

Sem vložte zadání Vaší práce.



ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE  
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ  
KATEDRA SOFTWAREVÉHO INŽENÝRSTVÍ



Diplomová práce

## **Children Usability Lab - management video streamů**

***Bc. Patrik Faistaver***

Vedoucí práce: Ing. Jiří Chludil

30. dubna 2018



---

## Poděkování

Rád bych poděkoval vedoucímu mé diplomové práce Ing. Jiřímu Chludilovi i oponentovi Ing. Jiřímu Melnikovovi za užitečné připomínky a mnoho cenných rad. Dále chci poděkovat své rodině a blízkým přátelům za podporu a motivaci při tvorbě této práce i během celého studia.



---

## Prohlášení

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

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

V Praze dne 30. dubna 2018

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2018 Patrik Faistaver. Všechna práva vyhrazena.

*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.*

### **Odkaz na tuto práci**

Faistaver, Patrik. *Children Usability Lab - management video streamů*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.



---

## Abstrakt

Tato diplomová práce navazuje na bakalářskou práci Karolíny Solanské s názvem Children Usability Lab - aplikace pro správu laboratoře. Její bakalářská práce je zde rozšířena o návrh a implementaci subsystému pro management veškerých záznamů nahrávaných v laboratoři Children Usability Lab. Součástí této práce je také analýza současného systému i existujících řešení pro požadovanou správu záznamů, dále pak návrh, realizace a její následné testování. Implementované rozšíření stávajícího systému plně pokrývá dohodnuté zadání i specifikované požadavky.

Závěrem této diplomové práce je podrobení subsystému integračním i akceptačním testům. Subsystém se podařilo úspěšně nasadit do počítačového prostředí laboratoře Children Usability Lab, kde usnadňuje a zefektivňuje tamní usability testování.

**Klíčová slova** testování použitelnosti, webová aplikace, laborator uživatelského testování, management záznamů, správa videa

---

## Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

**Keywords** Nahradte seznamem klíčových slov v angličtině oddělených čárkou.

---

# Obsah

<b>Úvod</b>	<b>1</b>
<b>1 Cíl práce</b>	<b>3</b>
<b>2 Analýza</b>	<b>5</b>
2.1 Definice pojmů . . . . .	5
2.2 Rozbor zadání . . . . .	5
2.3 Předchozí práce a aktuální stav . . . . .	7
2.4 Specifikace požadavků . . . . .	8
2.5 Existující komplexní řešení . . . . .	12
2.6 Výběr technologií . . . . .	16
2.7 Seznámení s Libyuri . . . . .	20
<b>3 Návrh</b>	<b>29</b>
3.1 Architektura systému . . . . .	29
3.2 Frontend . . . . .	30
3.3 Backend . . . . .	33
<b>4 Implementace</b>	<b>41</b>
4.1 Gitlab a CI . . . . .	41
4.2 Vybrané zajímavé funkce . . . . .	41
4.3 Instalační a uživatelská příručka . . . . .	41
<b>5 Testování</b>	<b>43</b>
5.1 Integrovaní testování . . . . .	43
5.2 Akceptační testování . . . . .	43
<b>Závěr</b>	<b>45</b>
<b>Literatura</b>	<b>47</b>

<b>A Seznam použitých zkratk</b>	<b>51</b>
<b>B Obsah přiloženého CD</b>	<b>53</b>

---

## Seznam obrázků

2.1	Libyuri – příklad orientovaného grafu . . . . .	21
2.2	Libyuri – příklad propojených podgrafů pomocí builderu . . . . .	23
2.3	Libyuri – graf vytvořený pomocí XMLBuilder . . . . .	24
3.1	Diagram nasazení . . . . .	30
3.2	Původní databázový model . . . . .	34
3.3	Původní databázový model . . . . .	38
3.4	Libyuri - konfigurace pro záznam v CHUL . . . . .	39
3.5	Libyuri - konfigurace zobrazení v SAGE a ukládání . . . . .	40



---

## Seznam tabulek

2.1	Porovnání technologií pro backend . . . . .	18
2.2	Porovnání technologií pro frontend . . . . .	18





---

# Úvod

Testování použitelnosti je proces při kterém se aplikace testuje přímo za pomoci vzorku koncových uživatelů a díky kterému se autoři aplikace dozvědí, jak jsou různé prvky uživatelského rozhraní srozumitelné a jak dobře jsou uživatelé schopni se v aplikaci orientovat. V současném světě technologií tento proces vyžaduje informační systém jako řešení pro usnadnění testování, plánování i komunikace mezi jednotlivými subjekty v laboratoři testování použitelnosti. Vývoj takového systému byl zahájen bakalářskou prací Karolíny Solanské s názvem Children Usability Lab - aplikace pro správu laboratoře (odkazem [1]). Její práce vyvinula systém, který slouží jako webová aplikace zajišťující plánování a organizaci testů použitelnosti. Zmíněná bakalářská práce měla neobvykle větší počet stran, avšak podařilo se implementovat zcela funkční systém, ale i přesto je v něm spousta možností pro vyvinutí nových funkcí. Na tomto základě se dá říci, že se jedná o celkem rozsáhlý systém. Jeden ze směrů, kterým by se systém mohl začít rozšiřovat byla veškerá práce s videem což bylo ostatně zmíněno v uvedené práci v kapitole o dalším vývoji.

Laboratoř použitelnosti na Fakultě informačních technologií ČVUT v Praze je vybavena spoustou zařízení, která jsou schopná zaznamenávat různorodá data. Jedná se nejen o všechna výpočetní zařízení, na kterých probíhají samotné testy, jako hlavní stolní počítač, tablet apod., ale také spousta kamer, mikrofónů či dokonce snímač pohybu očí po obrazovce. Není pochyb o tom, že data z těchto nahrávání schopných zařízení mohou značně přispět ke kvalitnější vyhodnocování testů použitelnosti. V aktuálním stavu je práce se záznamy velice obtížná a neúčinná, jak je uvedeno v kapitole o současném stavu (2.3). Z těchto důvodů je nutné rozšířit stávající systém o funkcionalitu, která práci s těmito záznamy značně ulehčí a zefektivní. Jedná se o to, aby uživatelé systému mohli jednoduše pouze přes webové rozhraní spustit a zastavit nahrávání z vybraných zařízení, záznamy roztrždit, uložit v požadovaném formátu a kompresi, upravovat zadaným výčtem způsobů, posílat přes síť do systému SAGE a podobně.



## Cíl práce

Cílem této diplomové práce je nejprve seznámení se s bakalářskou prací Karolíny Solanské s názvem Children Usability Lab - aplikace pro správu laboratoře ([1]). V uvedené práci je třeba analyzovat veškeré specifické požadavky zaměřené na správu videa. Tyto požadavky je nutné znovu probrat s koordinátory laboratoře Children Usability Lab, aktualizovat je a zakomponovat do specifických požadavků této práce. Vedle převzatých požadavků se nadefinuje i spousta dalších, nových funkčních i nefunkčních požadavků, které byly nashromážděny během posledních let běhu testů v laboratoři. Další částí analýzy je pak ponoření se do světa technologií, které usnadňují práci s videem a z existujících je třeba vybrat pokud možno ty nejvhodnější, které by se daly použít v praktické části této práce.

Výstup analytické části si poté převezme návrhová a implementační část, ve kterých se pomocí nástrojů softwarového inženýrství promění nabyté poznatky v reálný produkt, tedy modul pro správu videa, který půjde se stávajícím systémem provázet.

Se vzniklým testovatelným řešením se pak postoupí do testovací části, kde se otestují funkčnosti zadané specifikací, provede se nasazení, případné opravy nalezených chyb a dále akceptační testování. Součástí práce bude také dokumentace včetně instalační příručky pomocí které bude možné systém nasadit v různých laboratořích pro testování použitelnosti.



---

# Analýza

## 2.1 Definice pojmů

### Frontend

Jedná se o část aplikace, která je viditelná běžným uživatelům.

### Backend

Jedná se o část aplikace, uživatelům „schovaná“ za frontendem a slouží především ke zpracování dat.

### CHUL

Children Usability Lab - laboratoř testování použitelnosti na Fakultě informačních technologií ČVUT v Praze.

### CDN

CDN neboli Content Delivery Network je systém serverů rozmístěných po internetu, které spolupracují pro zajištění rychlého doručení dat klientovi.

### Hosting

Pronájem úložného prostoru uživateli na proprietárních serverech.

### Framework

Jedná se o softwarovou strukturu fungující jako knihovna funkcí a podpůrných nástrojů, která usnadňuje vývoj jiných softwarových projektů.

## 2.2 Rozbor zadání

1. Analyzujte a aktualizujte funkční i nefunkční požadavky zaměřené na správu videa, uvedené v předchozí práci.

Jak již bylo zmíněno, tak tato práce navazuje na bakalářskou práci Karolíny Solanské s názvem Children Usability Lab - aplikace pro správu laboratoře (odkazem [1]). Aby navázání na předchozí práci mělo ten správný směr, je nutné analyzovat specifické požadavky z předchozí práce, ty projednat s aktuálním vedoucím práce, aktualizovat jejich znění a zařadit do specifických požadavků této práce. Náplň tohoto kroku zadání je detailněji probrána v sekci zabývající se požadavky předchozí práce (vizte 2.3.1) a také v sekci s definicí specifických požadavků této práce (2.4.1).

### 2. Analyzujte nástroje pro zpracování videa použitelné v infrastruktuře SAGE.

Většinová část této práce se týká zpracování videa, proto je nezbytné nutné se s oblastí zpracování videa důkladně seznámit. Nejprve je nutné si stanovit, co se bude po pomocných technologiích chtít, tedy seznam obecných funkcí a vlastností a jejich priorit. Na základě tohoto seznamu je třeba provést rešerši existujících technologií a pak všechny nashromážděné poznatky je třeba využít v porovnávací metodice pomocí které se vyberou ty nejvhodnější technologie. Rešerši existujících technologií a různých řešení se zabývá podkapitola o existujících komplexních řešeních (2.5) a to, které technologie byly zvoleny je podrobně shrnuto v podkapitole o vybraných technologiích (2.6).

### 3. Pomocí nástrojů a metod softwarového inženýrství navrhnete rozšíření stávající aplikace o moduly pro správu videa s následující funkcionalitou: přenos po síti včetně ukládání, střih dle zadaných parametrů, změna rozlišení, možnost komprese, editace meta-informací, transformace obrazu atd.

Tím nejzajímavějším bodem zadání je právě návrh zadaného řešení. V kapitole o návrhu (3) se jedná především o sloučení poznatků nabytých v analýze se znalostmi softwarového návrhu. Za užití nástrojů a metod softwarového inženýrství je proveden návrh frontendové i backendové části řešení. Pro moduly zajišťující jednotlivé typy úprav videa je využito technologií studovaných v analytické kapitole (vizte 2.6).

### 4. Implementujte zmíněné rozšíření stávající aplikace (schopné kooperace s aplikací pro správu laboratoře CHUL).

V této produktivní části je na základě návrhu implementován prototyp rozšíření stávající aplikace se všemi zadanými funkcionalitami. Výstupem implementace je pak produkt splňující specifikované požadavky alespoň rámcově, ale zároveň tak, aby bylo možné jej začít testovat a v případě nalezení chyb je opravovat.

### 5. Aplikaci podrobte integračním a akceptačním testům.

Tímto bodem se zabývá celá kapitola o testování (vizte 5), ve které se

provedou nejprve zmíněné integrační testy, kterými se ověří bezchybná komunikace mezi jednotlivými komponentami uvnitř rozšíření stávající aplikace. V poslední řadě se provedou akceptační testy právě v CHUL laboratoři.

## 2.3 Předchozí práce a aktuální stav

Jak bylo řečeno v úvodu, bakalářská práce Karolíny Solanské vyvinula systém, který slouží jako webová aplikace zajišťující plánování a organizaci testů použitelnosti. Tento systém byl vytvořen jako plně funkční vůči zadaným požadavkům a je nasazen v laboratoři CHUL. Systém je celý implementován v jazyku PHP s použitím frameworku Nette. Systém je nasazený na externím serveru s přístupem přes protokoly HTTP a HTTPS. Technologie pro použití databázi je PostgreSQL. Aktuální stav správy videa je příliš nepohodlný a zdoluhavý. Webová aplikace nijak nekomunikuje s jakýmkoliv zařízením v laboratoři CHUL. Vše okolo videí se tedy musí dělat víceméně ručně. V tomto kontextu se pracuje s úžasnou knihovnou *libyuri* (vizte [2]), která za pomoci dvou konfiguračních souborů spustí nahrávání, sloučí obrazy a zvuk, uloží a případně zobrazí v systému SAGE. Naneštěstí existují pouze tyto 2 konfigurace, které se ještě ke všemu musí ručně pouštět z příkazové řádky. Nyní neexistuje žádná jednoduchá možnost, jak pohodlně pracovat se záznamy (například ořezávat, přidávat titulek apod.) a tím se tedy bude zabývat tato práce.

