
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# AUDIO A/NEBO VIDEO PRO SHOWROOM

**AUDIO AND/OR VIDEO FOR SHOWROOM**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **DENIS BUKOVINSKÝ**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **PAVEL ZEMČÍK, prof. Dr. Ing.**
**SUPERVISOR**



**BRNO 2020**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

# Zadání bakalářské práce


Student: **Bukovinský Denis**
Program: Informační technologie
Název: **Audio a/nebo video pro showroom**
**Audio and/or Video for Showroom**
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte postupy, možnosti a hardware pro distribuci videa a audia vhodné pro "showroomy" a obecně pro výstavní místnosti.
2. Navrhněte způsob řešení hardware, ideálně s užitím "off the shelf" komponentů, a software, ideálně s využitím open source zdrojů, pro výstavní místnost s několika monitory, na kterých se má zobrazovat video tak, aby vše šlo jednoduše instalovat a spouštět s tím, že audio se bude pouštět "on demand" individuálně do sluchátek. Jako hardware pro přehrávání můžete využít například i mobilní telefon se sluchátky.
3. Navrhněte postup realizace systému v showroom FIT a diskutujte dosažitelné vlastnosti.
4. Implementujte systém a demonstrujte jeho funkčnost na vhodné sadě videosekvencí.
5. Diskutujte dosažené výsledky a možnosti pokračování v práci.

Literatura:

- Dle pokynů vedoucího

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zemčík Pavel, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 5. listopadu 2019

# Abstrakt

Cieľom tejto práce bolo navrhnúť a implementovať systém pre distribúciu videa vo výstavných miestnostiach. V práci sú popísané a analyzované existujúce technológie a produkty. Po vykonaní analýzy bolo navrhnuté technické riešenie, ktoré bolo následne implementované. Implementovaný systém sa skladá z web serveru a používateľského rozhrania. Výsledok tejto práce je plne použiteľný na distribúciu videa v prezentačných miestnostiach.



# Abstract

The main goal of this work was to design and implement a system for video distribution in exhibition rooms. The work describes and analyzes existing technologies and products. After performing the analysis, a technical solution was proposed, which was subsequently implemented. The implemented system consists of a web server and a user interface. The result of this work is fully usable for video distribution in presentation rooms.



# Kľúčové slová

Video distribúcia, Python, Flask, PyQt5, Creative IT ShowRoom, HDMI, Raspberry Pi

# Keywords

Video distribution, Python, Flask, PyQt5, Creative IT ShowRoom, HDMI, Raspberry Pi



# Citácia

BUKOVINSKÝ, Denis. *Audio a/nebo video pro showroom.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Pavel Zemčík, prof. Dr. Ing.

# Audio a/nebo video pro showroom

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Pavla Zemčíka, prof. Dr. Ing. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Denis Bukovinský
14. augusta 2020


## Poďakovanie

Rád by som poďakoval môjmu vedúcemu, Pavlovi Zemčíkovi, prof. Dr. Ing., za jeho cenné rady, pripomienky a trpezlivosť.

# Obsah

# Kapitola 1

# Úvod

Rôzne inštitúcie si vo svojich priestoroch zriaďujú reprezentatívne miestnosti. V týchto miestnostiach sa nachádzajú napríklad ich aktuálne ale aj historické produkty, úspešné projekty alebo vyvinuté technológie. Fakulta informačných technológií Vysokého učenia technického v Brne má vo svojich priestoroch zriadenú miestnosť s názvom *Creative IT ShowRoom.* V tejto miestnosti sú vystavené rôzne exponáty, ktoré slúžia na propagáciu fakulty pre jej návštevníkov. Hostia si počas prehliadky vystavených predmetov môžu pozrieť na monitoroch, ktoré sú umiestnené nad každým jedným, sprievodné video sekvencie. Videá slúžia ako interaktívnejšia forma získania informácií o danom predmete. Použitím vhodných technológií a softvéru je potrebné dostať tieto multimediálne súbory na monitory pri jednotlivých exponátoch.

Cieľom tejto práce je realizovať systém využívajúci ľahko dostupný hardware a software na distribúciu mediálnych súborov do cieľových staníc. V rámci tejto práce je realizovaná len video distribúcia. V cieľových staniciach sa budú nachádzať vstavané počítače, ktoré budú dané súbory prehrávať. Výsledky tejto práce budú umiestnené v Creative IT ShowRoom, ale môžu byť použité kdekoľvek, kde je potrebné distribuovať audiovizuálny alebo aj iný obsah do viacerých zariadení.

