České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Makovec Marek

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: DVBGrabber - systém pro streamování a zaznamenávání televizního vysílání pro osobní použití

Pokyny pro vypracování:

Analyzujte systémy pro záznam a přehrávání (streamimg) televizního vysílání pro osobní použití jako je např. tv.sms.cz, O2 TV, nebo Horizon Go. Navrhněte a implementujte otevřený systém pro záznam a přehrávání televizního vysílání pro osobní použití, který umožní zpracování TV-streamů v rozlišení 576p a vyšším. Analyzujte a vyberte vhodné technologie pro realizaci, jak uživatelského rozhraní aplikace (frontend), tak pro aplikační vrstvu (backend) systému. Systém bude složený z mikroslužeb komunikujících definovaným rozhraním pomocí zprostředkovatele zpráv (message broker). Množství současně zaznamenávaných pořadů (různé TV kanály) bude možné horizontálně škálovat přidání transkodérů. Testování aplikace proveďte na 2 různých konfiguracích PC a navrhněte vhodný postup testování tak, aby ověřil rychlost kódování vybraných záznamů. Uživatelské rozhraní a funkčnost systému otestujte s alespoň 3-mi uživateli.

Seznam odborné literatury:

[1] The Essential Guide to Video Processing, Alan C. Bovik, Academic Press, 2009
[2] Real-Time Heterogeneous Video Transcoding for Low-Power Applications, Springer; 2014 edition (April 9, 2014), Ahmed Abdelgawad
[3] Next-Generation Video Coding and Streaming, Wiley; 1 edition (August 28, 2015)

Vedoucí: Ing. David Sedláček, Ph.D.

Platnost zadání do konce letního semestru 2017/2018

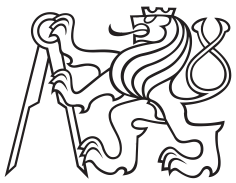
L.S.

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 27.2.2017

Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

DVBGrabber - systém pro streamování a zaznamenávání televizního vysílání pro osobní použití

Bc. Marek Makovec
Otevřená informatika

Únor 2017
Vedoucí práce: Ing. David Sedláček PhD.

Poděkování / Prohlášení

Chtěl bych poděkovat všem vyučujícím, které jsem během studia potkal, vedoucímu semestrálního projektu panu Ing. Martinovi Vaňkovi, vedoucímu této práce, panu Ing. Davidovi Sedláčkovi, PhD., svým kolegům a přátelům z Designeo Creative a hlavně svým rodičům a své přítelkyni.

Vám všem děkuji za to, že jste se mnou měli trpělivost a dali mi šanci dotáhnout to do konce.

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

V Praze dne 25. 5. 2017

.....

Abstrakt / Abstract

Tato práce se zabývá systémem pro šíření televizní signálu DVB-T do lokální sítě a následně možností tento audiovizuální obsah ukládat pro pozdější přehrání.

Klíčová slova: DVB-T, stream, konverze videa, docker, message broker, mencoder, ffmpeg

Goal of this thesis is to show a way how to stream DVB-T signal to LAN network and then store this content for future replay.

Keywords: DVB-T, stream, video conversion, docker, message broker, mencoder, ffmpeg

Title translation: DVBGrabber - system for streaming and recording of television broadcasts intended for personal use

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Jednou z prvních věcí, které jsem po nastěhování se na Strahovské koleje objevil, byl projekt Televize SH, který umožňoval streamování televizního obsahu do sítě LAN a jeho přehrávání na jakémkoliv zařízení, které má do této sítě přístup. Později jsem ještě objevil projekt DVBgrab, který umožňoval obsah televizního vysílání nahrávat a posléze si jej pohodlně stáhnout a zhlédnout později. Tyto možnosti mne fascinovaly.

Je pravda, že v dnešní době, není příliš velký problém se k audiovizuálnímu obsahu dostat i jinými cestami. Řada televizních stanic nabízí archivy pořadů (iVysílání České televize, iPrima.cz, Nova Voyo), jiné firmy se zabývají přímo streamováním filmů a seriálů (Netflix). Tato řešení ale typicky neposkytují živé vysílání a navíc jsou mnohdy značně nepohodlná - vyžadují nainstalované Flash pluginy apod.

Proto jsem si podobné řešení zprovozňoval pro vlastní potřebu doma. Streamování DVB-T dat do sítě jsem si vyřešil poměrně úspěšně už během bakalářského studia. Během magisterského jsem pak vymýšlel, jak k systému přidat ještě možnost obsah zaznamenat pro pozdější zhlédnutí a jak usnadnit přenositelnost celého systému mezi jednotlivými operačními systémy a servery.

Tato diplomová práce je tedy završením několikaleté snahy “mít televizi s možností záznamu kdekoliv, kde je internet”.

Kapitola 2
Analýza

2.1 Stávající řešení na trhu

Na trhu existuje několik různých řešení studované problematiky. Autor by rád podotkl, že s žádným z nich není nijak blízce spjat a nesnaží se je nikterak propagovat; ve všech případech byl nejvýše v roli uživatele a vychází z veřejně dostupných informací.

2.1.1 Televize SH

Televize SH je studentský projekt na kolejích Strahov, který umožňuje členům klubu Sillicon Hill přehrávat televizní vysílání. Detailní informace o projektu jsou k dispozici na webových stránkách projektu[1].

Pro přehrávání televize v lokální síti stačí registrovanému uživateli stáhnout playlist ve formátu m3u a otevřít jej v libovolném programu na přehrávání videa. Autoři konkrétně doporučují VideoLAN VLC[2]. Toto řešení je jednoduché a velice univerzální.

Kromě přehrávání televizního obsahu pro své členy nabízí tento projekt také veřejně dostupný televizní program ve strojově čitelném formátu XML.

Podle informací dostupných na webových stránkách projektu[3] se na stream DVB-T signálu do sítě používají utility DVBlast[4] a MuMuDVB[5]. O obou těchto softwarech přijde zmínka později. Televizní vysílání je pak v síti distribuováno pomocí multicastingu.

Výhoda tohoto řešení je úspora datového toku. Na webových stránkách projektu je zmíněno, že celkový datový tok serveru pro DVB-T je 115 Mbps. Uvážíme-li, že na kolejích bydlí až 4 000 studentů, v případě streamování pomocí metody unicast by teoreticky mohl datový tok vystoupat někam k 460 Gbps, což je absurdně vysoké číslo na úrovni menší páteřní sítě. Pro zajímavost: NIX.cz uvádí jako aktuálně nejvyšší zaznamenaný datový tok řádově 500 Gbps[6].

2.1.2 DVBgrab

Původní systém, který podle dostupných informací[7] fungoval na kolejích Strahov přibližně od roku 2007 do roku 2013. Umožňoval registrovaným členům klubu prohlížet televizní program, plánovat nahrávky a stahovat nahrané pořady. Jako zdroj dat využíval projekt Televize SH.

Z informací o serverech [8] lze částečně odvodit i architekturu aplikace. Systém byl rozdělen na jednotlivé části zodpovědné za obsluhu uživatelů, ukládání videostreamu, transkódování videa i jeho následné servírování uživateli. Vhodné komponenty byly zduplikovány pro zvýšení dostupnosti celé služby.

Podle dostupných informací nestály za koncem projektu problémy technické, ale personální. Projekt přišel o tehdejšího vedoucího a nenašel se nikdo, kdo by jej zastoupil. To je bohužel velká škoda, na Strahovských kolejích to byl velice používaný a oblíbený projekt.

DVBgrab je přímou inspiraci pro projekt DVBGrabber.

[1] `http://televize.sh.cvut.cz`

2.1.3 Grab 2.0

Grab 2.0[7] je duchovním a faktickým nástupcem projektu DVBGrab. Podle dostupných informací obsahuje modernější architektonické prvky (frontendová aplikace komunikuje s backendovou skrz API apod.), ale ve své podstatě poskytuje identickou funkcionalitu.

Byl vytvořen jiným týmem programátorů, než DVBgrab a provozován na kolejích Strahov mezi roky 2016 a 2017, kdy byl pozastaven kvůli problémům s tunelováním multicastových paketů do virtuálních serverů, na kterých byl projekt hostován. Tato práce mimo jiné zmiňuje i způsob, pomocí kterého by se tento problém nechal řešit - viz utilita udpxy v následující kapitole.

2.1.4 Lepší.tv

Mimo jiné také provozovatel televizního programu na adrese tv.sms.cz. V zadání této práce je mylně uveden provozovatel této aplikace. Tímto bych tento přehmat rád uvedl na pravou míru.

Pokud je autorovi známo, toto je jediný komerční projekt, který je fakticky dostupný všem uživatelům v České republice a na rozdíl od např. O2TV[1] není omezen na poskytovatele internetového připojení.

Za cenu 199,- měsíčně nabízí možnost nahrávání pořadů, jejich následné prohlížení, přístup k aktuálnímu TV programu a (skrze jejich vlastní aplikaci pro android) přehrávání televizního vysílání v reálném čase[9].

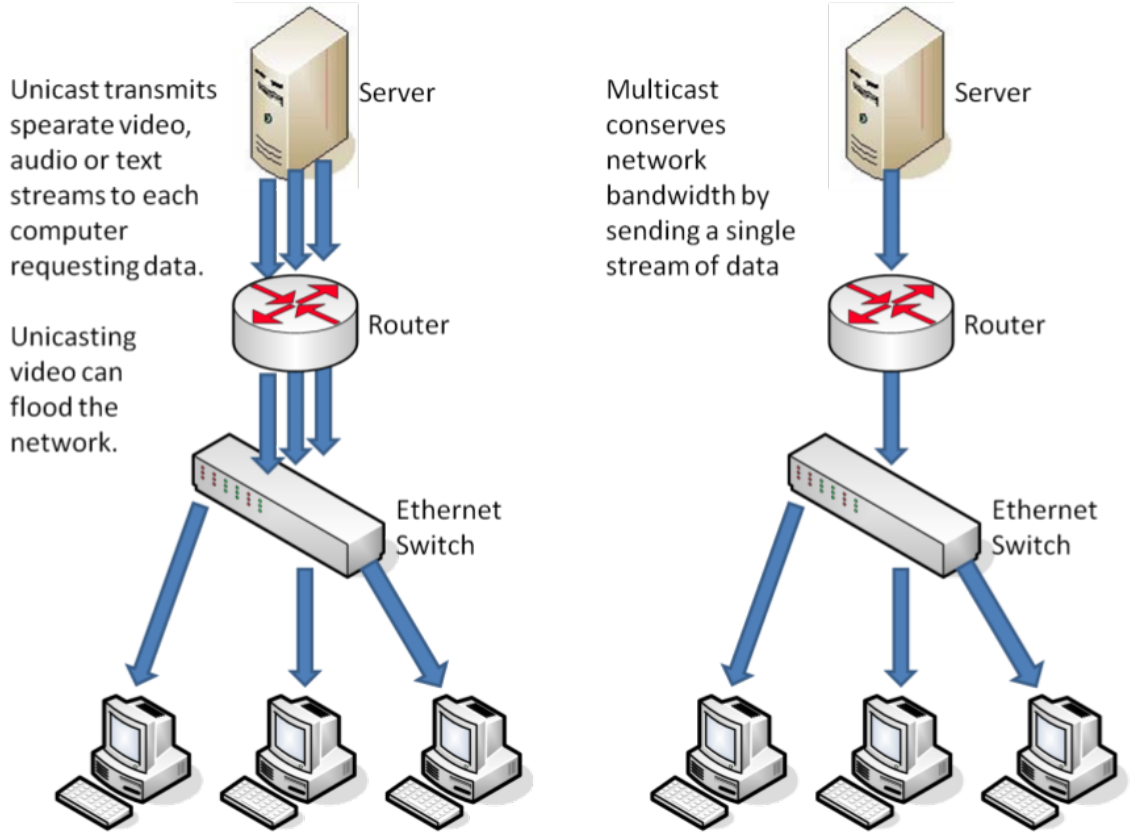
Navíc umožňuje přehrát libovolný pořad 30 dní zpětně. Architektura tohoto projektu tedy pravděpodobně umožňuje nahrávat kontinuální záznam celého dne a skladovat jednotlivé pořady, které jsou následně okamžitě k dispozici pro zákazníka. To je sice velice pohodlné, ale pro potřeby projektu DVBGrabber značně nepraktické. V podmínkách malé domácí sítě, kde je zamýšlen jeho provoz, není možné v reálném čase transkódovat a ukládat televizní vysílání byť jen jediné stanice, natož pak všechna dostupná.