### 2.3.1 Požadavky předchozí práce

Zde jsou uvedeny všechny funkční požadavky z předchozí práce, zaměřené na správu videa a které podpořily důvod vzniku této práce.

1. **Systém ke každému scénáři po proběhlém testu připojí video/-video.**

Před začátkem testu si moderátor zvolí scénář a testera, který test provádí. Následně zvolí spuštění testu a test začne. Po dokončení scénáře pak moderátor zvolí ukončení testu. Systém si zaznamená čas a dle toho připojí k jednotlivým experimentům výsek videa odpovídající časovým značkám začátku a konce experimentu.

2. **Moderátor a zadavatel mohou video přehrávat, zastavit a převíjet.**

Webové grafické rozhraní bude umožňovat zobrazení náhledu videa, které bude fungovat, jako jednoduchý video přehrávač.

3. **Moderátor a zadavatel mohou pořídit screenshot videa a okomentovat ho, následně uložit k danému testu.**

Bude možné vybrat konkrétní snímek videa (v rámci sekund), ten okomentovat a uložit k danému testu. K testu bude možné přiložit několik

snímků (kde každý snímek bude vždy možné utvořit pouze z videí od daného testu).

**4. Moderátor a zadavatel mohou video exportovat a stáhnout.**

Moderátor a zadavatel mohou video vyexportovat v několika různých formátech. Možné je stažení videa, ale také umístění na server Youtube.

**5. Moderátor a zadavatel mohou video konvertovat.**

Moderátor a zadavatel mají možnost video převést na jiný formát, především pro kompatibilitu s různými zařízeními a prohlížeči.

**6. Moderátor může k videu nahrát mluvený komentář.**

Uživatel bude moci nahrát zvuk k libovolnému videu ze vstupního zařízení.

## 2.4 Specifikace požadavků

### 2.4.1 Funkční požadavky

#### F1 – Nahrávání záznamů

**F1.1 – Zobrazení seznamu připojených nahrávacích zařízení.**

Webová stránka umožní zobrazení tabulky obsahující zařízení, která mohou být aktuálně použita pro nahrávání. Zařízení nemusejí být jen kamery, ale také zvuková či jiná zařízení, jejichž komunikační protokol je v aplikaci implementován (zařízení jsou tedy ta v laboratoři CHUL, ale také zařízení nacházející na počítači uživatele, který webovou aplikaci používá). Každý záznam v tabulce bude obsahovat jednoznačnou identifikaci zařízení a případně i aktuální náhled, bude-li se jednat o kameru.

**F1.2 – Možnost nahrávání záznamu z uživatelem vybraných nahrávacích zařízení.**

Tabulka (z předchozího případu) obsahující zařízení schopná nahrávání bude obsahovat checkboxy, které umožní vybrat ta zařízení, na kterých se nahrávání spustí. Při nahrávání bude vytvářeno nejméně tolik záznamů, kolik zařízení bylo vybráno (i více, pokud zařízení nahrává video i zvuk současně).

**F1.3 – Zobrazení seznamu již nahraných záznamů.**

Webová stránka umožní zobrazení tabulky obsahující záznamy, které byly v minulosti nahrané (a jsou v úložišti dostupném webovému serveru). Každý záznam v tabulce bude obsahovat jednoznačnou identifikaci nahrávky a případně i náhled, bude-li se jednat o obrazový záznam.

#### F2 – Přehrávání záznamů



**F2.1 – Možnost přehrát vybraný záznam.**

Existující záznamy zobrazené v tabulce zmíněné v případě výše bude možné přehrát ve webovém prohlížeči. Přehrávání bude možné pozastavit v libovolném čase a bude možný přesun kamkoliv v časové ose videa.

**F2.2 – Moderátor a zadavatel mohou změnit hlasitost přehrávaného záznamu.**

Uživatel bude moci změnit hlasitost zvuku přehrávaného záznamu. Uživatel si v nastavení přehrávaného videa zvolí hlasitost zvuku v rozsahu 0 - 100%.

**F2.3 – Moderátor a zadavatel mohou změnit rozlišení přehrávaného videa.**

Uživatel bude moci změnit rozlišení videa. Uživatel si v nastavení přehrávaného videa zvolí rozlišení, ze seznamem podporovaných rozlišení.

**F2.4 – Moderátor a zadavatel mohou změnit rychlost přehrávání videa.**

Uživatel bude moci ve webovém přehrávači změnit snímkovou frekvenci videa. Uživatel si v nastavení přehrávaného videa zvolí rychlost, ze seznamem podporovaných hodnot.

**F2.5 – Moderátor a zadavatel mohou přehrávání přepnout do celoobrazovkového režimu.**

Uživatel bude moci ve webovém přehrávači zapnout či vypnout přehrávání na celou obrazovku (tzv. full screen mód).

**F2.6 – Moderátor a zadavatel mohou vybrané video streamovat do systému SAGE.**

Uživatel bude moci spustit streamování konkrétního videa do systému SAGE. Systém díky pomocnému softwaru zajistí stream na uživatelem zadanou IP adresu se systémem SAGE.

**F3 – Úprava záznamů****F3.1 – Moderátor a zadavatel mohou sloučit několik videí paralelně do mřížky.**

Uživatel bude moci spojit několik videí vedle sebe do mřížky (např. do čtverce). Tyto videa se budou ve finálním složeném videu přehrávat souběžně. Pokud některé video bude kratší než ostatní, bude doplněno opakujícím se snímkem jednolitě barvy.

**F3.2 – Moderátor a zadavatel mohou video oříznout na menší časový úsek.**

Uživatel bude moci vybrat časy, na které chce aktuální video oříznout. Vzniklé oříznutí bude možné na video aplikovat a poté jej upravovat dále.

### **F3.3 – Moderátor a zadavatel mohou přidat či nahradit zvukovou stopu videa.**

Uživatelé budou zobrazeny existující zvukové záznamy, ze kterých si bude moci jeden vybrat. Vybraný záznam bude poté možné přidat k vybranému videu. Pokud video již zvukový záznam má, přepíše se za nový. Pokud bude zvukový záznam delší než video tak se ořízne na délku videa.

### **F3.4 – Moderátor a zadavatel mohou transformovat obraz všech snímků videa.**

Uživatel bude moci zvolit transformaci obrazu pro celé video. Uživatel na stránce s úpravou videa si vybere transformaci, kterou bude moci aplikovat na dané video. Systém bude podporovat základní transformace jako rotace, změna kontrastu, apod.

## **F4 – Export záznamů**

### **F4.1 – Moderátor a zadavatel mohou změnit rozlišení exportovaného videa.**

Uživatel bude moci změnit rozlišení videa na rozlišení nabízené systémem. Na stránce s exportem videa bude seznam podporovaných rozlišení, uživatel si vybere jedno a systém poté při exportu provede změnu rozlišení daného videa.

### **F4.2 – Moderátor a zadavatel mohou změnit rychlost exportovaného videa.**

Uživatel bude moci změnit snímkovou frekvenci videa. Na stránce s exportem videa bude seznam podporovaných hodnot, a při exportu se provede změna s vybranou hodnotou.

### **F4.3 – Moderátor a zadavatel mohou zvolit formát a kompresi videa.**

Uživatel bude moci zvolit formát a kompresní metodu, kterou systém použije ke kompresi vybraného záznamu.

### **F4.4 – Moderátor a zadavatel mohou editovat meta-informace u videa.**

Na stránce pro export záznamu bude uživatel moci navigovat na formulář s meta-informacemi k videu, kde bude moci tyto informace měnit.

### **F4.5 – Moderátor a zadavatel mohou video uložit na webový server nebo stáhnout.**

Video bude možné uložit do úložiště dostupné na webovém serveru nebo stáhnout do vlastního počítače.

### **F4.6 – Moderátor a zadavatel mohou video umístit na server youtube.**

Video bude možné pomocí komunikace s Youtube API umístit na server Youtube.

## **F5 – Ostatní**

### **F5.1 – Zobrazení volného místa pro nahrávání na lokálním úložišti.**

Webová aplikace bude uživateli zobrazovat vždy aktuální volné dostupné místo v lokálním úložišti pro nahrávání. Údaj bude zobrazen v jednotkách MiB/GiB a případně i minutách nahrávání při standardním rozlišení a formátu.

### **F5.2 – Moderátor a zadavatel mohou pořídit screenshot videa, okomentovat ho a následně uložit k danému testu.**

Bude možné vybrat konkrétní snímek videa (v rámci sekund), ten okomentovat a uložit k danému testu. K testu bude možné přiložit několik snímků (kde každý snímek bude vždy možné utvořit pouze z videí od daného testu).

## **2.4.2 Nefunkční požadavky**

### **N1 – Doba odezvy.**

Rozšíření stávající aplikace bude zaručovat nízkou dobu odezvy při jeho používání. Každá elementární operace, jako spuštění videa či změna hlasitosti přehrávání nebude trvat déle než 2 sekundy.

### **N2 – Udržitelnost.**

Frontend i backend tohoto rozšíření bude efektivně navržen a rozdělen do komponent a to tak, aby schopnost opravení nedostatků systému ovlivnila pouze tu komponentu, ve které se problém vyskytl a nikoliv celý systém.

### **N3 – Dostupnost.**

Rozdělení rozšíření do komponent bude zajišťovat celkovou dostupnost tak, že pokud některá komponenta přestane fungovat, ostatní budou stále dostupné.

### **N4 – Rozšiřitelnost.**

Řešení bude navrženo a vytvořeno tak, aby bylo snadno rozšiřitelné a modifikovatelné. Schopnost přidat novou funkcionalitu nebo modifikovat stávající funkcionalitu ovlivní minimální část celého systému.

### **N5 – Webové uživatelské rozhraní.**

Webové uživatelské rozhraní bude uživatelsky přívětivé, responzivní, jednoznačné, intuitivní a jednoduché. Úprava jeho případných nedostatků bude řešena akceptačními testy.

### **N6 – Prohlížeče a jejich verze.**

Implementované rozšíření bude fungovat nejméně na prohlížečích Firefox, Chrome a Internet Explorer, na jejich stabilních a stále podporovaných verzích.

### **N7 – Technologie použité pro frontend.**

Frontend webové aplikace používá framework Nette, značkový jazyk HTML(verze 5) a javascript.

### **N8 – Technologie pro backend.**

Jako pomocný software pro úpravu záznamů i přenos po síti bude použita knihovna Libyuri.

### **N9 – Provázání frontendu s backendovými moduly.**

Funkcionality frontendu, jako nahrávání, úprava či export videa budou realizovány moduly v backendu. Kontrakt mezi frontendem a těmito moduly bude jednoznačný, vhodně zobecněný a jednoduše rozšiřitelný.

### **N10 – Technologie pro databázový systém.**

Technologie databázového systému použitého pro celou webovou aplikaci je PostgreSQL.

### **N11 – Technologie pro tvorbu modulů.**

Moduly budou vytvořeny jako kompatibilní s knihovnou libyuri a případně budou použity existující moduly.

## **2.5 Existující komplexní řešení**

Při hledání pomocného softwaru, který by řešení značně ulehčil je třeba postupovat od komplexních řešení, které jsou již funkční a otestované komunitou a které by se vypořádaly s velkým počtem funkcionalit. Pokud se nepodaří najít řešení o vhodné úrovni komplexnosti, pak se musí úroveň rozsáhlosti snižovat a poohlížet se tak po technologiích řešící pouze jednotlivé funkcionality. Přednost mají open-source řešení, které je možné libovolně použít a rozšířit.

Pro úsporu místa jsou zde uvedeny především ta nalezená řešení, která byla při rešerši velice dobře hodnocena a měli největší šanci na to stát se kandidáty pro integraci do implementace.

### **2.5.1 Technologie pro backend**

V této části jsou uvedeny některé technologie, které byly rešerší nalezeny jako kandidátní technologie pro backendovou část aplikace. Je třeba nejprve zohlednit technologie, které byly zmíněny ve výčtu nefunkčních požadavků (2.4.2), pokud nějaké. V nefunkčních požadavcích N8 a N10 jsou zadány technologie pro backendovou část a to jsou knihovna Libyuri pro správu videa a databázový systém PostgreSQL. V rešeršní části je ze zmíněných dvou analyzována

pouze knihovna Libyuri, neboť databázový systém PostgreSQL byl již použit v předchozí bakalářské práci, je tedy zakomponován v existujícím systému a nevyskytl se závažný důvod to měnit.