Obsahom tejto práce je popis procesu tvorby systému, implementácia a nasadenie systému v showroome. Ku každému stanovišťu s exponátom je priradený monitor, ktorý má prehrávať video. Pôvod video signálu pre každý jeden monitor je práve jeden mikropočítač pripevnený na zadnej strane monitora pripojený do lokálnej siete. Tieto zariadenia slúžia viacerým účelom – okrem prehrávania samotného videa a vysielania zvuku na požiadanie, tak aj ako lokálne úložisko pre multimediálne súbory. Všetky tieto lokálne úložiská komunikujú s hlavným serverom, ktorý vystupuje ako prostredník medzi obsluhou a stanovištami. Obsluha používa hlavný počítač umiestnený v showroome, odkiaľ môže manažovať obsah na jednotlivých zariadeniach. Používateľ má k dispozícii aplikáciu s grafickým užívateľským rozhraním, kde si môže zobrazovať jednotlivé informácie o systéme alebo o konkrétnych zariadeniach. Môže pridávať/odoberať obsah, hromadne pridávať súbory do viacerých zariadení, premenovávať, synchronizovať zmeny, atď.

V nasledujúcej kapitole 2 je prebraný úvod do problematiky video distribúcie. Následne sú popísané jednotlivé požiadavky na takéto systémy. Ako posledné sú porovnané niektoré voľne dostupné technológie pre distribúciu videa v miestnosti. Kapitola 4 pojednáva o procese návrhu riešenia. Postupne sa preberajú ciele aplikácie, architektonický návrh celého systému, rozhranie web serveru, ktorý je srdcom celého systému až po užívateľské rozhranie a databázovú schému. V kapitole 3 sú popísané jednotlivé prostriedky použité na vytvorenie celého systému. Web server bol vytvorený hlavne pomocou frameworku Flask a Peewee.

Grafická aplikácia bola vytvorená pomocou knižnice PyQt5. Všetko bolo vytvorený pomocou jazyka Python. V časti 5 je popísaná implementácia tejto práce. Popísané sú časti, ktoré realizujú hlavnú funkčnosť a sú dôležité pre celkové fungovanie. V predposlednej časti 6 je popísané testovanie web serveru a používateľskej aplikácie.

# Kapitola 2

# Distribúcia videa

Súčasný technologický vývoj spôsobil aj v sektore múzeí a výstavníctva posun od tradičných textových brožúr k audio sprievodcom, až nakoniec dospel k prehrávaniu multimediálneho obsahu. Tento multimediálny obsah môže byť prehrávaný ako na veľkoplošných projekciách, tak aj na menších displejoch pri jednotlivých exponátoch. Nie všetky technológie prenosu videa je možné popísať v tejto práci kvôli maximálnemu možnému rozsahu práce.

## 2.1 Úvod do problematiky

Distribúcia videa, alebo všeobecne multimediálneho obsahu sa dostala z domácností aj do výstavných miestností a múzeí. Múzea hľadali nové možnosti ako priblížiť exponáty verejnosti, ako podať informácie [7]. Táto výzva bola náročná po technologickej ale aj finančnej stránke. No práve inštitúcie, ktoré mali dostatok financií a odvahy investovať do moderných technológií videli nárast v návštevnosti.

Tomuto nárastu pomohli napríklad aj aplikácie ponúkajúce jednotlivé múzeá/výstaviská pre svojich návštevníkov. Je v nich nahraný obsah, ktorý sa zobrazuje napríklad po načítaní QR kódu. Ďalšia technológia, ktorú múzeá využívajú pre zvýšenie atraktivity návštevy múzea je prehrávanie videa pri jednotlivých exponátoch. Či už využitie tzv. rozšírenej reality [12], interaktívneho videa alebo iba obyčajného statického videa.

Všetky tieto video projekcie ale majú jedno spoločné – obsah je treba vedieť jednoducho spravovať, najlepšie na diaľku. V tomto momente vstupuje do hry distribúcia videa, prípadne všeobecne multimediálneho obsahu. Prenos videa sa môže diať niekoľkými spôsobmi, použitím rôznych technológií. Medzi hlavné spôsoby patrí prenos video signálu pomocou kabeláže na to určenej, napríklad HDMI káble. Ďalej je potreba zmieniť využitie IP sietí [16], či už bezdrôtových alebo drátových, ktoré už môžu byť súčasťou priestorov. Poslednou technikou je použitie koaxiálnej siete, ktorá je už ale veľmi technologicky zastaralá.

