástí je podle zadání i průzkum možností implementace přehrávání obsahu chráněného DRM. Opět ne všechny platformy nabízí stejnou formu DRM a také neposkytují jednotné rozhraní při jejím použití.

Součástí této diplomové práce je rovněž testování aplikace uživateli, které probíhalo na všech zmíněných platformách. Připomínky, které vyústily z uživatelského testování byly zapracovány do výsledné aplikace.

Výsledná aplikace sdílí maximální množství společného kódu, je psána za pomocí moderní a robustní technologie a ukazuje směr, kterým se lze vydat při návrhu a realizaci potenciálně rozsáhlé aplikace pro VOD. Aplikaci lze snadno rozšířit do plnohodnotné VOD aplikace, která se může stát rovnocenným konkurentem zmíněných služeb, bude-li mít dostatečně bohatou knihovnu.

Kapitola 1

Specifikace zadání

V této kapitole je specifikované zadání diplomové práce, popis funkcí aplikace a formální požadavky. V závěru kapitoly jsou uvedena některá existující řešení, která se zabývají podobnou problematikou.

Dále se v textu práce používá názvů platforem s příponou TV i bez ve shodném významu, vždy je myšlena TV platforma. Podobně se takto v záměnném významu používá výrazů film a video.

1.1 Popis aplikace

Aplikace bude sloužit jako demonstrace toho, jak lze postavit funkční VOD aplikaci nad jednotným zdrojovým kódem pro předmětné platformy. Zároveň bude demonstrovat základní prvky VOD aplikace jako jsou knihovna filmů, detail filmu a video přehrávač. Každá z těchto základních částí bude mít svou roli v aplikaci a funkce, které jsou nastíněny v následujících odstavcích. Důraz bude kladen na dodržování guidelines (pravidel, směrnic) specifických pro danou platformu. Aplikace bude navržena s ohledem na snadné budoucí rozšíření a testovatelnost.

1.1.1 Přihlašovací obrazovka

Aplikace bude vyžadovat přihlášeného uživatele, proto nabídne jeho přihlášení pomocí této obrazovky.

1.1.2 Knihovna filmů

Knihovna filmů bude zobrazovat dostupné filmy ke shlédnutí spolu se stručnými informacemi o aktuálně vybraném filmu. Tyto informace budou obsahovat název filmu, příznak dosud neviděného filmu, rok vydání, žánr, dostupnou kvalitu, délku filmu, krátký popis a hodnocení v podobě pěti hvězd reflektující hodnocení jedna až pět spolu s počtem hodnocení. Protože se jedná zároveň

o první obrazovku celé aplikace (po přihlášení), kterou uživatel uvidí, musí splňovat několik požadavků. Mezi nejdůležitější požadavky vztažené na uživatele patří snadná orientace, přehlednost, rozpoznatelný relevantní obsah, nový obsah, např. ještě neviděné filmy, nové filmy, atd.

1.1.3 Detail filmu

Na detail filmu se uživatel dostane potvrzením výběru daného filmu z knihovny. Tato obrazovka bude zobrazovat spolu s informacemi viditelnými na úvodní obrazovce navíc dodatečné informace, jako jsou celý popis filmu a jména tvůrců a umělců. Nedílnou součástí detailu je zobrazení dostupného znění a titulků. Obrazovka detailu filmu bude umožňovat spustit přehrávač videa ve dvou režimech, buď od začátku nebo od času, kdy uživatel sledování přerušil. Později jmenovaným způsobem lze navázat na dříve sledovaný film.

1.1.4 Video přehrávač

U aplikace určené pro TV je nutné přizpůsobit vzhled a ovládání tomu, že uživatelovo primární použití aplikace je k odpočinku. Celkově budou zobrazeny jen nejnutnější informace, které uživatel potřebuje k přehrávání filmu, což je rozdíl oproti přehrávači určenému pro použití na počítači, kde je možné leccos nastavit a přizpůsobit. Na TV aplikaci by takové prvky byly rušivé a zbytečné.

Video přehrávač bude vyvolán z detailu filmu a bude přehrávat zvolený film buď od začátku nebo od zapamatované pozice. Přehrávač bude podporovat standardní funkce přehrávače videa a jejich snadné ovládání. Podrobný popis toho, jak přehrávač funguje je uveden v části 2.1.4, následující odstavce shrnují, co přehrávač podporuje za funkcionality.

1.1.4.1 Funkce

Hlavní funkcí přehrávače bude samozřejmě schopnost video načíst a spustit. Dále bude přehrávač podporovat pozastavení videa, posun ve videu vpřed, respektive vzad. Nedílnou součástí bude možnost změny znění a titulků.

Video přehrávač bude provádět automatický výběr audio a titulkové stopy podle systémového jazyka a podle dostupných stop. Způsob výběru je podrobněji popsán v odpovídající implementační části.

1.1.4.2 Ovládání

Kontrola přehrávače bude pokud možno jednotná a uživatelsky přívětivá. Uživatel by měl v každém okamžiku vědět, co má udělat, když bude chtít nějakou akci provést, to platí obecně pro celou aplikaci.

1.1.4.3 Stav

Přehrávač bude mít své vnitřní stavy a uživatel mezi nimi bude moci přecházet prostřednictvím dostupného ovládání přehrávače. Přehrávač bude indikovat uživateli svůj stav v každém okamžiku, ať se bude jednat o načítání, posun ve videu nebo samotné přehrávání.

1.2 Požadavky

Tato část se zabývá podrobnou definicí požadavků na aplikaci a rozšiřuje tak obecnější popis aplikace.

1.2.1 Funkční požadavky

1.2.1.1 Přihlášení a odhlášení uživatele

- Zobrazení přihlašovací obrazovky při prvním přihlášení a v případě vypršení platnosti přihlášení
- Možnost odhlášení či přihlášení jiného uživatele při odchodu z aplikace
- Po odhlášení odstranění uložených uživatelských voleb

1.2.1.2 Knihovna filmů

- Uživatel v každém okamžiku ví, co má udělat, když chce provést akci
- Vždy jen jeden jasně a zřetelně vybraný UI prvek
- Pohyb vertikálním (změna kategorie) a horizontálním (změna filmu v kategorii) směrem umožňující výběr filmu ke sledování
- Zapamatování vybraného filmu po návratu z detailu
- Zobrazení potvrzujícího dialogu při opouštění aplikace

1.2.1.3 Přehrávání videa

- Ovládání přehrávače
  - pozastavení a znovu spuštění videa
  - posun ve videu vpřed, respektive vzad
  - změna znění a titulků videa
- Zobrazení postupu videa (čas a lišta postupu)
- Zapamatování času postupu videa a možnost navázat při dalším spuštění

- Zapamatování vybraných titulků a znění pro další spuštění filmu
- Přednastavení audio stopy podle systémového jazyka a dostupných audio stop
- Přednastavení titulkové stopy podle systémového jazyka a dostupných titulkových stop
- Postup ve videu optimalizovaný vzhledem k omezenému výpočetnímu výkonu TV

1.2.2 Nefunkční požadavky

- Vývoj pro platformy Tizen TV (2.6), WebOS TV (3.0) a FirefoxOS TV (zařízení od 2016).
- Robustní základ pro vývoj potenciálně rozsáhlé VOD aplikace v moderní a pravidelně udržované technologii.
- Přirozené a snadné ovládání pomocí standardního dálkového ovládání 6-i tlačítky (OK, nahoru, dolu, vpravo, vlevo, zpět)
- Testovací playlist pro demonstraci funkčnosti aplikace
  - vygenerovaný v JSON formátu
  - vystavený na libovolném vzdáleném webovém serveru
  - obsahující testovací videa a titulky
- Nízká náročnost aplikace na úložný prostor v koncovém zařízení
- Minimální velikost aplikace pro rychlé načtení na koncovém zařízení v případě vystavení aplikace na vzdáleném serveru

1.3 Existující řešení

Existující řešení (VOD aplikace) lze rozdělit do dvou kategorií podle toho jakým způsobem je aplikace dodávána do koncového zařízení (TV). Prvním způsobem je Smart TV aplikace a tím druhým HbbTV standard. VOD může být poskytováno také jako dodatková služba poskytovatelů připojení k Internetu či kabelové TV, těmi se ale tato práce nezabývá. Další rozdělení konkurenčních řešení jsou možná, ale týkají se především finanční stránky aplikací či souvisejících služeb, jak již bylo v úvodu nastíněno (předplatné, kredit).

1.3.1 Smart TV aplikace

Nejrozšířenější službou v oblasti VOD je v současnosti Netflix se svými téměř 60 miliony předplatiteli v USA a 94 miliony po celém světě [3]. Netflix je dostupný i u nás. Tato služba nabyla nevídané popularity, má rozsáhlou knihovnu filmů a seriálů nejen z původní tvorby a nabízí také klasické půjčování DVD, což byla jejich původní služba. Netflix lze nainstalovat na všech třech probíraných platformách. Technické zpracování je na velmi vysoké úrovni, jak z pohledu UI, tak hlavně UX. Pohyb v knihovně filmů je vyřešen pomocí zafixovaného výběru, pod kterým se pohybují v horizontálním, resp. vertikálním směru miniatury filmů ze stejné kategorie, resp. různých kategorií. Jedinou zpozorovanou slabinou je reakce na odpojení Internetového připojení. Uživatel se v průběhu přehrávání ani při pohybu v aplikaci prakticky nedozví, že je odpojen, kontrola konektivity probíhá pouze při spouštění aplikace.

V českém prostředí jsou asi nejvýraznější služby VOYO od Nova TV a Stream.cz od Seznam.cz. Přičemž VOYO aplikace je podle [4] dostupná téměř na všechny Smart TV. Stream.cz má dostupnou Smart TV aplikaci pro Tizen TV a WebOS TV, nikoli však pro FirefoxOS TV [5]. Služba VOYO je stejně jako Netflix placená formou měsíčního předplatného, Stream.cz poskytuje video obsah zadarmo, ale za cenu reklamy.

Obě české aplikace se nemohou vyrovnat aplikaci Netflix především v UI a UX, které je sice funkční a svižné, ale v současnosti už ne příliš atraktivní. Tato domnělá vada, ale na druhou stranu umožňuje rozšíření na široké spektrum zařízení.

1.3.2 HbbTV aplikace

HbbTV standard znamená Hybrid Broadcast Broadband Television, neboli kombinaci TV vysílání a Internetu v TV. HbbTV je paralelním soupeřem Smart TV. Hlavním rysem HbbTV (stejně jako Smart TV) aplikací je možnost uživatelského vstupu oproti klasickému vysílání, kde je to technicky nemožné. Jeden z podstatných rozdílů HbbTV oproti Smart TV je v tom, že trh s aplikacemi není pod kontrolou dané platformy. Další rozdíl tkví v tom, jakým způsobem se aplikace načte do TV.

HbbTV aplikace jsou distribuovány zjednodušeně prostřednictvím TV vysílání speciálním streamem dvěma způsoby [6]. V prvním případě stream obsahuje odkaz na webovou stránku, kde je vystavená webová aplikace optimalizovaná pro HbbTV standard, odkaz se načte prostřednictvím Internetu. V druhém případě se aplikace vysílá v rámci DVB signálu. Druhý způsob nevyžaduje pro načtení aplikace Internet, nicméně jsou jím realizovány uživatelské akce. Pro úplnost, HbbTV aplikace se nespouští ze seznamu aplikací, nýbrž buď zmáčknutím tzv. červeného tlačítka ve správný okamžik (TV program uživatele upozorní – reklamní aplikace) nebo jsou vysílány na dedikovaném TV programu (kanálu).

Největší výhodou HbbTV je dostupnost na širokém spektru i velmi starých zařízeních, např. set-top-boxech. Dostupnost na starších zařízeních má také svou cenu a to v podobě nutnosti používat webové technologie z roku 2010, alespoň ve verzi 1.0 podle [7].

Téměř všechny přední české TV stanice nabízejí vlastní VOD HbbTV aplikaci, přičemž není myšleno TV programy (teletext) realizované pomocí tohoto standardu. Aplikace obsahují především pořady z vlastní tvorby, jmenovitě např. Česká televize a její iVysílání nebo také Stream.cz.

Dříve zmíněná aplikace Stream.cz vypadá totožně i pro HbbTV. Z toho lze usoudit, že je použit společný zdrojový kód, který je omezen právě standardem HbbTV. Ostatní VOD HbbTV aplikace vypadají velmi podobně a používají stejné principy při pohybu v aplikaci a nejsou nijak zvlášť tzv. brandované.

Výstupní aplikace této práce bude lepší oproti zmíněným aplikacím především v kombinaci atraktivního UI a pochopitelného UX. Logické a snadno pochopitelné UX je hlavní slabinou většiny konkurenčních aplikací, např. reakce na odpojení Internetu či samotný pohyb v aplikaci.

Kapitola 2

Analýza a návrh řešení

V této kapitole jsou rozebrány případy použití v části 2.1 výsledné Smart TV aplikace a jednotlivé komponenty celého řešení v části 2.2. Následuje část 2.3, která je věnována srovnání Smart TV platforem, jejich rozdílům a společným prvkům, v závěru této části jsou identifikované společné prvky pro vytvoření jednotného zdrojového kódu.

Další části kapitoly se věnují různorodému ovládání a omezení aplikací pro Smart TV vztaženo k návrhovým rozhodnutím pro výslednou aplikaci, části 2.4, resp. 2.6.

V kapitole přichází na řadu popis možností přehrávání obsahu chráněného DRM v části 2.7. Rozbor DRM je následován popisem zabezpečení aplikace v části 2.8.

Poslední části kapitoly jsou věnovány volbě technologií a zpracování vnitřního stavu aplikace v části 2.9, resp. 2.10.

2.1 Případy použití

V této části jsou uvedeny jen nejdůležitější případy použití a rozhodně se nejedná o jejich vyčerpávající výčet. Jednoduché, ale stěžejní případy použití jsou popsány pouze slovně. Případy použití jsou organizovány podle jednotlivých modulů aplikace. Podrobný návrh UI a UX aplikace je v části 2.5.

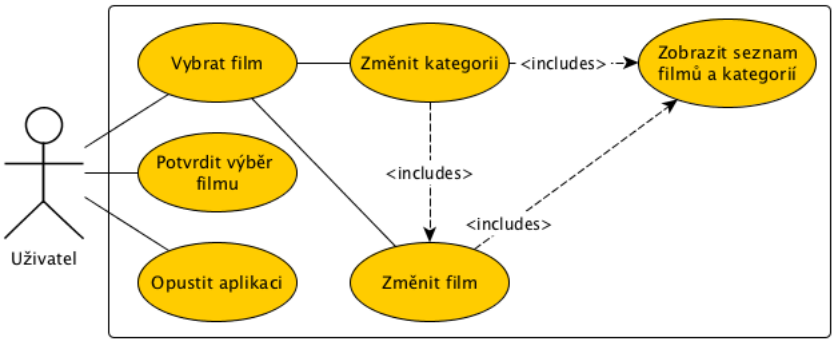
2.1.1 Rozdělení uživatelských rolí

V návrhu aplikace je uvažována pouze uživatelská role uživatel, neboli divák. Více uživatelský rolí nedává v kontextu této VOD aplikace příliš smysl, jedná se pouze o klientskou aplikaci pro uživatele. U VOD je důležité řízení přístupu k obsahu, které by mělo být řešeno na straně serveru poskytujícím webové služby aplikaci, tomu se tato práce nevěnuje. Aplikace bude však podporovat přihlášení a odhlášení uživatele.

Uživatel se nemusí vždy přihlašovat z domova, může být např. v hotelu, kde je dostupná Smart TV s aplikací. Kvůli podobným případům je nutné při odhlášení kompletně odstranit příp. uložená uživatelská data. Dalším případem použití může být hotelový uživatelský účet poskytovaný v rámci pokojového servisu, který by neukládal žádná uživatelská data, aby nabídl všem uživatelům stejný zážitek.

2.1.2 Modul – knihovna filmů

Knihovna filmů zahrnuje několik případů použití, zkráceně výběr filmu a opuštění aplikace, jak ilustruje diagram 2.1.



Obrázek 2.1: Diagram případů použití práce s knihovnou filmů

Vybrat film

Případ použití začíná, když uživatel chce vybrat film ke sledování. Aplikace zobrazí knihovnu dostupných filmů (provede se případ použití *Zobrazit seznam filmů a kategorií*, který předvybere první film první kategorie). Uživatel má možnost měnit výběr filmu pomocí navazujících případů použití *Změnit kategorii* a *Změnit film*. *Změnit kategorii* automaticky zahrnuje výběr filmu z kategorie. *Změnit film* znamená změna filmu ve vybrané kategorii.

Potvrdit výběr filmu

Případ použití začíná, když uživatel chce potvrdit vybraný film v kategorii. Potvrzením výběru aplikace zobrazí modul *detail filmu* 2.1.3. Rozdělení do případů použití *Vybrat film* a *Potvrdit výběr filmu* je z důvodu povahy ovládání dálkovým TV ovladačem, kde se nejprve směrovými tlačítky nasměruje nad požadovaný prvek a teprve následně provede potvrzení tlačítkem OK.

Opustit aplikaci

Případ použití začíná, když uživatel chce opustit aplikaci. Aplikace zobrazí potvrzovací dialog s možnostmi vrátit se zpět do aplikace, odhlásit se a opustit aplikaci, až na odhlášení je jejich chování zřejmé. Zvolí-li uživatel odhlášení, aplikace odhlásí přihlášeného uživatele a zobrazí přihlašovací obrazovku, jako ostatně i v případě úplně prvního vstupu do aplikace či vypršení platnosti uživatelského účtu.

2.1.3 Modul – detail filmu

Tento modul zahrnuje případy použití *Přehrát film od začátku*, *Přehrát film od zapamatované pozice*, *Návrat do knihovny filmů* a *Načíst podrobnosti*.
Při vstupu na detail filmu se provede automaticky případ použití *Načíst podrobnosti*. Volbou *Přehrát film od zapamatované pozice* se začne přehrávat film buď od začátku (film ještě nebyl přehráván) nebo od zapamatované pozice dřívějšího sledování. V obou případech přehrávání aplikace zobrazí modul *video přehrávač* 2.1.4.

2.1.4 Modul – video přehrávač

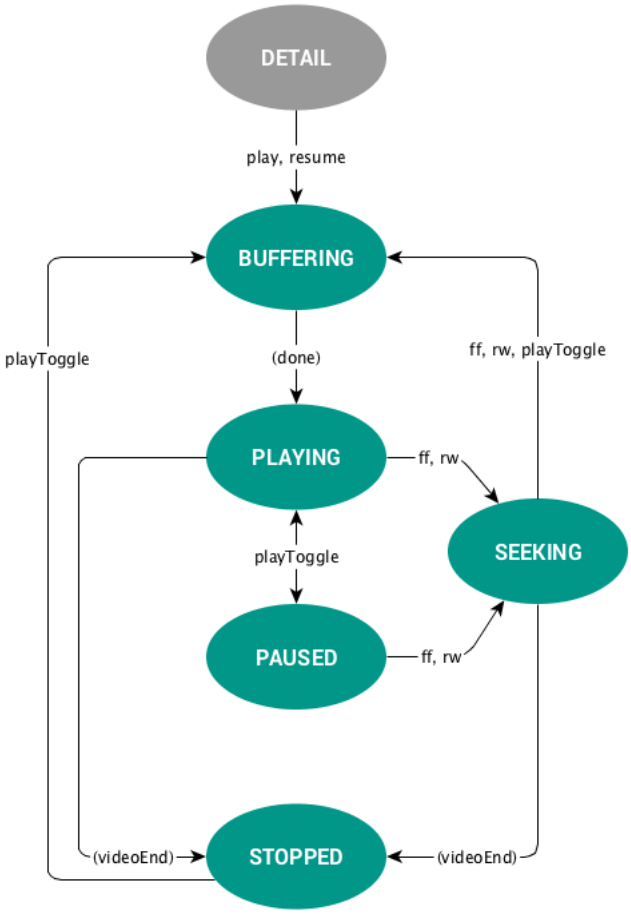
Video přehrávač umožňuje případy použití, které jsou slovně popsány v následujících odstavcích. Případy použití přehrávače úzce souvisí se stavy přehrávače (*player*) a jeho ovládacího panelu (*controls*). Stavy těchto komponent jsou vyobrazeny pomocí zjednodušených (kvůli přehlednosti) stavových automatů pro *player* na diagramu 2.2 a pro *controls* na diagramu 2.3. Diagramy slouží k lepší ilustraci daného případu použití, a proto je na ně odkazováno z popisu daného případu použití. Nicméně popis stavových automatů rozhodně není vyčerpávající a nezahrnuje určité krajní případy, např. posun videa před jeho začátek nebo za jeho konec. Tyto krajní případy jsou pak samozřejmě vyřešeny v implementaci. Stav *BUFFERING* je v popisu záměrně vynechán, jedná se o technický detail, pomocný stav, do kterého se může přehrávač dostat z několika možných stavů.

Přehrát, pozastavit

Případ použití začíná, když uživatel chce přehrát či pozastavit film (*playToggle*). Rozlišují se situace podle toho v jakém stavu je přehrávač a do jakého stavu se dostane akcí *playToggle*. Chování je patrné z diagramu 2.2. Do stavu *STOPPED* se dostane přehrávač v případě, že film skončil.

Posunout vpřed nebo vzad

Případ použití začíná, když uživatel chce posunout video vpřed, resp. vzad. Tato akce se běžně označuje jako *fast forward* (*ff*), resp. *fast rewind* (*rw*).