Je ovšem otázka, zda Lepší.tv umožňuje přístup k online vysílání nebo uloženému záznamu i z IP adres mimo Českou republiku. Kvůli autorským právům totiž často dochází k cíleným blokacím zahraničních rozsahů. Systém DVBGrabber by měl umožnit (při dostatečně kvalitním internetovém připojení provozovatele) například sledovat na obědě v lyžařském středisku v Alpách průběh aktuálního utkání Fed Cupu.

2.2 Multicast vs Unicast

Jak již bylo naznačeno, existují dva způsoby, jak streamovat audiovizuální obsah do sítě.

[1] `https://www.o2tv.cz/`



Obrázek 2.1. Schema fungování Unicast versus Multicast [1]

Jednodušší cesta je použít unicasting - tedy pro každého klienta jeden celý datový tok. Tato metoda je datově extrémně neefektivní - posílané pakety jsou stejné pro všechny klienty. To ale při dostatečně malém počtu klientů (řádově jednotky) není problém. Na druhou stranu není potřeba žádná sofistikovaná síťová infrastruktura, ani složitá konfigurace. Výsledné streamy se navíc dají jednoduše tunelovat do internetu.

Složitější, ale úspornější cestou, je pak použít multicasting. Multicasting umožňuje serveru zaslat do sítě jeden datový paket, který switche následně kopírují a distribuují připojeným klientům.

Pro domácí použití má multicasting několik nevýhod:

- Vyžaduje sofistikovanou a dražší síťovou infrastrukturu.
- Vyžaduje náročnější konfiguraci síťových prvků a důkladnější znalost síťových protokolů.
- Multicasting se ze své podstaty nedá tunelovat do internetu.

Tyto nevýhody mají dvě řešení:

- Pokud máme přístup k streamovacímu serveru a nehrozí nám vysoký datový tok, můžeme použít unicast.
- Pokud jsme pouze klientem ve velké síti, můžeme použít nějaký software na převod multicastu do unicastu. Autorovi se osvědčila linuxová utilitka udpxy.

Udpxy[10] umí obsah audiovizuálního streamu šířeného multicastingem pomocí UDP konvertovat do HTTP protokolu šířeného pomocí TCP paketů unicastingem. Díky nízkým hardwarovým nárokům této utility je například možné ji provozovat na routeru (existuje i hotový balíček pro oblíbený alternativní operační systém pro routery OpenWRT) a tím zpřístupnit multicastový stream uvnitř lokální sítě ven do internetu.

Drobná odbočka pro uživatele routerů Mikrotik: OpenWRT je možné na těchto routerech provozovat virtualizované přes funkci zvanou MetaROUTER a tak provozovat udpxy i přesto, že neexistuje její oficiální balíček pro Mikrotik. Snižuje se však stabilita routeru - je potřeba jej cca 1x za měsíc restartovat odpojením od napájení. Jedná se však o řešení "pro osobní potřebu", protože se zde opět setkáváme s problémem vysokého datového toku.

Během experimentálního testování v domácí síti se u multicastingu projevil ještě jeden neduh. Pokud jsou v síti jako access pointy routery Mikrotik ve výchozím nastavení, po zapnutí multicastingu je Wi-Fi síť naprosto zahlcena a nepoužitelná. Router se pravděpodobně pokouší multicastové pakety aktivně vysílat skrz Wi-Fi interface a ten nestačí svou propustností. Toto je hlavní důvod, proč bylo od dalších experimentů s multicastingem upuštěno a vývoj probíhal pouze s posílám dat přes unicast.

Pro úplnost zmiňme ještě broadcasting, tedy posílání všech dat všem klientům. Ten je ale z pochopitelných důvodů pro tuto potřebu naprosto nevhodný a nemá smysl se s ním nadále zabývat.

2.3 Podoba DVB-T

Následující scan2.1 proběhl 8. května 2015. Zdrojem dat je vysílač Kleť (okres České Budějovice).

#	Frekvence (MHz)	Nastavení QAM
1	698	QAM_AUTO
2	506	QAM_64
3	482	QAM_AUTO
4	618	QAM_64

Tabulka 2.1. Scan dostupných multiplexů DVB-T

Z tabulky vyplývá, že v dosahu přijímače jsou 4 multiplexy. Pro získání všech dostupných televizních kanálů jsou tedy zapotřebí 4 DVB tunery - každý naladěn na jednu konkrétní frekvenci a multiplex.

Tabulka 2.2 pak obsahuje Seznam TV kanálů, jejich multiplexů a nastavení tuneru a velikosti datových toků. Následující tabulka pochází ze dne 22. Května 2016. Zdrojem dat je opět vysílač Kleť (okres České Budějovice).

Během doby, která uplynula od mezi scanováním frekvencí a pořizováním vzorových dat z jednotlivých kanálů, se změnilo nastavení Jihočeské TV a Nova Cinema. Při zaznamenávání datových toků těchto stanic se tak nepodařilo pořídit žádná data. Ostatní kanály se všechny schodují v použití kodeků mp2 pro zvuk a mpeg2 pro video. Celkový datový tok pro nahrání těchto datových toků je přibližně 51200 kb/s = 6.2MB/s. Simultánní záznam všech těchto kanálů by pro standardní harddisk neměl být problém.

Název kanálu	Multiplex	PID	Bitrate kb/s	Audio kodek / video kodek	Velikost (GB)	Rozlišení	Délka záznamu (minuty)
ČT1	1	257	3966	mp2 / mpeg2video	1.8	720*576	60
ČT2	1	258	4401	mp2 / mpeg2video	2.0	720*576	60
ČT24	1	259	3255	mp2 / mpeg2video	1.4	720*576	60
ČT4 Sport	1	260	5348	mp2 / mpeg2video	0.388	720*576	10
Nova Cinema	2	514	-	-	-	-	-
Fanda	2	515	4805	mp2 / mpeg2video	2.1	720*576	60
Smíchov	2	517	2736	mp2 / mpeg2video	1.2	720*576	60
Telka	2	518	2666	mp2 / mpeg2video	1.2	720*576	60
6ST:D/Art	3	264	2426	mp2 / mpeg2video	1.0	720*576	60
Prima Love	3	772	2526	mp2 / mpeg2video	1.1	704*576	60
Óčko	3	1025	2244	mp2 / mpeg2video	1.0	544*576	60
Prima ZOOM	3	774	2425	mp2 / mpeg2video	1.0	704*576	60
Jihočeská TV	3	230	-	-	-	-	-
Nova	4	513	3105	mp2 / mpeg2video	1.3	720x576	60
Prima COOL	4	770	4070	mp2 / mpeg2video	1.8	720x576	60
Prima Family	4	773	3038	mp2 / mpeg2video	1.3	720x576	60
Barrandov	4	205	4146	mp2 / mpeg2video	1.8	720x576	60

Tabulka 2.2. Vzorky pořadů vysílaných dostupnými stanicemi

Tato hypotéza byla i experimentálně ověřena: Nahrávání vzorkových datových toku probíhalo paralelním spuštěním 17 instancí programu VLC Media a zápisem na jeden společný disk. Průměrný bitrate byl počítán z celkové doby záznamu. Informace byly získány pomocí utility ffmpeg 2.3.

```
ffmpeg -i ${INPUT_FILE}
```

Tabulka 2.3. Příkaz pro zjištění informací videosouboru.

2.4 Záznam datového toku

Pro záznam2.4 výše uvedených datových toků pro účel prozkoumání kvůli této práci bylo využito programu VLC Media Player[11].

```
vlc -vvv http://192.168.1.203:4022/rtp/239.255.0.63:5004 \
    --sout file/ts:Nova.stream
```

Tabulka 2.4. Příkaz pro nahrátí datového toku pomocí VLC Media Player.

VLC ale přináší zbytečný overhead - přestože není potřeba zobrazovat žádná data, tak vyrenderuje své GUI a může renderovat i aktuálně zaznamenávané video. Existuje několik konzolových variant, ze který je možné si vybrat. Pro potřeby nahrávání existují vhodnější aplikace.

2.4.1 ffmpeg

Ffmpeg je jeden z populárních konzolových nástrojů pro práci s audiovizuálním obsahem využívající knihovnu libavcodec[12]. Během vývoje knihovny došlo v roce 2011 k rozdělení skupiny vývojářů na dvě. Jedna stále pokračuje ve vývoji knihovny libavcodec, druhá začala vyvíjet knihovnu Libav[13].

Skupina za knihovnou libavcodec preferovala rychlý vývoj nových vlastností. Příznivci Libav naproti tomu preferují stabilní čistý kód a kvalitní API.

Krátce po rozdělení se příkaz k ovládání knihoven jmenoval stejně - ffmpeg. V některých argumentech se však lišil. To způsobovalo zmatení uživatelů, kteří se pokoušeli nastavit vstupní argumenty podle rad na fórech, ale výsledkem byla chybová zpráva aplikace.

Celé situaci nepomohl ani fakt, že se v repozitářích Ubuntu a Debian v roce 2011 zaměnil ffmpeg z libavcodecu za ffmpeg z Libav. Vývojáři Libav, vědomi si těchto přešlapů, krátce poté přejmenovali svou verzi ffmpeg na avconv.

Aby zmatkům nebyl konec, v roce 2015 byl ffmpeg z projektu libavcodec do zmíněných repozitářů opět navrácen[1].

Během vývoje projektu DVBGrabber bylo s utilitou ffmpeg experimentováno, ale výsledky nedopadly podle očekávání. Utilita má několik zásadních nevýhod:

- Trvá jí poměrně dlouho, než se zapne a začne nahrávat (řádově cca 15s).
- Chová se poměrně nestabilně. Nejednou se během nahrávání sama vypnula. Jednou pak zamrzla takovým způsobem, že ji nebylo možné ani vypnout příkazem SIGTERM, ani nebylo možné zrestartovat Dockerový kontejner, ve kterém byla spuštěna. Pomohl teprve restart celého serveru.

[1] https://stackoverflow.com/a/9477756

2.4.2 mplayer

Mplayer[14] je v první řadě přehrávač audiovizuálního obsahu, ale kromě toho obsahuje i šikovný nástroj mencoder pro manipulaci s ním. Narozdíl od ffmpeg / avconv výrazně rychleji začíná zaznamenávat stream (boot aplikace trvá cca 2-5 sekund) a chová se značně stabilněji. Na rozdíl od ffmpeg ale neumožňuje specifikovat dobu nahrávání streamu. Je potřeba použít malý trik a po uplynutí daného času jeho proces vypnout zasláním SIGKILL.

2.4.3 cvlc

CVLC je konzolová varianta VLC. V rámci rešerše ji nebyla věnována další pozornost, zejména kvůli nepřehledné dokumentaci a poměrně komplikovanému kombinování vstupních parametrů. V tomto výčtu je tedy zmíněna hlavně pro úplnost.

2.5 Benchmark pro odhad výkonu sestavy

Hlavním kritickým místem celého systému je proces zaznamenávání datového streamu. Pokud by datové úložiště nestihlo nahrávat, v zaznamenaném videu by byly výpadky v lepším případě artefakty, v horším případě kompletní výpadek nahrávky a záznam byl bezcenný.

V nejhorší možné, a velmi nepravděpodobné situaci, kdy si uživatel nechá nahrát všechny pořady najednou, je potřeba ukládat všechny streamy najednou. Podle výše uvedené tabulky kanálů by se muselo zaznamenávat 17 streamů o průměrné rychlosti 3000 kbit/s.

Pro simulaci zápisu 17 různých sekvenčních streamů lze použít utilita fio[15]. Konfigurace, která způsobí, že fio sekvenčně zapisuje 17 různých 1G velkých souborů vypadá velmi jednoduše2.5:

```
[sequence-writes]
rw=write
numjobs=17
size=1
```

Tabulka 2.5. Konfigurace benchmarkové utility fio.