## 2.2 Požiadavky na distribúciu videa

Niektoré požiadavky vychádzajú priamo od zákazníkov, respektíve od používateľov takýchto systémov. Príkladom môže byť návštevník showroomu má záujem vidieť kvalitné video, ktoré bude prehrávané plynulo [7]. Technologické detaily riešenia, ktoré stojí za prezentáciou, ho zaujímajú už menej. Pracovníci inštitúcie, ktorá používa vo svojich priestoroch tieto moderné technológie potrebujú, aby bolo jednoduché s nimi pracovať (napríklad pridávať nové monitory).

### Škálovateľnosť

Systémy používané na distribuovanie videa by mali byť jednoducho škálovateľné [14] v rôznych smeroch. Jednak by mali pohodlne prijímať nové koncové body, napríklad pridanie nového predmetu do expozície a ku nemu prislúchajúcu multimediálnu prezentáciu. Taktiež by tieto systémy mali zvládať ukladať a spravovať väčšie a väčšie objemy dát, pretože počas používania budú pribúdať nové súbory k už inštalovaným expozíciám alebo k novo vznikajúcim výstavám.

### Spoľahlivosť

Táto vlastnosť najviac ovplyvňuje návštevnícku skúsenosť z výstavy. Napríklad, ak návštevník zažije výpadok prehrávania v dôsledku slabého internetového pripojenia, nebude to pôsobiť želaný efekt. Priebeh premietania videa by mal byť bezproblémový za každých podmienok [14]. Nemalo by prichádzať k mrznutiu, zásekom alebo iným neželaným chybám počas prehrávania.

### Flexibilita

Múzeá a výstavy sa líšia od niektorých iných miest, kde je aplikované distribuované video tým, že sa po určitej dobe menia. Napríklad, múzeá presúvajú svoje exponáty, rušia niektoré staršie výstavy a prichádzajú nové. A tu vznikla požiadavka na flexibilitu [14], aby bolo možné časti infraštruktúry/zariadení jednoducho presmerovať na iné miesto.

### Kvalita videa

Zvýšené požiadavky na kvalitu videa majú inštalácie, ktoré zobrazujú napríklad umenie, u ktorého návštevník ocení jasné farby a vysoké rozlíšenie. Kvalitu videa [14] ovplyvňuje hlavne použitá technológia prenosu. Pri technológiách, kde neprichádza ku kompresii, je plne zachovaná kvalita videa. Naproti tomu, riešenia využívajúce kompresiu z rôznych dôvodov mierne strácajú kvalitu medzi zdrojom videa a cieľovým zariadením.

### Odozva videa

Odozva videa vyjadruje čas medzi odoslaním snímku videa zo zdroja do jeho zobrazenia na zariadení [14]. Tento parameter neovplyvňuje divácku skúsenosť ak sa jedná len o prehrávanie informácii, napríklad v cykle. Tento parameter má význam jedine ak si návštevník môže vyberať prehrávané video sekvencie. Vtedy je dôležitá celková odozva systému, ktorej je aj odozva videa.

## 2.3 Existujúce spôsoby distribúcie videa

V tejto časti budú prezentované niektoré všeobecné verejne dostupné technológie, pretože určitá časť systémov je obostrená rúškom firemného tajomstva.

### HDMI delič/prepínač/matica

Technológia HDMI je veľmi populárna najmä pre svoju schopnosť prenášať video formáty s vysokým rozlíšením bez nutnosti použiť kompresiu. Navyše ponúka prenos audio signálu v oddelenom kanáli. Od verzie 1.4 je dostupný Ethernet kanál s rýchlosťou 100 Mbit/s.

Najnovší štandard v čase písania tejto práce je 2.0 z roku 2017. Poskytuje možnosti prenášať video v rozlíšení až 8K pri obnovovacej frekvencii 30 Hz a s použitím kompresie až 120 Hz. Pri použití technológie HDMI je potreba okrem kabeláže vybrať aj vhodné zariadenia na reprodukciu alebo prepínanie signálu[19]. Na tieto účely sa používajú viaceré zariadenia s rôznymi funkciami.

Tzv. *delič (angl. splitter)* [19] je najjednoduchšie z dostupných zariadení a vykonáva jednoduchú funkciu. Používa sa v prípadoch ak je potrebné použiť signál z jedného zdroja a poslať ho do niekoľkých výstupov simultánne. V podstate len kopíruje vstupný signál na pripojené výstupy.

Tzv. *prepínač (angl. switch)* [19] je zariadenie, ktoré umožňuje používateľovi pripojiť viacero vstupov na jeden výstup. Z takto pripojených vstupov si môže používateľ vyberať, ktorý konkrétny si praje zobraziť na pripojenom výstupnom zariadení. Avšak, je možné si vybrať zobrazenie len jedného vstupu v jednom okamžiku. Takéto zariadenia väčšinou neposkytujú možnosť pozerať viacero vstupov na jednom výstupe.