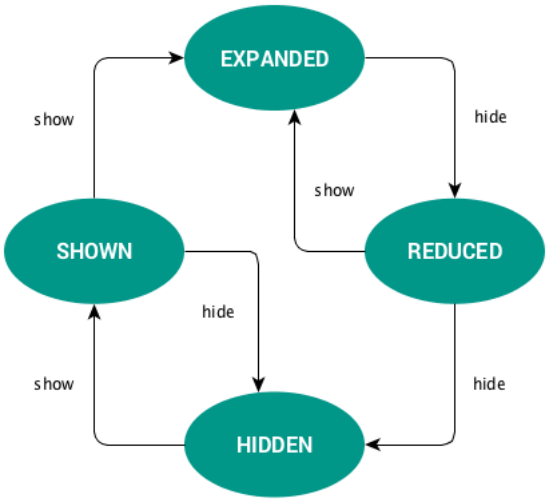
Obrázek 2.2: Diagram stavového automatu pro *player*

Přehrávač se vyvolanou akcí dostane do stavu *SEEKING* a probíhá posun (přetáčení) filmu. Posun se přeruší buď tím, že se přehrávač dostane na konec, resp. začátek videa nebo uživatel vyvolá případ použití *Přehrát, pozastavit* nebo *Zastavit, odejít*. Podrobnější popis mechanismu posunu, tzv. *seeking*, je v části 2.5.5.

Změnit znění nebo titulky

Případ použití začíná, když uživatel chce změnit znění nebo titulky filmu. Použije k tomu komponentu *controls* (podrobně vysvětlena v části 2.5.5) ve stavu *EXPANDED* (diagram 2.3), kdy se vyvolá případ použití *Zobrazit dostupná znění a titulky*. Uživatel vybere z nabídnutých možností a potvrdí výběr. Aplikace načte vybrané znění či titulky. Ve stavu *EXPANDED* nelze ovládat průběh videa, protože ovládací prvky jsou použity pro pohyb v komponentě *controls* a uživatelova pozornost je věnována výběru titulků či znění. Nicméně

lze v každém okamžiku odejít z přehrávače zpět na detail stisknutím tlačítka stop.



Obrázek 2.3: Diagram stavového automatu pro *controls*

Zastavit, odejít

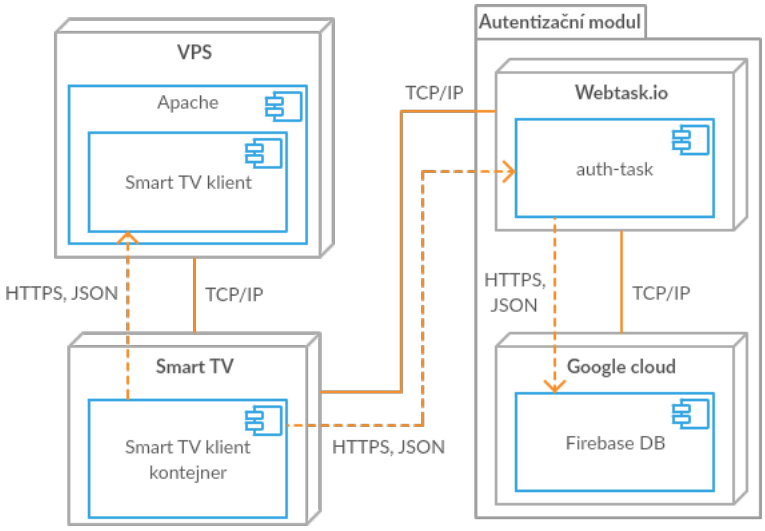
Případ použití začíná, když uživatel chce zastavit přehrávač neboli odejít z přehrávače. Tento případ použití inicializuje uživatel buď stiskem tlačítka stop kdykoli v přehrávači nebo stiskem tlačítka zpět ve stavu *HIDDEN* komponenty *controls*. Tento případ použití vyústí v návrat uživatele na detail fimu.

2.2 Komponenty systému

V níže uvedených částech se nachází popis jednotlivých komponent celého řešení tak, jak na sebe navazují a jak si předávají data. Celý tok dat mezi komponentami systému lze vyčíst z diagramu nasazení 2.4.

2.2.1 Webový server

Jako webový server je použit Apache HTTP Server od Apache Software Foundation. Tento open-source a multiplatformní webový server slouží v systému pro poskytovaní statických zdrojů. Mezi tyto zdroje patří simulovaná webová služba *playlist*, která poskytuje informace o kategoriích a filmech, multimediální soubory a samotné klientské Smart TV aplikace.



Obrázek 2.4: Diagram nasazení

2.2.2 Autentizační modul

Autentizační modul se jako logický celek skládá z *webtask* a databáze. Z důvodu absence serverové části aplikace je využito tzv. *serverless* metody pro přihlášení uživatele pomocí služby *Webtask.io* od Auth0 Inc. spolu s databází Firebase od Google Inc. *Webtask.io* sice nabízí úložiště, ale to je omezeno na pouhých 500 KiB.

2.2.3 Smart TV aplikace

Klientské aplikace pro jednotlivé platformy komunikující pomocí protokolu HTTPS s autentizačním modulem a s webovým serverem. Komunikace s autentizačním modulem probíhá při každém spuštění aplikace pro ověření platnosti uživatelského účtu, využívá se formátu JSON. Komunikace s webovým serverem probíhá v případě potřeby načíst statické zdroje jako jsou multimediální soubory, titulky a playlist ve formátu JSON.

2.3 Srovnání Smart TV platforem

Hlavním cílem této části je porovnání a nalezení společných prvků Smart TV platforem uvedených v zadání práce, Tizen TV, WebOS TV a FirefoxOS TV. Výsledek toho porovnání slouží jako základ pro volbu technologií a postupů při tvorbě společného kódu. Pro účely porovnání je využito oficiálních dokumentací platforem.

V první řadě je nutné zadefinovat stěžejní prvky, na které je potřeba se u jednotlivých platforem zaměřit, ty lze shrnout do těchto bodů:

- Možnosti webové aplikace (API) – všechny platformy nabízejí určitou formu podpory webových technologií
- Rozšiřující knihovny
- SDK – IDE, emulátory, simulátory, CLI
- Přehrávání médií – možnosti, formáty, DRM
- UI a UX
- Kompatibilita aplikací a verzí OS
- Verifikace a distribuce aplikace

2.3.1 Tizen TV

Společnost Samsung si přímo vyvíjí open-source platformu Tizen pro svá zařízení. OS Tizen běží na televizích, mobilech a hodinkách Samsung. Na webových stránkách pro vývojáře aplikací pro Tizen TV [9] je dostupný Software Development Kit (SDK) pro všechny hlavní operační systémy: Apple macOS, Linux a Microsoft Windows. Platforma nabízí vyvíjet aplikace dvěma způsoby, buď ve webových technologiích, HTML5, CSS a JavaScript, s využitím Tizen Web Framework k interakci s nativními subsystémy. Nebo může vývoj probíhat v programovacím jazyce C s využitím Native API. To se hodí pro zařízení s omezenými zdroji nebo tam, kde je potřeba využít HW komponenty zařízení jako jsou hodinky a mobily.

2.3.1.1 Možnosti webové aplikace

Tizen TV nabízí pro vývoj typy API [10]:

- W3C/HTML5 APIs – standardní webová API, která jsou kompatibilní s většinou webových prostředí. Tizen TV podporuje široký rozsah W3C specifikace, ale nepokrývá ji celou.
- Tizen Device Web APIs – specifická Tizen API kompatibilní s jinými Tizen zařízeními, jsou označována jako tizen.xxx a mohou být použita k prozkoumání Tizen funkcionalit.
- Samsung TV Product APIs - specifická pro Samsung Tizen TV, tzn. nejsou kompatibilní s ostatními Tizen zařízeními, jsou označována jako webapis.xxx a mohou být použita k přístupu k Samsung TV službám, např. network manager.

2.3.1.2 Rozšiřující knihovny

Tizen TV nabízí pro vývoj i rozšiřující knihovny [10]:

- NaCl – nativní browser pluginy v programovacích jazycích C, C++
- Smart View SDK – komunikace mezi mobilním zařízením a TV
- TOAST – Samsung open-source projekt založený na Apache Cordova s cílem psát jeden kód pro více platforem
- CAPH – webový aplikační framework, který cílí na usnadnění a urychlení vývoje webových Aplikací pro Tizen, obsahuje moduly pro jQuery a AngularJS (<2)

2.3.1.3 SDK

Oficiální IDE *Tizen Studio* vylepšuje předchozí pouze upravený klon Eclipse prostředí *Tizen IDE*. Byť se jedná opět o rozšíření Eclipse, tak toto nové IDE je cestou k čistšímu a přehlednějšímu prostředí. Do IntelliJ Idea však neexistuje zatím žádný Tizen plugin, ale SDK obsahuje CLI nástroje pro sestavení, zabalení i nasazení, takže oficiální IDE není pro vývoj vůbec potřeba.

Součástí SDK jsou i emulátor (virtuální stroj) a simulátor (NodeJS aplikace) pro testování aplikací bez nutnosti nasazovat je do reálného zařízení. Simulátor má být podle dokumentace používán pouze na ladění vizuální stránky aplikace, spouští se rychle, neobsahuje však některé pokročilejší funkcionality související s HW TV. Oproti tomu emulátor má být téměř plnohodnotným Tizen zařízením, s čímž souvisí i jeho pomalejší spouštění a nasazování nových verzí aplikací. Mezi chybějící funkcionality emulátoru patří např. kamera, tzv. *Smart interakce* (gesta, hlas), 3D, Bluetooth, UHD rozlišení či přehrávání DRM videa.

2.3.1.4 Přehrávání médií

Specifikace platformy Tizen se rozděluje podle modelových řad (2015, 2016 a 2017) [11]. Tato práce se zabývá kompatibilitou pro zařízení od roku 2016, jak je uvedeno v části 1.2.2. Tizen podporuje podle specifikace široké spektrum běžných hudebních a video formátů, mimo jiné technologie i *adaptive streaming* (vysvětleno v části 2.7). Podporované formáty titulků jsou uvedeny WebVTT až od 2017, SRT nebo Closed Caption. Podporované typy zabezpečení obsahu DRM jsou PlayReady, Widevine Classic a Verimatrix, podrobněji rozebráno v části 2.7.

2.3.1.5 UI a UX

Samsung předepisuje řadu pravidel, která by měla aplikace splňovat. Uživatelské prostředí by mělo být jednoduché, čisté, konzistentní, zobrazovat zpětnou

vazbu a být estetické. *UX Checklist* [12] a *Developer checklist* [13] předepisuje tyto vyžadované vlastnosti:

- Rozlišení aplikace 1920 x 1080 pixelů nebo 1280 x 720 pixelů
- Rozměr ikony aplikace 1920 x 1080 pixelů
- Velikost písma 20 – 56 pixelů
- Chování tlačítek dálkového ovladače musí být v souladu s pravidly [14], tzn. respektovat přirozenou funkci tlačítek a v případě tlačítka zpět dodržovat historii průchodu aplikací
- Uživatel musí v každém okamžiku vědět, co se na obrazovce odehrává, a když chce provést nějakou akci tak, co proto musí udělat
- Playback tlačítka dálkového ovladače musí pracovat správně podle očekávání uživatele
- V jednom okamžiku může být vybraný pouze jeden UI prvek, jasně rozlišitelný od ostatních v jednotném stylu
- V případě delší operace musí být zobrazen nějaký indikátor postupu – nahrávání
- V případě chyby musí být zobrazena uživateli s popisem na vyřešení
- Při přehrávání videa vypnout screensaver
- Aplikace by měla naběhnout do 10 sekund
- Potvrzovací dialog při opouštění aplikace

2.3.1.6 Kompatibilita aplikací a verzí OS

Důležitým aspektem při vývoji je zaručení zpětné a dopředné kompatibility aplikací pro různé verze OS. Během psaní této práce vyšla nová verze Tizen 3.0. Kompatibilita aplikací, které cílí na nejnovější verzi API je bez problémů (navíc omezeno konfiguračním parametrem pro ohraničení nejnižší podporované verze *required_version*, podle [15]). Stejně tak s dopřednou kompatibilitou aplikací, neboli starší aplikace na novější verzi OS bude funkční, protože cílila na starší verzi OS a tím pádem u ní nebudou vyžadovány věci, které jsou nutné u aplikací nových. Komplikace nastává až v případě přechodu cílení aplikace ze starší na novější verzi, kdy je potřeba dodržet všechny vyžadované náležitosti nově cílené verze, např. pro verzi 3.0 [16].

2.3.1.7 Verifikace a distribuce aplikace

Distribuci aplikace předchází povinná verifikace aplikace, která zahrnuje splnění předepsaných pravidel a doporučení a rozděluje se do tří kategorií: *UX checklist*, *Developer Checklist* a *Launch Checklist*. Verifikace aplikace navazuje na dodržování tzv. *design guidelines*.

UX Checklist obsahuje sadu pravidel souvisejících s uživatelským rozhraním a uživatelským prožitkem. Celkově se pravidla snaží doručit uživateli přehledné grafické prostředí aplikace bez matoucích nebo ztěžujících prvků uživatelského rozhraní [12].

Developer Checklist předepisuje pravidla dotýkající se funkčnosti aplikace, navrhuje testovací scénáře, které mohou navodit chybné chování aplikace a ukazuje typické defekty aplikací [13].

Launch Checklist vyjmenovává body týkající se správného zabalení, konfigurace všech požadovaných oprávnění a samotné konfigurace aplikace, která musí obsahovat veškeré nutné náležitosti [15].

2.3.2 WebOS TV

Open-source platformu WebOS, nebo také LG WebOS, Open WebOS, HP WebOS či Palm WebOS, používá společnost LG Electronics ve svých zařízeních. Webové stránky pro vývojáře [17] nabízejí SDK pro všechny hlavní OS. Platforma podporuje vývoj webových aplikací a nativních aplikací (v jazyce C), ty však pouze s partnery LG. Další cestou k běhu stávajících aplikací v C, C++ na WebOS může být Emscripten. Jedná se o projekt založený na LLVM, který kompiluje C, C++ do vysoce optimalizovaného JavaScript kódu v asm.js formátu.

2.3.2.1 Možnosti webové aplikace

Tato platforma nabízí dvoje API:

- Web API – HTML, CSS a JavaScript podporující W3C specifikace s určitými odchylkami, jmenovitě pouze částečná podpora HTML5, CSS3, Canvas, SVG, DOM 3. V dokumentaci [18] se uvádí, že „částečná“ podpora vyhovuje z 60 – 90 % specifikaci.

- Luna Service API – esenciální služby pro přístup k HW a jiným službám zařízení.

2.3.2.2 Rozšiřující knihovny

V dokumentaci se uvádí jako doporučený JavaScript framework Enyo JS Moonstone Framework. Dále se uvádí podpora NodeJS.

2.3.2.3 SDK

SDK obsahuje upravené Eclipse IDE, volitelně Sublime Text plugin, CLI zvané Ares a emulátor (virtuální stroj). Emulátor má svá omezení, proto je potřeba testovat aplikaci i na reálném zařízení. Mezi tato omezení patří hlavně ta související s HW, např. absence DRM, TV tuneru, částečná funkčnost tzv. *Magic Remote* (žádný senzor akcelerace), absence LG Content Store, ale i omezená podpora médiálních formátů, podle [19]. Také titulky nejsou v emulátoru oficiálně podporované.

2.3.2.4 Přehrávání médií

WebOS podporuje přehrávání běžných multimediálních formátů hudby, videa a obrázků. Pro streamování podporuje mimo jiné i *adaptive streaming*. Jediným využitelným titulkovým formátem je formát WebVTT. K zabezpečení obsahu DRM lze použít PlayReady, Widevide Classic a Verimatrix [20].

2.3.2.5 UI a UX

Podobně jako u Tizen, také WebOS předepisuje základní principy pro tvorbu aplikací jako jsou jednoduchost, živost, personalizace a kozistence UI. LG předepisuje sadu pravidel [21], [22], které by měl vývojář aplikace dodržovat, chce-li projít schvalovacím procesem a publikovat aplikaci. Tato pravidla lze shrnout do těchto bodů:

- Rozlišení aplikace 1920×1080 pixelů (dále FHD) nebo 1280×720 pixelů (dále HD)
- Rozměr ikony aplikace 80×80 a 130×130 pixelů
- Rozměr obrázku na pozadí ve FHD – použit OS při načítání aplikace, tzv. *splashscreen*
- Ovládací prvky je doporučené umístit min. 20 pixelů z každé strany obrazovky, tzv. *safe area*
- Rozměr tlačítek min. 75×75 pixelů pro rozlišení FHD, respektive min. 50×50 *pixel*ů pro rozlišení HD
- Velikost písma min. 20 pixelů pro rozlišení FHD, resp. min. 14 pixelů pro rozlišení HD
- Playback tlačítka by měla být zobrazena pokud nejsou přítomna na dálkovém ovladači
- Scrollbar zajišťuje aplikace a měl by být zobrazen v případě dlouhých seznamů

- Exit tlačítko dálkového ovladače musí zobrazit TV vysílání
- Aplikace musí vždy indikovat vybraný UI prvek

2.3.2.6 Kompatibilita aplikací a verzí OS

Podle dokumentace není popsána přesná formulace, jak zajistit dopřednou a zpětnou kompatibilitu aplikace v rámci různých verzi OS. V dokumentaci se pouze uvádí, že změnou web engine z WebKit na Blink (WebOS 2.0 na WebOS 3.0) je nutno zajistit, aby aplikace dodržela tři jednoduché podmínky [23] registrace posluchačů pomocí *addEventListener*, používání prefixovaných i neprefixovaných CSS pravidel, při změně zdroje videa nutné volat *load* metodu video elementu.

2.3.2.7 Verifikace a distribuce aplikace

Verifikaci aplikace předchází dodržování *design guidelines* platformy. Samotná verifikace aplikace začíná u zajištění zpětné kompatibility a následuje *UX checklist*. Tento checklist podobně jako u Tizen platformy předkládá požadované a doporučené body, které je potřeba splnit, aby mohla být aplikace vydána v rámci LG Content Store. Mezi požadované body patří např. popis chování aplikačních tlačítek a TV ovladače. Doporučené body pak ukazují různé nejednoznačné situace, ke kterým může dojít při vývoji aplikace, a jejich řešení.

2.3.3 FirefoxOS TV

Firefox OS, open-source platforma vyvíjená do verze 2.6 (léto 2016) organizací Mozilla Foundation [24], používá na svých TV společnost Panasonic a to na modelech od roku 2014. Vývojářské webové stránky obsahující dokumentaci [25] jsou nejstručnější z porovnávaných platforem a vyžadují přihlášení platným vývojářským účtem.

Po registraci účtu a nahlédnutí do dokumentace lze identifikovat, že Panasonic oficiálně nabízí možnost vyvíjet aplikace ve webových technologiích. Cestou ke spuštění stávajících aplikací v jazycích C, C++ může být Emscripten.

2.3.3.1 Možnosti webové aplikace

V dokumentaci se uvádí, že *HTML5 v2.0 SDK* adoptuje *Smart TV Alliance Specification 3.0.0* [26]. Tato specifikace je míněna hlavně pro výrobce TV. Nicméně z dokumentu lze vyčíst, že výrobce adoptující tuto specifikaci musí podporovat HTML5, CSS3 a JavaScript ECMAscript-262 verze 5 (ES5).

2.3.3.2 Rozšiřující knihovny

Panasonic nespecifikuje žádné doporučené rozšiřující knihovny, toto nechává čistě na volbě vývojáře. Mozilla uvádí jako doporučený JavaScript framework Ember.js.

2.3.3.3 SDK

Panasonic nenabízí žádné SDK ani emulátor ke stažení. SDK je zde chápáno jako sada služeb přístupných na reálném zařízení z JavaScript kódu, např. zmiňované PlayReady DRM. Doporučeným IDE je WebIDE – součást Firefox webového prohlížeče.

2.3.3.4 Přehrávání médií

Podle dokumentace [27] má vývojář omezenější možnosti výběru multimediálních formátů, které může na této platformě použít. Nechybí však podpora běžných multimediálních kontejnerů jako jsou mp4, MPEG2TS, ASF, fmp4, video kodeku standardu H264 a zvukového kodeku standardu AAC. Mimo jiné funkce lze využít i podporu PlayReady DRM a *adaptive streaming*. Jediným podporovaným formátem titulků je TTML.

2.3.3.5 UI a UX

Panasonic nepředepisuje zdaleka tolik pravidel jako předchozí dvě platformy, přece jen některá základní ano, následující body jsou jejich shrnujím [28], [29]:

- Rozlišení aplikace 1920×1080 pixelů
- Velikost písma min. 19 pixelů, doporučeno 48 pixelů
- Screensaver se chová přirozeně – při přehrávání videa se neaktivuje
- Video nesmí být mimo zobrazovanou oblast obrazovky
- Scrollbar zajišťuje aplikace a měl by být zobrazen v případě dlouhých seznamů
- Video se načte do 10 sekund pokud je to možné
- Přehrávač videa zobrazuje postup ve videu a čas
- Aplikace zvládne 3 hodiny přehrávání videa i nečinnosti
- V případě chyby připojení by měla být zobrazena příslušná zpráva uživateli

2.3.3.6 Kompatibilita aplikací a verzí OS

Podle odpovědi z emailové komunikace s technickou podporou [30], aplikace bude fungovat správně na televizích uvedených na trh od roku 2014 a vyhoví-li tzv. *Application Guidelines* v dokumentaci [28]. Odkazovaná doporučení jsou však velmi obecného charakteru, např. použití barev a velikost písma.