#### 2.5.1.1 Libyuri

Knihovna libyuri ([2]) je framework, poskytující prostředky pro vytváření vícevláknových aplikací zpracovávajících video, audio i jiná média. Knihovna je modulární, což znamená, že jednotlivá úprava videa je zpracovávána jedním modulem (například ořez videa). Zpracování videa a audia v Libyuri je pak možné pomocí vytvoření orientovaného grafu, kde jednotlivé fáze zpracování (moduly) jsou reprezentovány jako uzly grafu a datová propojení mezi moduly jako orientované hrany grafu. Modulů v této knihovně je již nepřeberná spousta, počínaje moduly, které přebírají obrazová data z různých zdrojů (například kamery, V4L zařízení, Decklink zařízení, ...) přes moduly pro zpracování obrazových i zvukových rámců a podobně. Knihovna je také velice snadno rozšiřitelná a díky dobré dokumentaci a šablonám pro vytvoření modulů je opravdu snadné rozšířit tento mocný nástroj o téměř libovolnou funkcionalitu.

Licence libyuri je dle ([2]) pod modifikovanou BSD licencí přičemž moduly používající technologie jiných autorů jsou pod licencemi dalšími, vesměs to však jsou open-source licence, což je další plus.

#### 2.5.1.2 Ultragrid

Ultragrid (odkazem [3]) je software vyvíjený lidmi ze sdružení CESNET z brněnské laboratoře SITOLA, pro nízko-latenční přenos video dat po síti. Software podporuje mnoho známých video standardů, jako například PAL/NTSC, HD, 2K i 4K. Pro zobrazování dat využívá technologie OpenGL a SDL. UltraGrid používá streamy videa bez komprese, nebo s velmi malou kompresí k zaručení až 8K rozlišení s až 100ms latencí mezi koncovými body. Software se již používá v mnoha oblastech jako jsou kolaborující prostředí, lékařská kinematografie, různé vzdělávací aktivity a jiné. Je schopný přenosového módu tzv. dual-link, což je ve zkratce posílání dat dvěma různými cestami pro zajištění větší šířky pásma. Ultragrid je dle ([4]) open-source software pod BSD licencí.

Tento software je také možné použít, jako komplexní řešení neboť podobně jako Libyuri funguje na základě propojování uzlů, které pracují s videem. Ultragrid má oproti Libyuri o trochu lepší dokumentaci, ale na druhou stranu má o mnohem menší počet modulů což je mnohem větší slabina než zmíněná dokumentace. Navíc se nepodařilo nalézt srozumitelný návod či šablonu pro vytvoření vlastních modulů.

### 2.5.1.3 FFmpeg

FFmpeg (odkazem [5]) je jeden z nejlepších a nejznámějších open-source multimediálních frameworků vůbec, který je schopný kódovat, dekódovat, konvertovat, streamovat, filtrovat i přehrávat nejen audio i video záznamy. Tento software podporuje téměř všechny známé formáty záznamů od zastaralých, po ty nejnovější. Je také velmi dobře portabilní, neboť se dá zkompileovat a spustit na distribucích operačních systémů typu Linux, Mac OS X, Microsoft Windows, BSD, Solaris, a tak podobně.

Díky známosti, spolehlivosti a komunitě tohoto frameworku by byla chyba jej nepoužít právě při zpracování videa. Naštěstí předchozí zmíněná softwarová řešení jako Libyuri a Ultragrid tento framework mají v sobě již zakomponován (pokud jsou ovšem zkompileovány s podporou pro FFmpeg).

### 2.5.1.4 GStreamer

GStreamer ([6]) je multimediální knihovna pro konstruování grafu z komponentů pro práci s multimediálním obsahem (podobně, jako Libyuri (2.5.1.1)). Knihovna je kompatibilní se všemi majoritními Linux, Windows, Mac OS X, iOS, stejně tak jako většina systémů BSD, také komerční jako Unixy, Solaris, ale také Android i Symbian. GStreamer má rozsáhlou dokumentaci ([7]) a ještě rozsáhlejší komunitu, takže je velká pravděpodobnost, že potenciální problém je již někde na diskuzním fóru vyřešen. Knihovna je modulární a je velice snadné napsat plugin pomocí poskytovaného generického rozhraní.

GStreamer je open-source a je šířen pod LGPL licenci.

### 2.5.1.5 Open Broadcaster Software

OBS neboli Open Broadcaster Software je dle ([8]) software pro nahrávání a streamování videa vyvíjený dobrovolníky z celého světa. OBS je distribuovaný open-source software pod GPLv2 licenci. Tento software umožňuje spolehlivé nahrávání videa i audia v reálném čase a jeho následné zpracování. Je zde také možné nahrávat současně z více zdrojů, jako například monitoru, kamery, speciální tzv. „grabovací“ grafické karty, webkamery a dalších.

OBS má dobrou dokumentaci (vizte [9]), ve které je mimo jiné zmíněno, že je možné software lehce přizpůsobit pomocí skriptovacích jazyků Lua či Python. Většina funkcionalit v OBS je přidána pomocí plugin modulů, které jsou reprezentovány typicky dynamickými knihovnami nebo skripty. Jelikož se jedná o open-source software, pak je možné moduly ve formě pluginů i libovolně vytvářet na což je v dokumentaci taktéž podrobný návod. Nevýhodou tohoto řešení je velice omezená manipulace s binární podobou programu z prostředí příkazové řádky, což je v případě potenciální automatizace nežádoucí.

## 2.5.2 Technologie pro frontend

Stejně jako v úvodu technologií pro backend (2.5.1) je při rešerši žádoucí respektovat frontendové technologie dohodnuté nefunkčními požadavky (2.4.2). V tomto případě je nefunkčním požadavkem N7 specifikován výčet tří technologií, které jsou framework Nette jazyku PHP, dále značkovací jazyk HTML5 a skriptovací jazyk JavaScript. Framework Nette není třeba zahrnout do porovnání technologií, protože je v něm napsán téměř celý systém předchozí bakalářské práce a bude tedy použit i nadále.

### 2.5.2.1 HTML, Javascript

Technologie HTML5 i JavaScript jsou nejběžněji používané technologie pro vývoj frontendu webových aplikací a v tomto kontextu se tedy nedají brát jako komplexní řešení pro nějakou podmnožinu specifikovaných úkolů a není tedy důvod je porovnávat s ostatními technologiemi pro frontend, které jsou ostatně na nich postaveny.

Všechny řešení analyzované dále se však snaží vyřešit jeden komplexní problém, což je webový video přehrávač/editor. Tento problém lze vyřešit nalezením komplexního řešení, tedy nějakého frameworku, nebo také napsat vlastní přehrávač od píky s pomocí těchto technologií. Nevýhodou tvoření od píky je, že se bude implementovat řešení, které již existuje, což ovšem může snadno vyvážit výhoda toho, že struktura řešení bude známá, a bude nejsnáze modifikovatelné i rozšiřitelné.

### 2.5.2.2 Video.js

Video.js je open-source knihovna pro práci s videem na webu (vizte [10]). Tato knihovna je vlastně obalení pro nativní webový videopřehrávač v technologii HTML5, avšak s přidáním užitečných funkcionalitami. Jelikož je tento software open-source a byl stavěný tak, aby byl dobře rozšiřitelný, tak se komunitě v průběhu času podařilo vyvinout nemalý počet užitečných pluginů (více zde [11]). Knihovna má výbornou dokumentaci ([12]), kde je mimo jiné i popsáno, jak lze napsat vlastní plugin, což je zvládnutelné i se základními znalostmi jazyku JavaScript.

Použití této knihovny je velice vhodné pro frontendovou část řešení této práce. Díky již zmíněným přednostem je lepší a snadněji rozšiřitelný než samotný přehrávač vestavěný v jazyku HTML5. Knihovnu je možné buď stáhnout a v případě potřeby i upravit před použitím nebo použít CDN Fastly ([13]), která poskytuje hosting všem nezbytným souborům knihovny Videojs. Knihovna je kompatibilní s prohlížeči se zabudovaným HTML5 nebo případně i s technologií Flash. Podporovanými prohlížeči je většina dnes nejčastěji používaných jako například Firefox verze 3.5 a vyšší, Internet Explorer 6 a vyšší, Chrome verze 3 a vyšší, Opera a další (více na wiki projektu [14]).

### 2.5.2.3 Afterglow

Afterglow je open-source HTML5 videopřehrávač pod MIT licencí. Dle ([15]) je jednoduše konfigurovatelný plně responzivní a je kompatibilní s širokou škálou webových prohlížečů a zařízení. Přehrávač má kvalitní dokumentaci ([16]) s ukázkovými příklady. Přehrávač je schopen přehrát Youtube i Vimeo video. Je možné mu dodat více stejných videí avšak o jiném rozlišení i v jiném formátu a uživatel si poté může v přehrávači ze zadaných formátů a rozlišení vybírat. Afterglow je díky obalení prohlížečů nativní HTML5 technologie kompatibilní s většinou dnes používaných prohlížečů a je taktéž k dispozici pomocí CDN jsDelivr.

Tato nadstavba nativního HTML5 videopřehrávače má však několik nevýhod. Je to velice jednoduchá nadstavba a neumožňuje dostatek nových funkcionalit nad nativním HTML5 přehrávačem, než které by bylo možno dopsat ručně zanedlouho. Komunita u tohoto softwaru není zdaleka tak silná jako u zmíněného Video.js a tak ani počet pluginů není tak velký (není zde například možné jednoduše udělat playlist videí).

### 2.5.2.4 Movie Masher

Movie Masher ([17]), přesněji knihovna moviemasher.js ([18]) je knihovna jazyku JavaScript pro editaci videa a audia v prohlížeči v reálném čase. Knihovna je open-source s licencí Mozilla Public Licence v2. Tento editor pracuje s HTML5 videopřehrávačem. Editor umožňuje vizuální kompozici videoklipů s přechody, mix audia za použití API WebAudio, Undo a Redo příkazy pro vrácení se zpět a vpřed v historii provedených úprav a další. Jednotlivé efekty jsou pak například vložení textu do videa, zkrácení videa (oříznutí pouze odzadu), změna barev, kontrastu a podobně. Movie Masher je možné buď stáhnout a nasadit na vlastním serveru nebo použít technologii Microsoft Azure ([19]), která jej podporuje avšak její hosting je placený.

Oproti požadovaným funkcionalitám tento video editor několik vymožeností postrádá (skládání videí vedle sebe do jednoho obrazu, ořez videa z obou stran, změna rozlišení, změna formátu a jiné). Editor také oficiálně nepodporuje pluginy ([19]) a v dokumentaci nebyl nalezen návod nebo šablony pro jeho snadné rozšíření.

## 2.6 Výběr technologií

Technologie nashromážděné řešerši je nutné zredukovat, neboť čím menší počet jich použijeme a čím větší část požadavků pokryjí, tím lépe. Nejprve je uvedena porovnávací metodika, která obsahuje výčet funkcí a charakteristik pro výběr technologií a pomocí priorit těchto charakteristik jsou pak technologie porovnány formou tabulek zvlášť pro backend 2.1) a pro frontend (2.2).



### 2.6.1 Porovnávací metodika

Aby výběr těch nejvhodnějších technologií proběhl kvalitně, je zapotřebí k tomu použít určitou metodiku, tedy souhrn postupů, které pokryjí proces výběru. Jednou z těchto metod je stanovení výčtu obecných vlastností a funkcionalit softwaru, jejichž úroveň a kvalita se změří na nalezených technologiích. Dalším postupem je shrnutí nashromážděných hodnocení, zde formou tabulek 2.1 a 2.2 a následné vyhodnocení (vizte 2.6.2), jehož výstupem bude redukovaný výčet technologií, které ve výběru uspěly a budou použity v implementační části této práce.

Nejprve je tedy výčtem uveden seznam obecných prioritních charakteristik softwaru, přičemž u každé z nich je definováno, co se pod ní, v kontextu této práce, myslí.

#### **Licence**

Preferováno open-source řešení nad proprietárním. Prioritní licence jsou například GPLv2, LGPLv2, BSD, MIT, Apache či dokonce WTFPL.

#### **Dokumentace**

Je technická i uživatelská dokumentace dostatečně specifická? Souhlasí verze dokumentace s verzí produktu a koresponduje s jejím aktuálním stavem?

#### **Podpora**

Je produkt stále ještě vyvíjen a alespoň některá z jeho verzí aktivně podporována? Existuje možnost snadné zpětné vazby od uživatele technologie k jejím udržovatelům?

#### **Portabilita**

Je technologie multiplatformní? Je možné ji bez větších problémů zprovoznit na novějších verzích operačních systémů Windows i na několika různých distribucích systému Linux?