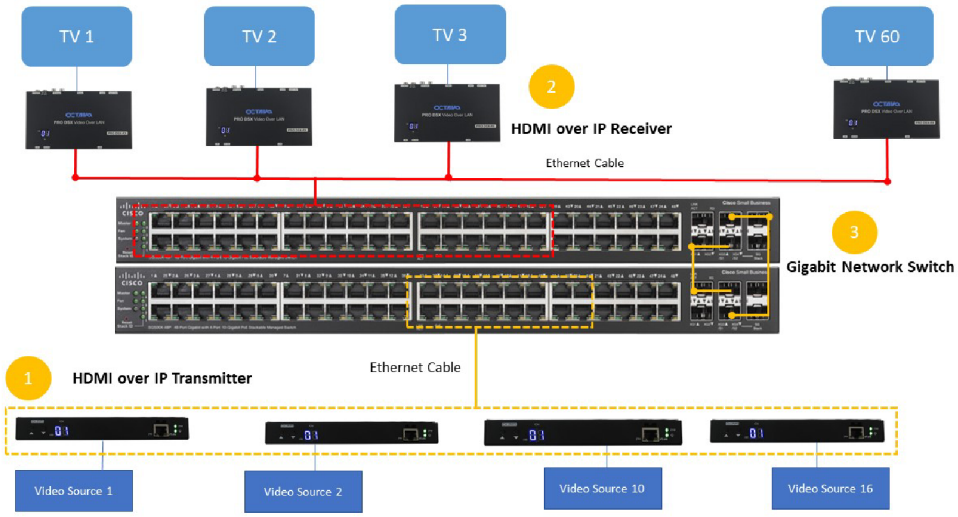
Tzv. *matica (angl. matrix)* [19] je najzložitejšie zariadenie z vyššie menovaných. S predchádzajúcimi má spoločných niekoľko vlastností – dokáže kopírovať jeden vstup na viaceré výstupy, dokáže prepínať viacero vstupov na jeden výstup. Čo ju však odlišuje od predošlých technológií je fakt, že dokáže tieto možnosti robiť podľa preddefinovaných vzorov. Môže si pamätať napríklad vzory použiteľné pre rôzne situácie. Niektoré zariadenia ponúkajú aj diaľkové ovládanie pre prácu s týmito prístrojmi.

### HDMI (video) cez IP

Už názov tejto technológie naznačuje, že sa určitým spôsobom bude jednať o kombináciu spomínanej HDMI technológie spolu s IP sieťami. Anglický názov je *HDMI over IP* [16]. V distribúcii videa pomocou HDMI dominuje HDMI matica, ktorá je srdcom celého systému, pretože ovláda a smeruje toky signálu. HDMI cez IP však túto maticu eliminovalo a namiesto matice používa ethernetový sieťový prepínač (angl. Ethernet Network Switch). Zdroje sa do takejto siete pripájajú pomocou špeciálnych zariadení nazývaných *vysielače* (angl. transmitter). Tieto komponenty majú okrem vstupu na napájacie napätie minimálne jeden HDMI vstup a jeden sieťový výstup. Vysielače vykonávajú kompresiu vstupného video signálu pred tým ako je video odoslané pomocou paketov po sieti, aby neprišlo k preťaženiu siete [16]. Algoritmy dostupné pre kompresiu videa už sú na takej úrovni, že dokážu priniesť vizuálne rovnaký zážitok ako keby bol prenášaný rovnaký obraz bez kompresie. Na prijímaní vysielaného signálu sa podieľajú zariadenia nazývané `prijímače` (angl. recievers). Tieto zariadenia prijímajú zo siete pakety obsahujúce komprimované video snímky a z nich potom vytvárajú výstupný signál. Zapojenie prijímačov a vysielačov do ethernetovej siete je možné vidieť na obrázku 2.1. Sieť, ktorá sa nachádza medzi zariadeniami funguje ako teoreticky nekonečne rozšíriteľná HDMI matica, pretože umožňuje aby niekoľko prijímačov prijímalo jeden signál, alebo aby bolo možné vytvoriť prednastavené vzory prijímania a vysielania signálu (ako to bolo pri HDMI matici).