2.3.3.7 Verifikace a distribuce aplikace

Pro úspěšné schválení a publikování aplikace je nutné vyhovět zmiňovaným *Application Guidelines* a projít body tzv. *Self-Checklist* [29] na reálném zařízení. Toto si kontroluje Panasonic nahlédnutím do seznamu přihlášených účtů k *HTML5 SDK TV* aplikaci – dostupná v obchodě s aplikacemi.

2.3.4 Společné prvky platforem

Z předchozí části 2.3 vyplývají některé společné prvky platforem naprosto jasně, jiné už nejsou tak zřetelné. V této části je uveden výběr a zdůvodnění společných prvků pro následnou volbu konkrétních technologií, frameworku a knihoven v části 2.9.

2.3.4.1 Jazyky a technologie

Jasným společným prvkem pro vývoj Smart TV aplikací je využití webových technologií, jmenovitě trojice HTML, CSS a JavaScript. Využití již hotových kódů v jazycích C a C++ je do jisté míry možné, ale pro potřeby této práce (VOD aplikace) zbytečné. Výsledná aplikace má splňovat požadavky z části 1.2, kterým lze dostát využitím výhradně webových technologií a to i v omezeních, která jednotlivé platformy kladou. Omezení platforem jsou blíže rozebrána v části 2.6.

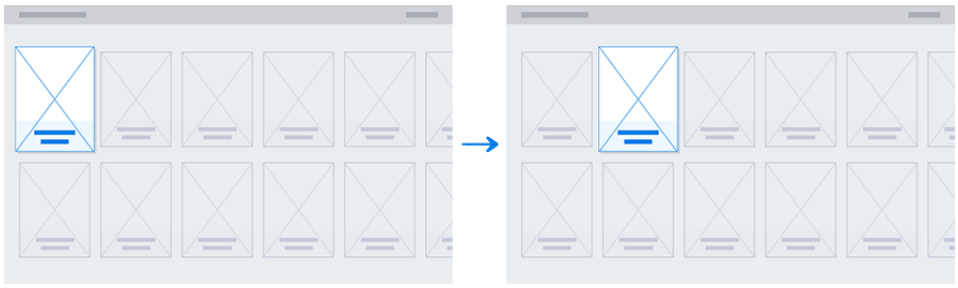
Nabízí se klíčová otázka, jaké verze technologií, resp. API, ze zmiňované trojice je bezpečné zvolit tak, aby byly v průniku všech platforem. Na základě předchozí rešerše dokumentace jednotlivých platforem je zřejmé, že nelze jednoduše vybrat verzi technologie. Některé platformy totiž dodržují W3C specifikaci [31], jiné jen částečně. Z těchto důvodů se jeví jako nutné pečlivě volit požadované vlastnosti implementované VOD aplikace a vždy kontrolovat jejich podporu napříč všemi platformami. Konkrétně je např. nutné používat CSS vendor prefixy i u vlastností, které současná jádra webových prohlížečů mají zvládat zpracovat bez nich.

2.3.4.2 UI a UX

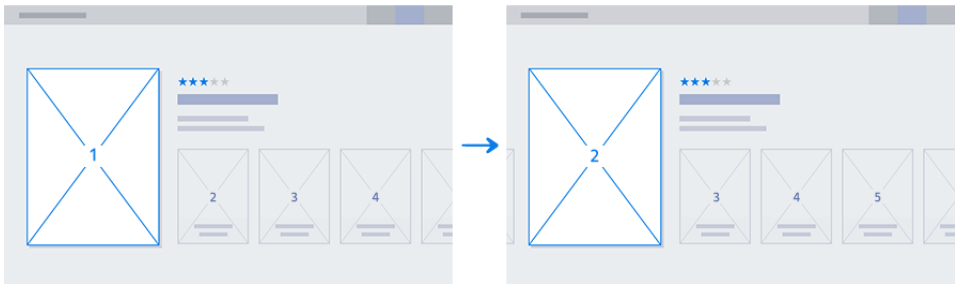
Z předchozích částí UI a UX u každé platformy je patrné, že průnik vlastností nalézt lze, ale není to bez kompromisů. Některé vlastnosti či chování je potřeba ošetřit pro každou platformu zvlášť, např. tlačítka ovladače, chování

tlačítka odchodu z aplikace, screensaver, statické zdroje aplikace (aplikační ikony, konfigurační soubory).

Některé principy návrhu lze uplatnit napříč všemi platformami, např. velikost obrazovky, velikost odsazení UI prvků od okrajů obrazovky, velikosti písma, pohyb v aplikaci. Pohyb výběru, tzv. focus, UI prvků lze řešit několika způsoby. Buďto se focus pohybuje nebo je zafixovaný a pohybují se UI prvky pod ním, jak ilustrují diagramy 2.5 a 2.6. V aplikaci je zvolen ten později zmíněný způsob pro *knihovnu filmů* z důvodu větší přehlednosti tohoto přístupu pro uživatele. Pohybující se focus nutí uživatele „běhat“ očima po obrazovce a nepřispívá k celkové přehlednosti a intuitivnosti použití aplikace. Samozřejmě tohoto přístupu nelze využít univerzálně všude, hodí se zejména pro tabulkové (mřížkové) rozložení vybratelných UI prvků.



Obrázek 2.5: Pohybující se focus [32]



Obrázek 2.6: Zafixovaný focus [32]

Nejobsáhlejší dokumentaci ze všech platforem má Tizen a to i ve směru UI návrhu. Z těchto rozsáhlých doporučení lze vycházet i pro ostatní platformy a vždy konzultovat příp. rozdíly ostatních platforem. Tímto způsobem společně s nějakým rozlišením platforem lze vytvořit abstraktní vrstvu nad platformami, která není závislá na konkrétní implementaci dané platformy, jako spíše zastřeší jejich rozličnosti pro jednotné použití v jiném modulu aplikace, blíže v části 3.2.2.

2.3.4.3 Požadavky pro publikováni aplikace

Tato práce se nezabývá publikováním aplikace v obchodech s aplikacemi jednotlivých platforem. Nicméně jsou dodržovány předepsané směrnice a pravidla, které velmi úzce souvisí především s UI a UX Smart TV aplikací.

2.4 Ovládání Smart TV aplikací

Tato část se zabývá shrnutím možných způsobů ovládání Smart TV aplikací. S různými způsoby ovládání souvisí další požadavky a směrnice, které je potřeba zohlednit při návrhu a vývoji aplikace. Stejně jako v předešlé části se i zde hledá použitelný průnik pro všechny tři platformy.

2.4.1 Dálkový ovladač

Prvním a klasickým způsobem ovládání TV je pomocí dálkového ovladače. Dálkové ovladače prošly roky inovací, ale některé jejich prvky zůstaly v zásadě nezměněné až do nynějších modelů. Bezesporu nejdéle přetrvávajícími prvky jsou numerická klávesnice a čtyři barevná tlačítka (červené, zelené, žluté a modré), později přibylo 6 tlačítek (OK, nahoru, dolu, vpravo, vlevo a zpět, dále navigační nebo směrová tlačítka). Tyto téměř neměněné prvky se začínají na některých novějších TV vytrácet ve prospěch tlačítka pro aktivaci naslouchání a jednoduší, čistší design. Nicméně i u těchto zjednodušených dálkových ovladačů přetrvává některá podoba navigačních tlačítek.

Pro obsluhu zmáčknutí tlačítka se v JavaScript používá obsluha standardních událostí, která nese mimo jiné i kód zmáčknutého tlačítka. Každá platforma má svou sadu kódů tlačítek, kterou je potřeba uvažovat zvlášť pro korektní ovládání aplikace. Událostem některých systémových tlačítek nelze naslouchat – nejdou z aplikace ovládat, např. tl. pro vypnutí TV. Platformy WebOS a Tizen poskytují různé akce při dlouhém stisku (*hold*) některých tlačítek.

Platforma WebOS poskytuje kromě klasického dálkového ovladače i *Magic Remote Control Unit* (*wand*, hůlka), který se z vývojářského hlediska chová stejně jako klasická USB myš, o té více v části 2.4.3. *Wand* má ve svém těle pohybový senzor a uživatelské pohyby ovladačem v prostoru jsou přeloženy do pohybu myši.

2.4.2 Klávesnice

Na Smart TV lze používat standardní USB klávesnici jako vstupní zařízení a využít tak celou obrazovku TV. Pro zadávání textových vstupů je pravděpodobně běžnější využít virtuální klávesnice přímo na obrazovce TV, po které se uživatel pohybuje navigačními tlačítky. Tento způsob ovládání má jeden dopad

na návrh UI u textových vstupních polí. Je potřeba UI navrhnout s ohledem, že klávesnice překryje přibližně spodní třetinu obrazovky.

2.4.3 Myš – pointer

Ovládání pomocí USB myši je podporováno na všech platformách s tím, že na Tizen a FirefoxOS lze příp. vypnout. Tento způsob ovládání se neliší od toho na PC. Podobně jako u klávesnice a ovladače si lze zaregistrovat tzv. *handler* vybrané DOM události.

2.4.4 Jiný způsob ovládání

Existují i další netradičnější způsoby ovládání Smart TV. Mezi tyto způsoby patří ovládání hlasem či gesty rukou. Oba zmíněné způsoby podporuje platforma Tizen, nicméně např. podle [33] nejsou tyto možnosti ještě připraveny pro široké použití. Jedním z důvodů je nepřesnost rozpoznávání gest v tmavé místnosti, což může být problém z povahy používání TV. Dalším možným způsobem ovládání TV může být v podobě tzv. *companion* mobilních aplikací. Tyto aplikace buďto rozšiřují schopnosti dálkového ovladače o možnost rozličných způsobů ovládání pomocí dotykové obrazovky nebo slouží jako rozšíření běžící aplikace na TV v podobě např. podrobných informací živého přenosu.

2.5 Návrh UI a UX aplikace

Tato část se zabývá návrhem UI a UX výsledné aplikace na základě srovnání platforem v části 2.3 a požadavků. Přiložené obrázky značí *low-fidelity* návrhy realizovaných obrazovek v aplikaci.

2.5.1 Společné prvky

V případě potenciálně delší práce na pozadí (síťová komunikace) zobrazí aplikace indikátor načítání (točící kolečko nebo jiný), který se bude animovat a zároveň bude animováno i jeho zobrazení, resp. zmizení.

Všechny obrazovky budou dodržovat odstup ovládacích prvků minimálně 20 pixelů od okrajů kvůli zajištění *safe area* z požadavků WebOS 2.3.2.5 a jistoty, že prvek bude viditelný celý.

Vybraný UI prvek bude obtažen jednotným bílým okrajem pro snadné rozlišení od okolních prvků.

Pohyb po vstupních polích bude možný přirozeně nahoru a dolu. Potvrzení hodnot bude možné při vybraném kterémkoli poli stiskem potvrzovacího tlačítka.

V aplikaci budou použité open-source *Material Design* ikony [34], které jsou vizuálně čisté a s jasným sdělením, co reprezentují.

2.5.2 Přihlašovací obrazovka

První obrazovka, kterou uvidí nepřihlášený uživatel, musí podporovat zadání jména a hesla a reakce na špatné uživatelské vstupy. Chybové hlášky se budou vypisovat přímo pod příslušná pole, aby uživatel přesně věděl, kde udělal chybu. Výsledný návrh je vidět na obrázku 2.7, pole jsou horizontálně vycentrovaná a dolní třetina obrazovky je volná pro zobrazení SW klávesnice na TV.

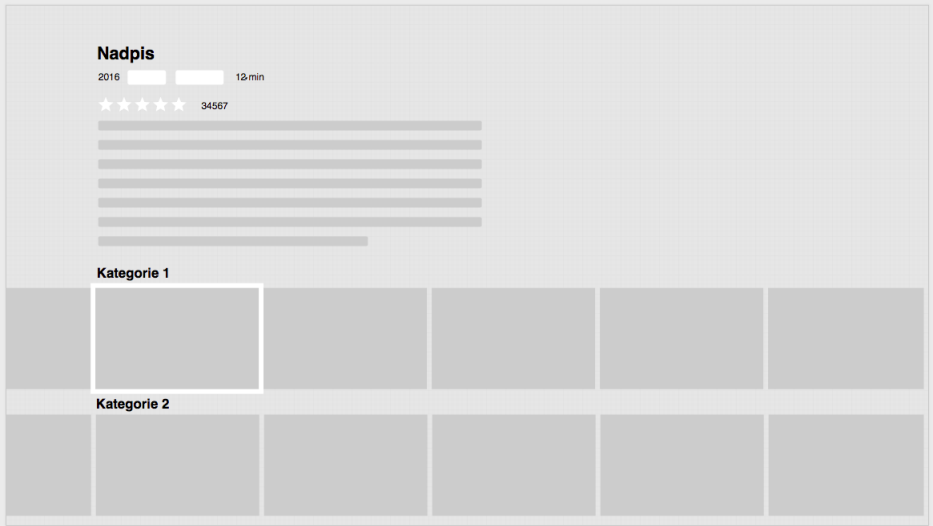
Zadávání jakýchkoli textových hodnot na TV je pro uživatele nepříjemné a zdlouhavé. Existují různé způsoby, jak se tomu vyhnout, např. kombinací mobilní aplikace s již přihlášeným účtem na stejné domácí síti jako je TV, více v části 2.6.2.

Obrázek 2.7: Přihlašovací obrazovka

2.5.3 Knihovna filmů

Knihovna filmů bude organizována jak lze vidět na obrázku 2.8 do dvou viditelných řádků (kategorií) a šesti sloupců, z toho levý sloupec bude částečně viditelný. Knihovna bude zobrazovat miniatury filmů (plakát, scény) a používat již diskutovaný fixní focus. Toto rozložení je zvoleno z důvodu, že miniatury budou dobře viditelné i z větší vzdálenosti, bude místo pro krátký popis vybraného filmu a pro náhled scény v pravém horním rohu.

Pohyb po miniaturách filmů bude možný pomocí směrových tlačítek, přičemž pohyb nahoru nebo dolu bude znamenat posun mezi kategoriemi. Pohyb směrem dolů u přiloženého obrázku 2.8 vyústí v přenesení dolního řádku na první řádek a do druhého řádku se přenesou miniatury dosud neviditelné kategorie, zároveň dojde ke změně vybraného filmu a aktualizaci podrobností o filmu. Obdobně to bude v případě pohybu vlevo či vpravo, pohybem vpravo



Obrázek 2.8: Knihovna filmů

se přenesou miniatury směrem doleva a dosud vybraný prvek bude vlevo částečně viditelný, obdobně při pohybu na druhou stranu, opět dojde k aktualizaci podrobností o vybraném filmu. Pohyb miniatur v horizontálním směru v kategorii bude simulovat nekonečný kolotoč.

Součástí změny vybraného filmu bude i změna pozadí celé aplikace odpovídající momentu z filmu. Pro dobrou čitelnost podrobností o filmu a částečnou viditelnost pozadí, bude použitý poloprůhledný přechod pod oběma řádky a zároveň pod podrobnostmi. Knihovna filmů je nejdůležitější obrazovkou aplikace, jedná se o domácí obrazovku, a proto je kladen důraz na vhodně zvolené rozložení a chování.

2.5.4 Detail filmu

Detail filmu bude zobrazovat dodatečné informaci o vybraném filmu a bude nabízet přehrávání od začátku (horní *play* tl.) nebo od zapamatované pozice (dolní *play* tl.). Na obrázku 2.9 je vidět dvě tlačítka umístěná vpravo (přehrávání) a jedno vlevo (zpět). Umístění tlačítek je záměrné a odpovídá myšlenému pohybu zleva doprava v aplikaci, tj. z knihovny filmů se uživatel dostane vpravo na detail a zpět se dostane nalevo umístěným tl. s šipkou směřující vlevo. Při výběru přehrávání se uživatel dostane dále vpravo v modelovém pohybu v aplikaci.

Pohyb po zobrazených třech tlačítkách bude probíhat přirozeným a očekávaným způsobem tak, že bude-li vybrané tl. zpět, tak tl. vpravo nebo dolu vybere horní *play* tl., následný pohyb dolů vybere dolní *play* tl., následný pohyb vlevo vybere tl. zpět. Předvybrané bude horní *play* tl., které reprezentuje nejčastější volbu uživatele v případě přehrávání filmu.

Obrázek 2.9: Detail filmu

2.5.5 Video přehrávač

Video přehrávač se bude skládat z horní lišty s nadpisem a zkráceným popisem videa, samotného videa ve střední části a dolní stavové a ovládací lišty, jak je vidět na obrázku 2.10. Horní a dolní lišty budou poloprůhledné z důvodu zachování lepší čitelnosti textu a zároveň plného zobrazení přehrávaného videa v situacích, kdy budou lišty viditelné (pozastavené video).

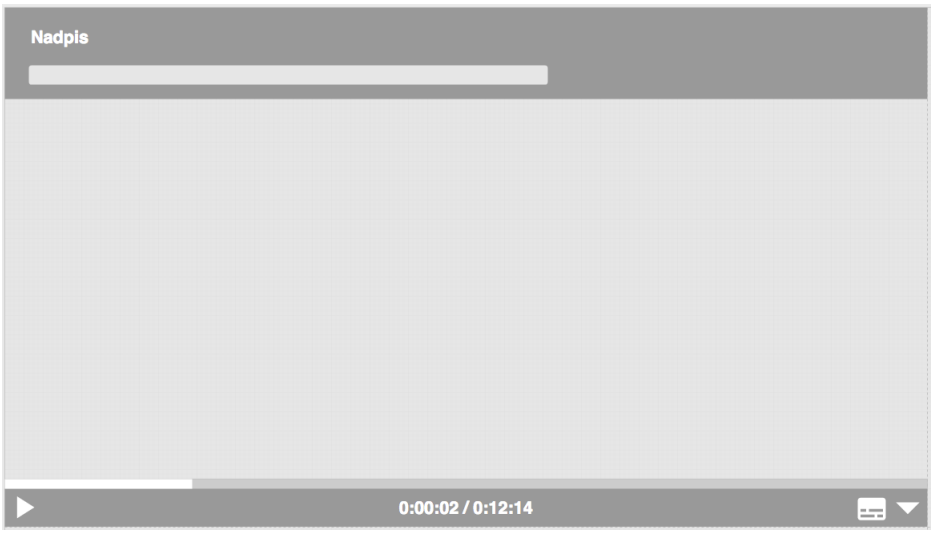
Dolní stavová lišta se bude automaticky skrývat při nečinnosti uživatele po 5 sekundách. Lišta bude sloužit jako indikátor stavu přehrávače – nahoře aktuální postup ve videu (tenká lišta bíle na obrázku 2.10) spolu s indikací již načtené části *buffered*, vlevo stav (přehrávání, pozastavení, posun dopředu a dozadu) a uprostřed uplynulý a celkový čas. Vpravo dolní lišty bude uživatel informován ikonou titulků a šipky dolů, že směrem dolů lze provést nějakou akci.

Ovládání video přehrávače

Ovládání videa bude možné především základními 6-i tlačítky, ale také odpovídajícími tlačítky pro ovládání multimediálního obsahu na dálkovém ovladači. Směrová tlačítka budou mít následující funkce a budou tak podporovat přirozené a snadné ovládání přehrávače.

- OK – přehrát, pozastavit
- šipka vpravo – bude-li skryta komponenta s výběrem znění a titulků (*controls*) – posun vpřed ve videu
- šipka vlevo – bude-li skryta komponenta *controls* – posun vzad ve videu

- šipka dolu
  - bude-li dolní lišta skryta – zobrazí ji
  - bude-li dolní lišta zobrazena – rozbalí komponentu *controls*
  - bude-li rozbalena komponenta *controls* – posune se výběr v seznamech znění a titulků směrem dolů
- šipka nahoru
  - bude-li dolní lišta skryta – neprovede žádnou akci
  - bude-li dolní lišta zobrazena – skryje ji
  - bude-li rozbalena komponenta *controls* a bude-li výběr na první položce znění – skryje ji
- zpět
  - bude-li dolní lišta skryta – opustí přehrávač
  - bude-li dolní lišta zobrazena – skryje ji
  - bude-li rozbalena komponenta *controls* – skryje ji



Obrázek 2.10: Video přehrávač – přehrávání

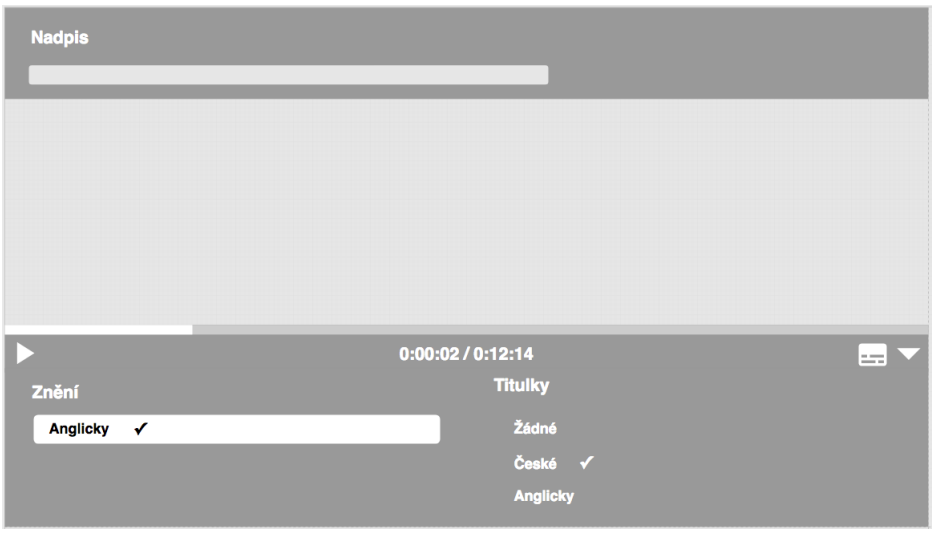
Změna znění a titulků

Samotná komponenta *controls* bude vysoká do třetiny obrazovky a organizována do dvou vertikálních seznamů vedle sebe s výběrem vlevo – znění, vpravo – titulků. Seznamy si budou pamatovat vybrané položky, půjde vybrat

vždy kombinaci jedno znění a jedněch titulků. V případě rozbalené komponenty *controls* nepůjde ovládat samotný přehrávač, ovládání bude dodržovat kontext a zaměření uživatele na danou činnost – při rozbalení této komponenty na výběr znění nebo titulků. *Controls* bude mít následující chování, vyjma případů zmíněných dříve.