#### **Rozšiřitelnost**

Je možné technologii snadno rozšiřovat a upravovat, pokud to licence umožňuje? Je například možné jej rozšířit o moduly s požadovanou funkcionalitou, pokud je zatím technologie nepokrývá?

#### **Integrace**

Zda a jak je možné technologii integrovat do systému této práce. Možnost integrace a kooperace technologie se stávající aplikací a s infrastrukturou SAGE.

#### **Komunita**

Velikost uživatelské komunity vybrané technologie. Existence dalších zdrojů informací o produktu např. webové stránky obsahující informace jiné než v technické dokumentaci.

## 2. ANALÝZA

---

	Libyuri	Ultragrid	FFmpeg	GStreamer
Licence	++	+	++	++
Dokumentace	+	+	++	++
Podpora	+	++	++	++
Portabilita	+++	+++	+++	+++
Rozšiřitelnost	+++	-	+	++
Integrace	+++	++	++	++
Komunita	-	+	++	++
Zkušenosti	+++	+	++	-

Tabulka 2.1: Porovnání technologií pro backend

	Video.js	Afterglow	Movie Masher
Licence	++	++	++
Dokumentace	++	+	-
Podpora	+	+	+
Portabilita	+++	+++	+++
Rozšiřitelnost	++	+	-
Integrace	++	++	++
Komunita	+++	+	-
Zkušenosti	++	-	+

Tabulka 2.2: Porovnání technologií pro frontend

### Zkušenosti

Má autor této práce nějaké zkušenosti s danou technologií, které by mu umožnily snadnější, rychlejší a efektivnější implementaci řešení?

Po porovnání všech nashromážděných technologií pomocí specifikovaných charakteristik bylo dosaženo výsledku, který je popsán tabulkou 2.1 pro backendové technologie a tabulkou 2.2 pro frontendové technologie. V buňkách těchto tabulek jsou buď znaky „+“ reprezentující výhodu či užitečnost a „-“ reprezentující nevýhodu či neužitečnost daného kritéria. Čím více znaků v buňce, tím větší intenzita užitečnosti respektive neužitečnosti daného kritéria.

### 2.6.2 Vybrané technologie

Výběr technologií je proveden na základě zmíněných hodnotících kritérií, které byly předem stanoveny a následně naměřeny na technologiích, které prošly rešerší. Provedené hodnocení je zaznamenáno v patřičných tabulkách pro technologie backendu (2.1) a frontendu (2.2). Zde jsou pak již uvedeny takové technologie, které se určitou mírou budou podílet na implementaci systému této práce. To jsou takové technologie, které byly buď již stanoveny zadáním

práce, nefunkčními požadavky nebo ty, které nejvíce uspěly v provedeném hodnocení.

### 2.6.3 Backend

Výběr technologií pro backend nejprve zohlednil zadání práce a nefunkční požadavky. V nefunkčních požadavcích (2.4.2, požadavek N10) je uvedena databázová technologie PostgreSQL, s jejímž použitím se již předem počítalo, neboť databáze stávající aplikace je již v této technologii dobře navržena. Dále je v těchto požadavcích zmíněna (požadavek N8) i multimediální knihovna Libyuri. Přestože tato knihovna byla podrobena porovnání s ostatními technologiemi nalezenými v rešerši (vizte tabulka 2.1) tak její hodnocení stejně dopadlo velice úspěšně a bude tedy taktéž použita. Z porovnávací metodiky však zbyly další tři technologie: Ultragrid, FFmpeg a GStreamer. Ultragrid nebude použit z toho důvodu, že v hodnocení dopadl o něco hůře než zmíněná knihovna Libyuri a většina funkčnosti, kterou by Ultragrid při implementaci poskytl dokáže Libyuri pokrýt. FFmpeg dopadl v hodnocení obdobně jako Libyuri a jeho použití se velmi dlouho zvažovalo, nicméně pro prototyp první verze systému této práce bude potřeba obstarat pouze funkcionality, jako je ořez videa a nanejvýš změna kontrastu a jasu, přičemž i tyto funkcionality Libyuri umí. Do budoucích verzí systému je však FFmpeg vhodnou kompenzací pro takovou práci s médii, kde Libyuri nebude tak efektivní nebo ji nebude neumět. GStreamer je velice dobrou alternativou Libyuri a tak nejvěrnějšími důvody pro upřednostnění Libyuri nad GStreamerem jsou autorovy slabé zkušenosti s touto technologií a navíc to, že Libyuri pokryje taktéž většinu funkcionalit, které by GStreamer poskytl. GStreamer je také velice snadno rozšiřitelný, avšak Libyuri obsahuje spoustu modulů, které implementaci této práce usnadní a urychlí.

### 2.6.4 Frontend

Frontendové technologie jsou taktéž částečně stanoveny v nefunkčních požadavcích (2.4.2). Konkrétně je to požadavek N7, který specifikuje PHP framework Nette, značkový jazyk HTML a skriptovací jazyk JavaScript. Začneme s frameworkem Nette jazyku PHP. Jak již bylo v úvodu sekce 2.5.2 stanoveno, tak s použitím frameworku Nette byl a napsána téměř celá aplikace z předchozí bakalářské práce a jelikož je zde návaznost řešení této práce na systém práce předchozí, pak samozřejmostí použití této technologie byla vyňata z porovnání s ostatními. Dnes již nativní technologie prohlížečů, HTML a JavaScript budou použity pro tu část této práce, která uživateli předkládá možnost práce s médii a zároveň transformuje uživatelský vstup do řeči backendu. Ostatní zbylé frontendové technologie nashromážděné rešerší jsou na těchto technologiích postaveny a bylo by tudíž nesmyslné dávat spolu do porovnání. Mezi nejlepšími kandidáty byly vybrány a porovnány technologie Video.js,

Afterglow a Movie Masher. Výhody technologie Video.js jsou již popsány v její rešerši. Mezi ně patří především to, že nativní HTML5 videopřehrávač obaluje užitečnými funkcionalitami, a pro vývojáře i tak zachovává rozsáhlý prostor pro rozšíření. Existuje také bezkonkurenční počet již vyvinutých pluginů. Afterglow nepokrývá zdaleka tolik funkcionalit jako Video.js a vylepšení, které má navíc, jako například přehrávání Youtube či Vimeo videí, není výhodou hodnou upřednostnění. Nakonec Movie Masher, jako nadějný webový video editor, by byl býval poskytl pokrytí téměř veškeré práce s videem na webu. Naneštěstí postrádá nemalé množství funkcionalit pravého video editoru a pro budoucí rozšíření by bylo obtížné jej modifikovat formou pluginů. Po porovnání technologií Video.js, Afterglow a Movie Masher je tedy vybrána Video.js.

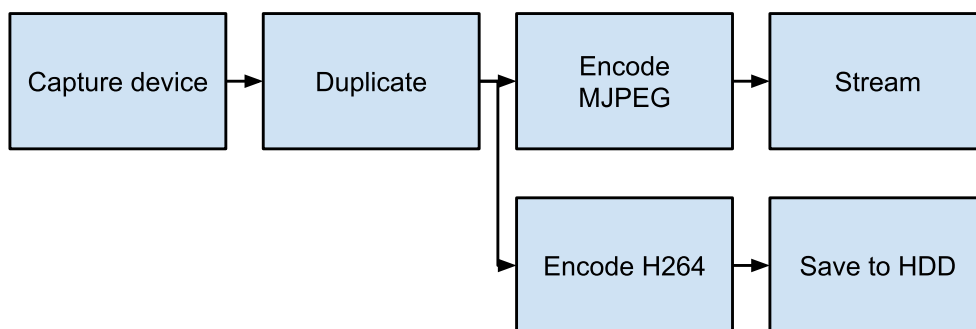
### 2.7 Seznámení s Libyuri

Jak již bylo zmíněno v analýze (2.5.1.1), knihovna Libyuri ([2]) je framework, poskytující prostředky pro vytváření vícevláknových aplikací zpracovávajících různá multimédia.

Framework se skládá ze 4 hlavních komponent:

- Jádru frameworku
  - V aktuální verzi reprezentováno knihovnou *libyuri2.8\_core.so*.
  - Poskytuje funkcionalitu pro vytváření/správu vláken.
  - Poskytuje základní datové typy pro přenos dat (audio, *raw* video, komprimované video).
  - Poskytuje infrastrukturu pro přenos dat mezi uzly grafu.
- Pomocné knihovny
  - Poskytují funkcionalitu, která může být využita více moduly, ale není vhodná do jádra.
  - Například knihovna *libyuri2.8\_helper\_gl.so*, obsahující společné funkce pro knihovnu OpenGL.
- Moduly realizující vlastní funkcionalitu
  - Každý modul může poskytovat libovolné množství uzlů použitelných v grafu.
- Ukázkové programy pro práci s frameworkem
  - Jedná se o programy pro dynamické vytváření aplikací podle konfigurací zapsaných v XML a na příkazové řádce a program pro testování modulů.

Zpracování dat v Libyuri je možné pomocí vytvoření orientovaného grafu, kde jednotlivé fáze zpracování jsou reprezentovány jako uzly grafu a datová propojení jako orientované hrany grafu. Ukázkou takového grafu je obrázek 2.1. Na tomto grafu je znázorněna jednoduchá konfigurace, kdy se berou data z kamery (*Capture device*), komprimovaná do MJPEG se streamují do sítě (*Encode MJPEG* a *Stream*) a zároveň se ukládají na disk v H264 (*Encode H264* a *Save to HDD*).



Obrázek 2.1: Libyuri – příklad orientovaného grafu

### 2.7.1 Uzly jako fáze zpracování

Každý uzel grafu má žádný nebo více vstupů a stejně tak výstupů. Existují typy uzlů, které nemají žádný vstup ani výstup (pokud se nepodílí přímo na zpracování dat), stejně jako uzly s pevně definovaným nebo i neomezeným počtem vstupů/výstupů. Každý uzel je charakterizován svým libovolným jménem a především třídou, která reprezentuje modul, který plní funkcionalitu daného uzlu. Tedy například v ukázkovém grafu 2.1 je jako první zleva uveden uzel se jménem „Capture device“ a může být realizován například modulem „v4l2source“, který dokáže načítat video z kamery typu *video4linux2*. Uzly mohou mít další množství parametrů, které jsou konkrétní každému modulu zvlášť (například zmíněný kamerový modul *video4linux2* může mít další parametry typu šířka, výška, snímková frekvence a jiné).

### 2.7.2 Hrany jako datová propojení uzlů

Orientované hrany v grafu konceptuálně představují fronty (často označované jako *pipe*). Tyto fronty, realizující datová propojení uzlů, se naplňují a vyprazdňují tzv. datovými rámci (co je to datový rámec je uvedeno v sekci 2.7.3).

V Libyuri je aktuálně 7 různých implementací hran s různými vlastnostmi:

### **Single**

Fronta o maximální délce 1 datový rámec. Má pouze 2 stavy – prázdná a plná. Pokud dojde k zápisu nového rámce do plné fronty, starý rámec se z fronty vyhodí a zapíše se do ní nový.

### **Single\_blocking**

Poskytuje stejné chování jako fronta *Single* s tím rozdílem, že zápis do plné fronty selže (a zablokuje uzel do fronty zapisující, dokud se ve frontě neuvolní místo).

### **Count\_limited, Size\_limited**

V těchto typech je velikost fronty omezena počtem rámců (předpona *Count\_*), případně jejich celkovou velikostí (předpona *size\_*).

### **Count\_limited\_blocking, Size\_limited\_blocking**

Varianty téměř totožné s předešlými, fungující analogicky jako fronta *Single\_blocking*, tedy zápis do plné fronty zablokuje zapisovací uzel do doby, než se ve frontě uvolní místo.

### **Unlimited**

Jedná se o nekonečnou frontu. Nemá žádné omezení, až na paměť počítače.

Uvedená množina front není omezená a je možné ji rozšiřovat.

## **2.7.3 Datové rámce**

Mezi uzly, po hranách grafu putují datové rámce (neboli *frames*). Ty představují základní jednotku dat. Libyuri obsahuje hierarchii typů datových rámců z nichž nejdůležitější typy rámců jsou:

### **Frame**

Obecný snímek - abstraktní třída reprezentující cokoliv, co může procházet po hranách.

### **RawVideoFrame**

Konkrétní třída reprezentující nekomprimovaný video snímek.

### **CompressedVideoFrame**

Třída pro komprimovaný video snímek.

### **RawAudioFrame**

Třída obsahující několik vzorků nekomprimovaného audia.

Množina typů datových rámců se dá taktéž rozšiřovat pomocí specializace třídy *Frame*.