### HDBaseT

*HDBaseT* je štandard popisujúci hlavne distribúciu zvuku a videa vo vysokom rozlíšení. Táto technológia môže používať rôzne typy prenosových médií, najčastejšie však obyčajnú sieťovú kabeláž Cat6 a lepšiu; ale aj optické káble, ktoré majú lepší dosah. Podobnosť s HDMI cez IP je v tom, že obe technológie používajú už zavedenú lokálnu internetovú sieť, avšak HDBaseT [24] využíva prenosové médium inak ako Ethernet. Dátová modulá-




Obr. 2.1: Obrázok znázorňuje zapojenie prijímačov a vysielačov do ethernetovej siete [14]. Žltou farbou je znázornené zapojenie vysielačov do ethernetového prepínača, typicky pomocou Cat 5e/6 kabeláže. Červenou farbou je zobrazené pripojenie prijímačov do ethernetového prijímača, aby mohli prijímať pakety obsahujúce snímky zo vstupného videa vysielaného vysielačmi. Číslami pri názvoch video zdrojov (Video Source X) je vyjadrená škálovateľnosť v smere pridávania nových video zdrojov. Rovnako tak pri číslach prijímačov (TV X) je vyjadrená škálovateľnosť v smere pridávania nových koncových staníc.

cia, ktorú HDBaseT používa je hustejšia v porovnaní s Ethernetom. Využíva viac úrovní pulznej amplitúdovej modulácie (angl. Pulse Amplitude Modulation – PAM) ako Ethernet, 16 PAM úrovní na všetkých štyroch dvojiciach versus 5 PAM úrovní na dvoch dvojiciach káblu. Zvolený počet PAM umožňuje väčšiu dátovú priepustnosť ako Ethernet pri použití rovnakého fyzického nosiča. HDBaseT využíva proprietárnu technológiu T-Packet na transport dát, narozdiel od IP paketov, ktoré prinášajú príliš veľké oneskorenie pri prenášaní vysoko kvalitného video obsahu. T-packet pridáva menej ako 10 mikrosekúnd na 100 m nosiča.

## 2.4 Existujúce produkty v distribúcii videa

### Blustream PLA88ARC-V2 a HEX100ARC-RX

Blustream [2] je austrálsky výrobca zariadení pre HDMI, HDMI over IP a HDBaseT distribúciu videa. Ako príklad je možné uviesť aplikáciu produktov PLA88ARC-V2 a HEX100ARC-RX, ktorý je možno vidieť na obrázku 2.2.

Spomenuté produkty umožňujú prenos videa do vzdialenosti až 100 m [13]. Je možné pomocou PLA88ARC-V2 HDBaseT matice 8x8 je možné vyberať z niekoľkých zdrojov video signálu a tie posielať nezávisle do prijímačov HEX100ARC-RX. Výhodou týchto produktov je, že používajú jednu spoločnú Cat6 sieť. Uvedená matica má aj niekoľko ďalších vlastností, ako je podpora pre HD audio s podporou pre Dolby Atmos, posielanie kontrolných dát a


Obr. 2.2: Na obrázku je jedna zo strán HEX100ARC-RX prijímača. Je možné vidieť RS-232, IR vstup/výstup, vstup HDBaseT RJ45 a napájanie[4].

automatizačných inštrukcií atď. Prijímače majú okrem RJ45 vstupu a HDMI výstupu ešte audio výstup, RS-232, ethernet konektory a IR vstup/výstup.

Tieto konkrétne komponenty sú použité vo video distribučnom systéme pre jachtu O'PTASIA [13], kde sa nachádza až 5 kino systémov a 23 integrovaných TV displejov. V tomto špecifickom prípade bolo použitých až 5 PLA88ARC-V2 matíc a 26 HEX100ARC-RX prijímačov. Cena jedného prijímača je 308 eur[1] a matica stojí 6 688 eur[2]. Po spočítaní nákladov na tento hardvér bez potrebnej kabeláže sa cena vyšplhá na sumu 41 460 eur, z toho matice stoja 33 640 eur a prijímače 8 010 eur. Ak sa cena prepočíta na jeden displej tak sa dostaneme na sumu 1 594 eur.

### HDMI matica KD-HD8x8Lite zapojená v domácej video distribúcii

Spoločnosť KeyDigital [3] je medzinárodná firma zaoberajúca sa dizajnom a vývojom video distribučných systémov. Ich produkty zahŕňajú rôzne technológie, od klasických HDMI distribúcií až po AV cez IP.

V prípadovej štúdii *Key Digital Brings HDMI Home with HDMI Matrix Switching in British Columbia* je popísané použitie HDMI matice [1] KD-HD8x8Lite, ktorú je možné vidieť na obrázku 2.3. Táto matica poskytuje 8 vstupov a 8 výstupov typu HDMI, RS-232 konektor a IR vstup/výstup. Ako vstupné zariadenia v tomto prípade figurujú tri satelitné prijímače a Apple TV. Výstupné zariadenia sú jeden projektor a sedem monitorov. Cena takejto novej matice je približne 4 857 eur [7], čo pri prepočte na jeden displej je približne 606 eur.