Výsledný report poskytuje informace o každém vlákně zvlášť. Na konci následuje souhrn, kde je pod atributem aggrb schovaná průměrná rychlost zápisu. Je-li větší než požadovaných 3000kbit/s, dá se předpokládat, že dané datové úložiště ustojí i zápis všech streamů najednou.

Pro porovnání, SSD disk Kingston OCZ Vertex 2 (128 GB) dosahuje rychlosti zápisu aggrb=94899KB/s.

2.6 Kodeky pro uložení záznamu

Po uložení surových dat ze streamu, je vhodné je zpracovat - převést do nějakého formátu s lepším poměrem kvality obrazu ku velikosti výsledného záznamu. Nahrané vzorky trvají cca 60 minut a jejich velikost dosahuje v některých případech téměř 2 GB.

Abychom se vyhnuli znovuvynalézání kola, inspirujeme se s pro začátek volbou kodeků u audiovizualních děl, které je možné si pro vlastní potřebu legálně stáhnout z veřejných úložišť.

Pro ukázku jsem prozkoumal jistý nejmenovaný, přibližně 2 hodiny a 16 minut trvající film v rozlišení 1920x1080px. Jako audio kodek využívá AAC s bitrate 128 kbit/s. Video kodek je pak h264, bitrate dosahuje 2049 kbit/s. Celková velikost souboru je pak 2.2GB. To je stejná velikost, jakou má hodinový stream stanice Fanda.

Využití audio kodeku AAC má ale jednu zásadní nevýhodu: autor této práce má osobní skutečnosti s tím, že někteří výrobci televizí (například LG) kodek AAC záměrně nepodporují, jelikož je údajně často využíván piráty. Jako vhodnější se tedy jeví využít kodek mp3 nebo ogg.

Podle wikipedie ArchLinux[16] je možné konverzí do h264 u mencoder dosáhnout následujícím příkazem 2.6:

```
COMMON=fast_pskip=0:tune=film:frameref=15:bitrate=3000:threads=auto

mencoder ${SOURCE} -oac copy -ovc x264 -x264encopts \
pass=1:preset=veryslow${COMMON} -o /dev/null

mencoder ${SOURCE} -oac copy -ovc x264 -x264encopts \
pass=2:preset=veryslow:${COMMON} -o ${OUTPUT}
```

Tabulka 2.6. Dvouprůchodový transcoding videa do h264.

Druhou možností je inspirovat se řešením, které bylo použito v projektu DVBgrab. Autorovi se v archivu podařilo najít starý záznam, který si kdysi nechal DVBgrabem nahrát. Po inspekci pomocí utility ffmpeg se dozvíme následující informace2.7:

```
Metadata:
    encoder         : MEncoder dev-SVN-r26940
  Duration: 00:31:00.20, start: 0.000000, bitrate: 1797 kb/s
    Stream #0.0: Video: mpeg4 (Simple Profile), yuv420p,  \
        720x576 [PAR 64:45 DAR 16:9], 25 tbr, 25 tbn, 25 tbc
    Stream #0.1: Audio: mp2, 48000 Hz, stereo, s16, 192 kb/s
```

Tabulka 2.7. Inspekce záznamu z DVBgrab

Tedy použitý audiocodec je mp2, videocodec je mpeg4. Zajímavostí je, že autoři DVBgrab se také přiklonili ke konverzi videa pomocí utility mencoder.

Podle stejného zdroje[17], kde jsme našli způsob konverze do h264, můžeme převzít i konverzi do mpeg42.8:

```
#!/bin/bash

$COMMON1=mbd=2:mv0:trell:v4mv:cbp:predia=6:dia=6:precmp=6:cmp=6:subcmp=6:preme=2:qns=2:vbi

mencoder ${SOURCE} -oac copy -ffourcc DX50 \
   -ovc lavc -lavcopts vpass=1:${COMMON1}  \
   -subfont-text-scale 3 -o ${OUTPUT}
mencoder ${SOURCE} -oac copy -ffourcc DX50 \
   -ovc lavc -lavcopts vpass=3:${COMMON1}  \
   -subfont-text-scale 3 -o ${OUTPUT}
mencoder ${SOURCE} -oac copy -ffourcc DX50 \
   -ovc lavc -lavcopts vpass=3:${COMMON1}  \
   -subfont-text-scale 3 -o ${OUTPUT}
```

Tabulka 2.8. Tříprůchodový transcoding videa do mpeg4

Kapitola 3
Návrh řešení

3.1 Funkční požadavky

- Možnost přehrávat televizní vysílání v reálném čase.
- Možnost naplánovat zaznamenání definovaného pořadu.
- Automatická konverze nahraného pořadu do úspornějšího kodeku / formátu.
- Automatická notifikace o úspěšném nahrání pořadu.
- Možnost stáhnout nahraný pořad do lokálního počítače.
- Automatické smazání nahraných dat, pokud uživatelé dají najevo, že již o ně nemají zájem.
- Možnost škálovat množství aktivních transcoderů.
- Možnost prohlížet televizní program na nejbližší dny.

3.2 Nefunkční požadavky

- Zjednodušená instalace: během instalace systému nebude potřeba manuálně instalovat jednotlivé komponenty (message broker, sql databáze, podpůrné nástroje pro zaznamenání / konverzi).
- Podpora OS Linux.
- Provozování na jednom nebo několika serverech zároveň - možnost rozdělit jednotlivé úlohy na servery podle jejich hardwarové konfigurace.
- Obsluha řádově jednotek uživatelů.
- Nasazení primárně v rámci intranetu.

3.3 Získání dat z DVB-T

3.3.1 Hardware - DVB tunery

Se streamováním DVB-T do počítačové sítě bylo dlouhodobě experimentováno. Po hardwarové stránce se osvědčily USB tunery PCTV 73e. Jejich hlavní výhoda spočívá v nativní podpoře v linuxové distribuci Debian (minimálně od edice Debian Wheezy).

Všechny čtyři tunery jsou k počítači připojeny jedním USB hubem 3.1. Dlouhodobě se osvědčil sedmiportový, napájení USB 2.0 hub Belkin F5U307-BRN.

Obrázek 3.1. Fotografie reálného zapojení antény DVB tuneru

3.4 Scan éteru

Po připojení DVB-T tunerů k počítači je potřeba zjistit, jaké jsou dostupné vysílací frekvence. Na to lze s úspěchem použít utilitu w_scan[18]. Na zapnutí scanování stačí jednoduchá konfigurace3.2:

```
w_scan -c CZ > frequencies.log
```

Obrázek 3.2. Příklad spuštění utility w-scan

Výstupem je pak seznam kanálů nalezených na dostupných frekvencích:

```
...
CT 1;Ceska televize:698000:...:T:27500:257=2:273=cze,275=cze:289:...
CT 2;Ceska televize:698000:...:T:27500:513=2:529=cze,531=cze:545:...
CT 24;Ceska televize:698000:...:T:27500:769=2:785=cze,787=cze:801:...
CT 4;Ceska televize:698000:...:T:27500:1025=2:1041=cze,1043=cze:1057:...
...
```

Obrázek 3.3. Ukázka výstupu programu w-scan (zkráceno)

Pro další potřebu jsou nejzajímavější první tři parametry. Prvním z nich je název kanálu, druhý parametr obsahuje název provozovatele multiplexu a třetím parametrem je frekvence multiplexu. S tou budeme pracovat v následujících programech, které umožňují streamovat DVB-T signál do sítě.

Ani jeden z nich nemá nijak vysoké hardwarové nároky. Jejich testování probíhalo na virtuálním počítači s procesorem Intel Xeon o taktu 2 GHz a 8 GB RAM. Zátěž procesoru a využití RAM se ve všech případech pohybovala v jednotkách procent. Server na stream televizního signálu tedy může mít například podobu Intel Compute Sticku umístěného na půdě pod anténou.

3.4.1 Software - Getstream

Autorem projektu[19] je Florian Lohoff. Domácí stránka projektu[1] je na informace poměrně skoupá a najít dokumentaci, která by do hloubky popisovala konfiguraci aplikace, je poměrně problém.

Tento program byl testován několik let (přibližně od roku 2013 do května 2015). Zásadní nevýhodou byla nestabilita - pokud některý z tunerů nemá opravdu kvalitní signál, ve výsledném streamu jsou značné artefakty a několikrát do měsíce dochází k pádům celé aplikace.

Na druhou stranu bylo výhodou, že dokáže všechny tunery obsluhovat v jedné instanci a nativně umožňuje unicasting.

```
adapter 3 {

        packet-buffer 50;
        stat-interval 120;
        stuck-interval 200;

        dvb-t {
                frequency 482000000;
                bandwidth 8;
                transmission-mode 8;
                guard-interval 8;
                hierarchy none;
                modulation auto;
        };

        stream {
                name "zoom";
                input {
                        pnr 774;
                };
                output-http {
                        url /zoom;
                };
        };
};
```

Obrázek 3.4. Příklad konfigurace aplikace getstream

[1] http://silicon-verl.de/home/flo/projects/streaming/

3.4.2 Software - DVBlast

Utilita DVBlast[4] pochází od tvůrců VLC Media Player. Větší vývojářská základna v tomto případě přislibuje i kvalitnější a stabilnější software. DVBlast je výrazně méně náchylný na nekvalitní signál a artefakty se projevují jen minimálně.

Narozdíl od getstreamu neumožňuje unicasting a pro každý tuner je potřeba spustit separátní instanci programu.

Výhodou je, že je výrazně stabilnější. Za rok provoze jej bylo třeba ručně restartovat asi jednou. Pravděpodobně tomu ale předcházela výrazná manipulace s anténním kabelem a tedy možná ztráta signálu. Za to ovšem nelze DVBlast nijak vinit.

```
; RUN: dvblast -a 0 -c multiplex.1.cfg -f 698000000 -m qam_auto -b 8 -e
; CT1
239.255.0.50:5004    1    257
; CT2
239.255.0.51:5004    1    258
; CT24
239.255.0.52:5004    1    259
; CT 4
239.255.0.53:5004    1    260
```

Obrázek 3.5. Příklad konfigurace aplikace DVBlast

3.4.3 Software - MuMuDVB

Posledním zkoušeným softwarem je aplikace MuMuDVB[5] (Multi Multicast DVB). Na rozdíl od předchozích aplikací není potřeba definovat každý kanál zvlášť, ale je možné nechat aplikaci daný multiplex oscanovat a vygenerovat pro něj m3u playlist. Navíc podporuje multicasting i unicasting. Nevýhodou je nutnost pro každý multiplex spustit jednu instanci programu. Po stránce stability je aplikace srovnatelná s aplikací DVBlast.

Díky nativní podpoře unicastingu a nejsnazší konfiguraci byl MuMuDVB použit pro získávání DVB-T dat v systému DVBGrabber.