- OK – potvrzuje výběr
- zpět – skryje *controls*
- šipka vpravo – posune focus do pravého sloupce s titulky, pokud tam již není
- šipka vlevo – posune focus do levého sloupce se zněním, pokud tam již není
- šipka nahoru – posune focus nahoru v aktuálním seznamu, příp. na poslední položku levého seznamu
- šipka dolu – posune focus dolů v aktuálním seznamu, příp. na první položku pravého seznamu

Na obrázku 2.11 je vidět jak bude vypadat komponenta *controls*. Focus bude odlišen bílým pozadím a černou barvou textu, zde bude odlišnost oproti označení výběru jinde v aplikaci. Rámeček by vypadal rušivě, tvořil by přechod tmavého pozadí, bílého rámečku, tmavého pozadí a bílého textu vybraného prvku.



Obrázek 2.11: Rozbalená komponenta *controls*

Posun ve videu

Na obrázku 2.12 je vidět jak bude uživateli zobrazen stav posunu ve videu uprostřed obrazovky poloprůhledným boxem s uplynulým časem filmu a rychlostí posunu. Alternativní způsob zobrazení posunu bývá ve formě podobného boxu s časem, ale pohybujícím se s průběhem videa nad lištou průběhu. Tento alternativní způsob nutí uživatele v případě jeho krátké nepozornosti zbytečnému hledání aktuálního stavu.

Rychlosti posunou budou 2x až 128x v posloupnosti jako mocniny čísla 2. Mechanismus ovládání posunu ve videu naznačují následující body.

- OK – zruší posun, spustí přehrávání
- šipka vpravo – první stisknutí vyvolá posuv vpřed rychlostí 2x (zruší příp. posun vzad), další zvýší rychlost
- šipka vlevo – bude fungovat analogicky šipce vpravo

Obrázek 2.12: Video přehrávač – posun vpřed ve filmu rychlostí 2x

Tento způsob posunu ve video byl zvolen z důvody, že není dostupné žádné API, které by zobrazovalo náhledy a zároveň umožňuje rychlý posun oproti alternativnímu opakovanému přeskakování o fixní časový úsek. Tento alternativní způsob lze sice rozšířit o možnost držet tlačítko pro posun a tím ho zrychlit, ale tento způsob může způsobovat zmatení uživatele.

2.5.6 Obrazovka voleb

Na tuto obrazovku se uživatel dostane z knihovny filmů tl. zpět na ovladači. Obrazovka bude sloužit jako potvrzovací bod pro uživatele, že skutečně chce

opustit aplikaci nebo se chce odhlásit ze svého účtu. Na obrázku 2.13 je vidět tři tlačítka s funkcemi zleva: zpět (levé šipka), odhlásit (symbol přepnutí uživatelů) a odejít (vypínací symbol). Pozadí obrazovky bude částečně průhledné a bude tak evokovat zobrazení dialogu. S přidanými funkcemi v další verzi aplikace může tato obrazovka bez problémů pojmout např. tlačítko nastavení.

Obrázek 2.13: Obrazovka voleb, zleva – zpět, odhlásit, odejít

Pohyb po třech zmíněných tl. bude možný očekávaným směrem vlevo (nahoru) a vpravo (dolu). Potvrzení vybraného zpět vrátí uživatele zpět na knihovnu filmů. Potvrzením prostředního tl., aplikace uživatele odhlásí a zobrazí přihlašovací obrazovku. Potvrzením pravého tl. se aplikace vypne. Předvybrané bude tl. zpět z důvodu, že chceme uživatele udržet v aplikaci.

2.6 Omezení Smart TV aplikací

Následující části poukazují na některá hlavní omezení vývoje Smart TV aplikací, vztaženo na předmětné platformy. Zároveň se opět hledá společný jmenovatel platforem uplatnitelný při návrhu či realizaci aplikace.

2.6.1 Velikost displeje a UI prvků

Smart TV aplikace musí být navrhovány s ohledem na to, že běží na TV, které jsou běžně několik metrů od uživatele, přibližně 3 m a mají úhlopříčku průměrně do 1,5 m. Z těchto důvodů je nutné zajistit dostatečně velké ovládací prvky a písma, v neposlední řadě pak uživatele nemást komplikovaným UI.

2.6.2 Uživatelské vstupy

Uživatelé si chtějí zpravidla u TV odpočinout a nechtějí řešit nic komplikovaného, což by měly aplikace respektovat. Jednou z nejnepříjemnějších věcí, která tak může uživatele potkat při používání aplikace, je hned na začátku přihlašovací obrazovka, kde je uživatel nucený zadávat textový vstup. Někdy je tento krok nevyhnutelný a měl by být požadovaný pouze jednou na začátku používání aplikace. Existuje mnoho způsobů, jak lze uživateli jeho přihlášení zjednodušit. Některé služby jako Spotify toho úspěšně využívají, byť se jedná o přehrávač hudby. Přihlášení vypadá tak, že uživatel je přihlášen ke službě v aplikaci na svém telefonu a je-li TV na stejné Wi-Fi síti, pouze jednou klikne, příp. vybere z dostupných zařízení na TV. Podobné způsoby přihlašování, které využívají mobilní telefon je možné založit na tzv. Zeroconf [35], popis tohoto protokolu je nad rámec rozsahu této práce.

2.6.3 Přepínání mezi aplikacemi

Na TV stejně jako na chytrém telefonu může běžet několik aplikací a uživatel mezi nimi může přepínat. Každá ze tří platforem se k tomuto přepínání chová jinak, ale společnými prvky jsou to, že je potřeba zajistit, aby se při návratu do aplikace (pokud není spuštěna znovu) obnovil focus a zkontrolovala konektivita.

2.6.4 Dlouhý běh aplikací

Určení Smart TV aplikací je hlavně do obýváků ke sledování multimediálního obsahu, ať už filmů, TV přenosů či hraní her. Všechny tyto aktivity mají společné to, že u nich stráví uživatel poměrně dlouhou dobu oproti většině webových aplikací, kromě např. e-mailových klientů jako Gmail od Google. Z předchozího plyne, že aplikace musí zvládat dlouhý běh bez přerušení, což má několik praktických důsledků. Asi nejdůležitějším je ten, že aplikace musí správně uvolňovat alokovanou paměť, správně vypínat a zapínat screensaver, kontrolovat připojení k Internetu nebo obnovit focus při návratu z jiné aplikace.

2.6.5 Omezený výpočetní výkon

Současné TV se dobře vypořádají s přehráváním velkých video souborů, některé i 4K, tzn. video o rozlišení 3840×2160 pixelů. Ve specifikacích TV lze mnohdy nalézt i čtyřjádrové procesory, které by mohly zajišťovat dostatečný výpočetní výkon, není tomu tak vždy, vyzkoušeno v části 4.3.2.

2.6.6 Omezený úložný prostor

Většina aplikací potřebuje ukládat data nutná pro personalizaci aplikace, např. uživatelské volby v nastavení. Společným použitelným úložištěm napříč všemi platformami je pouze LocalStorage. Kapacitu tohoto webového úložiště garantuje WebOS a Tizen na 5 MiB, v případě překročení je vyvolána chyba (Tizen). LocalStorage na FirefoxOS má kapacitu pouhých 524288 bajtů, tj. 0,5 MiB.

2.6.7 Ladění aplikací

K ladění aplikací poskytují platformy různé úrovně podpory. Ladění Smart TV aplikací není jednotné a většinou nelze použít *alert* ani *console.log* pro jednoduchý výpis hodnot. Například na FirefoxOS při použití vzdáleného výpisu pomocí služby *console.re* (využívá technologii WebSockets), se aplikace okamžitě po startu zavře. Platformy Tizen a WebOS deklarují ladění (nejen výpis) pomocí Web Inspector, který je založen na Google Chrome Developer Tools. Propojení s Developer Tools velmi často nefunguje nebo jen velmi krátkou dobu, než dojde ke ztrátě spojení s aplikací, příp. nelze aplikace ani spustit v ladícím režimu kvůli chybám SDK. Nicméně univerzálním návodem na ladění Smart TV aplikací je konvenční a velmi neflexibilní výpis hodnot přímo na obrazovku.

2.7 Možnosti přehrávání DRM obsahu

Většina současného multimediálního obsahu je zatížena autorskými právy, která jsou standardně zajišťována pomocí DRM ochrany. DRM je ochrana multimediálního obsahu, která kombinuje symetrické a asymetrické šifrování na straně serveru a dešifrování na straně klienta.

Klíčovými prvky ve způsobu této ochrany jsou většinou API server, licenční server, mediální server a klient, který DRM šifrovaný obsah dešifruje a posílá surová data do přehrávače.

2.7.1 Dostupné DRM technologie na Smart TV

Každá platforma podporuje nějakou formu DRM ochrany. Těchto technologií existuje celá řada a tabulka na stránce [36] zachycuje přehled jejich dostupnosti na různých, nejen Smart TV, platformách. Tizen nabízí PlayReady od Microsoft, Widevine Classic od Google a Verimatrix. WebOS podporuje PlayReady, částečně Widevine Classic a Verimatrix.

Panasonic je součástí SmartTV Alliance, která spojuje několik výrobců TV (LG, Philips, Toshiba, Panasonic) a snaží se být protiváhou takovým gigantům jako je např. Samsung. Tato iniciativa má za cíl vytvořit standardy, aby

tak sjednotila roztříštěný svět TV výrobců a nabídla vývojářům aplikací jednoduší a jednotné prostředí pro vývoj aplikací. SmartTV Alliance předepisuje nutnou podporu PlayReady a volitelnou podporu Widevine [26], FirefoxOS na Panasonic TV podporuje pouze PlayReady. Společnou DRM technologií těmto třem platformám se jeví PlayReady.

Provozovatel VOD služby musí většinou uvažovat nejen Smart TV, ale i jiné platformy a různá zařízení, mobilními zařízeními počínaje a herními konzolemi konče. Aby došlo k pokrytí co největšího počtu zařízení, je potřeba podporovat minimálně Playready a Widevine. Podle [36] tím ale provozovatel přijde o uživatele produktů firmy Apple, chce-li provozovatel zahrnout i tyto uživatele, musí podporovat i technologii FairPlay od Apple.

Nedílnou součástí VOD aplikací je kromě DRM ochrany i adaptivní streamování, které zajišťuje nepřerušené přehrávání za cenu nižší kvality obsahu. Typ DRM většinou determinuje i typ video streamu. Podle [10] Tizen podporuje MPEG-DASH (ISO standard [37]), HLS (HTTPS Live Stream) od Apple a Smooth Streaming od Microsoft. WebOS zvládá přehrát podle [20] pouze HLS a FirefoxOS zase jen Smooth Streaming, podle [27].

Adaptivní streamovací formát společný všem platformám nalézt nelze. Tizen má nejvíce možností, takže výběr ohraničují platformy WebOS a FirefoxOS, které podporují pouze HLS (Verimatrix) resp. Smooth Streaming (PlayReady), Tizen podporuje obě technologie.

Tizen

Na Platformě Tizen je nutno použít speciální objekt *avplay* z *webapis* a také přidat potřebná DRM oprávnění do konfiguračního souboru (obdoba oprávnění v Google Play obchodě). Dokumentace Tizen poskytuje podrobný návod [38], jak přehrávání DRM v Tizen funguje a jak ho zprovoznit. Na stavovém diagramu v návodu je vidět, že ve stavu IDLE jsou dostupné metody `webapis.avplay.setDrm` a `webapis.avplay.setStreamingProperty`, které slouží pro nastavení vlastností DRM (URL adresu licenčního server, *customData*, URL adresu playlistu, dále jen atributy DRM).

WebOS

WebOS rovněž publikuje podrobný tutoriál korektního iniciování přehrávání DRM obsahu [39]. V tomto případě je využíván HTML video element, kterému se nastaví správná hodnota atributu *type* začínající například takto: `video/mp4;mediaOption=`. Hodnota je získána z Luna Service API sérií asynchronních volání, která jsou iniciována s atributy DRM.

FirefoxOS

Platforma FirefoxOS neposkytuje žádný tutoriál zabývající se DRM. Nicméně SmartTV Alliance poskytuje jednoduchý přehled [26], ze kterého je patrné,

že průběh inicializace přehrávání DRM obsahu je podobný WebOS. Rovněž je použit HTML video element, rozdíl nastává v použití tzv. *OIPF DRM Agent* objektu. Tento speciální objekt je naplněn a lze referencovat až na reálném zařízení pomocí interního Panasonic SDK.

2.8 Zabezpečení aplikace

Obsahuje-li aplikace přihlašovací modul, do kterého uživatel zadává citlivé údaje, zabezpečení přenosu dat je naprostou nutností. Dalším aspektem je pak zabezpečení zdrojových kódů aplikace.

2.8.1 Synchronizace dat

Z diagramu 2.4 je patrné, že aplikace komunikuje protokolem HTTPS. Na straně webového serveru je zabezpečené spojení podloženo certifikátem vystaveným pro potřeby této práce pomocí otevřené a automatizované certifikační autority *Let's Encrypt*, která je zdarma [40]. Na straně autentizačního modulu implementovaného systému je zabezpečené spojení realizováno prostřednictvím služby *Webtask.io*.

2.8.2 Zabezpečení zdrojových kódů

Webové aplikace nebo konkrétněji Single Page Applications (SPA) jsou JavaScript aplikace renderované na klientské straně a trpí jedním podstatným neduhem. Standardně se celý zdrojový kód nachází u uživatele, kde se před použitím nahrají a spustí potřebné skripty. Tímto způsobem se může útočník snadno zmocnit zdrojových kódů. Samozřejmě by měly být užito metody „minify“ (minimalizace) na zdrojové kódy, už kvůli rychlosti stažení zdrojů. Zpětné složení kódů není extrémně složité, ale proměnné a metody pak nedávají bez mapovacího souboru příliš smysl, nicméně struktura programu může být obnovena. Dalším krokem jsou metody tzv. „uglify“ či „scramble“, které přejmenují proměnné na něco podobného jako `t[0x567]` a ještě kód „zamíchají“, což může mít za důsledek jeho nefunkčnost a nesnadné ladění.

Výsledná aplikace užívá metod „minify“ a „uglify“, které jsou součástí sestavovacího procesu oficiálního CLI nástroje vybraného frameworku, podrobněji v části 2.9.

2.9 Výběr technologií

V této části jsou zdůvodněny volby použitých technologií, jako konkretizace části 2.3.4, kde je nastíněno kudy se lze vydat při výběru. Smart TV aplikace jsou v naprosté většině webové aplikace a toto prostředí, konktrétně JavaScript (JS), v posledních pěti až sedmi letech prodělal extrémní vývoj. Po dlouhé době stagnace a statutu doplňkového nástroje pro tvorbu webu se

objevilo ohromující množství knihoven, frameworků nebo jiných nástrojů. Rapidní vývoj odstartovalo nejspíše serverové JS běhové prostředí NodeJS. NodeJS je open-source, multiplatformní systém založený na enginu V8 od Google a několika standardních knihovnách pro komunikaci s HW.

Pravděpodobně druhým motorem, který nastartoval takové tempo vývoje je *npm* – výchozí NodeJS *package manager*, na který může kdokoli publikovat svou knihovnu a ostatní ji mohou používat ve svých projektech. Tato otevřenost má však i svou stinnou stránku pro uživatele knihoven. Čím více se do projektu přidává tzv. *dependencies* neboli *závislostí* na knihovnách třetích stran, tím více roste riziko budoucích problémů. Český ekvivalent vcelku výstižně pasuje do tohoto kontextu a podtrhuje důležitost rozhodování o přidání další knihovny. Každá přidaná knihovna do aplikace s sebou přináší kromě výhod i zvýšení komplexity a nároky na údržbu. Toto ostatně potvrzuje nedávný případ, kdy tvůrce jedné malé knihovny po sporech o její název, knihovnu odstranil z repozitáře npm a tisíce závisejících projektů bylo nefunkčních [41]. Zmíněnou situaci už v npm vyřešili v podstatě znemožněním balíček po 24 hodinách od publikování odstranit.

2.9.1 Jazyk a framework

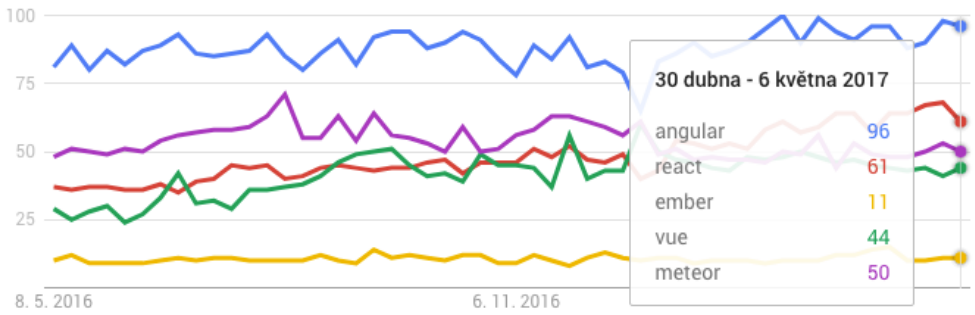
Výstupem této práce je klientská aplikace v JS, jde o skutečnou dynamickou aplikaci, nikoli webovou prezentaci. Proto je vhodné vybrat vhodnou JS knihovnu nebo framework pro tvorbu SPA. V současné době se nabízí celá řada JS knihoven a frameworků, např. ReactJS[1] [42] od společnosti Facebook nebo Angular (od verze 2) [43] od Google. Bezpochyby těmito dvěma výčet nekončí, ale alespoň podle diagramu 2.14 jsou to nejživější technologie v porovná s dalšími JS frameworky. Tyto trendy odráží do jisté míry popularitu technologie, ale neměl by se na nich zakládat výběr. Každý projekt má svá specifika, která je nutno mít při výběru na paměti.

2.9.1.1 Nastavení požadavků

Podrobnému shrnutí výběru mezi Angular a ReactJS se věnují i články [44] a [45], které předkládají stěžejní body volby a popis obou technologií. Avšak konečný výběr je ponechán na vývojáři a jeho promítnutí požadavků na projekt do zmíněných bodů. První zmiňovaný článek shrnuje otázky, které by se měly při výběru klást. Otázky se týkají dílem projektu samého, dílem sebereflexe vývojáře či týmu.

U samotného projektu by nás mělo zajímat, jak je framework vyspělý a kdo za ním stojí, jaké nabízí funkcionality, jaká jsou užitá programovací paradigmata a jaký má ekosystém. V případě sebereflexe lze zvažovat, jak

[1] ReactJS je sice jen knihovna, avšak ve spojení s dalšími potřebnými knihovnami ho lze považovat za rovnocennou technologii frameworku Angular.



Obrázek 2.14: Google trends: porovnání vyhledávání výrazů v kategorii Internet a telekomunikace za poslední rok. Čísla představují relativní zájem ve vyhledávání vzhledem k nejvyššímu bodu grafu pro danou oblast a dobu. Hodnota 100 představuje nejvyšší popularitu výrazu. Hodnota 50 znamená, že měl výraz poloviční popularitu. Skóre 0 značí popularitu nižší než 1 % nejvyšší popularity. Zdroj [46]

snadno se bude schopen vývojářský tým nový nástroj naučit a jaká bude vývojářská zkušenost.

Vyspělost porovnávaných frameworků je podobná a za oběma stojí velké společnosti, které je využívají ve svých kritických aplikacích.

2.9.1.2 Funkcionality

Některé nabízené funkcionality oba frameworky sdílí, např. ochrana proti XSS, jednotkové testovací nástroje, UI založené na komponentách nebo CLI nástroje. V dalších případech se liší, např. Angular používá *dependency injection*, které je vhodné u větších projektů, oproti ReactJS. Angular rozšiřuje HTML o vlastní DSL, ReactJS staví na JSX. Oboje má své výhody i nevýhody. Čitelný deklarativní způsob, který zvolil Angular od sebe odděluje vzhled a logiku komponent, což zní správně, na druhou stranu je potřeba další nástroje do *build chain* (součátí *angular-cli*) a učit se nové direktivy. JSX je oproti tomu čistý JavaScript nebo TypeScript s dodatečnými funkcemi, tím pádem se lze spolehnout na statickou analýzu kódu. JSX kód je součástí logiky komponenty, což je pravděpodobně největší nevýhoda, ale zároveň lze využít funkcionálního přístupu a skládat komponenty z krátkých znovupoužitelných funkcí.