[1] https://www.futureshop.co.uk/blustream-hex100arc-rx-hdbaset-receiver-100m-4k-up-to-70m-bi-directional-ir

[2] https://www.futureshop.co.uk/blustream-pla88arc-v2-8x8-hdbaset-matrix-100m-4k-upto-70m-bi-directional-ir-poh-poe

[4] https://www.blustream.co.uk/hex100arc-rx?rq=hex100

[6] https://www.amazon.com/Key-Digital-KD-HD8x8Lite-Switcher-Receivers/dp/B010QCJJDA

[7] https://www.amazon.com/Key-Digital-KD-HD8x8Lite-Switcher-Receivers/dp/B010QCJJDA


Obr. 2.3: Na obrázku je možné vidieť KD-HD8x8Lite maticu, ktorá má 8 vstupných, 8 výstupných HDMI portov, jeden RS-232 konektor, IR vstup/výstup a možnosť rozšíriť audio[6].

# Kapitola 3

# Nástroje používané vo video distribúcii

Kapitola sa zaoberá vybranými nástrojmi používanými pri distribúcii videa. Z dôvodu maximálneho možného rozsahu práce nie sú uvedené všetky nástroje ktoré súvisia s distribúciou videa.

## 3.1 Python

Programovací jazyk *Python* [17] je objektovo orientovaný a štruktúrovaný jazyk podporujúci funkčné a aspektovo orientované programovanie. Mnoho ďalších aspektov podporuje pomocou rozšírení. Python 3.0 bol vydaný v roku 2008 a navrhnutý tak, aby bol značne rozšíriteľný. Vďaka modularite sa teda stal veľmi obľúbeným prostriedkom na pridávanie programovateľných rozhraní do existujúcich aplikácií. Oceniteľná je taktiež jeho dostupnosť pre všetky platformy – je bez problémov spustiteľný na Microsoft Windows, operačnom systéme Macintosh a všetkých distribúciách Linuxu. Programy sú takto ľahko prenosné medzi rôznymi platformami. V jazyku Python sa na štruktúrovanie kódu využíva medzera (napríklad v porovnaní so zátvorkami v jazyku C) a odsadenie, ktoré značí, kde bloky kódu začínajú a končia. Projekt *Raspberry Pi*[1], počítač o veľkosti kreditnej karty, prijal Python ako svoj hlavný programovací jazyk pre používateľov.

### Flask

*Flask* [9] je webový mikro framework, distribuovaný v podobe Python modulu, ktorý umožňuje tvorbu webových aplikácií. Má malé, jednoducho rozšíriteľné jadro, teda ide o mikro framework ktorý neobsahuje *ORM (Object Relational Manager)*. Je založený na *Workzeg WSGI toolkit* a *Jinja2 template engine*, ktorý ale v tejto bakalárskej práci nie je použitý. Jednoduchý príklad Flask koncového bodu, ktorý odošle HTML stránku s nadpisom *Hello World* vyzerá nasledovne:

[1] https://www.raspberrypi.org/

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
   return '<html><h1>Hello, World!</h1></html>'
```

Na tomto jednoduchom príklad je možné vidieť, že Flask je mirko framework, kompaktný a jednoduchý.

*WSGI (Web Server Gateway Interface)* [8] je špecifikácia spoločného rozhrania medzi webovými aplikáciami a webovými servermi a používa sa ako štandard pre vývoj Python webových aplikácií.

### PyQt5

*PyQt* [27] spája framework Qt verzie 5.0 z jazyka C++ s interpretovaným jazykom Python. PyQt5 obsahuje najrôznejšie abstrakcie, napríklad sieťových socketov, vlákien, SQL, XML, vlákien a tiež obsahuje rozsiahlu kolekciu grafických používateľských widgetov. Jeho hlavnou výhodou je možnosť použitia na všetkých populárnych operačných systémoch ako je Windows, Unix, alebo Mac OS. Celé implementácia Python rozhrania k Qt C++ knižniciam je rozdelená do niekoľkých modulov:

- QtCore
- QtGui
- QtWidets
- a mnoho ďalších ...

Je ich o dosť viacej, ale tieto sú v práci používané najčastejšie.

Modul *QtCore* [6] ponúka hlavnú funkcionalitu pre tvorbu GUI. Je možné ho použiť pre prácu s časom, súbormi, adresami URL, vláknami, a tak ďalej. *QtGui* ponúka triedy pre správnu integráciu systému okien, spracovanie udalostí, základné zobrazovanie, a tak ďalej. *QtWidgets* obsahuje klasické prvky pre tvorbu GUI aplikácií v štýle desktopových používateľských rozhraní.

## 3.2 HTTP

*Protokol HTTP* (Hypertext Transfer Protocol – Hypertextový prenosový protokol) [10] je bezstavový aplikačný protokol pre distribuované, spolupracujúce hypertextové informačné systémy. Je základom akejkoľvek výmeny dát na webe a umožňuje získavať zdroje ako napríklad HTML dokumenty. Pracuje pomocou požiadavok a odpovedí v klient-server modeli – klient (napríklad webový prehliadač) vytvorí HTTP požiadavok a server na ňu odpovie zaslaním výsledného dokumentu v tele správy a/alebo stavovou informáciou (kódom).

HTML dokumenty [26] sú identifikované pomocou *URL* (Uniform Resource Locators – URL), ktoré určuje miesto uloženia zdroja a pomocou *URI* (Uniform Resource Identifier – URI) schém, ktoré identifikujú konkrétny zdroj.

### HTTP Authentication

*HTTP Autentifikácia* [10] slúži na overenie identity klienta. HTTP ponúka mnoho rôznych možností autentifikácie, napríklad bežné prístupové overenie (angl. Basic access authentication) alebo hashované prístupové overenie (angl. Digest access authentication). Server pri poskytovaní obsahu využíva mechanizmus *výzva-odpoveď*. Princíp mechanizmu spočíva v tom, že ak klient od servera požaduje obsah, ktorý je určený len konkrétnym používateľom, vyzve server klienta na zaslanie identifikačných údajov. Server po prijatí týchto údajov vykoná overenie a zašle odpoveď klientovi.

### Formát HTTP správy

*HTTP správa* [10] predstavuje komunikačný prostriedok medzi klientom a serverom. Klient posiela požiadavku serveru a server na ňu odpovie odpoveďou.

Prvý riadok požiadavky popisujúci metódu, url a použitý protokol musí byť od hlavičiek oddelený párom neviditeľných znakov `<CR><LF>`, kde `<CR>` je ASCII znak `Carraige Return` a `<LF>` `Line Feed`. Každá jedna hlavička je práve na jednom riadku, ktorý je ukončený znovú dvojicou znakov <CR><LF>. Na konci segmentu s hlavičkami sa nachádza znovu dvojica <CR><LF> a za ňou už nasleduje telo správy. HTTP definuje metódy popisujúce akcie, ktoré sa majú vykonať s daným zdrojom. V názvoch metód sa rozlišujú veľké a malé znaky, narozdiel od názvov polí v HTTP hlavičke. Príklady metód definovaných HTTP sú `GET, POST, PUT` a `DELETE` [23]. Príklad požiadavky aj s telom správy je nasledovný:

```
GET /doc HTTP/1.1
Host: www.example.org

docId=123&timStamp=1596555248
```

Server odpovedá [23] na požiadavku od klienta správou, ktorá má podobnú štruktúru ako požiadavka. Na prvom riadku sa nachádza protokol a jeho verzia, za ním nasleduje číselný kód spolu s reťazcom znakov, ktorý reprezentujú stav spracovania požiadavky. Číselné kódy majú svoj presne daný význam sa sú rozdelené podľa prvej číslice do niekoľkých skupín. Napríklad číselný kód 4xx reprezentuje chybu na strane klienta, číselný kód 5xx reprezentuje, že nastala chyba na strane serveru. Príklad odpovede je nasledovný:

```
HTTP/1.1 200 OK
Date: Sun, 02 Aug 2020 05:25:12 GMT
Content-Length: 27
Content-Type: text/html