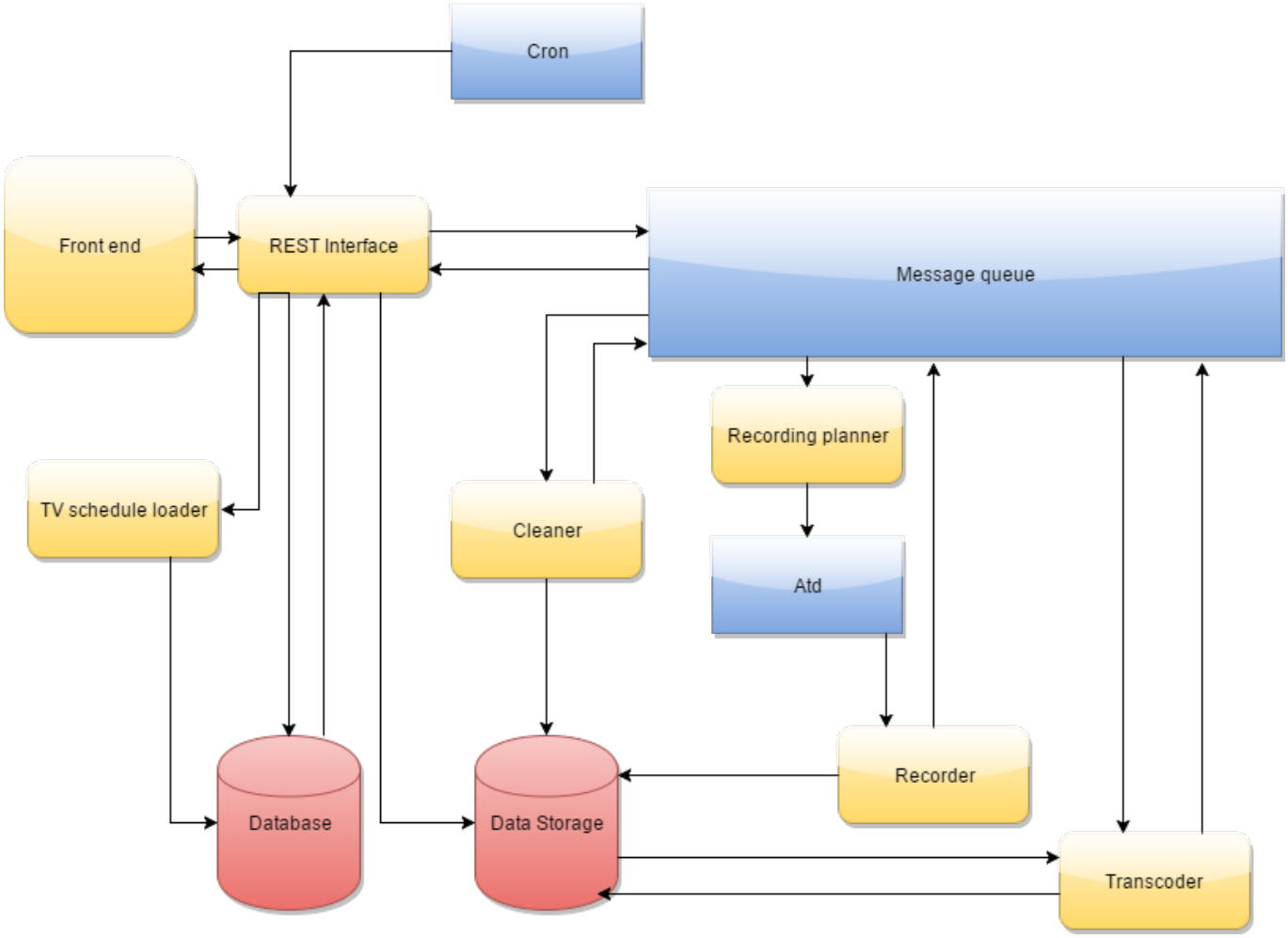
```
timeout_no_diff=100
dvr_thread
rewrite_pat=1
rtp_header=1
check_status=1
pol=h
srate=27500
autoconfiguration=full
autoconf_radios=1
autoconf_pid_update=0
autoconf_ip4_header=239.10
multicast_ttl=2
multicast_auto_join=1
unicast=1
ip_http=192.168.1.203
sap=1
sap_default_group=MCAST
card=1
freq=506
port_http=91
multicast_ipv4=0
multicast_ipv6=0
```

Obrázek 3.6. Příklad konfigurace aplikace MuMuDVB

Tento program byl následně zvolen a provozován v referenční instanci systému DVB-Grabber. Jeho konfigurace však není součástí systému, je potřeba, aby ji každý potencionální uživatel provedl sám na vhodném počítači, který má přístup k signálu DVB-T.

3.5 Architektura systému pro záznam dat

Celý systém je navrhnut modulárně, aby bylo možné podle potřeby jednotlivé komponenty měnit, případně škálovat. Navrhovaná struktura je na následujícím schématu3.7.



Obrázek 3.7. Schéma systému

Modře označené části jsou existující systémové aplikace. Message queue (konkrétně RabbitMQ) je použit na doručování zpráv mezi systémy a k horizontálnímu škálování v případě transcoderu, cron a atd slouží pro plánované spouštění úloh - načítání TV programu a zahájení nahrávání.

Žlutě jsou vyznačeny části systému, které jsou vytvořeny na míru. Jde o různé ovládací a plánovací skripty pro nahrávání a transcoding videa, čištění nepotřebných dat, backend s implementovanou logikou a životním cyklem entit a REST API a v neposlední řadě také frontendová část - v tomto případě webový klient napsaný v HTML5 a JavaScriptu.

Červeně jsou pak vyznačena datová úložiště. V SQL databázi jsou uložené informace o uživatelích, nahrávaných pořadech a televizním programu. V Data Storage jsou uloženy samotné audiovizuální soubory.

3.5.1 Prostředí

Celý systém je navržen s úmyslem vyvíjet a provozovat jej v Dockeru. Docker je projekt umožňující provozovat software v uzavřených kontejnerech. Nejedná se o typickou virtualizaci, ale spíš o oddělení uživatelských prostorů jednotlivých kontejnerů.

Tento způsob nasazení má několik výhod:

- Přenositelnost: Ačkoliv je nativním prostředím pro Docker Linux, je možné jej provozovat i na Mac OS X a na Windows.
- Škálování v rámci jednoho počítače: Docker s rozšířením Docker Compose umožňuje velice jednoduché spuštění několika instancí stejného kontejneru na jednom počítači.
- Unifikovaný vývoj: nezávisle na operačním systému vývojáře můžou všichni pracovat ve stejném prostředí, které lze velice rychle a jednotně vytvořit a udržovat na základě připojených manifestů zvaných Dockerfile.
- Snadné nasazení: Každý kontejner obsahuje všechno potřebné softwarové vybavení. Pokud aplikace potřebuje nějakou novou externí závislost, například na PostgreSQL databázi, stačí přidat kontejner s PostgreSQL a přes Docker Compose jej propojit s kontejnerem s aplikací.
- Watchdog: Při použití Docker Compose je možné nastavit automatický restart kontejneru, pokud jeho hlavní proces skončí. To se dá použít jako jednoduchý watchdog.

Samozřejmě, že nasazení v Dockeru má i nějaké nevýhody:

- Mimo prostředí Linuxu není možné promapovat do Docker containeru žádné USB zařízení. Toto je jeden z důvodů, proč systém DVBGrabber neobsahuje MuMuDVB přímo v sobě, ale je potřeba jej zprovoznit separátně.
- Každý kontejner udržuje naživu právě jeden proces. Pokud tento proces skončí, kontejner se automaticky vypíná (včetně všech dalších procesů, které v něm běžely).
- Nižší výkon. Aplikace vysoce náročná na HDD a CPU bude v Dockeru o něco zpomalena. Toto zpomalení nelze generalizovat, jelikož pro každou platformu existují jiné triky, jak dopad na konkrétní zdroj (ať už na procesor či pevný disk) zmenšit.

Zároveň je ale možné jednotlivé části systému (např. transcoder) pustit přímo a připojit je na instance Message brokeru. V takovém případě je však potřeba doinstalovat závislosti ručně. Potřebné kroky lze odvodit z konkrétního Dockerfile dodávaného k dané komponentě.

3.6 REST API provider

3.6.1 Použité technologie

Provider je napsaný v jazyce PHP s využitím frameworků Symfony 2, Doctrine 2, relační databází PostgreSQL a servírovaný webovým serverem Nginx.

Jazyk PHP asi není potřeba niktera zvlášť představovat. Jedná se o jednoduchý interpretovaný jazyk s oblibou používaný na tvorbu webových aplikací. Přestože se na něj spousta vývojářů dívá "skrz prsty", autor se domnívá, že obecně nezáleží na daném jazyce, ale na jeho použití a že schopnost psát čitelný kód není vlastností jazyka, ale programátora.

Framework Symfony[20] je jeden z celosvětově nejrozšířenějších MVC frameworků pro PHP. Tento framework má velké množství rozšíření (takzvaných Bundles) které přidávají nové vlastnosti a umožňují integraci s různými službami. Projekt DVBGrabber nejvíce využívá následující rozšíření:

- FOSUserBundle[1]: Balík usnadňující práci s entitou uživatele, jeho autentifikaci a autorizaci.
- DoctrineBundle[2]: Balík nabízející propojení s ORM knihovnou Doctrine 2.
- DoctrineMigrationsBundle[3]: Balík rozšiřující Doctrine 2 o možnost databázových migrací a integraci těchto migrací do frameworku.
- NelmioCorsBundle[4]: Balík řešící konfigurace HTTP hlaviček pro Cross Origin Request Sharing.
- OldSoundRabbitMQBundle[5]: Integrace s RabbitMQ.

O vyspělosti tohoto frameworku svědčí, že si jej pro provozování své platformy zvolila i společnost Spotify.

Doctrine 2[21] je jedno z nejrozšířenějších ORM (Object Relational Mapper - knihovna pro mapování relačních databází na objekty) pro PHP. Doctrine přiznává, že se svou logikou výrazně inspiruje od Hibernate ze světa Javy. Mapování entit na tabulky lze realizovat anotacemi nebo konfiguračními soubory. Entity je možné z databáze získávat přes přichystané metody v repozitářích, nebo pomocí takzvaných querybuilderů. Zde se používá dotazovací jazyk DQL - Doctrine Query Language. Tento jazyk je částečně podobný SQL, ale obsahuje jen zlomek jeho funkcionality. Z pochopitelných důvodů například úplně chybí RIGHT JOIN nebo LEFT OUTER JOIN.

Další nevýhodou DQL je pak malé množství implementovaných databázových funkcí. Doctrine se snaží podporovat pouze takové funkce, které jsou společné mezi všemi podporovanými databázovými stroji (MySQL, PostgreSQL, Oracle a SQLite). Některé speciální vlastnosti databází se dají řešit pomocí rozšiřujících pluginů. Například GIS funkce v rozšíření PostGIS pro PostgreSQL.

Pokud je potřeba, například kvůli reportingu, napsat nějaký ještě složitější dotaz, umožňuje Doctrine vytvořit takzvané Native Query - dotaz psaný v SQL daného databázového stroje. Výsledek je pak možné buď získat jako pole dat, nebo pomocí takzvaných ResultSetMapperů namapovat zpátky na entity a dále s nimi pracovat, jak je ukázáno níže3.8.

[1] https://github.com/FriendsOfSymfony/FOSUserBundle
[2] https://github.com/doctrine/DoctrineBundle
[3] http://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html
[4] https://github.com/nelmio/NelmioCorsBundle
[5] https://github.com/php-amqplib/RabbitMqBundle

```
/**
 * @return Show[]
 */
public function findShows(
    string $channelId,
    string $showTitle,
    \DateTimeInterface $start,
    \DateTimeInterface $stop
) {
    $rsm = new ResultSetMappingBuilder($this->_em);
    $rsm->addRootEntityFromClassMetadata(
        'AppBundle:Show', 'SHOW', [], $rsm::COLUMN_RENAMING_NONE
    );

    $sql = '
        SELECT SHOW.*
        FROM shows SHOW
        JOIN channels CHANNEL ON CHANNEL.id = SHOW.channel_id
        WHERE CHANNEL.xml_id = :xmlId
        AND SHOW.title = :title
        AND extract(epoch FROM SHOW.start)
            BETWEEN :startBegin AND :startEnd
        AND extract(epoch FROM SHOW.stop)
            BETWEEN :stopBegin AND :stopEnd
    ';

    $nativeQuery = $this->_em->createNativeQuery($sql, $rsm);
    // ... definice $startBegin|end a $stopBegin|end byla vynechána

    $nativeQuery->setParameter('startBegin', $startBegin);
    $nativeQuery->setParameter('startEnd', $startEnd);
    $nativeQuery->setParameter('stopBegin', $stopBegin);
    $nativeQuery->setParameter('stopEnd', $stopEnd);
    $nativeQuery->setParameter('xmlId', $channelId);
    $nativeQuery->setParameter('title', $showTitle);

    return $nativeQuery->getResult();
}
```

Obrázek 3.8. Příklad použití Native Query a resultSetMapperu

Relační databáze byly zvažovány následující:

- MySQL
- PostgreSQL
- Oracle

MySQL databáze obecně trpí problémy v oblasti ochrany integrity dat. Oproti tomu Oracle je ohledně ACID[1] naprosto striktní, ale jeho instalace do prostředí Dockeru a následná distribuce takového kontejneru porušuje licenční podmínky Oracle a obecně administrace tohoto datového stroje je poměrně obtížná.