Angular využívá silně typového systému TypeScript (TS), ReactJS zase Flow anotací. Migrace netypovaného JS bývala jednodušší s Flow, kde se postupně přidávaly typy, TS od verze 2.3 dokáže totéž [47]. TS je nadmnožina současného JS, ve které lze modelovat složitou business logiku a procesy.

Angular masivně adoptuje open-source knihovnu RxJS (Reactive Extensions for JavaScript). Bez alespoň základná znalosti této knihovny se v Angular nelze obejít. Tato velmi rozsáhlá knihovna pracuje na principu, že téměř vše lze považovat za stream neboli proud dat. Taková úvaha dává dobrý smysl,

zvláště v prostředí front-end, kde jako stream mohou být interpretovány uživatelem vyvolané události (kliknutí, pohyby myší), postupně přicházející data (WebSockets, apod.) či různé časovače (*setInterval*). RxJS knihovna poskytuje velké množství transformačních funkcí a operátorů pro usnadnění zpracování dat, přesto je nutné počítat s komplexitou knihovny.

2.9.1.3 Ekosystém

Oba zmíněné frameworky mají kolem sebe ohromnou komunitu lidí, kteří produkují skvělé nástroje a knihovny. ReactJS by se bez takové podpory ani neobešel. React jako knihovna neobsahuje řadu naprosto nutných funkcionalit pro tvorbu SPA, např. *router*, *forms*, *animations*, *http*, nicméně díky silné komunitě nechybí kvalitní a ověřené knihovny, jen je nutné vybrat tu správnou, což vyžaduje čas na evaluaci různých možností. Angular v základu nabízí vše potřebné a změny klíčových modulů by byly kontraproduktivní.

V ReactJS prostředí se pro zpracování stavu aplikace a komponent hojně používá knihovny Redux. Úvaha je taková, že aplikace má svůj jeden centralizovaný stav, od kterého se odvíjí všechno v aplikaci. Tento stav může být poměrně rozsáhlý, a proto ho lze rozdělovat na menší celky. Stav se mění pomocí vyvolání akce, která je zpracována v příslušné *pure* funkci – *reducer*, daného podstavu. Mezi největší výhody takového zpracování stavu patří jednoduší vývoj, testovatelnost, snadná podpora *undo*, apod. Protějškem Redux ve světě Angular je rozáhlá knihovna *@ngrx/store*, která poskytuje mimo jiné i vývojářské nástroje pro ladění. Zpracování stavu se věnuje více část 2.10.

V případě klasických webových stránek je důležité SEO, u Smart TV aplikací nikoli. To je dáno tím, že jsou distribuovány přímo v obchodu s aplikacemi daného výrobce, proto se tím tato práce ani nezabývá. Nicméně pro správné SEO je zjednodušeně nutné renderovat aplikaci na straně serveru, *server-side rendering*, a posílat klientovi v podstatě předpřipravenou HTML stránku, kterou mohou vyhledávače správně indexovat. Oba rivalové tento způsob renderování podporují, nikoli však v základu. Pro ReactJS lze využít knihovnu Next.js a pro Angular knihovnu Angular Universal nebo Angular-SSR.

2.9.1.4 Shrnutí rozdílů porovnávaných frameworků

Předchozí popisné části obou frameworků se mohou shrnout do následujících bodů, podle [45] je doporučován ReactJS pokud:

- je aplikace středního až většího rozsahu,
- je preferováno dělat vlastní specializovaná rozhodnutí o tom, jak SW pracuje,
- je preferováno v případě chyby tzv. „fail fast and early“,
- nevadí psaní více kódu,

- není obava z patentové žaloby v případě střetu zájmů s firmou Facebook.

Naproti tomu je doporučován Angular pokud:

- je preferován silně typový systém s využitím TypeScript od Microsoft,
- je aplikace rozsáhlá a náročná na výkon,
- není v týmu JS purista a nevadí učit se proprietární direktivy,
- je možno věnovat více času učení pokročilého frameworku,
- nevadí větší velikost sestavené aplikace (blíže v části 2.9.3),
- je požadována pravidelně udržovaná a vyvíjená technologie.

2.9.1.5 Zdůvodnění výběru

Na základě požadavků v části 1.2 na vytvoření robustního základu pro potenciálně rozsáhlou VOD aplikaci v moderní technologii a na základě předchozích odstavců se jeví jako správné rozhodnutí zvolit framework Angular. Tato volba má řadu výhod v podobě kompletního balíku potřebných funkcionalit, např. *dependency injection*, *router*, *forms*, *animations*, *http* modul, adopce RxJS apod. Další výhodou[2], u této volby je fakt, že je možné používat silně typový systém TypeScript, který je pochopitelnější pro Java či C# vývojáře než klasický JS. TS lze tzv. „transpilovat“ až na ES3. Transpilace znamená převedení zdrojových kódů z TS do JS určité verze specifikace EcmaScript, např. zmiňovaná verze 3. Tento aspekt se hodí právě při vývoji Smart TV aplikací, kdy aplikace běží ve starších běhových prostředích. Aby bylo možné používat některé novější funkce a zároveň cílit na starou specifikaci, je nutné zakomponovat do sestavovacího procesu i *polyfills* [3]. Více se sestavovaní aplikace věnuje část 2.9.3.

Komponenty

Angular podobně jako ostatní současné JS frameworky staví na konceptu komponent[4], které mají mezi sebou úzké vazby a lze je snadno skládat. Tímto způsobem je možné rychle vybudovat komplexní UI, kde komponenty mají jasně definované rozhraní, a proto jsou snadno znovupoužitelné. Komponenty

[2] Nikoli rozhodující výhodou, protože ReactJS lze taktéž použít s TS

[3] *Polyfills* spolu s *shims* jsou knihovny, které přináší novou funkcionalitu do starších prohlížečů. *Shim* přináší nová API pomocí existujících prostředů. *Polyfill* je *shim*, který kontroluje kompatibilitu prohlížeče a příp. instaluje vlastní implementaci nového API. [48, str. 405]

[4] Komponenta v Angular je JS třída, která je dekorovaná dekorátorem *@Component()*, který umožňuje nastavit metadata komponentě, jako jsou HTML šablona, CSS styly, animace atd.

tvoří strom komponent, na kterém se kontrolují případné změny modelů prostřednictvím tzv. binding (vazby). Vazba komponent je dvojího typu: atributů (vstupy) a událostí (výstupy).

Výkonnost

Komponent v aplikaci může být velké množství, k tomu ještě hluboko zanořených ve stromu komponent. Dojde-li ke změně v některé komponentě, musí se v Angular ve výchozím stavu zkontrolovat všechny komponenty, jestli se jich změna nedotkla. Tato operace může být velmi náročná. Angular nabízí možnost, jak toto optimalizovat. Lze nastavit, aby se komponenta měnila jen na základě jejích vstupů, tzv. *ChangeDetectionStrategy.OnPush*. Detekce stavu pak neprobíhá na všech komponentách při jakékoli změně, ale jen při změně vstupů. Tuto optimalizaci výsledná aplikace využívá. Ke změnám vstupů se často využívá tzv. immutability (neměnnost), kdy se místo modifikace atributů objektům vytváří objekty nové s pozměněnými vlastnostmi.

2.9.2 Knihovny

Aplikace využívá minimální počet knihoven třetích stran. Jednou z použitých knihoven je *@ngrx/store* spolu s *ngrx-store-localstorage*, o který se pojednává v části 2.10. Další použitá knihovna je *@ngx-translate*, která slouží k překladu aplikace do různých jazyků. Kromě těchto knihoven, jejich přidružených modulů a samozřejmě Angular frameworku spolu s knihovnami zajišťujícími zpětnou kompatibilitu (*polyfills*) pro starší webové prohlížeče, nejsou použity žádné další knihovny. To umožňuje právě volba frameworku Angular, který obsahuje téměř všechny potřebné funkcionality nutné pro vývoj SPA už ve svém základu na rozdíl např. od ReactJS.

2.9.3 Vývojové a sestavovací prostředí

Vývoj aplikace zahrnuje povinný překlad zdrojových kódů. Z důvodu vývoje v TS a SCSS (jazyk pro styly) je nutný překlad do JS a CSS, aby aplikace mohla fungovat i na Smart TV. Výhody použití TS jsou nastíněny v dřívější části. Výhody SCSS oproti běžným kaskádovým stylům jsou např. v možnosti deklarovat proměnné, čehož výsledná aplikace hojně využívá, aby se vyhnula syndromu magických konstant. Dalším důvodem proč je nutný překlad stylů, je ten, že styly komponent v Angular dodržují princip enkapsulace.

Oficiální nástroj *angular-cli* umožňuje sestavení (build) pro vývoj i produkci. Vývojový build aplikace sleduje změny ve zdrojových kódech, které automaticky vyvolají sestavení a pomocí technologie WebSockets znovu načtou webovou stránku s aplikací. Produkční build podporuje *tree shaking* a AOT (Ahead-of-time kompilace) bez nutnosti cokoli nastavovat. Tree shaking je metoda sestavení aplikace, které zajišťuje, že se do výsledného balíčku nedostanou metody a třídy, které aplikace nepoužívá. AOT je v Angular protipól

JIT (Just-in-time kompilace). Při použití AOT, Angular kompilátor kompiluje zdrojový kód dopředu pouze jednou, oproti JIT. AOT skýtá výhody mimo jiné v tom, že výsledná aplikace má menší velikost a rychleji se renderuje. CSS vendor prefixy jsou taktéž automaticky přidány do výsledných stylů.

2.10 Volba zpracování vnitřního stavu aplikace

Každá aplikace má nějaké vnitřní stavy, mezi kterými přechází. Tyto stavy mohou být definovány buďto implicitně stavem objektů v systému nebo explicitně centrálním stavovým úložištěm či repozitářem (v operační a/nebo persistentní paměti). K přechodu mezi stavy může opět docházet implicitně změnami atributů objektů v systému nebo explicitně např. pomocí tzv. *reducers*. Reducer je *pure*[5] funkce, která zajišťuje dobře definovaný přechod z jednoho stavu do druhého, značí se signaturou `(previousState, action) => newState`. Důležitou podmínkou pro správnou funkci je neměnnost stavu, neboli stav se nesmí měnit, nýbrž je potřeba vytvářet vždy nový. Úložiště může být rozděleno na menší logické celky (podstavy), které se vzájemně neovlivňují.

Oba přístupy mají své výhody i nevýhody. Zjevnou výhodou implicitního přístupu je počáteční snadná implementace, která se ale s přidáváním dalších funkcionalit do aplikace komplikuje. Použití konceptu *pure* funkcí pro zpracování vnitřního stavu vyžaduje zpočátku více času na osvojení, protože se jedná o změnu programovacího paradigmatu. Jedním z důvodů nesnadného přechodu na nové paradigma je velká popularita objektově orientovaného přístupu, který se nadužívá paušálně všude a upozaďuje v mnohých případech vhodnější funkcionální přístup.

Použití centrálního stavového úložiště poskytuje mnoho výhod, např. snadnou implementaci pohybu v historii provedených akcí v aplikaci, persistenci samotného stavu či jeho menších částí nebo jednodušší testovatelnost.

Výsledná aplikace využívá přístupu centrálního úložiště stavu pro stěžejní části aplikace jako je video přehrávač, knihovnu filmů a další, podrobněji v části 3.2.4. Rozhodně se ale nejedná o vyčerpávající použití tohoto přístupu a přístupů v částech 2.10.1, 2.10.2 a 2.10.3 jako spíše o selektivní použití na nejvhodnějších místech aplikace.

2.10.1 Změny stavu založené na událostech

Zajímavým způsobem pohledu na nejen uživatelské aplikace je pohled, který považuje veškeré změny stavu za vyvolané události nad nějakým zdrojem dat. Zdroj dat je zde chápán jako *stream* neboli kolekce dat v čase. Zdroje událostí mohou být např. akce uživatele, časové funkce nebo odpovědi ze serveru.

[5] *Pure* funkce je taková funkce, která nemá vedlejší efekty, pro daný vstup vrací vždy stejný výstup a není závislá na žádném externím stavu.

Tento pohled sjednocuje a zjednodušuje zpracování na první pohled odlišných dat z různých zdrojů.

2.10.2 Reakce na události

Reakce na vyvolané události lze agregovat jako akce vyvolané právě nad centrálním úložiště stavu. Zpravidla ne každá událost vyvolá akci na změnu stavu, naopak události se nějakým způsobem předzpracovávají.

Typickým příkladem je vyhledávací pole, které našeptává výsledky. Zde každý úhoz uživatele vyvolá příslušnou událost, a kdyby každá událost vyvolala dotaz na server, tak by byl server zbytečně zatěžován nerelevantními dotazy. Z tohoto důvodu se uživatelské události filtrují, používá se zpoždění, poslední ze série dotazů atd. K tomuto zpracování je vhodná zmiňovaná RxJS knihovna, která nabízí příslušné funkce a operátory. V příkladu vyhledávání by se akce na změnu stavu vyvolala po příchodu odpovědi ze serveru.

Podobně jako při použití centrálního úložiště stavu se jedná o změnu programovacího paradigmatu.

2.10.3 Reakce na změny stavu

Registrací na změny nad stavem lze reagovat na změny stavu, které vyvolaly nějaké akce. Typickou reakcí je nějaká změna UI, např. zobrazení našeptávaných hodnot u vyhledávání. Změny UI souvisí s vazbami, které pojí UI se stavem aplikace a na základě změn vazeb se vyvolají i změny UI.

2.10.4 Perzistence dat

Jedna ze zmiňovaných výhod použití centrálního úložiště stavu byla snadná perzistence stavu. Toho lze dosáhnout zapojením *middleware* funkce po změně stavu. V tomto případě se funkce stará o uložení nového stavu na perzistentní úložiště. Perzistence stavu využívá aplikace pro uložení stavu jednotlivých filmů (čas přerušení sledování, vybrané znění a titulky).

Kapitola 3

Realizace

3.1 Příprava

Samotné implementační části aplikace předcházela nutná příprava prostředí – výběr vhodných filmů, konfigurace webového serveru a vytvoření generátoru playlistu pro účely naplnění aplikace daty a simulace komunikace s API serverem.

3.1.1 Open-source filmy

Jako vhodné testovací filmy byly zvoleny krátké open-source filmy *Sintel* [49] a *Tears of Steel* [50] v rozlišení 1080p zaštítěné projekty *Durian Open Movie Project*, resp. *Mango Open Movie Project. Blender Foundation*, která zaštiťuje oba projekty, licencuje oba filmy licencí *Creative Commons Attribution 3.0 Unported (CC BY 3.0)* [51], která dovoluje díla volně šířit a upravovat i pro komerční účely a zároveň předepisuje označení vydavatele licence a deklarování provedených změn. Pro účely aplikace byly použity video soubory ve formátu MP4 v nezměněné podobě, snímky z filmů dostupné v galeriích projektů v nezměněné podobě a dále příslušné titulky ve formátu SRT, které byly dále konvertovány do TTML a WebVTT formátů pro zajištění přehrávání na Smart TV platformách. Všechny použité multimediální soubory jsou vystaveny na testovacím webovém serveru pro zajištění dostupnosti.

Neexistuje mnoho open-source filmů, které by nabízely video soubory ve vysokém rozlišení, když už takové filmy existují, jedná se o krátké filmy s délkou trvání přibližně 10 minut. Existují různé archivy filmů, kterým už vypršela licenční práva, jež nedovolovala šíření. Tyto filmy jsou velmi staré a tím pádem ve velmi špatné obrazové i zvukové kvalitě, proto nebyly zvoleny pro testování, byť jsou to filmy s délkou kolem 1 hodiny. Kratší délku testovacích filmů kompenzuje pro účely testování zatížení aplikace jejich vysoká kvalita.

3.1.2 Testovací server

Testovací webový server Apache byl nakonfigurován, aby správně vystavoval potřebné zdroje, obzvlášť pak výslednou aplikaci (statické HTML, JS, CSS) a multimediální soubory. Důraz byl kladen na konfiguraci správných MIME typů vystavovaných souborů (modul *mod_mime*), aby se správně interpretovaly na cílových platformách. Dále byl kladen důraz na co nejmenší velikost výsledné klientské aplikace, která se přenáší do TV, toho bylo na straně serveru dosaženo konfigurací Apache, konkrétně modulu *mod_deflate*, aby podporoval kompresi všech souborů, kromě video a obrázkových souborů. V neposlední řadě byl server nakonfigurován pro použití šifrovaného HTTPS spojení, jak je zmíněno v části 2.2. Byl použit zmiňovaný *Let's Encrypt* certifikát konfigurovaný pro automatikou obnovu jako *cron* úloha běžící každý den a kontrolující možnost obnovení před vypršení platnosti certifikátu.

Je potřeba zdůraznit, že se jedná o testovací server, který rozhodně není připraven na velké množství uživatelů. Produkční server by měl využívat techniky horizontálního škálování spolu s předřazeným *load balancer* pro optimální využití několika instancí. Rovněž využití ochrany DRM vyžaduje dodatečné kroky, jako je zapojení licenčního serveru, více v části 2.7.

3.1.3 Generování playlistu

Vytvořený jednoduchý generátor playlistu zohledňuje filmy v kategoriích a náhodně vybírá atributy filmů z předem připravených polí možných alternativ hodnot pro konkrétní atribut. Dále jsou do výsledného playlistu nastaveny atributy testovacích filmů včetně odkazů na odpovídající obrazové a textové materiály pro maximální autentičnost aplikace. Ostatní položky playlistu slouží jako zástupné s generovanými atributy, nicméně mají náhodně nastaveny odkazy na zmiňované krátké filmy včetně odpovídajících titulkových souborů. Jedním z atributů je např. barva pozadí a barva miniatury zástupného filmu, tyto dvě hodnoty se nesmí u stejného filmu rovnat, aby nesplývala vybraná miniatura filmu s jeho pozadím.

3.2 Implementace aplikace

V této části jsou popsány důležité prvky aplikace a nalezené problémy, které se vyskytly v průběhu implementace spolu s jejich řešením.

Dále v textu jsou použity některé prvky RxJS knihovny, následující seznam je stručně definuje:

- *Observer* (konzument) a *Observable* (producent) jsou rozhraní, která poskytují mechanismus známý jako *observer design pattern*. *Observer* může implementovat metody *onNext*, *onError* a *onCompleted*, metoda *onNext* může být volána několikrát nad sekvencí dat.

- *BehaviorSubject* – *Subject* se chová zároveň jako *Observable* a *Observer* lze na něm zaregistrovat posluchače a přijímat hodnoty a zároveň do něj lze posílat hodnoty. *BehaviorSubject* je *Subject* s iniciální hodnotou.

3.2.1 Autentizační modul

Jak již bylo uvedeno v části 2.2.2, autentizační modul se skládá ze *serverless* úlohy a Firebase DB pro uložení uživatelský dat –uživatelské jméno, hashované heslo (SHA1) a datum expirace účtu (nastaveno 30 dní po přihlášení). *Webtasks.io* úloha implementuje přihlášení, odhlášení a kontrolu validity přihlášení. Validita uživatelského účtu se kontroluje při každém spuštění aplikace službou *AuthService*[6], jestli nedošlo k jeho expiraci, v případě expirace nebo při prvním spuštění se zobrazí přihlašovací dialog.

Webtasks.io úloha se vytváří nezávisle na aplikaci, jedná se vlastně o jeden JS soubor, jehož strukturu lze vidět na ukázce 3.1. Funkce *login* a *logout* volají HTTP PATCH metodu pro aktualizaci záznamu ve Firebase DB. V případě chyby se posílá rovnou odpovídající řetězec pro následný překlad chybové zprávy v aplikaci. Ukázka je velmi zkrácená kvůli úspoře místa.

```
module.exports = function (context, callback) {
  // extrahovani uziv. jmena, hesla ... z parametru context (POST data)
  // dotaz Firebase DB ...
  request(requestOptions, (err, res) => {
    // reakce na chybu ...
    const userFB = JSON.parse(res.body);
    if (!userFB) return callback('LOGIN.BAD_USERNAME', null);
    switch (cmd) {
      case 'login': {
        if (!hashesMatch(user, userFB)) return callback('LOGIN.BAD_PASSWORD',
            null);
        login(requestOptions, callback);
        break;
      }
      case 'logout': ...
      case 'auth': {
        if (userFB.loggedIn && notExpired(userFB.expiresAt)) { callback(null,
            'ok');
        else callback('LOGIN.LOGGED_OUT', null);
        break;
      }
      default: callback('unknown cmd (login, logout, auth)', null);
    }
  });
};
```

Listing 3.1: *auth-task*: *serverless* autentizační modul

[6]Služba v Angular je JS třída, která je dekorována *@Injectable()* dekorátorem, který umožňuje následné použití DI mechanismem.

3.2.2 Platformově závislý kód

Každá z podporovaných Smart TV platforem vyžaduje použití určité části platformově závislého zdrojového kódu. Platformy Tizen a WebOS vyžadují nahrání dodatečného JS souboru pro použití jejich API. Nejvíce rozdílností u platforem je v případě použitých kódů tlačítek ovladače, vypínání aplikace, kontroly konektivity, ovládání screensaver atd.