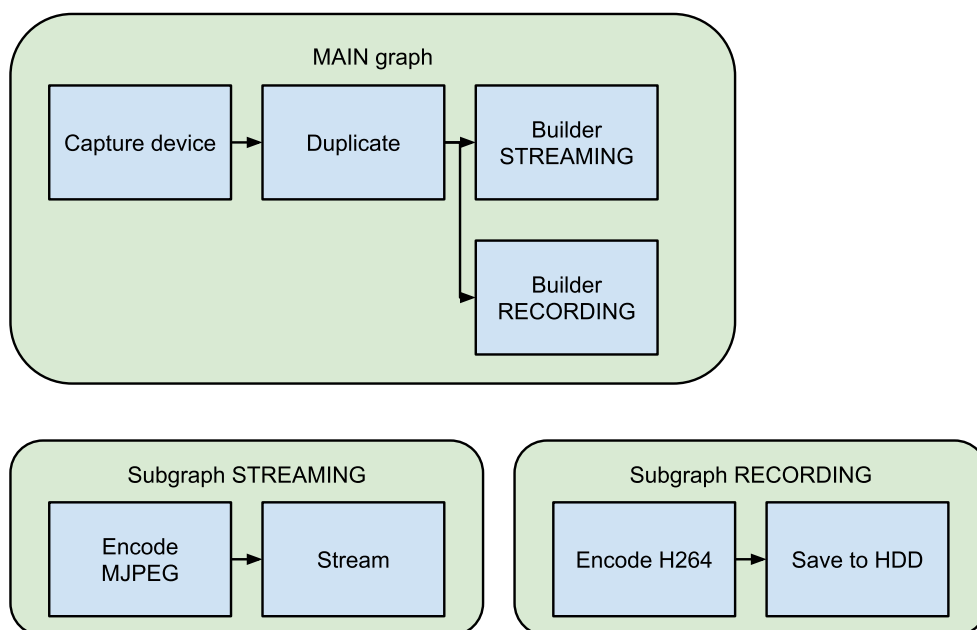
### 2.7.4 Vytvoření grafu

Pro vytvoření grafu je možné přímo využít funkcí z API knihovny Libyuri. Pro snadnější vytváření je nicméně vhodné použít takzvaný builder. To je koncept třídy, která vytváří graf na základě nějakého vstupu.

#### 2.7.4.1 GenericBuilder

Jádro obsahuje obecnou implementaci builderu pod názvem „GenericBuilder“, usnadňující vytváření vlastních builderů, jako příklad je možné použít vytvořený builder „SimpleBuilder“ v aplikaci `yuri_simple` (více o aplikaci `yuri_simple` v sekci 2.7.7). GenericBuilder vytváří graf specifikovaný množinou uzlů a množinou hran.

Konceptuálně představuje builder graf a zároveň je možné jej chápat jako samostatný uzel, který může být součástí většího grafu. Například předchozí ukázkový graf (2.1) je možné rozdělit na 3 propojené podgrafy, každý vytvořený pomocí jednoho builderu, jak je patrné na grafu 2.2.



Obrázek 2.2: Libyuri – příklad propojených podgrafů pomocí builderu

#### 2.7.4.2 XMLBuilder

V jádře Libyuri je také implementace builderu „XMLBuilder“, který načítá XML soubory popisující graf a datové cesty a z nich pak výsledný graf vytvoří. Použití tohoto builderu je demonstrováno dále, kde je nejprve uvedena

## 2. ANALÝZA

konfigurace grafu v jazyku XML a poté vygenerovaný graf.

```
<?xml version="1.0" ?>
<app name="webcam" xmlns="urn:library:yuri:xmlschema:2001"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <variable name="device">/dev/video0</variable>

  <node class="v4l2source" name="webcam">
    <parameter name="path">@device</parameter>
  </node>
  <node class="sdl_window" name="sdl"/>

  <link name="yuyv_image" class="single" source="webcam:0" target="sdl:0"/>
</app>
```

V uvedené XML konfiguraci je uzel se jménem „webcam“ typu „v4l2source“, který představuje zdroj z webkamery. Uzel nazvaný „sdl“ typu „sdl\_window“ je okno, které zobrazuje video. Mezi těmito uzly je orientovaná hrana (v Libyuri označovaná jako *pipe* nebo také fronta), pojmenovaná „yuyv\_image“, typu „single“. Ukázková konfigurace zároveň obsahuje proměnnou (značka <variable>) obsahující systémovou cestu ke kameře. Tato proměnná je pak použita pro nastavení hodnoty parametru „path“ u uzlu „v4l2source“. Výhody oproti přímému nastavení hodnoty (které je samozřejmě také možné) jsou dvě. Jednak je možné snadno zajistit stejnou hodnotu pro více parametrů (i v různých uzlech), ale také je možné tuto proměnnou nastavovat při spuštění yuri aplikace (více v sekci 2.7.7).

Zmíněná XML konfigurace vytvoří graf znázorněný obrázkem 2.3.



Obrázek 2.3: Libyuri – graf vytvořený pomocí XMLBuilder

XMLBuilder se chová jako každý jiný uzel a je možné ho použít v grafu. Podporuje i směrování rámců a událostí mezi vnějším a vnitřním grafem. Je tak možné napsat například složitý graf, kde vstup je reprezentován subgrafem, který je možné snadno přepnout na jiný, jak již bylo ukázáno na obrázku 2.2. Stejně tak je možné takto vyměňovat libovolnou část grafu. Pokud mám například složitě zapojení kamery a potřebuji výsledek někdy streamovat a někdy ukládat, tak stačí mít jednu konfiguraci pro kameru, která má



finální zpracování specifikované jako subgraf. Ten pak mohu měnit (ukládání, streamování). Název XML souboru se pro XMLBuilder uvádí jako parametr „filename“. Ten je možné samozřejmě specifikovat pomocí proměnné, která se dá při spuštění nastavit na jinou hodnotu.

Kompletní popis builderu XMLBuilder je v souboru *doc/XMLBUILDER.txt* v distribuci Libyuri.

### 2.7.5 Události

Propojení uzlů v grafu pomocí hran definuje datové cesty pro audio/video data. Kromě toho je ale také možné mezi uzly posílat události (tzv. *events*). Každá událost má svůj typ a hodnotu (například *bool* a *false*, nebo *double* a *3.14*). V Libyuri existují následující typy událostí:

<b>Bool</b>	Hodnota <i>true</i> nebo <i>false</i> .
<b>Int</b>	Celočíselná hodnota, omezená na 64 bitů. Volitelně může mít i specifikovaný rozsah do kterého patří.
<b>Double</b>	Hodnota s plovoucí desetinnou čárkou, omezená na 128 bitů. Volitelně může mít specifikovaný rozsah.
<b>String</b>	Textová hodnota.
<b>Time</b>	Časová známka.
<b>Vector</b>	Uspořádaná množina událostí (heterogenních typů)
<b>Dictionary</b>	Slovník mapující hodnoty typu <i>String</i> na události (heterogenních typů)
<b>BANG</b>	Speciální typ, který nemá hodnotu.

Uzel, který implementuje koncept „BasicEventProducer“ může generovat (*emit\_event*) dvojice jméno a událost. Uzel sám neřeší kam se událost posílá. Oproti tomu uzel, který implementuje koncept „BasicEventConsumer“ umí dvojice jméno a událost přijímat. Jaká událost se kam posílá se specifikuje zvlášť v konfiguraci.

Dále je uveden jednoduchý příklad XML kódu demonstrující události.

```
<event>  
  route(dump:sequence) -> info:progress;  
</event>
```

## 2. ANALÝZA

---

Pokud se zanesse tento příklad do existující XML konfigurace grafu, tak ve chvíli, kdy uzel pojmenovaný „dump“ vygeneruje událost s názvem „sequence“, tak se ta událost pošle do uzlu „info“ pod názvem „progress“.

Specifikace směrování podporuje i funkcionální jazyk pro modifikaci událostí nebo kombinující více událostí dohromady. Je tak možné vytvářet i složitější konstrukce jak v příkladu uvedeném dále.

```
<event>
  route(gt(dump:sequence, var:max_frames)) -> generate:stop;
</event>
```

Tento trochu komplexnější příklad vezme hodnotu události „sequence“ z uzlu „dump“, porovná ji (funkce *gt*) s hodnotou události „max\_frames“ z uzlu „var“ a výsledek (hodnota *true* nebo *false*) se pošle na uzel „generate“ jako událost s názvem „stop“.

Detailnější popis událostí je v souboru *doc/EVENTS.txt* v distribuci Libyuri.

### 2.7.6 Moduly

Jak již bylo zmíněno v sekci o uzlech grafu zpracování audia/video (2.7.1), každý uzel je charakterizován třídou, která reprezentuje modul plnící funkcionality daného uzlu. Jako modul je označována dynamická knihovna (například *yuri2.8\_module\_v4l2\_source.so*), která může obsahovat dodatečné funkce pro Libyuri. Typický takový modul obsahuje jeden nebo více uzlů do grafu, může ale obsahovat i jiné funkce (například modul *yuri2.8\_module\_yuri\_udp.so* obsahuje implementaci konceptu *DatagramSocket* pomocí UDP socketu).

Moduly se kompilují samostatně, mohou být umístěny i mimo distribuci Libyuri. Pro kompilaci modulu je nutné pouze dostupné API Libyuri a knihovna *yuri2.8\_core*, se kterou se linkují. Aplikace, které využívají Libyuri (vizte 2.7.7) se linkují pouze s jádrem knihovny. Po spuštění aplikace, jádro (pomocí builderu) vyhledá moduly v adresářích napevno uvedených v kódu a případně dalších, specifikovaných uživatelem a načte je.

### 2.7.7 Aplikace používající libyuri

Jako součást knihovny Libyuri je distribuováno několik aplikací, které tuto knihovnu používají. Dále jsou uvedeny 3 nejdůležitější pro implementaci řešení této práce.

#### Yuri\_simple

Program pro vytváření grafů bez konfiguračního XML souboru. Všechny

potřebné informace k vytvoření grafu se předají argumenty při spuštění programu. Například následující příkaz spustí stejnou konfiguraci jako ukázkový příklad grafu ze sekce XMLBuilder (2.7.4.2).

```
./yuri_simple v4l2source[path=/dev/video0] sdl_window
```

### Yuri\_test\_module

Aplikace testující validitu modulů. Následující příkaz otestuje, jestli je modul *yuri2.8\_module\_null.so* validní.

```
./yuri_test_module yuri2.8_module_null.so
```

### Yuri2

Tento program načítá konfigurační XML soubor (vizte popis XMLBuilderu 2.7.4.2) a spouští aplikaci v něm popsanou. Následuje nejjednodušší příklad spuštění tohoto programu.

```
./yuri2 config.xml
```

Pokud jsou v konfiguračním souboru uvedeny proměnné (značky `<variable>`, stejné, jako v příkladu 2.7.4.2), je možné jim nastavit hodnotu na příkazové řádce tak, jako v následujícím příkladu, kde se nastaví proměnná „device“.

```
./yuri2 config.xml device=/dev/video0
```

Aplikace *yuri2* má množství dalších parametrů, pro jejich seznam stačí spustit aplikaci bez parametrů a zobrazí se základní nápověda. Mezi důležité parametry patří:

- a Vypíše informace o zadaném konfiguračním souboru.
- v Detailnější výpisy, -q je méně detailní výpis.
- L class Vypíše informace o třídě „class“.
- l [what] Bez uvedeného argumentu *what* (nebo pokud je tento argument „classes“) vypíše seznam registrovaných tříd. Pokud je argument „pipes“, tak vypíše seznam tříd pro *pipes* neboli fronty.
- C format1:format2 Vypíše, pokud je možná automatická konverze z formátu „format1“ do „format2“ a vypíše, jaké uzly by tuto konverzi realizovaly.



# Návrh

Návrh nejprve rozebere architekturu systému a pak ve dvou podkapitolách proběhne návrh frontendové (prezentační vrstva aplikace, která systému předkládá vstupy od uživatele a zároveň uživateli zobrazuje výstupy) a backendové (kde je uvedeno, jaká je struktura té části aplikace, která je uživateli odstíněna, ale která také zajišťuje většinu funkcionalit) části řešení.