[1] ACID: Atomicity, Consistency, Isolation, Durability

Jako vhodný kompromis byl proto zvolen databázový stroj PostgreSQL; není problém najít přichystané dobré Docker kontejnery s připravenou databází a stále zůstane zachována výhoda kvalitního databázového stroje respektujícího ACID.

3.6.2 Autentifikace

REST API provider přijímá klasické HTTP požadavky. Autentifikace klienta probíhá metodou Basic access authentication - tedy každý požadavek obsahuje hlavičku Authorization obsahující uživatelské jméno a hesla klienta zakódované pomocí base64. Nejedná se o nijak bezpečnou variantu autentifikace, ale tento problém lze částečně vyřešit nasazením HTTPS, navíc je celý systém primárně zamýšlen pro použití v intranetu.

3.6.3 Popis endpointů

```
GET /api/schedule/full
```

Vrací kompletní televizní program od aktuálního okamžiku.

```
POST    /api/show/scheduled/{id}
```

Naplánuje nahrání televizního pořadu s daným identifikátorem pro aktuálního uživatele.

```
DELETE  /api/show/scheduled/{id}
```

Zruší nahrávání televizního pořadu s daným identifikátorem pro aktuálního uživatele.

```
DELETE  /api/show/recorded/{id}
```

Dá systému pokyn, že aktuální uživatel již nepotřebuje na serveru uchovávat nahraný pořad. Pokud všichni zájemci o daný pořad odešlou tento příkaz, dojde ke smazání nahraných dat.

```
GET     /api/user/scheduled
```

Vrací seznam pořadů naplánovaných k nahrání pro daného uživatele.

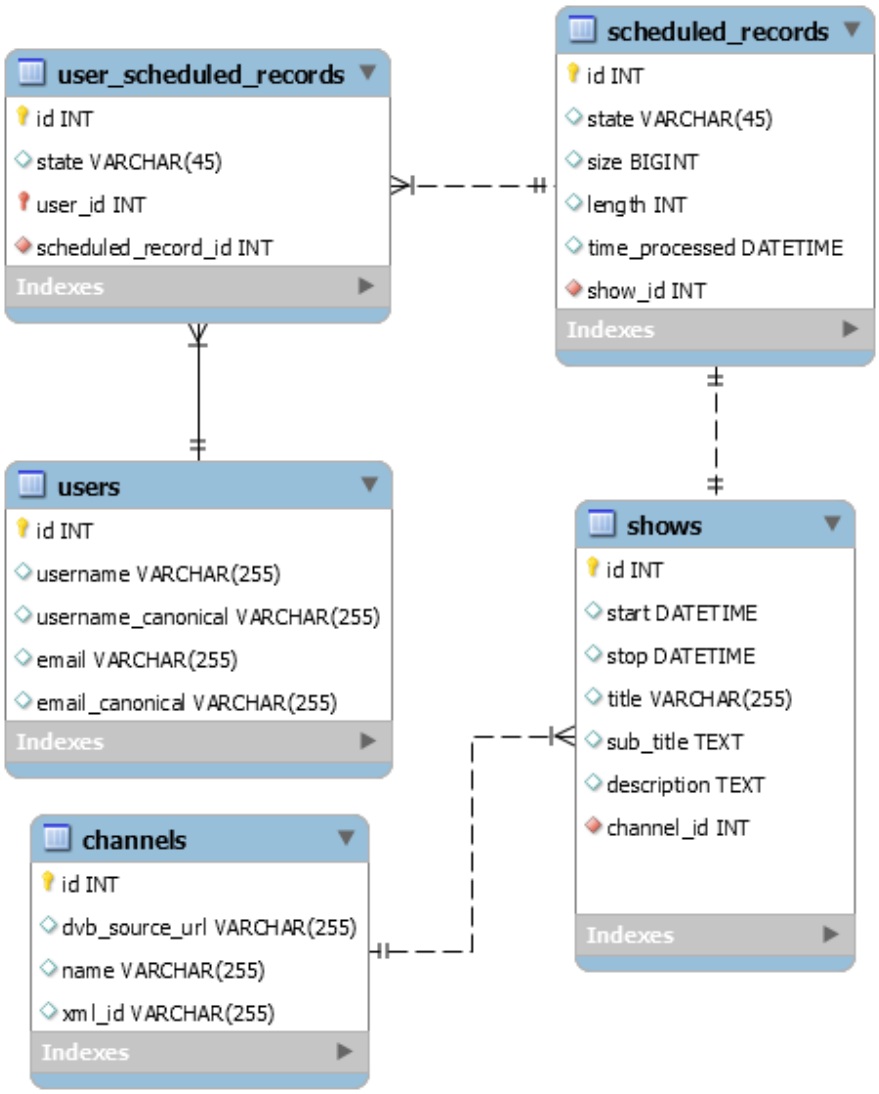
```
GET     /api/user/recorded
```

Vrací seznam nahraných pořadů pro daného uživatele.

Server ve všech případech vrací data ve formátu JSON a v kódování UTF-8.

3.6.4 Databázové schéma

Na databázovém schématu3.9 jsou zachyceny nejdůležitější tabulky a atributy databáze projektu DVBGrabber. Oproti reálné databázi byly odebrány zbytečné sloupečky z tabulky users (velká část z nich aktuálně nemá v aplikaci význam, ale jsou vyžadována rozšířením FOSUserBundle) a také chybí servisní tabulka doctrine_migrations, která je spravována rozšířením DoctrineMigrationsBundle a do které zaznamenává historii provedených migrací.



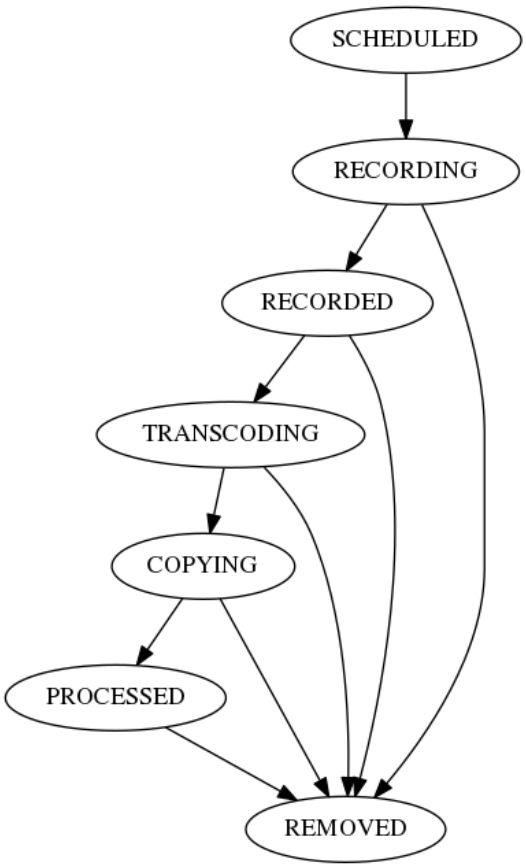
Obrázek 3.9. Schéma databáze

3.6.5 Entity systému

Entity svými názvy vycházejí z databázových tabulek, kde jsou uloženy jejich hodnoty. Aplikace dodržuje konvenci, kdy databázové tabulky jsou pojmenovány pomocí snake_case a v množném čísle, zatímco entity jsou zásadně v jednotném čísle a v PamelCase. Jednotné číslo symbolizuje fakt, že konkrétní instance jedné entity je jeden konkrétní záznam z databáze.

Entita ScheduledRecord reprezentuje žádost o nahrání daného pořadu. Tato entita prochází postupně následujícími stavy:

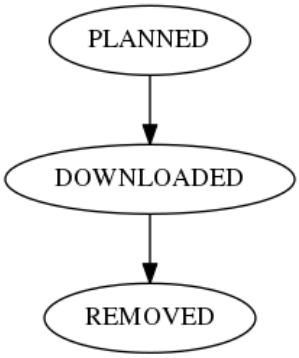
- SCHEDULED - Pořad je naplánován k zaznamenání.
- RECORDING - Pořad se právě nahrává.
- RECORDED - Pořad byl nahrán a čeká na transcoding.
- TRANSCODING - Pořad právě prochází procesem konverze.
- COPYING - Pořad byl nahrán a kopíruje se na veřejnou část aplikace. Tento stav byl doplněn pro možné budoucí rozšíření, aktuálně se však nepoužívá.
- PROCESSED - Pořad byl zpracován a je možné jej stáhnout.
- REMOVED - Pořad byl smazán ze systému. Informace o něm zůstává v databázi kvůli možným budoucím statistikám.



Obrázek 3.10. Schéma přechodů stavů entity ScheduledRecording

Entita UserScheduledRecord propojuje konkrétního uživatele s konkrétní žádostí o nahrání pořadu. Její životní cyklus je výrazně jednodušší. Stavy, kterými může postupně projít:

- PLANNED - Pořad je naplánován k zaznamenání.
- DOWNLOADED - Pořad byl alespoň jednou stažen.
- REMOVED - Pořad může být smazán ze systému.



Obrázek 3.11. Schéma přechodů stavů entity UserScheduledRecording

3.6.6 Aktualizace televizního programu

Kromě poskytování API a business logiky řeší REST API provider ještě aktualizaci televizního programu stanic a stavu entit. Aktualizace TV programu probíhá načítáním XML souboru ve formátu XML TV. Aktuálním poskytovatelem těchto dat je projekt Televize SH, která tato data poskytuje na svých webových stránkách.[1]

Aktualizace TV programu byla původně zamýšlena jako samostatná komponenta, ale z důvodu výrazného propojení s databází bylo vhodnější vyhnout se duplikaci entit a spojit obojí na jedno místo. Výhodou tohoto řešení pak je, že veškerá komunikace s SQL databází probíhá skrze komponentu REST API providera.

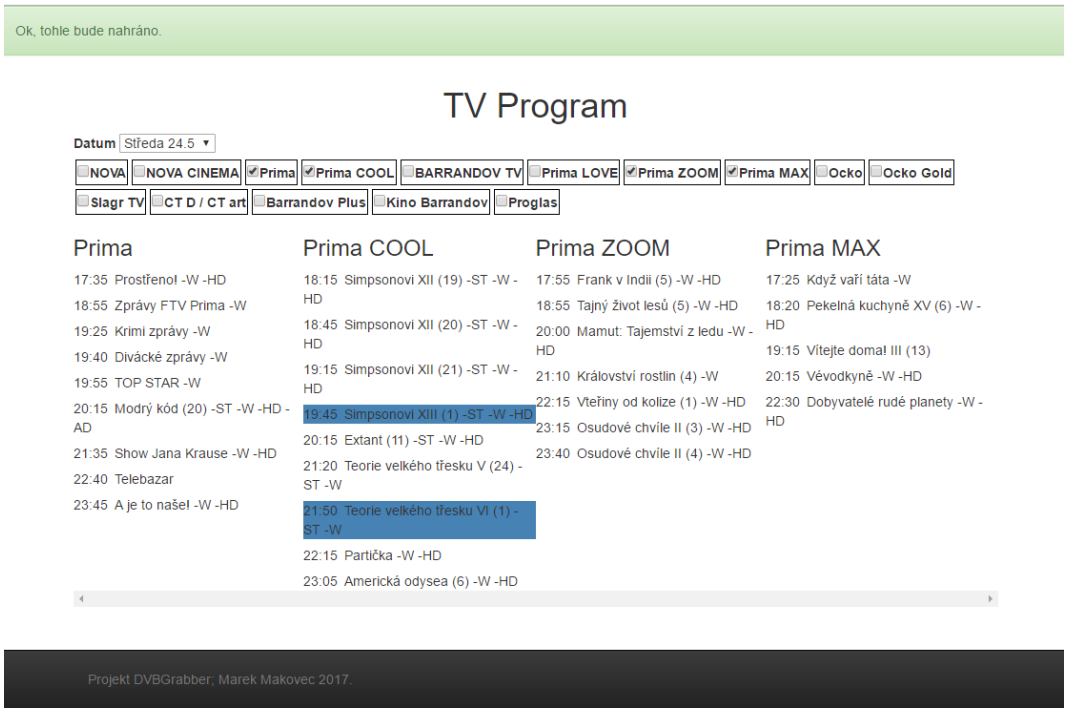
Aktualizace stavu entit je řešena pomocí consumera RabbitMQ fronty naprogramovaného jako Symfony Command. Protože jazyk PHP není vhodný pro dlouhodobě běžící procesy z důvodu náchylnosti k takzvaných memory leakům, je tento consumer spouštěn každou minutu znovu. Pokud tento consumer nedostane po dobu 40 vteřin žádnou zprávu, automaticky se vypne. Zároveň se automaticky vypne, pokud zpracoval více jak 5 zpráv.