Rozlišení platforem je docíleno použitím URL *query* parametru s hodnotou řetězce identifikujícím danou platformu a následným využitím programování proti rozhraní, kdy se použije správná instance třídy. Nesprávná hodnota či chybějící parametr zn. běh mimo Smart TV. K tomuto účelu bylo vytvořeno rozhraní *PlatformInterface* a Angular služby pro každou platformu a jedna zaštiťující *PlatformService*, která vystavuje metody, které na pozadí volají příp. platformově závislý kód. Všechny zmíněné služby implementují *PlatformInterface* a *PlatformService* implementuje navíc rozhraní *VisibilityChange*, protože služba je volána v případě *visibilitychange* DOM události. Základní prvky aplikace lze vidět v ukázce 3.2 zapsané v TS.

Důležitým aspektem této implementace je využití služby *ServiceLocator*, která využívá uloženou referenci Angular DI *Injector* ve statické proměnné pro manuální injektování služeb mimo typický konstruktor. Tento konstrukt byl zvolen z důvodu, že není potřeba inicializovat služby jiných platforem, pouze té, která je zrovna potřeba, pak se např. ani nenahrávají příp. nerelevantní skripty.

```
interface PlatformInterface {
  init(): void;
  close(): void;
  screensaverOn(): void;
  screensaverOff(): void;
}

interface VisibilityChange {
  onAppHide(): void;
  onAppShow(): void;
}

interface NetworkStatusChangeListener {
  onOnline();
  onOffline();
}
```

```
class ServiceLocator {
  static injector: Injector;
}

interface KeyDownListener {
  handle(code: number, event: Event):
      boolean;
  getName(): string;
}

interface KeyCode {
  KEY_BACK: number;
  KEY_CANCEL: number;
  KEY_BACK_SPACE: number;
  ...
}
```

Listing 3.2: Důležité prvky aplikace

3.2.3 Navigace v aplikaci

Navigace v aplikaci je řešena pomocí *NavigationHandler* služby a již zmiňovaného rozhraní *KeyCode*, které jednotlivé platformy implementují a tím

nastavují specifické číselné kódy jednotlivým tlačítkům. *NavigationHandler* implementuje rozhraní *KeyDownListener* stejně jako komponenta *CommonComponent*, od které dědí komponenty, které umožňují ovládání. *CommonComponent* řeší automatické registrování a odregistrování komponent jako posluchačů v *NavigationHandler*, jedinou podmínkou je přepsání metody *handle* v komponentě.

DOM událost *keydown* vyvolá metodu *handle* služby *NavigationHandler*, která projde všechny své registrované posluchače a provolá na nich taktéž metodu *handle* (přepsanou), kde se na základě *keyCode* atributu události rozhodne o další akci v komponentě. Odregistrování je důležité, protože v jeden okamžik může být registrována pouze jedna komponenta schopná obsloužit daný *keyCode*, jinak by docházelo k nečekanému chování. Registrování, resp. odregistrování komponenty probíhá v metodách životního cyklu komponenty v *ngOnInit*, resp. *ngOnDestroy*.

Focus UI prvků je řešen pomocí implementace vlastní direktivy[7] *focusable*, která nastavuje navázanému HTML elementu atribut *tabIndex*, což dovoluje focus i jiných elementů než jen těch formulářových. Direktiva dále umožňuje automatický focus – po inicializaci HTML elementů, tj. v životním cyklu komponenty v okamžiku provolání metody *ngAfterViewInit* provolané frameworkem, zavolá metodu focus navázaného HTML elementu. Direktiva se používá např. takto `<div focusable="{{autofocus}}">...</div>`, přičemž *autofocus* je boolean vazba do komponenty.

3.2.4 Vnitřní stav aplikace

Stavy stěžejních částí aplikace jsou řešeny pomocí centrálního úložiště stavu – *store*, spolu s jednotlivými transformačními funkcemi – *reducers*, řešeno pomocí knihovny *@ngrx/store*. Každý podstav je popsán pomocí rozhraní a je kompozicí příslušné *reducer* funkce pro transformaci stavu a akcí nad podstavem. Např. podstav *platform* je popsán pomocí rozhraní *PlatformState* a příslušné *platformReducer* funkce spolu s množinou tříd reprezentující akce dostupné nad podstavem, např. *SavePlatformAction* – vyvolá uložení platformy do LocalStorage. Každá akce implementuje rozhraní *Action* z použité knihovny, akce musí implementovat (v TS přepsat) minimálně *type* atribut, který označuje typ akce a slouží pro rozlišení v *reducer*. Dále může akce implementovat *payload* atribut, který lze použít pro posílání hodnot při vyvolání akce, hodnoty jsou pak dostupné v *reducer*.

Ve zkrácené ukázce 3.3 je vidět definice a použití podstavu *platform*, čísla v závorce reprezentují pořadí volání. Všechny *reducer* funkce jsou volány při jakékoli změně v *store*, proto je nutné rozlišení akcí pomocí *type*. Pokud akce neodpovídá dané *reducer* funkci, *switch* vrátí původní stav, příp. stav výchozí. Pokud se daný podstav nezměnil, není ani zavolán příslušný *observer*.

[7] Direktiva je v Angular komponenta, která nemá šablonu (*template*).

Pro správné fungování je nutné dodržovat neměnnost stavu, tzn. neměnit atributy stavu, ale vždy vytvářet nový, resp. hlubokou kopii objektu. Perzistentní podstavy na začátku vrací místo výchozího stavu stav uložený, tento mechanismus dovoluje *store* zapojením *middleware* (knihovna *ngrx-store-localstorage*) pro ukládání do LocalStorage.

```
// platform.state.ts
export interface PlatformState {
  readonly platformName: Platform;
}
// platform.actions.ts
export const ActionTypes = {
  SAVE: type('[Platform] SAVE'),
};
export class SavePlatformAction implements Action {
  type = ActionTypes.SAVE;
  payload: Platform;
  constructor(payload: Platform) {
    this.payload = payload;
  }
}
// platform.reducer.ts
const defaultState: PlatformState = {platformName: 'web',};
export const platformReducer: ActionReducer<PlatformState> = (state =
    defaultState, action: Action) => {
  switch (action.type) { // (2)
    case ActionTypes.SAVE: {
      return {
        platformName: action.payload
      };
    }
    default: return state;
  }
};
// vyvolani akce
this.store.dispatch(new SavePlatformAction(platformName)); // (1)
// reakce na zmenu stavu
const platformState$ = store.select('platform') as Observable<PlatformState>;
platformState$
  .subscribe((state: PlatformState) => /* pouziti stavu (3) */);
```

Listing 3.3: Ukázka stavu, jeho komponent a zpracování

Následující body stručně popisují jednotlivé podstavy implementované v aplikaci, ty nejdůležitější jsou blíže popsány v odpovídajících částech.

- *videoPlayer* – aktuální stav přehrávače videa (přehrávání, pozastavení, posun ve videu)
- *videoControls* – stav *controls* (*hidden*, *shown*, *expanded*, *reduced*)
- *videos* – perzistentní stav všech videí (čas ukončení sledování, vybrané znění a titulky)

- *gridVideos* – stav knihovny (kategorie, nadpisy, samotná videa a jejich pozice v mřížce)
- *platform* – uložená konkrétní platforma
- *auth* – uložený autentizovaný uživatel

3.2.5 Knihovna filmů

Knihovna filmů se skládá z komponenty částečného detailu filmu a komponent jednotlivých miniatur filmů organizovaných do struktury rozebrané v návrhové části 2.5.3. Zmiňované prohazování filmů v rámci dvou viditelných řádků miniatur filmů je realizován kombinací Angular **ngFor*[8] direktivy a *gridVideosReducer* funkce, která mění stav *GridVideosState*. *gridVideosReducer* mimo jiné řeší simulaci nekonečného kolotoče miniatur filmů pomocí rotace pole, kdy při posunu vpravo se položky pole posunou doleva – první položka zaujme nově vzniklé místo na konci pole.

Knihovna filmů dostává playlist z *PlaylistService*, která musí došlá data transformovat tak, aby obsahovala správné URL titulků podle typu, který je na dané platformě podporován. Jak je vidět na ukázce 3.4, každá titulková stopa obsahuje pole URL řetězců – *src*, vedoucí na různé formáty (VTT, TTML, SRT), zde je potřeba vybrat ten pro danou platformu pomocí indexu *subSrcIdx*. K transformaci je použito funkce *Array.prototype.map*, kterou zmiňuje je i kniha [48, str. 293]. *Map* je ve funkcionálním stylu programování, je čitelnější a jednodušší, než klasický *for* cyklus, zároveň podporuje snadnou kompozici funkcí.

```
const playlist = data.playlist
    .map((category: Category) => {
      category.videos = category.videos.map((video: Video) => {
        const newVideo = {...video};
        newVideo.subTracks = newVideo.subTracks
          .map((track: SubtitleTrack) => {
            const newTrack = {...track};
            if (Array.isArray(newTrack.src)) {
              newTrack.src = newTrack.src[subSrcIdx];
            }
            return newTrack;
          });
        return newVideo;
      });
      return category;
    });
```

Listing 3.4: Ukázka transformace playlistu, aby obsahoval URL na titulky ve formátu pro danou platformu

[8] Direktiva **ngFor* z předloženého seznamu vygeneruje příslušné HTML elementy.

3.2.6 Video přehrávač

Video přehrávač podporuje navrhované funkčnosti a ovládání z části 2.5.5. Změna stavu přehrávače je implementována pomocí zmiňovaného centrálního úložiště. Nicméně některé stavy tak řešeny nejsou. Jedná se o případy, kdy by měl mít příslušný *reducer* vedlejší efekty, což je v rozporu s definicí *pure* funkce. Např. změna znění a titulků využívá centrálního stavu jen pro zapamatování voleb uživatele, nikoli k samotnému ovlivnění HTML video elementu.

Posun ve videu je realizován kombinací akcí nad podstavem a periodické aktualizace aktuálního času video elementu. Čas je aktualizován pomocí vztahu `const time = state.seeking.forward * state.seeking.speed / coef`.
Podstav obsahuje směr *forward* a rychlost *speed* posunu, směr nabývá hodnot `-1` a `1`, doleva resp. doprava a rychlost zmiňovaných 0, 2, 4, ..., 128. Parametr *coef* je definován vztahem `const coef = 1000 / TICK`, přičemž *TICK* je konstanta, reprezentující četnost aktualizace, hodnota `1000 / 60` značí 60 FPS. Akce vyvolané nad podstavem mění koeficienty vztahu – *forward* při změně směru a *speed* při změně směru a opakovaném stisku tlačítka pro posun ve stejném směru – změna rychlosti.

Posun je optimalizován tak, že během posunu je video pozastavené a nový čas se nastaví video elementu až při přerušení posunu. Zvolený způsob posunu přičítáním času je záměrný, protože zdaleka ne všechny implementace video elementu podporují běh ve zrychleném režimu.

Titulky jsou vystaveny pod správnými MIME typy, v kódování UTF-8 a ve formátech pro každou platformu zvlášť, protože zde průnik neexistuje. Přestože podle dokumentací platforem má být možné titulky přehrávat na všech platformách, nepodařilo se je na platformách po četných pokusech zprovoznit, nicméně v nejrozšířenějších prohlížečích ano. Byl položen dotaz na technickou podporu každé platformy, nicméně před dokončením této práce nebyla obdržena konstruktivní odpověď s řešením problému.

3.2.7 Podpora více jazyků

Aplikace podporuje multijazyčnost, konkrétně jsou dostupné české a anglické překlady statických popisků, nicméně hodnoty (názvy kategorií apod.) v testovacím playlistu jsou pouze anglicky. K překladu bylo využito zmiňované knihovny *@ngx-translate*, která detekuje i zvolený systémový jazyk a podle toho se nahrají správné překlady při každém spuštění aplikace. Je-li systém v českém jazyce, nahrají se odpovídající překlady, v ostatních případech se nahrají anglické překlady. Překlady jsou strukturované ve formátu JSON pro každý jazyk ve vlastním souboru, což zajišťuje snadnou rozšiřitelnost podpory dalších jazyků. Knihovna *@ngx-translate* nabízí několik možností jak

provést překlad, z toho aplikace používá *TranslatePipe*[9] a *TranslateService*. *TranslatePipe* se používa přímo v HTML takto:
`<p>'DETAIL.DIRECTORS' | translate</p>`, kde DETAIL.DIRECTORS je překladový klíč, podobně se používá *TranslateService*. Jak již bylo zmíněno autentizační modul vrací přímo překladové klíče.

3.3 Plánované a neočekávané události

3.3.1 Přepínání aplikací

Aplikace reaguje na událost *visibilitychange*, jde-li aplikace do pozadí, uvolní se nepotřebné zdroje a při návratu do aplikace se opět alokují. Probíhá-li při přepnutí aplikace přehrávání videa, tak se přehrávání pozastaví, uloží se aktuální čas videa, uvolní se zdroje a nastaví se prázdné video jako zdroj HTML video elementu. Prázdné video bylo vygenerované pomocí nástroje *ffmpeg* z důvodu absence jakékoli metody pro zrušení nahrávání videa v API HTML video elementu. Bez tohoto mechanismu by se video nahrávalo i v případě aplikace na pozadí. Podobně se aplikace chová v případě návratu z video přehrávače na detail.

3.3.2 Výpadek internetového připojení

Smart TV aplikace jsou založeny na streamování multimediálního obsahu a tím pádem vyžadují trvalé dostatečně rychlé internetové připojení. Každá z platforem nabízí různé API pro kontrolu konektivity zařízení z aplikace, např. FirefoxOS nenabízí žadné API, WebOS *ConnectionManager* vrací neustále status připojení jako `isInternetConnectionAvailable: false`. Nelze se ani spolehnout na provolání zaregistrovaných funkcí na události *online* a *offline* ani na aktuální stav atributu *navigator.onLine*, tak jako při vývoji pro většinově používané prohlížeče.

Nespolehlivé API platforem vyústilo v jednotnou kontrolu změny stavu připojení. Jedná se o jednoduchý *ping* každých 5 sekund na webový server, který obsahuje proměnnou hodnotu v URL query parametry, aby se zabránilo cachování požadavku na straně klienta. Tento způsob je nevhodný v produkčním prostředí z důvodu vytěžování web serveru.

Při realizace bylo využito stavebních bloků z RxJS knihovny, konkrétně *BehaviorSubject* a *Observable*. Hodnota *BehaviorSubject* v implementaci figuruje jako poslední známý stav připojení. Jak je vidět na zjednodušené ukázce 3.5, algoritmus na začátku předpokládá dostupné připojení a metoda *isOnline* vrací *Observable<boolean>*, ke kterému lze registrovat *Observer*. Zároveň lze vidět, že se registruje zmiňovaný posluchač na změnu stavu připojení, který

[9] *Pipes* jsou v Angular třídy implementující *PipeTransform* rozhraní s jedinou metodou `transform(value: any, ...args: any[]): any;`.

se volá pouze v případě změny. Mimo provolání metod posluchače stavu připojení se aktualizuje stav připojení, neboli se vloží nová hodnota do sekvence *BehaviorSubject* a *isOnline* emituje aktualizovanou hodnotu posluchačům.

Aplikace podporuje cachování statického obsahu na straně klienta pomocí HTML technologie *AppCache* v případě ztráty připojení k Internetu.

```
private onlineStatus$ = new BehaviorSubject<boolean>(true);
private wasOnline = true;

isOnline(): Observable<boolean> {
  return this.onlineStatus$.asObservable();
}

registerListener(listener: NetworkStatusChangeListener): void {
    Observable.interval(INTERVAL)
      .subscribe((res) => {
       this.http.get(PING_URL)
         .subscribe(
          res => {
            if (!this.wasOnline) listener.onOnline();
            this.wasOnline = true;
            this.onlineStatus$.next(true);
          },
          err => {
            if (this.wasOnline) listener.onOffline();
            this.wasOnline = false;
            this.onlineStatus$.next(false);
          });
     });
  }
```

Listing 3.5: Ukázka kontroly konektivity

3.4 Dodatky

3.4.1 Lazy loading

Angular podporuje tzv. *lazy loading*, neboli nahrávání zdrojů až, když je to nevyhnutelné. Standardně sestavené aplikace toto v základu nevyužívají, ale konfigurace pro zprovoznění není složitá a vyžaduje pouze správné nastavení *RouterModule*. Konkrétně je potřeba nastavit pro požadované cesty příslušné moduly, což je standardní, ale navíc je zde přidání absolutní cesty k modulu, který má být nahrán později spolu s názvem třídy modulu[10],
např. `loadChildren: 'app/detail/detail.module#DetailModule'`. Nastavení *lazy loading* vyústí v očekávaný menší výsledný *bundle*, přičemž dodatečné

[10]Modul v Angular je JS třída dekorovaná dekorátorem *@Module()*, který nastavuje metadata jako importy, exporty, providery atd.

soubory reprezentující dotahované moduly jsou prefixovány čísly ve formátu: `#.hash.chunk.js`, např. `0.2436c797cbdf80f850f6.chunk.js`.

S touto optimalizací spolu s GZIP kompresí nastavenou na webovém serveru a povoleným AOT při sestavování se podařilo stáhnout původní velikost *bundle* z 1,6 MiB na 320 KiB.

3.4.2 Uvolňování zdrojů

Aplikace hojně používá *Observables* z knihovny RxJS, jejichž použití skýtá úskalí ve správném uvolňování posluchačů. Typicky se používá konstrukt `mySubs = Observable.interval(100).subscribe(observer)`, kde se uloží *Subscription* pro následné uvolnění, jinak dochází k neuvolňování paměti. Tento přístup vyžaduje přítomnost mnoha *Subscription* v komponentě a uvolňování všech v metodě *ngOnDestroy*, která se volá na konci životního cyklu komponenty.

V aplikaci je tento problém vyřešen pomocí již zmiňované *CommonComponent*, od které dědí ostatní komponenty, které potřebují využívat *Observable* nebo navigaci. *CommonComponent* obsahuje atribut *ngUnsubscribe* typu *Subject<void>*, který se předsazuje každému volání *subscribe* metody (v potomcích), jak je vidě v ukázce 3.6. Operátor *takeUntil* spolu s implementací *ngOnDestroy* metody zajistí tak, jak je ve zmíněné ukázce, odregistrování posluchače a tím také správné uvolňování paměti.

Podobným způsobem je stejný problém vyřešen u služeb, rozšiřují *CommonService*, nicméně služby nepodléhají životnímu cyklu komponent, tudíž se nelze spoléhat na provolání příslušných metod frameworkem. Opět se využívá atributu *ngUnsubscribe*, tentokrát se služby registrují v konstruktoru do pole, které se v případě potřeby pročistí `subscribedServices.forEach(service => service.destroy());`, přičemž metoda *destroy* je shodná s metodou *ngOnDestroy* v *CommonComponent*.

```
// CommonComponent
ngOnDestroy(): void {
    this.ngUnsubscribe.next();
    this.ngUnsubscribe.complete();
  }
// MyComponent
Observable.interval(INTERVAL)
  .takeUntil(this.ngUnsubscribe)
  .subscribe(observer)
```

Listing 3.6: Ukázka uvolňování paměti při použití *Observables*

3.4.3 Docker

Byť je aplikace nasazena na zmiňovaném webovém serveru, byla taktéž testována v *docker* image. Aplikace obsahuje jednoduchý *Dockerfile* s návodem na zprovoznění a *nginx* konfigurační soubor, který mimo jiné povoluje GZIP statických zdrojů.

Kapitola 4

Testování

V této kapitole je zkráceně prezentováno softwarové a uživatelské testování včetně vyhodnocení výsledků. Pro softwarové testování bylo využito *white box* a *black box* metodiky. Cílem bylo odhalit co nejvíce softwarových chyb vzniklých při implementaci aplikace. Testování probíhalo průběžně při vývoji aplikace v několika fázích a průběžně nalezené problémy byly rovnou zapracovány. Uživatelské testování mělo za cíl odhalit UI a UX problémy a diskutovat možná vylepšení.

4.1 *White box* testování

Testování softwarovou metodikou *white box* testování, neboli testování bílé skříňky, je způsob testování, který je prováděn při znalosti vnitřní struktury aplikace. Znamená to, že jsou dobře známy programové a datové struktury testované aplikace a je tedy možné odhalit jemné nedostatky snadno přehlédnutelné při *black box* testování.

Testování probíhalo tak, že se postupně nejprve izolovaně, poté v sekvencích zkoušeli jednotlivé klíčové části aplikace. Velká pozornost byla věnována zpracování stavu přehrávače, zejména postupu ve videu, aby fungoval správně i v krajních případech a svižně i na slabších zařízeních. Dalším důležitým prvkem bylo testování chování aplikace v případě výpadku konektivity a neméně důležitým bylo ověření uvolňování alokovaných zdrojů, zejména posluchačů událostí. Samozřejmostí bylo odstranění nežádoucích ladících výpisů.

White box testování odhalilo už v průběhu implementace četné nedostatky, které se projevovali pouze na reálných Smart TV. Ty mají zpravidla stará jádra prohlížečů, kde ani použití *polyfill* nebylo dostatečné a musel se zvolit jiný přístup, více v části 4.3.

4.2 *Black box* testování

Softwarová metodika *black box* testování, neboli testování černé skříňky, spočívá v neznalosti vnitřních struktur aplikace a testování pouze aplikace tak, jak ji bude používat uživatel. Testeři dostali k dispozici Smart TV s předinstalovanou aplikací a scénáře, které měli ověřit.