## 3.1 Architektura systému

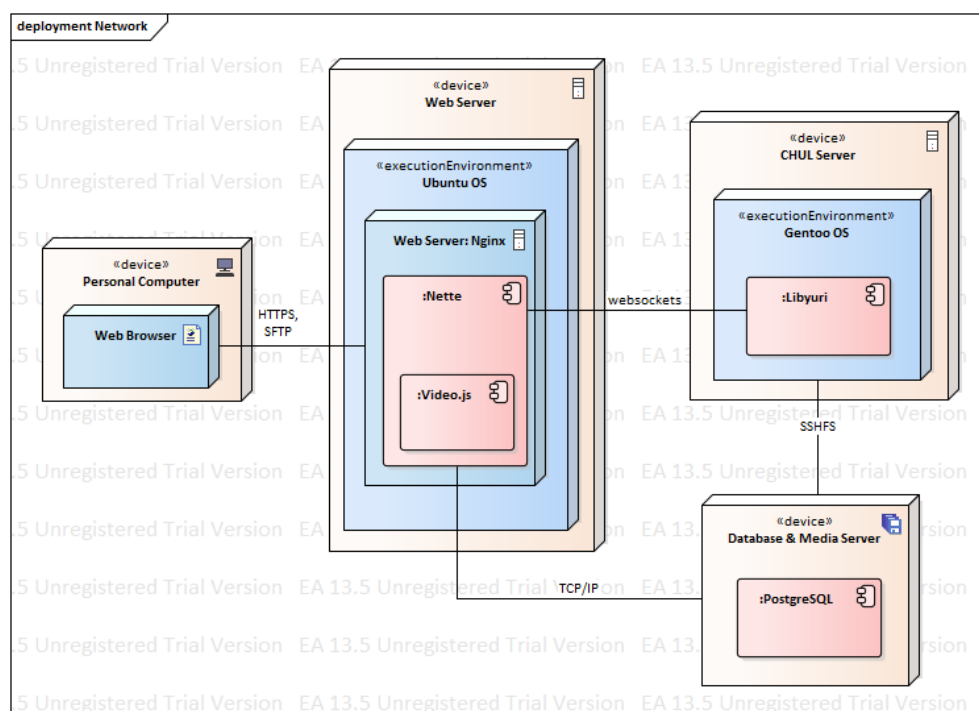
Řešení bude navrženo, jako typická třívrstvá architektura skládající se z prezentační aplikační a datové vrstvy. Prezentační vrstva zahrnuje frontend 3.2 a jeho zprostředkování pomocí webového serveru. Aplikační vrstva pojímá aplikační server (server v laboratoři CHUL) včetně nahrávacích zařízení a komunikace mezi nimi pomocí knihovny Libyuri. Datová vrstva se pak skládá z PostgreSQL databáze a úložiště veškerých nahrávek.

### 3.1.0.1 Diagram nasazení

Schéma na obrázku 3.1 uvedeném dále představuje přibližný model nasazení v provozu. Tento model se skládá ze čtyř hlavních částí (zařízení), z nichž první představuje klienta a jeho webový prohlížeč. Ten komunikuje s aplikací Nginx, která představuje webový server. Samotná komunikace mezi klientem a webovým serverem probíhá pomocí protokolů HTTPS a SFTP pro procházení webové aplikace a případný download videí. Aplikace Nginx hostuje samotnou webovou aplikaci, jejíž podstatná část je tvořena PHP frameworkem Nette a dále Javascriptovou knihovnou Video.js (2.5.2.2), která je odpovědná za práci s videem na webu. Další částí diagramu je CHUL server, neboli server, nacházející se v CHUL laboratoři, na kterém běží instance knihovny Libyuri (2.5.1.1). Komunikace mezi Libyuri a webovou Nette aplikací je zajištěno technologií websocketů, pomocí kterých se například z webové aplikace pošle příkaz na spuštění nahrávání z různých zařízení v laboratoři. K operačnímu systému

### 3. NÁVRH

CHUL serveru je pomocí technologie SSHFS připojen souborový systém dalšího a posledního zařízení v diagramu – databázového a multimediálního serveru (Dále jen „DM server“). Tento DM server obsahuje zmíněnou databázi PostgreSQL a také úložiště pro nahrané záznamy. Databázový systém PostgreSQL na DM serveru obsahuje především data potřebné webovou aplikací (jako jsou data o uživateli, experimentech, scénářích a jiné) a tudíž je se samotnou webovou aplikací spojena komunikačním protokolem TCP/IP.



Obrázek 3.1: Diagram nasazení

## 3.2 Frontend

### 3.2.1 Návaznost na současný systém

### 3.2.2 Uživatelské rozhraní

Tato sekce se věnuje návrhu a tvorbě UI (User Interface) frontendové části řešení. Se zužitkováním znalostí z [20] je využito ověřených postupů a je kladen důraz na principy dobrého UI jako jsou jednoduchost, předpověditelnost, zobrazování pouze užitečných informací, tolerance chyb a další.

Návrh uživatelského rozhraní rozdělen na několik fází. Prvním krokem je seznam požadavků a vlastností kladených na uživatelské rozhraní. Tyto požá-

žadavky jsou již stanoveny v analýze (2.4). Zmíněné požadavky a vlastnosti jsou poté, po poradě s vedoucím práce, transformovány do takzvaného task listu, jehož finální podoba obsahuje uspořádaný seznam základních funkcionalit UI z uživatelského pohledu. Dalším krokem, dle uznávaných postupů, by byl výběr cílové skupiny uživatelů tohoto systému a uživatelský výzkum. Cílová skupina uživatelů je však již známa díky již existující webové aplikaci, jejíž koncepty budou pro první prototypy UI této práce postačující. Uživatelské testování je tak tedy provedeno až na prvních prototypech řešení a je částečně zmíněno v kapitole o testování (5.2.1). Pro zvýraznění struktury závislostí mezi funkcionalitami task listu bude vytvořen tzv. task graf (nebo také task model), který znázorní vztahy mezi jednotlivými funkcionalitami. Nakonec se vytvoří hi-fi prototyp, který bude reprezentovat grafickou a interaktivní podobu uživatelského rozhraní.

### 3.2.2.1 Task List

Zde je uveden seznam většiny úkolů, které by mělo uživatelské rozhraní splňovat, jakožto požadavky uživatelů. Pokud se ve vytvořeném seznamu nedopatřím neobjevily věci, které tam být mají, tak by se měly ukázat při testování použitelnosti uživatelského rozhraní.

- Zobrazení zařízení schopných nahrávání.
- Označení skupiny zařízení pro nahrávání
- Volba streamu do systému SAGE
- Spuštění nahrávání (streamování)
- Zastavení nahrávání (streamování)
- Zobrazení streamovaného obrazu
- Spuštění editoru/přehrávače nahrávek
- Zobrazení existujících nahrávek
- Přehrání vybrané nahrávky
- Změna hlasitosti přehrávání
- Změna rychlosti přehrávání
- Změna rozlišení přehrávání
- Spuštění přehrávání na celou obrazovku
- Sloučení nahrávek paralelně do mřížky

### 3. NÁVRH

---

- Volba časového ořezu videa
- Rotace obrazu videa
- Změna kontrastu obrazu videa
- Volba rozlišení pro export
- Volba formátu pro export
- Volba komprese pro export
- Volba uploadu záznamu na server Youtube
- Zobrazení editovatelných meta-informací
- Návrat z editoru do streamování
- Zobrazení volného místa v úložišti

Dále je nutné tento neorganizovaný seznam funkcionalit seskupit tak, aby to co nejlépe pomohlo s návrhem UI. Tasky se seskupují například podle toho, zda-li se jedná o vstup od uživatele, oznámení chyb, jak velkou část obrazovky zaujímají a podobně. Některé položky v seznamu mohou být syntaxí skryté duplicity jiných. Zároveň je žádoucí seřadit položky dle jejich důležitosti (například chybové hlášky mají vysokou prioritu).

- Nastavení streamování
  - Zobrazení zařízení schopných nahrávání.
  - Označení skupiny zařízení pro nahrávání
  - Volba streamu do systému SAGE
  - Sloučení streamů paralelně do mřížky
  - Spuštění nahrávání (streamování)
  - Zastavení nahrávání (streamování)
- Zobrazení streamovaného obrazu
- Spuštění editoru/přehrávače nahrávek
- Zobrazení existujících nahrávek
- Přehrání vybrané nahrávky
  - Změna hlasitosti přehrávání
  - Změna rychlosti přehrávání
  - Změna rozlišení přehrávání



- Spuštění přehrávání na celou obrazovku
- Úprava videa pro export
  - Volba časového ořezu videa
  - Změna kontrastu obrazu videa
  - Volba rozlišení pro export
  - Volba formátu pro export
  - Volba komprese pro export
  - Rotace obrazu videa
  - Zobrazení editovatelných meta-informací
  - Volba uploadu záznamu na server Youtube
- Zobrazení volného místa v úložišti
- Návrat z editoru do nastavení streamování

Z již uspořádaného task listu je patrné, že pro realizaci UI budou potřeba 2 až 3 webové stránky (jedna pro přehrávání a úpravu záznamů, další pro nastavení a spuštění nahrávání/streamování a případná třetí pro sledování průběhu nahrávání/streamování). .... "tolik elementů pro control (tlacítek), tolik main display elementu, tolik status lajn, tolik kamer ..... atd -> tím padem se tahle obrazovka rozdeli na tolik casti, ktere obsahujou to a to , atd"

#### 3.2.2.2 Task Graf

#### 3.2.2.3 Prototypování

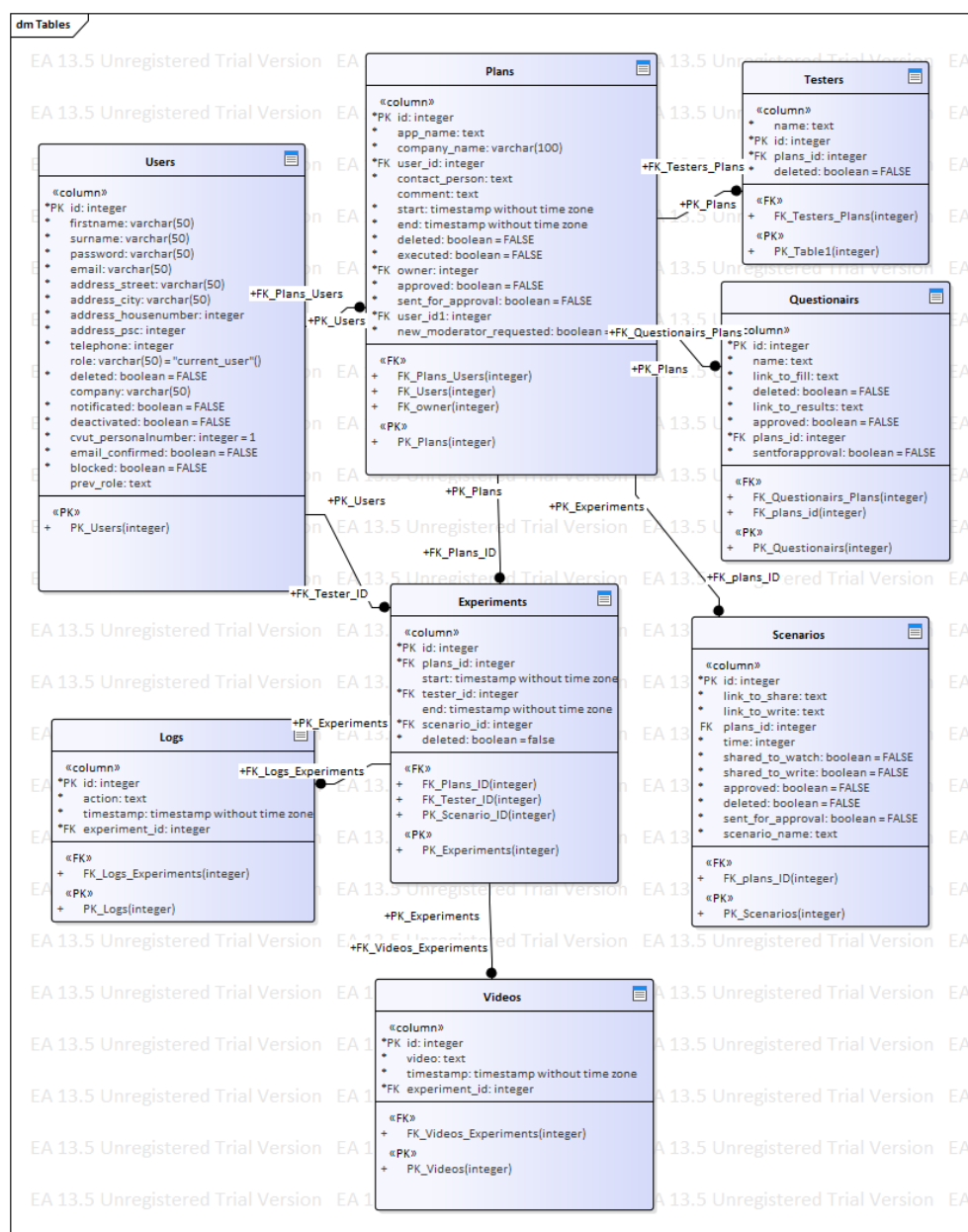
hifi prototypes (interactive wireframes)

### 3.3 Backend

#### 3.3.1 Databázový model

Při návrhu relačního databázového modelu se vycházelo z již existujícího modelu. Předchozí práce ([1]) totiž již se systémem pro správu videa a streamů částečně počítala ve specifických požadavcích, ale i během vytváření tehdejšího databázového modelu, který již obsahoval tabulku pro videa. Jak bylo zmíněno, tak rozšíření pro správu videí a streamů se v předchozí práci neimplementovalo a tedy i tato funkcionality byla v databázovém modelu pouze jakýmsi základním prototypem, který bylo nutné řádně poupravit. Na obrázku 3.2, který znázorňuje právě databázový model při konci předchozí závěrečné práce, je patrná zmíněná tabulka s videi (*Videos*), která však zdaleka neobsahuje všechny potřebné sloupce.