3.7 Frontend

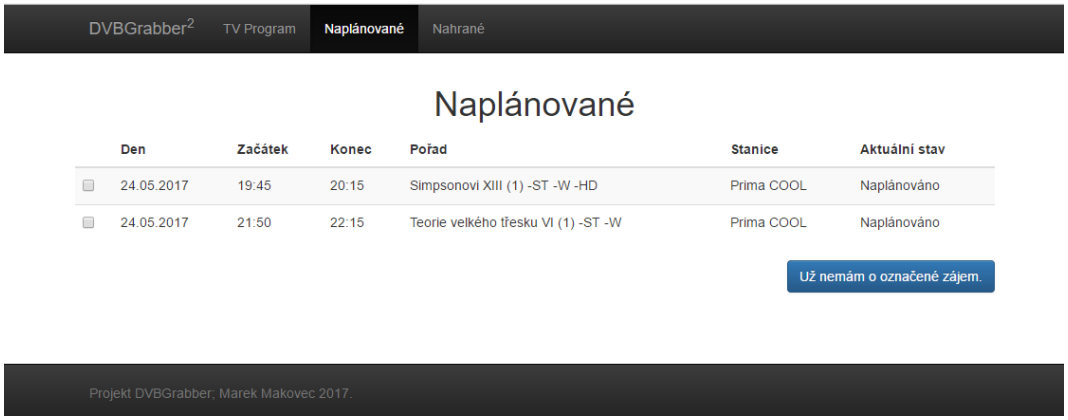
Frontendový klient je realizován na míru psanou HTML5 single-page aplikací bez použití Javascriptových frameworků. Vzhled a základní layout je řešen skrze CSS framework Bootstrap. Díky němu je aplikace částečně responzivní a je tedy možné nastavit nahrání například i z mobilního telefonu. Pro manipulaci s časem a časovými zónami je využita knihovna MomentJS.

Frontendový klient nabízí tři obrazovky. Přehled TV programu s možností nastavení nahrávání pořadu3.12, přehled naplánovaných pořadů3.13 a přehled nahraných pořadů3.14.

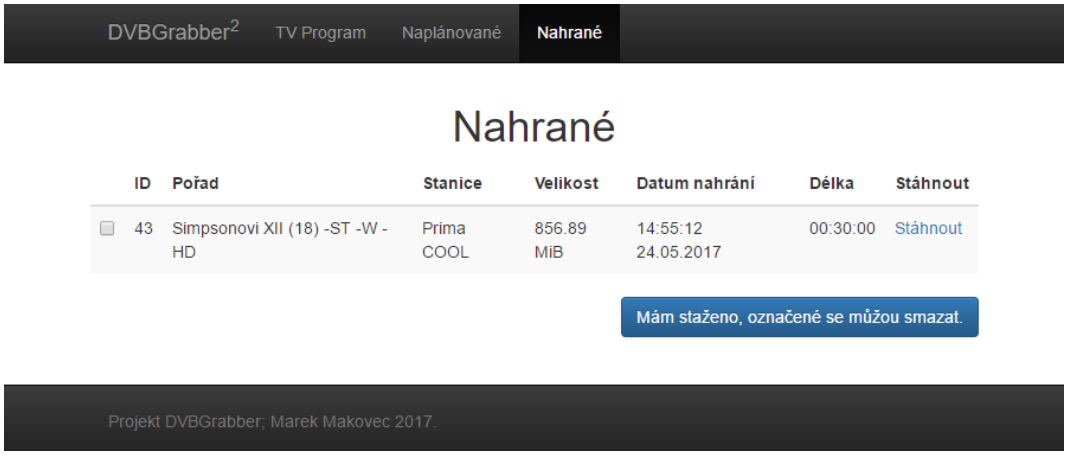
[1] `http://televize.sh.cvut.cz/xmltv.php`



Obrázek 3.12. Obrazovka s televizním programem a možností naplánování nahrávky



Obrázek 3.13. Obrazovka se seznamem pořadů naplánovaných k nahrání



Obrázek 3.14. Obrazovka se seznamem nahraných pořadů

Televizní program je po načtení držen v paměti prohlížeče Local Storage, aby se předešlo častému načítání tohoto poměrně velkého souboru.

3.8 Recorder

Recorder se skládá ze dvou skriptů v jazyce Python. Jazyk Python dostal přednost před PHP, protože má lepší předpoklady pro dlouhodobý běh procesu. Konkrétně nemá takové problémy s memory leaky, jako PHP.

První část je připojena na Message broker a sleduje příkazy o přidání / odebrání pořadu ze seznamu nahrávání. Pokud takovou zprávu obdrží, naplánuje spuštění druhé části skrze linuxového daemona atd. Atd je podobný nástroj, jako cron, ale narozdíl od něj slouží pouze pro jednorázové spouštění aplikací; neumí daný příkaz periodicky opakovat.

Druhá část je spouštěna zmíněným daemonem atd. Po spuštění vytvoří adresář pro uložení nahraného pořadu a spustí nahrávání pomocí příkazu mplayer. Jelikož mplayer nemá přepínač po vypnutí nahrávání streamu po N sekundách, je potřeba si zapamatovat zapnutý pid a po uplynutí daného časového úseku poslat proces mplayeru signál SIGTERM3.15. Po nahrání pořadu je do Message brokeru zaslána notifikace o dokončení nahrávání.

```
#!/usr/bin/env bash

SOURCE="$1"
OUTPUT="$2"
DURATION="$3"

mplayer -cache 4096 "${SOURCE}" -dumpstream -dumpfile "${OUTPUT}" &
RECORDER_PID=$!
sleep ${DURATION}s
kill ${RECORDER_PID}
```

Obrázek 3.15. Skript pro nahrání $DURATION sekund ze streamu $SOURCE do souboru $OUTPUT

3.9 Transcoder

Transcoder je skript napsaný v jazyce Python, který je připojený na Message broker a při obdržení zprávy spustí transcoding videa pomocí aplikace mencoder. Po dokončení nahrávání posílá zpátky zprávu do Message brokeru o úspěšném zpracování videa.

Pokud by došlo k pádu transcoderu, je jeho kontejner automaticky zapnut znovu díky direktivě “restart: on-error” v konfiguračním souboru docker-compose.yml.

3.10 Cleaner

Cleaner je skript napsaný v jazyce Python, který je rovněž připojený na Message broker a při obdržení příslušné zprávy smaže celý adresář z komponenty Storage, ve kterém se nacházely údaje o pořadu. V tomto adresáři se kromě surového záznamu z DVB-T a hotového videa po konverzi nachází také různé záznamy o nahrávání a procesu konverze a další debugovací informace.

3.11 Storage

Storage není komponentou v pravém slova smyslu. Jde pouze o společný adresář, který je skrz docker-compose namapovaný do kontejnerů REST API Provider (backend), Transcoder, Recorder a Cleaner a ve kterém se nachází data nahraného pořadu.

Adresářová struktura je triviální. V první úrovni se nachází adresář, jehož názvem je ID pořadu, jehož data jsou v něm uložena3.16. Uvnitř jsou pak dva hlavní soubory: input.raw a output.raw, které obsahují data nahraného pořadu. Input.raw jsou surová data z DVB-T. Output.raw jsou pak data přichystaná pro uživatele ke stažení.

Kromě těchto dvou souborů se pak v adresáři nacházejí ještě další soubory vzniklé během procesu vzniku záznamu, například soubory s metadaty, které vytváří mencoder při konverzi videa apod.

```
-rw-r--r--  1 root  root    92 May 24 14:25 cmd.txt
-rw-r--r--  1 root  root  857M May 24 14:55 input.raw
-rw-r--r--  1 root  root  857M May 24 14:55 output.raw
-rw-r--r--  1 root  root   149 May 24 14:55 rabbit.queue.log
-rw-r--r--  1 root  root   160 May 24 14:55 record.stderr.log
-rw-r--r--  1 root  root   53K May 24 14:55 record.stdout.log
-rw-r--r--  1 root  root    62 May 24 14:55 transcode.sh
-rw-r--r--  1 root  root    59 May 24 14:55 transcode.stderr.log
-rw-r--r--  1 root  root     0 May 24 14:55 transcode.stdout.log
```

Obrázek 3.16. Obsah adresáře po nahrání pořadu

3.12 Message broker

Jako Message broker byl v projektu použit RabbitMQ. Skrz něj jsou jednotlivé části aplikace propojené a zároveň řeší i potřebu škálování počtu transcoderů.

Každý klient připojený k RabbitMQ může vystupovat v jedné ze dvou rolí:

- Producer: Takový klient vytváří zprávy, které posílá do front v Message brokeru.
- Consumer: Tento klient zprávy přijímá a nějakým způsobem zpracovává.

Některé části aplikace mohou být připojeny v obou rolích. Například z jedné fronty (kde jsou v roli consumer) dostanou zprávu s úkolem a po zpracování daného úkolu do jiné fronty (kde jsou v roli producera) pošlou zprávu o jeho dokončení.

V Message brokeru jsou definovány následující fronty zpráv:

3.12.1 Fronta schedule-recording

Producerem této fronty je REST API provider. Consumerem je Recorder. Tato fronta slouží k naplánování pořadu pro nahrání. Zprávy3.17 se skládají z textových řetězců ve formátu JSON s následujícími atributy: start, stop, id, source, action. Atributy start a stop obsahují timestamp od kdy a do kdy se má nahrávat. Source obsahuje HTTP URL streamu, který má být nahráván. ID obsahuje ID entity Show, která má být nahrávána. Action má jednu z hodnot queue_insert a queue_remove. Pokud action nabývá hodnoty queue_insert, pak je do daemona atd naplánováno nahrání daného pořadu. Pokud má action hodnotu queue_remove, pak consumer projde všechny naplánované úlohy v daemonu atd, najde takovou, která má Show ID shodné s ID v této zprávě a takovou úlohu odebere.

```
{
    id: 42,
    start:  "1495639195"
    stop:   "1495641195",
    source: "http://192.168.1.203:90/bysid/774",
    action: "queue_insert"
}
```

Obrázek 3.17. Ukázka zprávy z fronty

3.12.2 Fronta start-transcoding

Producerem je REST API provider, consumerem je transcoder. Tato fronta je použita v okamžiku, kdy REST API provider dostane zprávu z fronty recording_status, že pořad byl nahrán (dostal se do stavu RECORDED) a dává komponentě transcoder pokyn pro započnutí procesu transcodingu. Zprávou je text ve formátu JSON obsahující klíče id a action. Hodnotou id je ID pořadu, který má být zkonvertován a atribut action má hodnotu transcode.