Testování probíhalo především za účelem ověření zadávání krajních a nesmyslných hodnot do vstupních polí, reakce na neočekávané události jako je i výpadek připojení k Internetu či ověření správné reakce při rychlém pohybu aplikací.

Black box testování neodhalilo žádné závažné nedostatky a v případě chyby, aplikace vždy správně zobrazila odpovídající chybovou hlášku uživateli s návrhem na řešení.

4.3 Testování na jednotlivých platformách

Tato část testování se věnuje popisu testování, které bylo realizováno uplatněním obou předchozích metodik a směrnic platforem.

4.3.1 Emulátor, simulátor, reálné zařízení

Testování probíhalo postupně nejprve na simulátorech a emulátorech, následně na reálných zařízeních, kde se odhalilo nejvíce nedostatků, které se předtím na simulátorech a emulátorech vůbec neprojevily. K testování byly dostupné modely všech tří Smart TV platforem, od každé jeden zástupce, jednalo se o modely ve střední cenové kategorii dostupné v polovině roku 2016.

Jedním z objevených nedostatků na reálných TV byla nefunkčnost aplikace při použití protokolu HTTPS. TV jsou dodávány s předinstalovanými certifikáty několika certifikačních autorit a tento seznam již nelze rozšířit jinak, než aktualizací firmware, která musí nové certifikační autority obsahovat. *Let's Encrypt* ale nepatří mezi podporované certifikační autority ani na jedné z dostupných testovacích TV. Řešením je využít placeného certifikátu některé z ověřených certifikačních autorit.

Dalším nedostatkem byla reakce na výpadek připojení k Internetu, který TV většinou vůbec nedetekovaly. Řešení toho problému je nastíněno v závěru implementační části.

Přestože implementace dodržovala dostupnou dokumentaci a titulky fungovaly správně na většinových prohlížečích, tak na reálných TV se je zobrazit při přehrávání videa nepodařilo. Např. ukázkové příklady v dokumentaci WebOS vůbec nefungují podle četných dotazů na téma titulky ve fóru technické podpory, FirefoxOS příklady se zase titulkům úplně vyhýbají a Tizen zařízení by měla zvládat zobrazení titulků pouze při použití *avplayer*, který se používá především pro přehrávání obsahu chráněného DRM. Jak již bylo zmíněno v části 3.2.6, byl položen dotaz na technickou podporu.

Obecně emulátory a simulátory mají problém přehrávat video obsah oproti reálným zařízením, což je uvedeno v dokumentacích. Přesto se podařilo přehrát testovací videa na Tizen simulátoru, emulátor už ale vykazoval delší odezvy a video přehrát nezvládl, podobně WebOS emulátor.

4.3.2 Rychlost aplikace

Před samotnou implementací byl proveden jednoduchý test výkonnosti reálných TV. Vytvořená jednoduchá testovací aplikace obsahovala 1024 vygenerovaných HTML div elementů na jedné stránce, které bylo možno vybrat. Po spuštění aplikace byl proveden automaticky focus na první element. Test spočíval v pohybu výběru po div elementech. Byly patrné dlouhé odezvy na všech třech testovaných platformách (modely TV ve střední třídě, v pořizovací ceně cca 12 000 Kč). Samsung TV si vedla nejlépe. Počet elementů byl ubírán metodou půlení intervalů a až po páté iteraci, tj. při 16 elementech byla zaznamenán svižná interakce. Není to nijak překvapivý výsledek, byť v prohlížečích na běžném dnešním PC nebyl rozdíl v počtu elementů patrný, odezva byla stále poměrně svižná. Z tohoto naivního testu ale plyne, že počet elementů, které lze vybrat by měl být nízký, tzn. vykreslovat pouze ty, které jsou ve viditelné oblasti obrazovky. Poznatky z tohoto testu byly zapracovány v implementaci.

Testování probíhalo tak, že se zkoušelo rychlé přepínání obrazovek a změna položek. Aplikace reagovala poměrně svižně a správně i při několikanásobném a rychlém přepínání, historie obrazovek taktéž. Jedině přepínání mezi filmy a mezi kategoriemi mělo byť malou, ale znatelnou odezvu.

4.3.3 Testování podle směrnic jednotlivých platforem

Směrnice jednotlivých platforem lze kromě zmíněných UI a UX pravidel sjednotit do následujících stěžejních bodů, nuance platforem jsou vyřešeny zmíněným platformově závislým kódem, např. screensaver u Tizen. Tyto body výsledná aplikace splňuje.

- Aplikace je plně funkční bez pádů a chyb.
- Tlačítko zpět správně dodržuje historii průchodu aplikací a na domácí obrazovce může zobrazit potvrzovací dialog, který se po dalším stisku zpět skryje. Vypínací tlačítko potvrzovacího dialogu korektně ukončí aplikaci.
- Aplikace je odolná vůči tzv. stres testu, tj. opakované rychlé stisknutí tlačítka.
- Aplikace plně naběhne do 10 sekund, stejně tak příp. video.

- Aplikace funguje správně při přepínání aplikací, tj. zkontroluje konektivitu a v případě přehrávání multimédií naváže.
- Přehrávání a ovládání multimédií funguje bez problémů, má synchronizovaný zvuk a obraz. Přehrávací tlačítka fungují podle očekávání.
- Aplikace zobrazuje nahrávací indikátor v případě práce na pozadí.
- Aplikace uživatele v případě chyby vždy upozorní a nabídne řešení, např. v případě ztráty spojení.

4.3.4 Výsledky softwarového testování

Během softwarového testování byla odhalena řada menších nedostatků a ty byly průběžně zapracovány. Celkově aplikace ze softwarového hlediska funguje dobře a bez větších problémů. Jedině zmiňované titulky se musí dořešit s technickou podporou platforem.

4.4 Uživatelské testování

Uživatelské testování obnášelo přípravu testování včetně přípravy a vyhodnocení *interview*, výběr participantů, vyhotovení testovacích scénářů, samotné testování, jeho vyhodnocení a rozbor nálezů s návrhy na jejich řešení.

4.4.1 Příprava testování

Samotnému testování předcházela příprava čítající vyhotovení dotazníku – *screener*, který obsahuje několik jednoduchých otázek a odpovídajících odpovědí. Odpovědi na otázky určily, kteří z participantů jsou vhodní pro samotné testovaní ale hlavně dokreslily profily participantů. V závorkách u odpovědí je vyznačena četnost volby participanty.

Screener

1. Jaký je Váš věk?
   a) do 18
   b) 18 – 29 *(1x)*
   c) 30 – 39 *(2x)*
   d) 40 – 49 *(1x)*
   e) více než 50

2. Máte doma Smart TV?
   a) Ano *(4x)*
   b) Ne
3. Jak často TV sledujete?
   a) Méně než 3 h týdně
   b) 3 – 7 h týdně *(2x)*
   c) 1 – 2 h denně *(2x)*
   d) Více než 2 h denně
4. Vyberte odpověď, která Vás nejlépe charakterizuje.
   a) Sleduji převážně TV vysílání *(1x)*
   b) Ke sledování používám převážně vlastní domácí knihovnu *(2x)*
   c) Ke sledování používám převážně nějakou Smart TV aplikaci *(1x)*
5. Máte zájem zúčastnit se testování aplikace?
   a) Ano *(4x)*
   b) Ne

Popis participantů

Testování se zúčasnili 4 participanti, jedna žena a tři muži. Následuje stručný popis participantů.

- Participant 1 – žena (30) – přímo Smart TV nevlastní, ale doma má Apple TV a monitor a sleduje vlastní domácí knihovnu asi 2 h denně. Participantka používá pro sledování aplikaci Plex, kterou ale sama nekonfigurovala. Jedná se o mírně pokročilou uživatelku.
- Participant 2 – muž (33) – vlastní Smart TV, na které sleduje hodinu denně seriály v aplikaci Netflix. Participant je povoláním programátor, jedná se tedy o velmi pokročilého uživatele, který má zkušenosti s podobnou aplikací.
- Participant 3 – muž (27) – nevlastní Smart TV, ale doma používá ke sledování vlastní domácí knihovny herní konzoli Xbox One v kombinaci s aplikací Plex a promítacím plátnem. Participant je programátor a tudíž ho lze považovat za pokročilého uživatele.
- Participant 4 – muž (48) – vlastní starší Smart TV, na které sleduje převážně TV vysílání, které si často nahrává pomocí Apple Mac Mini a sleduje později. Participanta lze považovat za mírně až středně pokročilého uživatele.

4.4.2 Průběh testování

Testování předcházelo sestavení běžných úkolů v aplikaci. Samotné testování probíhalo buďto přímo na testovacích Smart TV nebo na emulátoru, tj. na počítači s emulovaným TV ovladačem ovládaným myší. Kvůli vyzkoušení změny titulků proběhl daný scénář i v prohlížeči Chrome (verze 58). Po dokončení úkolů následovalo shrnutí testování a byly doptávány příp. návrhy na vylepšení, jestli a kde se uživatel cítil nejistě a celkový pocit z aplikace.

Seznam testovacích scénářů

- Přihlaste se do aplikace.
- Vyberte první film z kategorie nejvíce populárních a zjistěte dostupná znění a titulky.
- Spusťte přehrávání vybraného filmu, nechte přehrávat 10 sekund a vraťte se do knihovny, následně navažte na sledování.
- Při přehrávání filmu se posuňte na poslední minutu.
- Při přehrávání filmu změňte titulky.
- Odhlaste se z aplikace.

4.4.3 Přehled nálezů a jejich řešení

Uživatelské testování odhalilo některé problémy aplikace. Následuje jejich rozbor a návrhy na zlepšení. Nálezy mají přiřazeny jednu ze tří priorit závažnosti a jsou podle nich i seřazeny.

Funkce video přehrávače (vysoká priorita)

Funkce video přehrávače nejsou na první pohled zřetelné, např. jeden participant správně předpokládal, že posun ve videu se realizuje směrovými tl. doleva, resp. doprava, ale držel je místo postupného stiskání. Řešením může být průvodce prvního spuštění nebo popisky při viditelné dolní stavové liště.

Přihlášení (střední priorita)

Při přihlašování na platformě WebOS není jasné jak se posunout z pole jméno na pole heslo. SW klávesnice postrádá šipky a směrová tlačítka ovladače fungují pro posun po písmenech *qwerty* klávesnice, po vyplnění jednoho pole je nutné stisknout tl. zpět, což nebylo pro participanti intuitivní. Řešením může být využití barevných tl. pro posun mezi poli a potvrzení, zároveň by tato funkčnost měla být nějak ihned viditelná, např. vyobrazením tl. a popiskem.

Detail filmu (střední priorita)

Na detailu filmu není zřetelný rozdíl dvou *play* tlačítek. Padl návrh na ponechání pouze jednoho *play* tl. s funkcí navázání na předešlé sledování. Po diskuzi si participant, který návrh předložil, uvědomil, že kdyby chtěl spustit film od začátku, musel by se ve videu zbytečně posouvat. Každopádně symboly tl. nejsou úplně jasné a je třeba je doplnit popiskem.

Výběr znění a titulků (nízká priorita)

Při potvrzení výběru znění se rovnou začne načítat zvolený video soubor a komponenta *controls* se skryje, oproti potvrzení výběru titulků, kdy komponenta *controls* zůstane zobrazena. Toto chování je potřeba sjednotit buďto okamžitým skrytím komponenty *controls* po potvrzení výběru titulků nebo naopak neskrývat ani po potvrzení výběru znění, ale až když zvolí uživatel zpět nebo po uplynutí časového limitu.

Odhlášení (nízká priorita)

Symboly obrazovky voleb nemají zřetelnou funkčnost. Jeden z participantů pochopil vypínací symbol jako odhlášení místo symbolu přepnutí uživatelů. Opět by pomohly popisky jednotlivých tlačítek.

4.4.4 Návrhy na zlepšení

Diskuze s participanty ihned po provedeném testování nabídla další body ke zlepšení. Např. již zmiňované průvodní popisky nebo místo zobrazeného příznaku *unwatched* u dosud neviděných filmů, otočit logiku a zobrazovat naopak příznak *continue watching* indukující nedokončené sledování.

4.4.5 Závěr uživatelského testování

Testování odhalilo několik nedostatků, ale celkově se participanti shodli na kladném hodnocení aplikace. Konkrétně participanti ocenili čistý a přehledný design aplikace a jednoduché intuitivní ovládání.

Závěr

Aplikace splnila zadání práce a všechny požadavky, které měl zadavatel a zároveň vedoucí této práce. Rovněž byly splněny požadavky a nuance jednotlivých platforem, které byly nashromážděny a analyzovány v rozsáhle návrhové části diplomové práce. Výsledná VOD aplikace se společným zdrojovým kódem pro Smart TV platformy Tizen TV, WebOS TV a FirefoxOS TV obsahuje knihovnu filmů v kategoriích, detail filmu a video přehrávač. Přehrávač podporuje běžné funkce video přehrávače, tedy snadné a intuitivní ovládání. Součástí aplikace je i autentizační modul pro přihlašování uživatelů. Aplikace podporuje multijazyčnost a je ovladatelná dálkovým ovladačem pomocí směrových a multimediálních tlačítek stejně na všech platformách.

Samotné implementaci předcházelo vypracování obsáhlé analytické a návrhové části, která se zabývala rozpracováním požadavků do případů použití a komponent systému. Následovalo podrobné srovnání Smart TV platforem včetně shrnutí různých možností ovládání. Srovnání platforem pomohlo zúžit výběr potenciálních technologií pro implementaci na webové technologie. Srovnání také pomohlo při návrhu UI a UX aplikace podle směrnic a doporučení jednotlivých platforem. Návrh byl podrobně analyzován a hledaly se alternativy, ze který byly vybrány ty nejvhodnější. Rovněž byl vypracován souhrn omezení Smart TV platforem. Součástí analýzy byl i rozbor možností přehrávání multimediálního obsahu chráněného DRM na všech třech Smart TV platformách.

Analýza a návrh se zabývaly také volbou vhodných technologií pro implementaci aplikace s ohledem na limitace Smart TV platforem a budoucí rozšiřitelnost. Po důkladné úvaze a diskusi konkurenčních možností byl zvolen framework Angular. Další úvahy se týkaly zpracování stavu aplikace, kde byl zvolen přístup centrálního stavu aplikace a *pure* transformačních funkcí. Také rozbor možností zabezpečení webové aplikace není opomenut.

Realizační část diplomové práce se zabývala nejprve nutnou přípravou – výběrem testovacích filmů, simulací webové služby a konfigurací webového serveru pro vystavení aplikace a jejích zdrojů. Implementace dále popisovala

nejdůležitější části aplikace, diskutovala případné nalezené problémy při implementaci aplikace a ukazovala jejich řešení. V implementaci bylo například uvedeno, jak byl vyřešen problém s platformově závislým kódem, s navigací v aplikaci či zpracování vnitřního stavu aplikace. Realizace se dále zabývala reakcí na plánované a neočekávané události, ke kterým může při provozu aplikace dojít. Závěr realizační části se věnoval tématům spojeným s nasazením aplikace.

Aplikace byla podrobena softwarovému a uživatelskému testování včetně testování na reálných TV jednotlivých platforem. Softwarové testování na reálných zařízeních probíhalo v několika fázích a tak došlo k odhalení a zapracování několika skrytých nedostatků už v rané fázi vývoje. K uživatelskému testování bylo přizváno několik participantů, kteří přispěli k mnoha užitečným připomínkám, které byly zapracovány a přidaly další funkce zpříjemňující používání aplikace. Během testování nebyly odhaleny žádné závažné nedostatky, naopak byla aplikace hodnocena velmi kladně.

Výsledná VOD aplikace pro zmíněné platformy může být použita jako robustní základ rozsáhlé aplikace poskytující obsáhlou knihovnu multimediálního obsahu a to nejen VOD charakteru.

Před další případnou verzí aplikace by mělo proběhnout rozsáhlejší uživatelské testování, které by buď potvrdilo nebo vyvrátilo nálezy finálního uživatelského testování. Nesporným rozšířením další verze je plná podpora titulků na všech platformách a vystavení placeného certifikátu ověřenou certifikační autoritou pro možnost využití HTTPS protokolu pro aplikaci na Smart TV platformách. Pro uživatele přínosným rozšířením může být přidání podobných filmů do detailu filmu.

Literatura

[1] Premium Content Helps Grow US Connected TV Audience. *eMarketer* [online].
eMarketer Inc., ©2016, 17.11.2016. [cit. 2.4.2017].
Dostupné z: https://www.emarketer.com/Article/Premium-Content-Helps-Grow-US-Connected-TV-Audience/1014731

[2] Global Smart TV Market 2015-2019. In: *Research and Markets* [online].
Research and Markets, ©2010-2017, březen 2017. [cit. 8.4.2017].
Dostupné z: http://www.researchandmarkets.com/research/4l9tl2/global_smart_tv

[3] DUNN, Jeff. Here's how huge Netflix has gotten in the past decade. In: *Business Insider* [online].
Business Insider Inc, ©2017, 19.1.2017. [cit. 9.4.2017].
Dostupné z: http://www.businessinsider.com/netflix-subscribers-chart-2017-1

[4] Podporovaná zařízení. *VOYO* [online].
TV Nova s.r.o. [cit. 10.4.2017].
Dostupné z: http://voyo.nova.cz/devices

[5] Stream.cz pro chytré televize, Samsung a LG. *Internetová televize Stream* [online].
Seznam.cz a.s., ©1996-2017 [cit. 9.4.2017].
Dostupné z: https://www.stream.cz/tv/samsung-lg

[6] HbbTV Wiki. *HbbTV Developer Page* [online].
HbbTV Developer Page. Poslední změna 30.10.2015 [cit. 1.2.2017].
Dostupné z: http://www.hbbtv-developer.com/site/wiki/index.php/HbbTV_Wiki

[7] Resource Library, Specifications. *HbbTV* [online].
Hybrid broadcast broadband TV (HbbTV), ©2016. [cit. 22.2.2017].
Dostupné z: https://www.hbbtv.org/resource-library/#specifications

[8] *Webtask.io* [online].
Auth0 Inc., ©2013-2017 [cit. 15.3.2017].
Dostupné z: https://webtask.io

[9] Smart TV. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 18.4.2017].
Dostupné z: http://developer.samsung.com/tv

[10] API References. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 19.4.2017].
Dostupné z: http://developer.samsung.com/tv/develop/api-references

[11] General Features. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 19.4.2017].
Dostupné z: http://developer.samsung.com/tv/develop/specifications/general-features

[12] UX Checklist. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 19.3.2017].
Dostupné z: http://developer.samsung.com/tv/design/ux-checklist

[13] Development Checklist. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 19.3.2017].
Dostupné z: http://developer.samsung.com/tv/develop/development-checklist

[14] Input Methods. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 20.4.2017].
Dostupné z: http://developer.samsung.com/tv/design/input-methods#basic-controls

[15] Launch Checklist. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 18.3.2017].
Dostupné z: http://developer.samsung.com/tv/distribute/launch-checklist

[16] Tizen 3.0 Milestones. *SAMSUNG Developers* [online].
SAMSUNG, ©2010-2016 [cit. 20.3.2017].
Dostupné z: https://source.tizen.org/release/tizen-3.0-milestones

[17] *webOS TV Developer* [online].
LG Electronics, ©2016 [cit. 20.4.2017].
Dostupné z: http://webostv.developer.lge.com/

[18] Web Engine. *webOS TV Developer* [online].
LG Electronics, ©2016 [cit. 20.4.2017].
Dostupné z: http://webostv.developer.lge.com/discover/specifications/webos-tv-platform/web-engine/

[19] Introduction to Emulator. *webOS TV Developer* [online].
LG Electronics, ©2016 [cit. 20.4.2017].
Dostupné z: http://webostv.developer.lge.com/sdk/emulator/introduction-emulator/

[20] Supported Media and DRM Formats. *webOS TV Developer* [online].
LG Electronics, ©2016 [cit. 20.4.2017].
Dostupné z: http://webostv.developer.lge.com/discover/specifications/webos-tv-platform/supported-media-formats/

[21] UX Checklist. *webOS TV Developer* [online].
LG Electronics, ©2016 [cit. 20.4.2017].
Dostupné z: http://webostv.developer.lge.com/design/ux-checklist/

[22] Key App Assets. *webOS TV Developer* [online].
LG Electronics, ©2016 [cit. 20.4.2017].
Dostupné z: http://webostv.developer.lge.com/develop/app-developer-guide/app-packaging-guide/key-app-assets/

[23] Remaining Backward Compatible. *webOS TV Developer* [online].
LG Electronics, ©2016 [cit. 20.4.2017].
Dostupné z: http://webostv.developer.lge.com/develop/remaining-backward-compatible/

[24] BRYANT David, JAAKSI Ari. B2G OS and Gecko. In: *Skupiny Google* [online]. 27.9.2016.
Google Inc. [cit. 20.4.2017].
Dostupné z: https://groups.google.com/forum/#!msg/mozilla.dev.fxos/FoAwifahNPY/Lppm0VHVBAAJ

[25] HTML5 v2.0. *Panasonic IPTV Apps Developers* [online].
Panasonic, ©2017 [cit. 21.4.2017].
Dostupné z: http://developer.vieraconnect.com/dev-guide/html5-v2.0

[26] API Reference. *Panasonic IPTV Apps Developers* [online].
Panasonic, ©2017 [cit. 21.4.2017].
Dostupné z: http://developer.vieraconnect.com/dev-guide/html5-v2.0/api-reference

[27] Audio a Video. *Panasonic IPTV Apps Developers* [online].
Panasonic, ©2017 [cit. 21.4.2017].
Dostupné z: http://developer.vieraconnect.com/dev-guide/html5-v2.0/capabilities/audio-and-video

[28] Application Guidelines. *Panasonic IPTV Apps Developers* [online].
Panasonic, ©2017 [cit. 21.4.2017].
Dostupné z: http://developer.vieraconnect.com/dev-guide/html5-v2.0/development-guide/application-guidelines

[29] Self Check List. *Panasonic IPTV Apps Developers* [online].
Panasonic, ©2017 [cit. 21.4.2017].
Dostupné z: http://developer.vieraconnect.com/dev-guide/html5-v2.0/submitting-apps-for-qa/self-check-list

[30] The Panasonic IPTV Apps DEVELOPERS Team. *version compatibility* [elektronická pošta]. Message to: Petr Parek. 15.6.2016 [cit. 21.4.2017].