### 3. NÁVRH



Obrázek 3.2: Původní databázový model

Horní část původního databázového modelu (3.2) obsahuje tabulky (např. *Users*, *Logs*, *Plans*,...), které se správou videí a streamů příliš nespojují, jsou však implementované v původní webové aplikaci a proto jsou zachovány v nezměněné formě novém modelu. Bylo tedy třeba přidat část pro správu videí a informací o nich. Na obrázku 3.3 je uveden nově vytvořený databázový model, který je pak dále popsán.

Z nového databázového modelu je patrné, že zmizela tabulka s videi a byla nahrazena tabulkou *VideoRecords* a *AudioRecords*. Krom videí bylo také třeba udržovat informace o audio souborech, aby bylo možné s nimi pracovat odděleně (například přidávat a odebírat audio stopy k videu). Z důvodu nemožnosti implementace dědičnosti v relačním databázovém modelu nebyla vytvořena třída *Records*, která by zapouzdřovala možné nahrávky. Dále je z diagramu patrná tabulka *Jobs*, do které se budou ukládat jednotlivé úlohy pro úpravu záznamů (více v sekci 3.3.2). Tato tabulka obsahuje vstupní záznamy (relace z tabulek *VideoRecords* a *AudioRecords*), které reprezentují záznamy připravené k nějaké úpravě a dále výstupní záznam realizovaný tabulkou *OutputRecord*. Tabulka *OutputRecord* obsahuje záznamy o médiích vzniklých úpravou jiných po dokončení některé ze zmíněných úloh. Jako poslední je také přidána samostatná tabulka *LabConfiguration*, která obsahuje informace o nahrávacích zařízeních (které jsou připojené a podobně), IP adresy a informace o zařízení SAGE, webserveru, CHUL serveru, úložišti a další.

V tabulkách *VideoRecord*, *AudioRecord*, respektive *OutputRecord* jsou sloupce reprezentující požadované meta-informace nahraných, resp. upravených, záznamů. Veškeré nahrané záznamy se nebudou ukládat do databáze přímo, ale do souborového systému, a databázové tabulky pak budou obsahovat sloupec s lokací, kde se daný záznam v souborovém systému nachází.

### 3.3.2 Správa úloh

Při popisování vytvořeného databázového modelu byla zmíněna tabulka *Jobs*, která obsahuje informace o úlohových zpracování záznamů. Dále je popsáno, co za touto tabulkou stojí.

Backendový subsystém pro zpracovávání záznamů upravených ve webové aplikaci je navržen jako dávkové zpracování úloh. V systému se nachází fronta úloh, obsahující úlohy připravené ke zpracování (například převod videa do jiného formátu). Dále je v systému démon, tedy proces běžící na pozadí, který tuto frontu obhospodařuje. Tento proces si hlídá existující zpracovávané úlohy knihovnou Libyuri a pokud se uvolní výpočetní prostředky, pak se z fronty úloh vezme ta s nejvyšší prioritou a začne se zpracovávat. Informace o těchto úlohách jsou uloženy právě v tabulce *Jobs*. Na této tabulce je také v PostgreSQL databázi nastaven tzv. trigger (pomocí příkazu `CREATE TRIGGER`), který, po zápisu nové úlohy do tabulky, probudí či notifikuje zmíněného démona pro správu úloh, který nově zapsanou úlohu zaregistruje.

### 3.3.3 Libyuri

V backendové části aplikace se na CHUL serveru nachází instance knihovny Libyuri. Tato knihovna zde bude sloužit jako prostředník mezi nahrávacími zařízeními v CHUL laboratoři a multimediálním úložištěm, případně webovou

### 3. NÁVRH

---

aplikací. Reference a návod pro pojmy a různé postupy je k nalezení v analýze v sekci 2.7, kde je zevrubně popsáno, jak Libyuri, potažmo aplikace Yuri2 pracuje.

Nejprve je zde, na obrázku 3.4, uveden příklad konfigurace Libyuri pro spuštění nahrávání z vybraných zařízení v CHUL laboratoři. Tato konfigurace slouží jednak pro ilustraci toho, které moduly a v jakém pořadí by se měly propojit, a zadruhé jako vzor pro to, jak budou vypadat konfigurace vytvářené za běhu, na základě uživatelského vstupu přes webovou aplikaci. Tato konfigurace se nachází na stroji CHUL server (vizte diagram nasazení 3.1) při tamní instanci knihovny Libyuri.

Uvedený obrázek 3.4 obsahuje kolečka spojené šipkami, kde každé kolečko reprezentuje uzel a každá šipka reprezentuje orientovanou datovou hranu v grafu konfigurace Libyuri. Každý uzel uvnitř obsahuje svůj název, který je libovolný a pod ním je v závorce přesný název modulu knihovny Libyuri (více v sekci o zpracovávajících uzlech 2.7.1), která funkcionalitu tohoto uzlu realizuje, tedy například druhý uzel má název „webcam“ a je realizován modulem „v4l2source“ (*v4l2* je linuxový ovladač pro práci se zařízeními typu webkamera). Každá orientovaná hrana je popsána obdobným způsobem. Nejprve je uvedeno libovolné jméno hrany a pod ním je v závorce typ datové fronty (více v sekci o datových hranách 2.7.2), která jej realizuje (například první hrana shora zleva má název „screen\_delay“ a je typu „single“).

V takovéto konfiguraci prezentované obrázkem, který se interpretuje odshora dolů, je patrných pět nahrávacích zařízení, které jsou reprezentovány první řádkou uzlů. Mezi těmito zařízeními jsou zleva: obrazovka počítače (screen), webkamera (webcam), dvě kamery připojené přes rozhraní HDMI (hdmi0 a hdmi1) a nakonec jeden mikrofón (audio\_in). Z kamer putují datové rámce do modulů pro upravení FPS (uzly *fps0* – *fps2*), dále všechny rámce putují do modulu *delay*, který upraví zpoždění jednotlivých streamů dle potřeby, obrazové streamy pak navštíví modul *jpeg\_encoder* pro kompresi do formátu JPEG čímž se zefektivní přenos dat po síti a nakonec je všech pět streamů modulem *uv\_rtp\_sender* vysláno sítí směrem k cílové IP adrese a portu kde tyto data přebírá jiná instance knihovny Libyuri (například do multimediálního serveru nebo do infrastruktury SAGE).

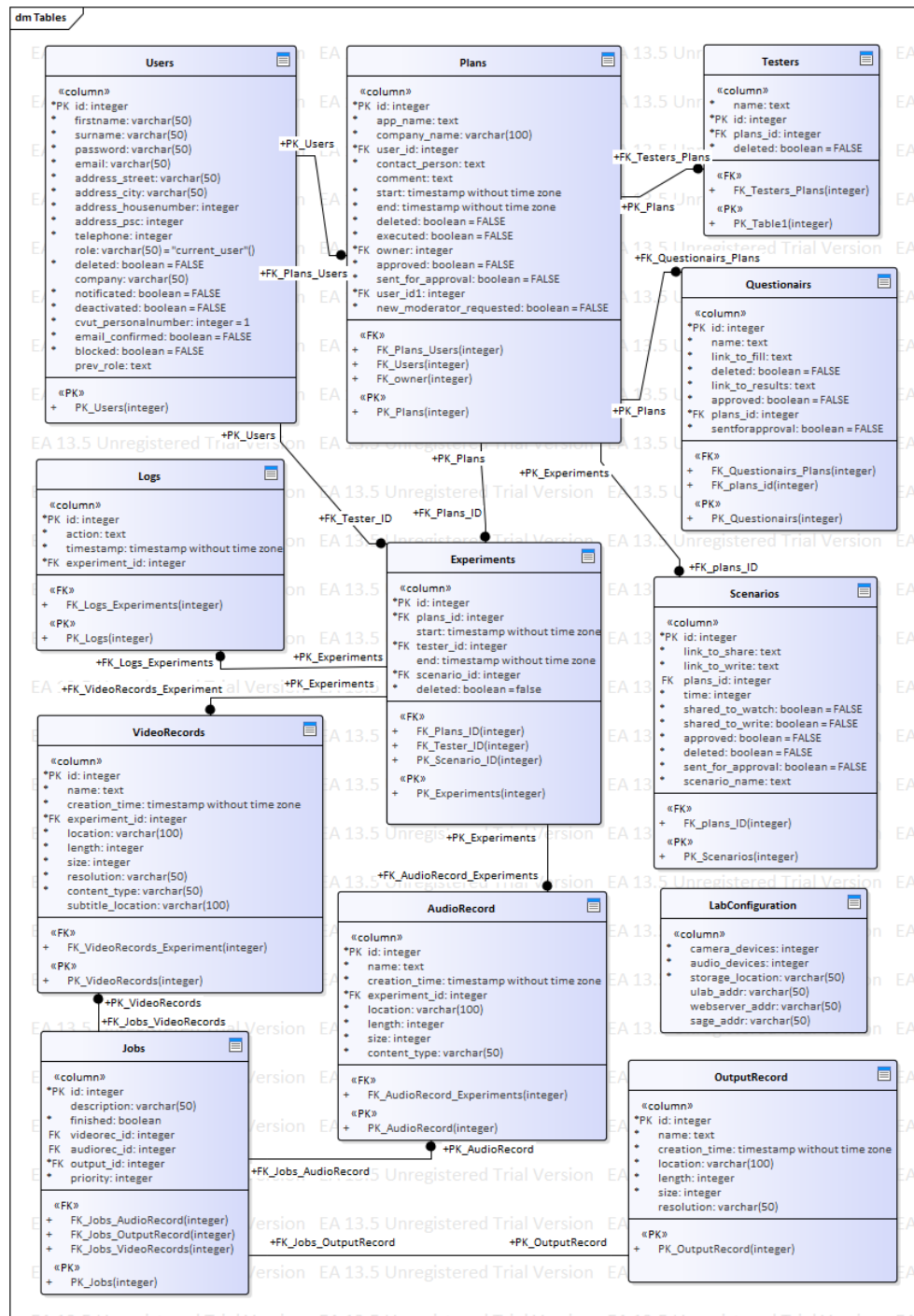
Nahrávaná data v laboratoři CHUL a odesílaná po síti pryč by měla být někde přijímána. K tomu slouží další instance knihovny Libyuri s pozměněnou konfigurací pro příjem dat a následné zpracování. Pro ukázkou je zde na obrázku 3.5 uvedena konfigurace, která se nachází na serveru infrastruktury SAGE, a která přijatá data souběžně ukládá na disk a zároveň streamuje do systému SAGE.