3.12.3 Fronta remove-recorded-recording

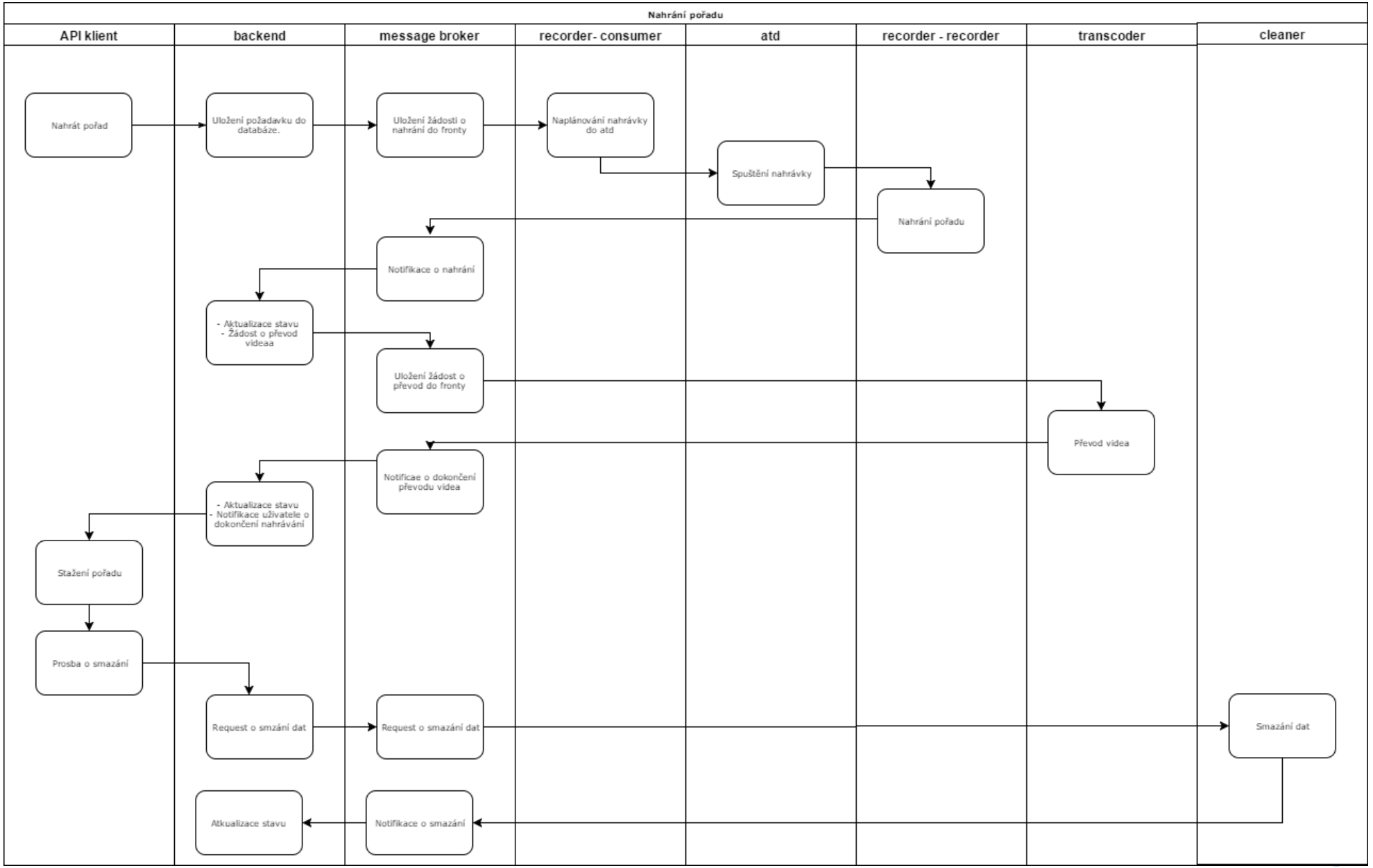
Producerem je REST API provider, consumerem je cleaner. Touto frontou se posílá informace o smazání fyzických dat požadovaného pořadu. Zpráva obsahuje pouze id pořadu, které má být vymazáno. Po smazání cleaner odpovídá zprávou do recording_status, kde spolu s klíčem id (ID pořadu) posílá ještě klíč status s hodnotou REMOVED.

3.12.4 Fronta recording-status

Producerem jsou transcoder, recorder a cleaner, kteří do této fronty zasílají informace o změnách stavu nahrávaného pořadu. Consumerem je v tomto případě REST API provider.

Zprávou je JSON obsahující klíče id a status. Pole status obsahuje jednu z hodnot, které může nabývat entita ScheduledRecording.

3.13 Workflow nahrávání pořadu



Obrázek 3.18. Workflow nahrání pořadu

Kapitola 4
Implementace a testování

Tato kapitola se věnuje testování vzniklého systému a problémům, které bylo potřeba řešit během implementace samotné. Zároveň zmiňuje i jejich řešení či workaroundy.

4.1 Uživatelské testování

4.1.1 Testovací prostředí

Ve všech případech se testovalo na počítači daného participanta a ve webovém prohlížeči Google Chrome, na který jsou všichni participanti zvyklí.

4.1.2 Testovací scénář

Testovací scénář byl pro všechny participanty stejný. Jeho cílem bylo projít všechny uživatelsky dostupné funkce aplikace v přirozeném pořadí.

- Vstupte do aplikace DVBGrabber a přihlašte se pomocí následujících příhlašovacích údajů: Uživatelské jméno: *admin@dvbgrabber.local* Heslo: *secret*.
- V pátek 26. května dávají na stanici Nova seriál *Hvězdná Brána*. Naplánujte si jeho nahrání.
- Přesvědčte se o tom, že pořad bude nahrán.
- 22. května Jste si naplánoval nahrání napořadu *Ano, šéfe!*. Touto dobou by už měl být nahraný a k dispozici. Stáhněte si jej.
- Rozmyslel jste si nahrávání pořadu *Hvězdná Brána*. Zrušte jeho nahrání.
- Pořad *Ano, šéfe!* se vám už dostahoval. Dejte systému pokyn, že jej už nemusí skladovat.

4.1.3 Participant 1

Profil participanta: Muž, 64 let, vysoce IT gramotný (povolání: programátor a manažer IT firmy).

Participant prochází scénářem bez zadrhnutí.

4.1.4 Participant 2

Profil participant: Žena, 26 let, průměrně IT gramotná.

- Participant objevuje chybu - pokud odebere pořad ze seznamu naplánovaných, tento je v televizním programu stále vyznačen jako naplánovaný.
- Participant podotýká, že by u televizního programu očekával stejnou logiku, jako je v seznamech naplánovaných a nahraných - tedy tlačítko na potvrzení akce.

4.1.5 Participant 3

Profil participanta: Muž, 23 let, průměrně IT gramotný

- Participant chtěl pořad k nahrání najít přes funkci vyhledávání.
- Participant měl problém kliknout v navigaci - text není odkazem.
- Participant hledal tlačítko na smazání - možnost hromadného smazání našel až po chvíli.

4.1.6 Nálezy

Následuje seznam nálezů podle priority (řazeno sestupně).

- Chyba: Odkazy v navigaci se špatně klikají - samotný text nefunguje pro přepnutí.
- Chyba: Po odebrání naplánovaného pořadu je v televizním programu pořad stále zvýrazněný, jako by byl naplánovaný.
- Vylepšení: Checkbox na vybrání řádku s pořadem, který se může smazat, by měl být větší nebo by mělo být možné klikat na celý řádek.
- Vylepšení: V seznamech pořadů ke shlédnutí a ke stažení by bylo vhodné přidat tlačítko na smazání jednoho pořadu.
- Nová vlastnost: V systému chybí možnost vyhledávat v televizním programu.

4.1.7 Výsledek uživatelského testování

Po uživatelském testování byly opraveny obě nalezené chyby. Možná vylepšení a nové vlastnosti jsou ponechány jako možnosti rozšíření do další verze aplikace.

4.2 Zátěžové testy

Zátěžových testů byly zvoleny dva různé typy. První z nich testuje rychlost zápisu na disk. Ověřuje tak, zda na daném počítači je možné vůbec stihnout zapisovat data televizního vysílání v reálném čase. K tomuto testu byla použita v předchozích kapitolách zmíněná utilita fio, která sekvenčně zapisovala na disk 17 různých soubor, každý o velikosti 1 GB. Je-li průměrný bitový tok jednoho vysílání přibližně 3000 kbit/s, při souběžném zápisu 17 potřebujeme propusnost alespoň 6375 kB/s.

Test probíhal spuštěním utility fio v prostředí docker kontejneru recorder v takovém adresáři, že vytvářená data byla zapisována do datové oblasti mapované do storage - jde o stejnou oblast, kam se v případě nahrávání ukládá televizní vysílání.

Druhý typ testů spočíval v rychlosti, jakou dokáže daný počítač transkódovat video o následujících parametrech4.1. Jde o konkrétní záznam vysílání pořízený systémem DVBGrabber.

Název atributu	Hodnota
Velikost souboru	601 MB
Délka	24 minut 59 vteřin
Kodek	mpeg2video / mp2

Tabulka 4.1. Vzorky pořadů vysílaných dostupnými stanicemi

Test porovnává rychlost konverze a velikost výsledného souboru při konverzi do dvou různých kodeků. První z nich je konverze do kodeku mpeg4, druhá pak do kodeku h264.

Systém byl testován na dvou sestavách. Sestava zvaná Server má následující parametry4.2:

Parametr	Hodnota
CPU	Intel Xeon E5606 @ 2 GHz virtuální server má přiřazeno jen jedno jádro.
RAM	8 GB
HDD	100 GB v RAID 5 nad šesti 7200rpm HDD
Operační systém	Ubuntu Server 16.04

Tabulka 4.2. Parametry sestavy Server

Druhá sestava zvaná Desktop má parametry následující4.3:

Parametr	Hodnota
CPU	Intel Core i7 6700K @ 4 GHz
RAM	32 GB
HDD	200 GB SSD HDD (Samsung 850 Evo)
Operační systém	Windows 10 Educational

Tabulka 4.3. Parametry sestavy Desktop

4.3 Rychlost zápisu na disk

V případě provedení testu na sestavě Server změřilo fio následující parametry4.4:

```
Run status group 0 (all jobs):
  WRITE: io=17408MB, aggrb=192688KB/s, minb=11334KB/s,
         maxb=11584KB/s, mint=90514msec, maxt=92511msec
```

Tabulka 4.4. Výsledek měření utility fio na sestavě Server

Průměrná rychlost zápisu na disk (aggrb) je tedy přibližně 188 MB/s. To je pro potřeby ukládání více než dostačující.

Na testovacím prostředí Desktop byly očekávány výrazně horší výsledky. Zejména kvůli tomu, že zatímco na OS Linux funguje Docker téměř nativně, v případě Windows je provozován ve virtuálním stroji pomocí HyperV a na sdílené diskové úložiště zapisuje pravděpodobně skrz protokol SMB. Výsledek testu tak byl nadmíru překvapivý4.5:

```
Run status group 0 (all jobs):
  WRITE: io=17408MB, aggrb=196215KB/s, minb=11542KB/s,
         maxb=12480KB/s, mint=84016msec, maxt=90848msec
```

Tabulka 4.5. Výsledek měření utility fio na sestavě Desktop

Průměrná rychlost zápisu na disk (aggrb) je v tomto případě ještě vyšší než na sestavě Server - přibližně 191 MB/s.

4.4 Rychlost konverze videa

Parametry vstupního videa popisuje tabulka 4.1 v předchozí sekci. Výsledky tohoto testu popisuje následující tabulka 4.6:

Kodek	Sestava	Odhadovaná délka konverze	odhadovaná výsledná velikost souboru
mpeg4	Server	~300 minut	331 MB
h264	Server	~260 minut	508 MB
mpeg4	Desktop	~140 minut	334 MB
h264	Desktop	~55 minut	512 MB

Tabulka 4.6. Výsledky konverze videa

Údaje vychází z odhadů, které mencoder během konverze zobrazuje. Kvůli extrémně dlouhým dobám neproběhly konverze až do konce. I tak je ale z tabulky patrné, že kodek h264 nebyl vhodnou volbou. Konverze do něj je sice o něco rychlejší, ale výsledný soubor je téměř stejně velký, jako vstupní.

Pokud by měl Server silnější procesor, dalo by se uvažovat o využití konverze do mpeg4, který evidentně zmenšuje velikost výsledného souboru přibližně na polovinu. Bohužel, doba konverze je přibližně desetinásobná oproti délce trvání videa. Čekat celý den na konverzi dvou a půl hodinového snímku zní absurdně.

4.5 Vybrané problémy při implementaci a jejich řešení

4.5.1 Synchronizace televizního programu

Prvním z nepříjemných problémů během implementace byla synchronizace televizního programu s databází. Formát XML TV, ze kterého jsou strojově čitelná data do systému importována, sice obsahuje jednoznačný identifikátor televizní stanice, ale už neobsahuje žádný identifikátor pořadu.

Prvotní implementace se pokoušela hledat kolize ve vysílacích časech a pokud měl být nějaký pořad ve vysílacím slotu, kde kolidoval s jinými, pokoušela se rozhodnout, který z pořadů odebrat popřípadě neimportovat. Toto řešení bylo určeno hlavně pro situace, kdy se například místo zrušeného tenisového utkání objeví v televizním programu dva jiné, kratší pořady.