[31] WEB DESIGN AND APPLICATIONS. *W3C* [online]. W3C, ©2016 [cit. 22.4.2017]. Dostupné z: https://www.w3.org/standards/webdesign/

[32] Apps screen. *SAMSUNG Developers* [online]. SAMSUNG, ©2010-2016 [cit. 22.4.2017]. Dostupné z: http://developer.samsung.com/tv/design/apps-screen

[33] Specifications details for Smart TVs platforms. *mautilus* [online]. Mautilus, ©2017. 17.12.2013 [cit. 22.4.2017]. Dostupné z: https://www.mautilus.com/blog/specifications-details-for-smart-tvs-platforms/

[34] Material icons *Material Design* [online]. Google Inc. [cit. 10.4.2017]. Dostupné z: https://material.io/icons/

[35] *Zero Configuration Networking (Zeroconf)* [online]. Stuart Cheshire [cit. 23.4.2017]. Dostupné z: http://www.zeroconf.org/

[36] DRM Platform Comparison *.DRMtoday* [online]. castLabs, ©2013-2017. Poslední změna květen 2016 [cit. 24.4.2017]. Dostupné z: https://drmtoday.com/platforms/

[37] ISO/IEC 23009-1:2014. *Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats*. International Organization for Standardization, 2014.

[38] AVPlay Guide. *SAMSUNG Developers* [online]. SAMSUNG, ©2010-2016 [cit. 22.4.2017]. Dostupné z: http://developer.samsung.com/tv/develop/tutorials/multimedia/avplay-guide

[39] Playing DRM Content. *webOS TV Developer* [online]. LG Electronics, ©2016 [cit. 23.4.2017]. Dostupné z: http://webostv.developer.lge.com/develop/app-developer-guide/playing-drm-content/

[40] *Let's Encrypt* [online]. Google Inc., ©2010-2017 [cit. 17.4.2017]. Dostupné z: https://letsencrypt.org/

[41] SCHLUETER, Isaac Z. Kik, left-pad, and npm *The npm Blog* [online]. The npm Blog, ©2013-2017. 23.3.2016 [cit. 24.4.2017]. Dostupné z: http://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm

[42] *React. A JavaScript library for building user interfaces* [online]. Facebook Inc., ©2017 [cit. 10.4.2017]. Dostupné z: https://facebook.github.io/react/

[43] *One framework. Angular* [online]. Internet Security Research Group [cit. 11.4.2017]. Dostupné z: https://angular.io/

[44] JELISEJEVS, Pavels.Choosing between React and Angular: An in-Depth Comparison. In: *SitePoint* [online]. SitePoint Pty. Ltd., ©2000-2017, 19.4.2016. [cit. 27.4.2017]. Dostupné z: https://www.sitepoint.com/choosing-between-react-and-angular/

[45] O'SHAUGHNESSY, Kevin. Choosing a JavaScript Framework in 2017. In: *Medium* [online]. Medium, 18.6.2016. [cit. 22.4.2017]. Dostupné z: https://medium.com/@ZombieCodeKill/choosing-a-javascript-framework-535745d0ab90

[46] *Google Trends* [online]. Google Inc. [cit. 22.4.2017]. Dostupné z: https://trends.google.com/

[47] ROSENWASSER, Daniel. Announcing TypeScript 2.3. In: *TypeScript* [online]. Microsoft Corporation, ©2017, 27.4.2017. [cit. 28.4.2017]. Dostupné z: https://blogs.msdn.microsoft.com/typescript/2017/04/27/announcing-typescript-2-3

[48] RAUSCHMAYER, Axel.*Speaking JavaScript*. An in-depth guide for programmers. United States of America: O'Reilly Media, Inc., 2014. 460 s. ISBN 1449365035.

[49] *Sintel, the Durian Open Movie Project* [online].
Blender Foundation [cit. 8.5.2017].
Dostupné z: https://durian.blender.org

[50] *Tears of Steel, Mango Open Movie Project* [online].
Blender Foundation [cit. 8.5.2017].
Dostupné z: https://mango.blender.org/

[51] *Creative Commons Attribution 3.0 Unported (CC BY 3.0)* [online].
Creative Commons [cit. 8.5.2017].
Dostupné z: https://creativecommons.org/licenses/by/3.0/legalcode

Příloha A

Seznam použitých zkratek

VOD Video On Demand

DRM Digital Rights Management

UI User Interface

UX User Experience

HbbTV Hybrid Broadcast Broadband Television

DVB Digital Video Broadcasting

CLI Command Line Interface

LLVM Low Level Virtual Machine

ES EcmaScript

ASF Advanced Systems Format

AAC Advanced Audio Coding

SRT SubRip Text

TTML Timed Text Markup Language

DOM Document Object Model

MPEG Moving Picture Experts Group

DASH Dynamic Adaptive Streaming over HTTP

HLS HTTPS Live Streaming

ISO International Organization for Standardization

SPA Single Page Applications

XSS Cross-site scripting

JSX JavaScript XML

TS TypeScript

AOT Ahead-of-time compilation

JIT Just-in-time compilation

Příloha B

Instalační a uživatelská příručka

URL aplikace na testovacím serveru je `http://sange.hukot.net/hvp/client/?platform=web`, přičemž parametr `platform` nabývá hodnot: `tizen`, `webos`, `firefoxos` a `web`. Testovací účet – jméno: `t`, heslo: `t`.

B.1 Tizen TV

Pro spuštění aplikace je nutná instalace Tizen SDK a *TizenStudio* a provedení následujících kroků, které dobře popisuje dokumentace [9], zde jsou uvedeny pouze ty zásadní. Pokud se spouští aplikace na TV, musí být z podporovaných modelů od 2016 uvedených v dokumentaci.

1. Rozbalit archiv *tizen.zip* a otevřít v *TizenStudio*. Archiv obsahuje kontejner Tizen aplikace, ta po spuštění provede přesměrování na testovací server, kde je vystavená VOD aplikace.
2. *(přeskočit v případě běhu aplikace na TV)* Spustit aplikaci jako *Tizen Web Simulator Application (Samsung TV)*
3. *(přeskočit další kroky v případě běhu aplikace na simulátoru)*
4. Založit si vývojářský účet.
5. Připravit podporovaný model TV – připojit TV a počítač s SDK na stejnou síť a v obchodu s aplikacemi zadat „magickou sekvenci“ čísel – `1, 2, 3, 4, 5`, která otevře dialog, kde lze povolit vývojářský mód. Poté je potřeba TV restartovat.
6. V *TizenStudio* prohledat dostupná zařízení a připojit. Vytvořit certifikát pro podepisování aplikací svým vývojářským účtem a pomocí kontextového menu na připojeném zařízení zvolit *permit install apps*.
7. Spustit aplikaci jako *Tizen Web Application*

B.2 WebOS TV

Aby mohla běžet aplikace na platformě WebOS, je nutná nejprve instalace SDK a WebOSIDE. Následující body popisují pouze klíčové kroky pro úspěšné spuštění aplikace, podrobněji popsáno v dokumentaci [17]. Pokud se spouští aplikace na TV, musí být z podporovaných modelů od 2016 uvedených v dokumentaci.

1. První dva kroky obdobně jako u Tizen, tentokrát pro archiv *webos.zip*.
2. *(přeskočit v případě běhu aplikace na TV)* Spustit emulátor a aplikaci jako *webOS Application* a vybrat jako cíl spuštěný emulátor.
3. *(přeskočit další kroky v případě běhu aplikace na emulátoru)*
4. Založit si vývojářský účet.
5. Připravit podporovaný model TV – připojit TV a počítač s SDK na stejnou síť a z obchodu s aplikacemi nainstalovat a spustit aplikaci *Developer Mode*.
6. V aplikaci se přihlásit ke svému vývojářskému účtu a povolit *Dev Mode Status*. Restartovat TV. Znovu spustit aplikaci a povolit *Key Server*.
7. V *WebOSIDE* založit nové spojení s IP adresou TV.
8. Vytvořené zařízení spojit s TV pomocí kontextového menu volbou *Generate Key*, do pole *Passphrase* zadat odpovídající hodnotu z aplikace *Developer Mode*. Pomocí kontext. menu se připojit.
9. Spustit aplikaci jako *webOS Application* a vybrat jako cíl vytvořené zařízení.

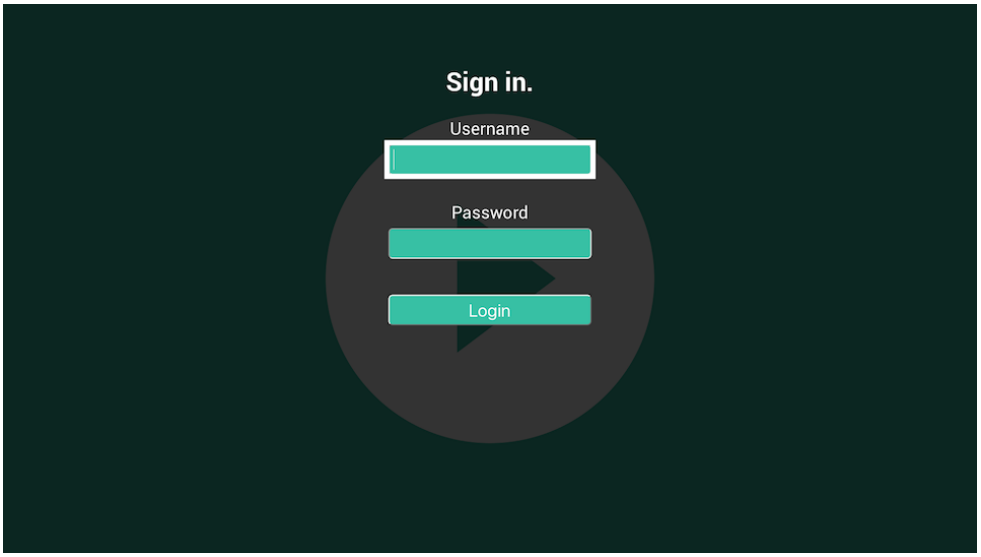
B.3 FirefoxOS TV

U FirefoxOS je situace nejjednodušší –Panasonic nenabízí žádné SDK. Prerekvizitou pro běh aplikace na TV je samotná Panasonic TV, která je z množiny podporovaných modelů [25]. Dále je nutný založený vývojářský účet pro pozdější přihlášení. Spuštění aplikace vyžaduje následující kroky.

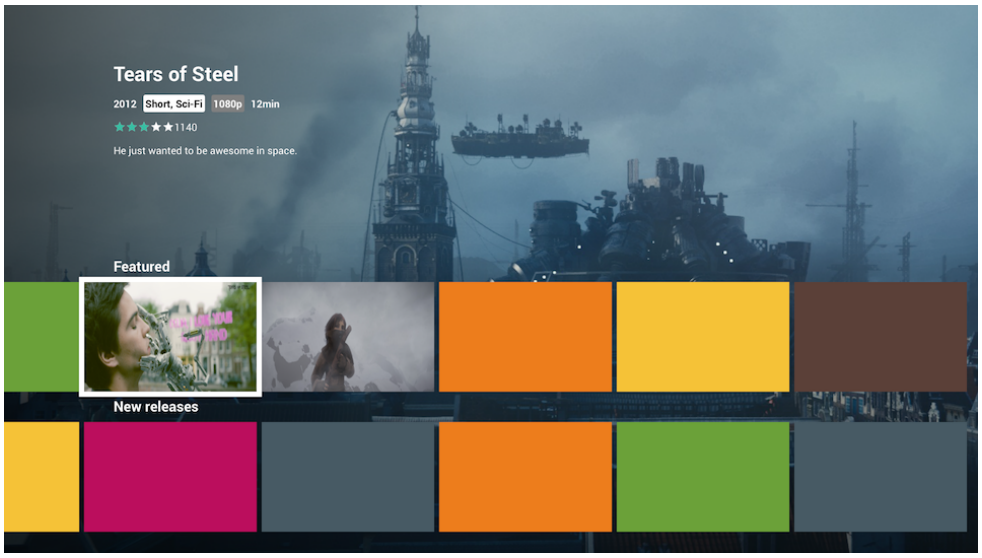
1. Nainstalovat a spustit aplikaci *Panasonic IPTV Apps DEVELOPERS (HTML5)* z obchodu s aplikacemi.
2. V aplikaci vyplnit přihlašovací údaje k vývojářskému účtu.
3. Do pole *Direct* vložit URL testovací aplikace s hodnotou parametru `platform=firefoxos`. Spustit.

Příloha C

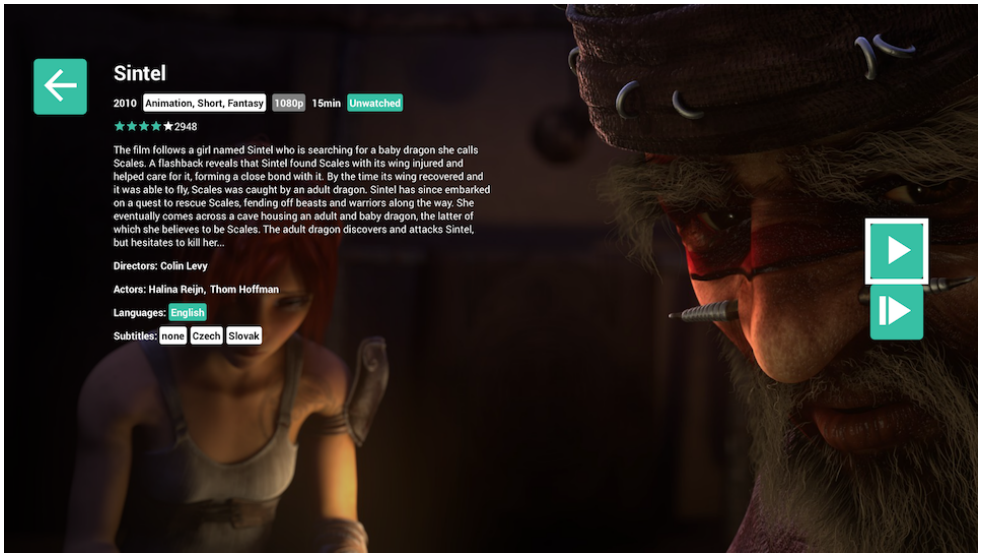
Ukázka aplikace



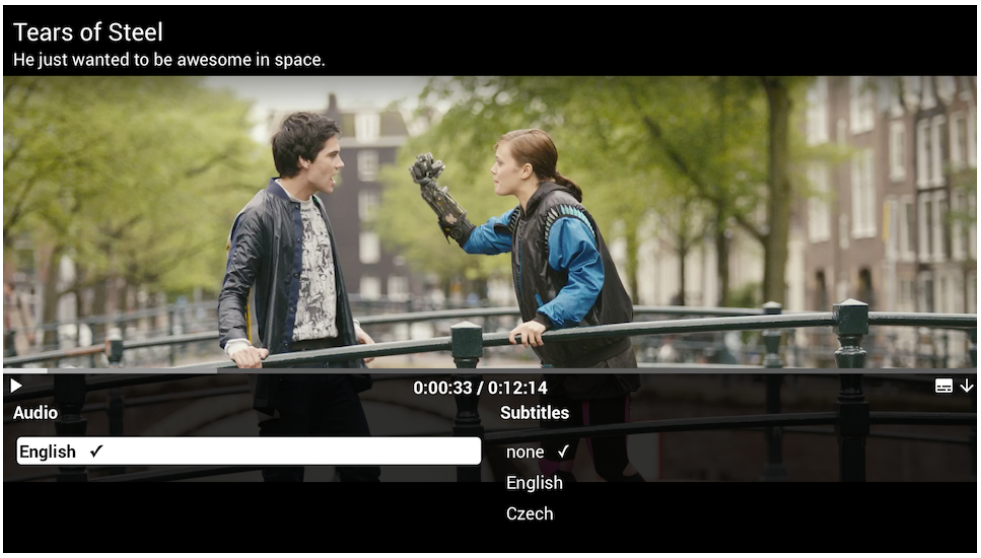
Obrázek C.1: Přihlašovací obrazovka



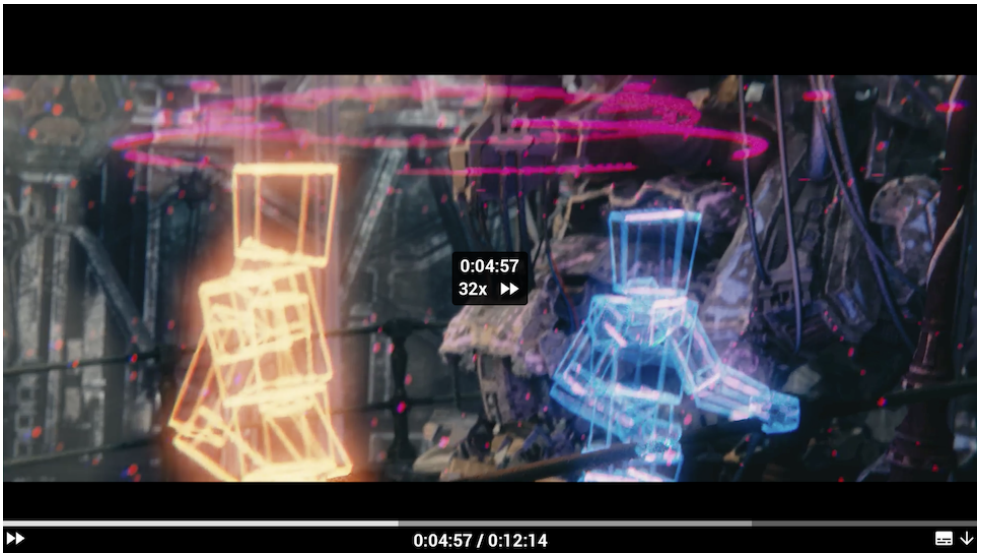
Obrázek C.2: Knihovna filmů



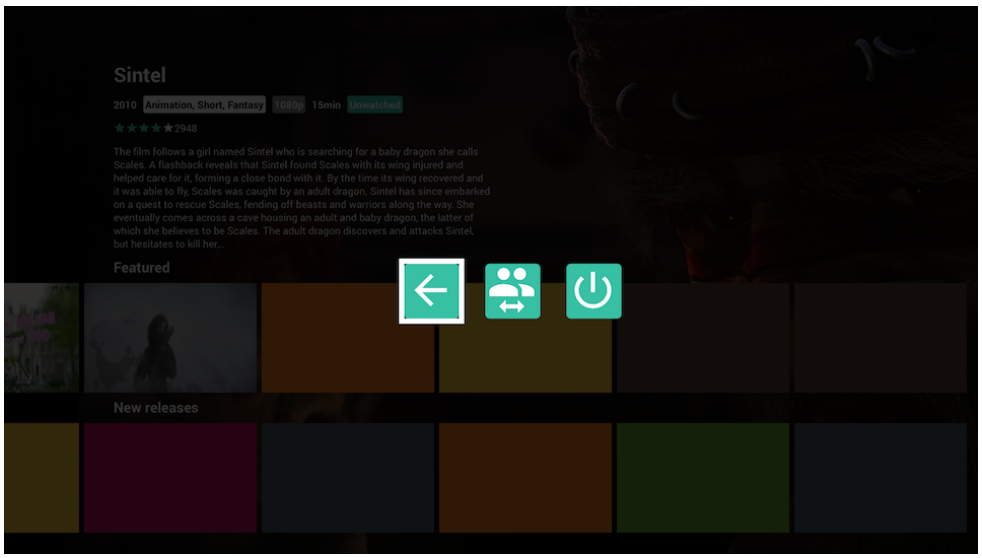
Obrázek C.3: Detail filmu



Obrázek C.4: Video přehrávač se zobrazenou komponentou pro výběr znění a titulků



Obrázek C.5: Video přehrávač při posunu rychlostí 32x vpřed



Obrázek C.6: Obrazovka voleb

Příloha D

Obsah přiloženého CD

- `abstract` ..... abstrakt práce
  - `abstract-cs.txt`
  - `abstract-en.txt`
- `readme.txt` ..... popis obsahu CD
- `src` ..... zdrojové kódy
  - `app.zip` ..... zdrojové kódy aplikace
  - `DP_Marek_Petr_2017.zip` .... zdrojová forma práce ve formátu LaTeX
  - `tizen.zip` ..... Tizen kontejner aplikace
  - `webos.zip` ..... WebOS kontejner aplikace
- `text`
  - `DP_Marek_Petr_2017.pdf` ..... text práce ve formátu PDF