Konfigurace na obrázku 3.5 má podobnou strukturu, jako předchozí konfigurace. Opět se interpretuje odshora dolů. V první řadě je patrných pět uzlů (uzly *rtp\_video0* – *rtp\_video3* a *rtp\_audio\_in*), které se starají o síťový

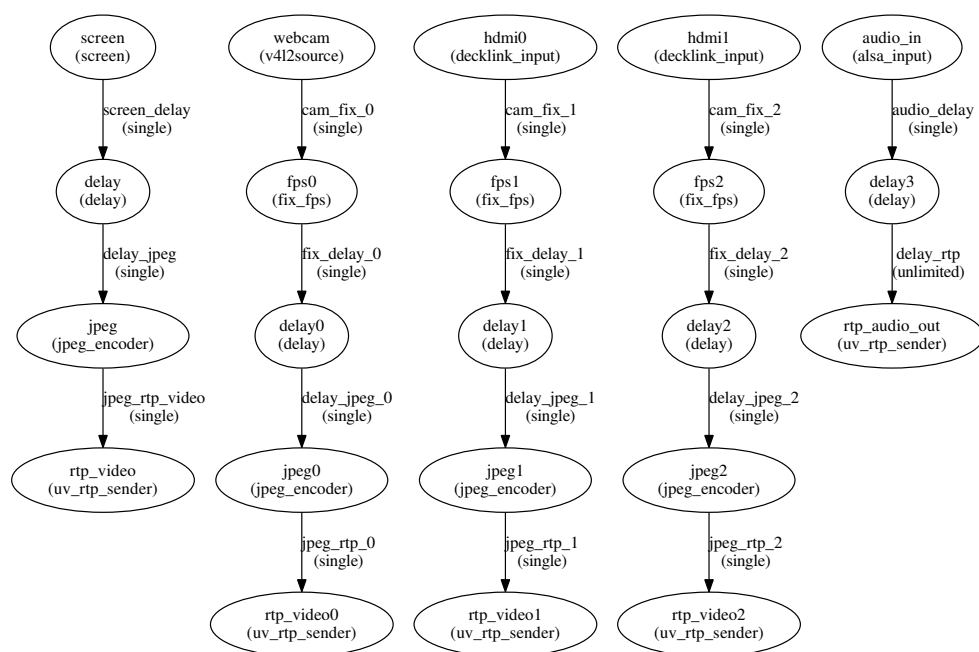
příjem dat, které byly předchozí konfigurací odesílány pomocí RTP streamů. Obrazová data se ze síťových přijímačů posílají do modulů *convert*, které provedou konverzi obrazů do shodného prostoru barev, dále se pomocí modulu *scale* zmenší či zvětší na požadované rozlišení a všechny se poté spojí vedle sebe do obrazové mřížky v jediném modulu *combine*. Z tohoto modulu dále postupuje pouze jeden stream (který je kombinací všech čtyř obrazových dat) do modulu *render\_text*, který na každý obrazový rámec přidá libovolný text (v tomto případě časovou značku) a nakonec jdou data do modulu *dup*, který stream zduplikuje a jeden z nich pošle do zařízení SAGE (modul *sage\_output*) a druhý po konverzi do formátu JPEG a upravení FPS uloží do souborového systému pomocí modulu *filedump*. Nezmíněná pravá část obrázku 3.5, která zajišťuje zvuková data začíná taktéž uzlem pro příjem RTP streamu. Z něj pak audio data postupují do již zmíněného modulu *dup* a po duplikaci se jeden stream posílá na výstup zvukového zařízení (v tomto případě pomocí modulu *jack\_output*) a druhý se ukládá do souboru modulem *wav\_dump*.

TODO..a jsou 2 verze práce - xml pro nahrávání to je dany a pak xml generovány onthefly pro upravu videí

### 3. NÁVRH

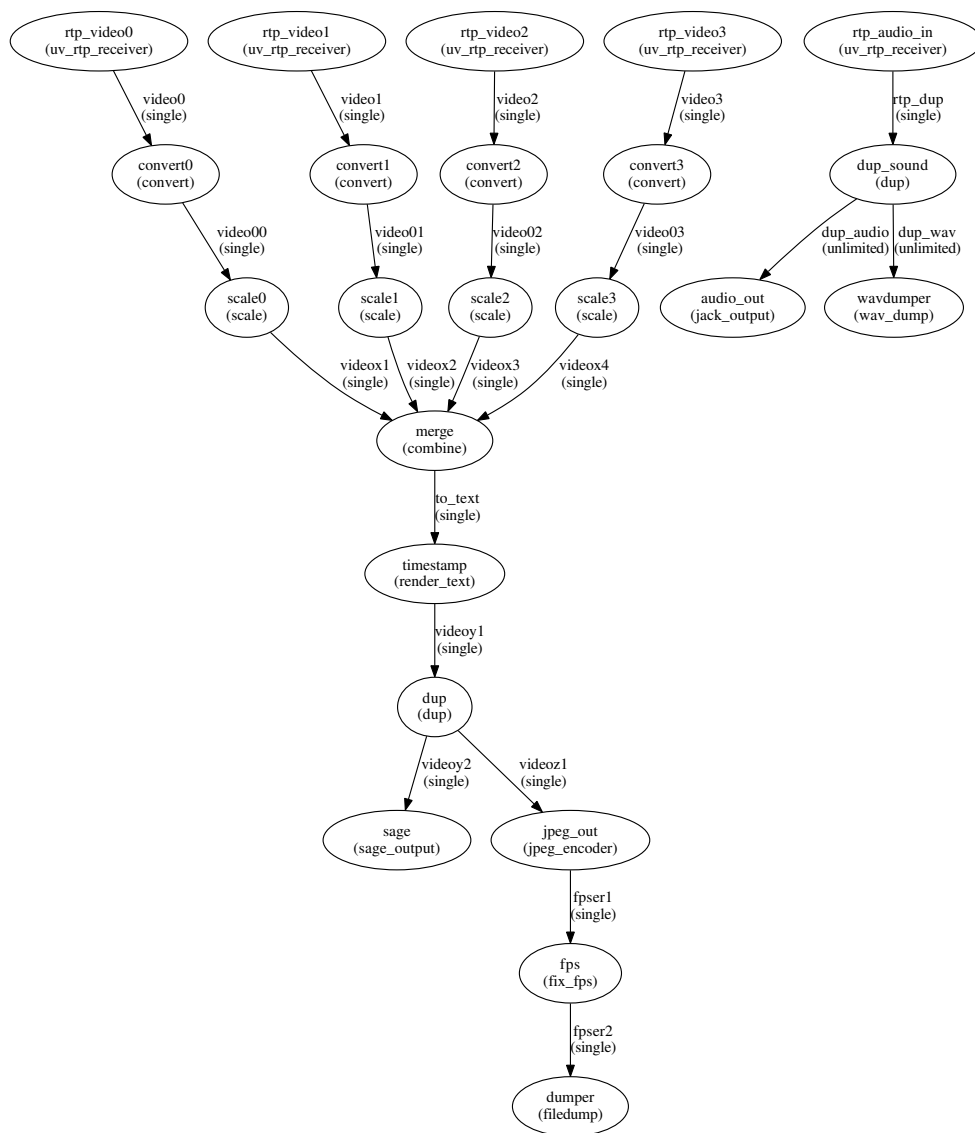


Obrázek 3.3: Původní databázový model



Obrázek 3.4: Libyuri - konfigurace pro záznam v CHUL

### 3. NÁVRH



Obrázek 3.5: Libyuri - konfigurace zobrazení v SAGE a ukládání



## Implementace

### 4.1 Gitlab a CI

### 4.2 Vybrané zajímavé funkce

### 4.3 Instalační a uživatelská příručka

TODO .. mozna rozdelit na instalaci a uziv. prirucku



# Testování

## 5.1 Integrační testování

TODO ... sem dat asi kapku zminku o tetovacim skriptu v CI

## 5.2 Akceptační testování

### 5.2.1 Probíhající usability testování

TODO ... použít něco z NUR - usability testing (první přednáška)



---

## **Závěr**



---

## Literatura

- [1] Solanská, K.: Children Usability Lab - aplikace pro správu laboratoře. 2016.
- [2] Yuri [projects.iim.cz]. 2013, [cit. 2018-03-11]. Dostupné z: <http://projects.iim.cz/yuri>
- [3] Ultragrid [online]. [cit. 2018-03-12]. Dostupné z: <http://www.ultragrid.cz/>
- [4] Ultragrid - wiki [online]. [cit. 2018-03-12]. Dostupné z: <https://github.com/CESNET/UltraGrid/wiki>
- [5] FFmpeg [online]. [cit. 2018-03-13]. Dostupné z: <https://www.ffmpeg.org/about.html>
- [6] GStreamer [online]. [cit. 2018-03-14]. Dostupné z: <https://gstreamer.freedesktop.org/>
- [7] GStreamer - Documentation[online]. [cit. 2018-03-14]. Dostupné z: <https://gstreamer.freedesktop.org/documentation/>
- [8] Open Broadcaster Software [online]. [cit. 2018-03-13]. Dostupné z: <https://obsproject.com>
- [9] Open Broadcaster Software - Documentation [online]. [cit. 2018-03-13]. Dostupné z: <https://obsproject.com/docs/>
- [10] VideoJS - The Player Framework [online]. [cit. 2018-03-12]. Dostupné z: <https://videojs.com/>
- [11] VideoJS - Community plugins [online]. [cit. 2018-03-12]. Dostupné z: <http://videojs.com/plugins/>

- [12] VideoJS - Documentation [online]. [cit. 2018-03-12]. Dostupné z: <http://docs.videojs.com/>
- [13] Fastly - Content delivery and image optimization [online]. [cit. 2018-03-12]. Dostupné z: <https://www.fastly.com/products/web-and-mobile-performance>
- [14] VideoJS - Wiki, browser compatibility [online]. [cit. 2018-03-12]. Dostupné z: <https://github.com/videojs/video.js/wiki>
- [15] Afterglow player [online]. [cit. 2018-03-14]. Dostupné z: <http://afterglowplayer.com/>
- [16] Afterglow player - Documentation [online]. [cit. 2018-03-14]. Dostupné z: <http://docs.afterglowplayer.com/>
- [17] Movie Masher - Website [online]. [cit. 2018-03-14]. Dostupné z: <http://moviemasher.com>
- [18] Movie Masher - moviemasher.js [online]. [cit. 2018-03-14]. Dostupné z: <https://github.com/moviemasher/moviemasher.js>
- [19] Movie Masher - Microsoft Azure [online]. [cit. 2018-03-14]. Dostupné z: <https://azuremarketplace.microsoft.com/en-gb/marketplace/apps/moviemasher.moviemasher>
- [20] Žikovský, P.: MI-NUR - Přednáška č.č - Návrh UI, prototypy. 2015, [cit. 2018-04-09].
- [21] Duncan, D.: GRASP Patterns [online]. 2012, [cit. 2018-03-08]. Dostupné z: <http://www.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/duncan.pdf>
- [22] SAGELab – Síťová multimediální laboratoř [online]. [cit. 2018-03-08]. Dostupné z: <https://sagelab.cesnet.cz/>
- [23] Kočička, P.; Blažek, F.: *Praktická typografie*. Brno: Computer Press, 2004.
- [24] Faistaver, P.: NEC-Controller [software]. 2015. Dostupné z: <http://dx.doi.org/10.6084/m9.figshare.1408922>
- [25] The Linux man-pages project - ldd [online]. 2008, [cit. 2015-04-01]. Dostupné z: <http://man7.org/linux/man-pages/man1/ldd.1.html>
- [26] Padala, P.: Ncurses introduction [online]. 2001, [cit. 2015-04-05]. Dostupné z: <http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/intro.html>



- 
- [27] Mitchell, M.; Oldham, J.; Samuel, A.; aj.: *Advanced Linux Programming*. První vydání, ISBN 07-3571-043-0.
  - [28] NEC Display Solutions: External Control Codes for P.V.X-series [online]. 2013, [cit. 2015-04-02]. Dostupné z: [http://www.necddisplay.com/documents/UserManuals/External\\_Control\\_P.V.X-series.pdf](http://www.necddisplay.com/documents/UserManuals/External_Control_P.V.X-series.pdf)
  - [29] The GNU C Library - dlfcn.h [online]. 2008, [cit. 2015-04-01]. Dostupné z: [https://fossies.org/dox/glibc-2.21/dlfcn\\_2dlfcn\\_8h.html](https://fossies.org/dox/glibc-2.21/dlfcn_2dlfcn_8h.html)
  - [30] NEC Display Solutions: NaViSet Administrator 2<sup>TM</sup>[online]. 2014, [cit. 2015-05-03]. Dostupné z: <http://www.necddisplay.com/support-and-services/naviset-administrator/Overview>
  - [31] Padala, P.: Ncurses programming HOW-TO [online]. 2005, [cit. 2015-04-06]. Dostupné z: <http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/>
  - [32] Úřad pro technickou normalizaci, metrologii a státní zkušebnictví: *ČSN ISO 690 Informace a dokumentace – Pravidla pro bibliografické odkazy a citace informačních zdrojů*. 2011.



## Seznam použitých zkratk

**UI** User Interface

**GUI** Graphical User Interface

**API** Application Programming Interface

**XML** eXtensible Markup Language

**HTML** Hypertext Markup Language



## Obsah přiloženého CD

	readme.txt.....	stručný popis obsahu CD
	exe .....	adresář se spustitelnou formou implementace
	src	
	impl.....	zdrojové kódy implementace
	thesis .....	zdrojová forma práce ve formátu L <sup>A</sup> T <sub>E</sub> X
	text .....	text práce
	thesis.pdf .....	text práce ve formátu PDF
	thesis.ps .....	text práce ve formátu PS