Bohužel byla tato myšlenka opravdu naivní, protože experimentálním testováním bylo zjištěno, že kolize v televizním programu nejsou výjimečným stavem, ale realitou všedního dne. Je zcela běžné, že v případě dvou po sobě jdoucích pořadů začíná pozdější pořad půl minuty předtím, než předchozí skončí. Děje se tak hlavně v období nočního vysílání.

Výsledný algoritmus synchronizace je tedy mnohem jednodušší a připouští jakékoliv kolize. Televizní pořad se nyní importuje téměř pokaždé s jedinou výjimkou - tedy pokud ve stejném časovém období (+- 5 minut na začátku i na konci) už není v databázi jiný záznam, který je vysílán na stejné televizní stanici a pod stejným názvem. Konkrétní implementace v této práci naznačena výše 3.8.

Nevýhoda tohoto řešení je patrná na první pohled; uživatel si naplánuje nahrát televizní pořad, místo něhož se na poslední chvíli vysílá například živý vstup z nějaké významné událost. Ve výsledku tak bude pravděpodobně zklamán, protože záznam, který dostane, vůbec nechtěl.

Možným řešením tohoto problému by bylo ještě před začátkem nahrávání číst EPG[1] a kontrolovat, zda právě vysílaný pořad je skutečně to, co má být nahráno. Budiž toto jedno z možných rozšíření funkcionality do budoucna.

4.5.2 Přenášení času mezi komponentami systému

Přenášení časové informace mezi jednotlivými komponentami systému bylo problémem hlavně kvůli roztříštěnosti jednotlivých kontejnerů. Různé kontejnery staví na různých linuxových distribucích a navíc by bylo potřeba zásahu uživatele, aby pro každý kontejner nastavil správnou časovou zónu.

Zvolené řešení tedy vychází z toho, že všechny kontejnery mají časovou zónu UTC a čas je mezi nimi přenášen v podobě unixového timestampu, což není nic jiného, než počet sekund který uplynul od 1.1.1970 UTC. Na frontendu je pak čas pro uživatele zobrazen v časové zóně jeho prohlížeče.

4.5.3 Rychlost konverze videa

Jak vyplynulo z kapitoly Zátěžové testy, rychlost konverze videa na sestavě zvané Server, na které je i plánováno další provozování systému DVBGrabber, trvá v případě krátkého, zhruba třicet minut dlouhého záznamu, řádově několik hodin a úspora datového prostoru je přibližně poloviční. To je značné zklamání. Z toho důvodu byl do systému DVBGrabber dodělán třetí, ve výchozím stavu zapnutý filtr pro komponentu transcoder, který pouze překopíruje vstupní soubor na výstup. K tomuto kroku bylo přistoupeno z důvodu, že čekat na konverzi videa je značně nepraktické. Například iPrima.cz uvolňuje pořady ve svém archivu ke zhlédnutí s přibližně hodinovým zpožděním. Nač tedy čekat mnoho hodin na konverzi pořadu, která navíc příliš mnoho místa neušetří, když už je mezitím k dispozici v archivu televize. Referenční instance systému[2] tedy aktuálně neprovádí transcoding, ke stažení dává surová data z vysílání.

[1] Electronic Programme Guide.[22]
[2] `http://dvbgrabber.spadnul.net`

Kapitola 5
Zhodnocení

Implementace systému DVBGrabber se zdá být poměrně úspěšná. Systém, za předpokladu že je k dispozici kvalitní televizní signál, umožňuje pohodlně sledovat televizní vysílání z různých zařízení připojených do domácí sítě. Kromě toho umožňuje procházet televizním programem na nejbližší dny a naplánovat nahrání požadovaných pořadů.

Velkým zklamáním je pak neschopnost rozumné konverze videa do nějakého úspornějšího formátu. Autor je přesvědčen, že z časového hlediska se nevyplatí čekat na konverzi. Je nepravděpodobné, že po shlédnutí záznamu pořadu bude takový pořad potřeba uchovávat dlouhodobě na disku a pokud ano, je možné provést konverzi ručně a dodatečně.

Aplikaci v aktualním stavu není možné považovat za vhodnou k nasazení do menší komunity. Na to by bylo potřeba minimálně rozšířit správu uživatelů a dodat nějaké administrátorské prvky a monitoring. Ale v domácím použití v intranetové síti je použitelná a ozkoušená.

Kapitola 6
Možnosti rozšíření

Během implementace systému se objevilo několik zajímavých požadavků a potřeb, které nebyly v původních funkčních požadavcích na aplikaci a nejsou tedy prozatím ani součástí systému. Bylo by ale zajímavé o tyto vlastnosti systém v budoucnu rozšířit.

6.1 Statistiky

Pro potřeby jednoho uživatele jde pravděpodobně o zbytečnou vlastnost. V případě menší komunity by však šlo o cenný zdroj dat. Z aktuálních dat v databázi lze vyčíst minimálné následující hodnoty:

- Zabraný diskový prostor.
- Počet nahraných pořadů.
- Celková délka nahraného záznamu.
- Počet pořadů aktuálně dostupných ke stažení.
- Počet pořadů čekajících na konverzi.

Dalším zajímavým ukazatelem by bylo zaplnění datové oblasti na disku. Tento údaj se však získává komplikovaněji, než dotazem do databáze. Nejsnazším způsobem, jak jej realizovat, by bylo zprovoznit další Docker kontejner s přístupem na komponentu storage a periodicky se dotazovat na její parametry (velikost, aktuální zaplnění) a tyto informace následně posílat do Message brokera k dalšímu zpracování - pravděpodobně uchování do databáze..

Neméně vhodnou veličinou by pak mohlo být aktuální vytížení transcoderů a případné spouštění nových v okamžiku velkého množství pořadů čekajících na konverzi. Tuto problematiku bych však v této práci pro její komplikovanost již nerozváděl.

6.2 Vyhledávání nad programem

Z rozhovorou s participanty po uživatelském testování vyplynul zájem o možnost nad televizním programem vyhledávat - ideálně pomocí fuzzy search. Na tomto místě se autor přiznává, že implementace této funkcionality jej nikdy nenapadla, protože celá obrazovka televizního programu byla koncipována v podobě "papírového katalogu toho, co je dostupné" a s jako takovou s ní vždy i pracoval. Rozhodně ale nic nebrání zprovoznění této vlastnosti dobudoucna.

6.3 Pipeline pro transpilaci javascriptových závislostí

Velkou bolístkou aplikace je načítání HTML5 klienta. Při načtení je totiž provedeno přes 30 HTTP požadavků, valná většina z nich pak stahuje javascriptové soubory obsahující jednotlivé části systému. Tento problém by se dal řešit pomocí automatizované pipeline, která by transpilovala a ideálně i minifikovala javascriptové soubory, ze kterých se klient skládá, do jednoho.

6.4 Čtení dat z EPG

Jednou z výhod streamovací utility MuMuDVB je poskytování EPG dat. Komponentu recorder by tedy bylo možné rozšířit o utilitu, která by oddalovala spuštění samotného nahrávání do okamžiku, dokud v EPG nepřijde informace o tom, že je požadovaný pořad právě přehráván.

Bohužel, najít utilitu, která by uměla pouze podávat informace o aktuálních datech v EPG, je komplikované. Většinou je taková funkcionalita součástí nějakého většího celku. Pravděpodobně by bylo potřeba vytvořit vlastní implementaci.

Kapitola 7
Závěr

V rámci této práce byl úspěšně prozkoumán, ověřen a zdokumentován postup, jak v domácích podmínkách zajistit přehrávání televizního signálu v reálném čase v rámci lokální sítě.

Kromě toho tato práce nastiňuje způsob, jak obejít problémy s přenosem multicastingu skrz síťové prvky, které multicasting nepodporuje.

Dále byl implementován systém, který dokáže uživateli předložit aktuální televizní program, ve kterém si uživatel navolí takové pořady, které chce zaznamenat. Tyto pořady se následně skutečně nahrají na disk a uživatel si je později může stáhnout a přehrát.

Tento systém je nezávislý na operačním systému hostitele, umožňuje horizontální škálování výpočetně náročných komponent a poskytuje programové API pro snadnou tvorbu vhodného klienta poskytujícího GUI.

V neposlední řadě byl naimplementován jednoduchý HTML5 single-page JavaScriptový klient demonstrující funkčnost celého systému.

Co se nepodařilo je vyřešení otázky konverze nahraných dat do datově úspornějšího kodeku. Tento problém se ale netýká tolik zvolené architektury a implementace, jako spíše konkrétního hardwaru, na kterém byl systém vytvořen a na který je aktuálně nasazen.

Literatura

[1] *Technical terms*.
http://www.lantechcom.tw/global/eng/learning-center-technical-terms.html.

[2] Televize SH team. *DIGITAL VIDEO BROADCASTING*.
http://televize.sh.cvut.cz/navod.html.

[3] Televize SH team. *DIGITAL VIDEO BROADCASTING*.
http://televize.sh.cvut.cz/index.html.

[4] The VideoLAN non-profit organization. *DVBlast*.
http://www.videolan.org/projects/dvblast.html.

[5] Brice Dubost. *MuMuDVB*.
http://mumudvb.net/.

[6] NIX.cz. *NIX.CZ - Neutral Internet Exchange*.
https://www.nix.cz/cs.

[7] *DVBgrab – wiki.siliconhill.cz*.
http://wiki.siliconhill.cz/DVBgrab.

[8] *Servery dvbgrab – wiki.siliconhill.cz*.
http://wiki.siliconhill.cz/Servery_dvbgrab.

[9] goNET s.r.o. *Nová dimenze sledování televize s nahráváním — Lepší.TV*.
https://www.xn--lep-tma39c.tv/vyhody/.

[10] *Introduction*.
http://www.udpxy.com/index-en.html.

[11] The VideoLAN non-profit organization. *VLC Media Player*.
http://www.videolan.org/index.html.

[12] The FFmpeg developers. *Libavcodec*.
https://ffmpeg.org/libavcodec.html.

[13] Libav community. *Libav*.
https://libav.org/.

[14] The MPlayer Project. *MPlayer - The Movie Player*.
http://www.mplayerhq.hu/design7/news.html.

[15] Jens Axboe. *fio*.
http://freecode.com/projects/fio.

[16] *MEncoder - ArchWiki*.
https://wiki.archlinux.org/index.php/MEncoder#Two-pass_x264_.28very_high-quality.29.

[17] *MEncoder - ArchWiki*.
https://wiki.archlinux.org/index.php/MEncoder#Three-pass_lavc_.28very_high-quality_mpeg4.29.

[18] Winfried Koehler. *w_scan*.
http://wirbel.htpc-forum.de/w_scan/index2.html.

[19] Jens Axboe. *getstream*.
`http://silicon-verl.de/home/flo/projects/streaming/`.

[20] SensioLabs France Fabien Potencier. *w_scan*.
`https://symfony.com/`.

[21] Doctrine team. *w_scan*.
`http://www.doctrine-project.org/`.

[22] "European Telecommunications Standards Institute 2003. European Broadcasting Union 2003.". "Electronic Programme Guide (EPG); Protocol for a TV Guide using electronic data transmission". "2003", DOI "http://dx.doi.org/10.1016/j.dss.2006.08.006".

Příloha A
Zadání práce

- Analyzujte systémy pro záznam a přehrávání (streamimg) televizního vysílání pro osobní použití jako je např. tv.sms.cz, O2 TV, nebo Horizon Go.
- Navrhněte a implementujte otevřený systém pro záznam a přehrávání televizního vysílání pro osobní použití, který umožní zpracování TV-streamů v rozlišení 576p a vyšším.
- Analyzujte a vyberte vhodné technologie pro realizaci, jak uživatelského rozhraní aplikace (frontend), tak pro aplikační vrstvu (backend) systému.
- Systém bude složený z mikroslužeb komunikujících definovaným rozhraním pomocí zprostředkovatele zpráv (message broker). Množství současně zaznamenávaných pořadů (různé TV kanály) bude možné horizontálně škálovat přidání transkodérů.
- Testování aplikace proveďte na 2 různých konfiguracích PC a navrhněte vhodný postup testování tak, aby ověřil rychlost kódování vybraných záznamů.
- Uživatelské rozhraní a funkčnost systému otestujte s alespoň 3-mi uživateli.