<h1>Welcome to my Page</h1>
```

### HTTPS

*Protokol HTTPS* (Hypertext Transfer Protocol Secure) [23] rozširuje protokol HTTP. Jeho úlohou je zabezpečiť bezpečnú komunikáciu na internete. Komunikačný protokol je šifrovaný pomocou *TLS* (Transport Layer Security), ktorý nahradil pôvodný *SSL* (Secure Socket

Layer) – protokol sa teda nazýva aj *HTTP over TLS*, alebo *HTTP over SSL*. Význam protokolu HTTPS rastie v prípade nezabezpečených sietí, v ktorých sa môžu vyskytovať útoky na komunikáciu.

Cieľmi HTTPS [23] sú autentifikácia webovej stránky, na ktorú sa pristupuje; ochrana súkromia a zabezpečenie integrity vymieňaných dát počas komunikácie. Protokol HTTPS vyžaduje na strane serveru podpísaný digitálny certifikát. Certifikáty sú podpísané dôveryhodnou certifikačnou autoritou, aby sa zabezpečila ich pravosť. Prehliadače alebo iní klienti overujúci pravosť certifikátov majú už prednastavený zoznam dôveryhodných autorít.

### REST API

*REST* (Representational State Transfer) [21] je spôsob architektonického návrhu rozhraní aplikácii komunikujúcich prostredníctvom HTTP protokolu. Definuje súbor obmedzení pri vytváraní webových služieb. Takéto webové služby sa nazývajú *RESTful webové služby* a poskytujú jednoduchú vzájomnú výmenu informácií medzi počítačovými systémami na internete. Taktiež umožňujú vykonávať operácie nad textovou reprezentáciou webových zdrojov.

## 3.3 Prenos súborov

Pre prenos súborov existuje nespočetne veľa rôznych aplikácií, frameworkov, protokolov atď. Z dôvodu rozsahu práce sú uvedené len niektoré.

### Rsync

*Rsync* [20] je flexibilná a rýchla štandardná linuxová utilita pre kopírovanie súborov. Vyznačuje sa svojou vysokou efektivitou, čo v praxi znamená, že prenáša malé objemy dát pri synchronizácii. Našla svoje použitie napríklad pri synchronizácii dát medzi lokálnym diskom a diskom uloženým v sieti. Táto utilita sa preslávila svojím delta prenosovým algoritmom, ktorého hlavným cieľom je znížiť objem prenášaných dát. Rsync posiela len zmeny medzi zdrojovým súborom a cieľovým súborom. Najčastejšími a najrozšírenejšími použitiami sú zálohy alebo zrkadlenie dát (angl. *mirroring*).

Súbory, ktoré sú predané ako argumenty sú porovnávané na základe veľkosti a dátumu poslednej zmeny. Porovnávanie podľa iných kritérií je možné vynútiť pomocou zadania vhodných parametrov.

Pre pripojenie používa Rsync [20] dva rôzne spôsoby – rsync démon používajúci priamo TCP alebo vzdialený shellovský program ako je `SSH`[2] alebo `RSH`[3]. Pre použitie vzdialeného shellu je potrebné špecifikovať príkaz, ktorý sa na to má použiť. Na tento účel slúži parameter príkazovej riadky `--rsh=COMMAND` [20]. Ďalšou možnosťou pre vytvorenie SSH spojenia je vložiť ako parameter celý príkaz. Pre ilustráciu príklad, ktorý je aj následne použitý v tejto práci:

```
rsync -zvh -e "ssh -oStrictHostKeyChecking=no -i key"fileA.mp4 test@rpi1:/home
```

Tento príkaz sa pokúsi synchronizovať súbor `fileA.mp4` s rovnakým súborom v zaria-

[2] https://www.dsl.cz/jak-na-to/jak-na-ssh
[3] https://docs.microsoft.com/cs-cz/windows-server/administration/windows-commands/rsh

dení `rpi1` v adresári `/home`, na prihlásenie použije `SSH` protokol s privátnym kľúčom `key` a prihlási sa pod užívateľom `test`.

### SCP

Utilita *SCP* [4] je štandardnou súčasťou väčšiny Linuxových distribúcií. Jej účelom je prenášať súbory z lokálneho disku na vzdialený disk, prípadne prenos medzi dvoma vzdialenými stanicami. Posiela vždy celé súbory. Prenáša dáta pomocou SSH protokolu, zároveň tento protokol používa aj na zabezpečenie autenticity a diskrétnosť prenášaných dát.

### SFTP

*Secure File Transfer Protocol* [25] alebo SFTP je protokol umožňujúci prenos, manažment a prístup k dátam vo vzdialených zariadeniach. V porovnaní so SCP SFTP klient umožňuje napríklad vrátenie sa k prerušenému prenosu súborov alebo zobrazenie zoznamu súborov na vzdialenom disku. Server je väčšinou súčasťou SSH implementácie na konkrétnom zariadení. Tento protokol je menej závislý na platforme ako SCP, ktoré je častejšie implementované pre Unix.

## 3.4 Relačná databáza PostgreSQL

*Databázový systém* je systém slúžiaci na evidovanie a spracovanie údajov. Práca s ním nie je zložitá, ide o bežné operácie, ktoré možno zhrnúť do niekoľkých oblastí:

- vkladanie údajov do databázy,
- úpravy – mazanie, prepisovanie,
- triedenie – usporiadanie podľa číselných hodnôt, abecedy, ...
- výber, prehľadávanie (filtrovanie údajov),
- štatistické operácie a iné.

Najdôležitejším prvkom sú *údaje*. Z nich sa získavajú *informácie*. Údaje sa ukladajú do tabuliek, ktoré obsahujú *polia* (stĺpce) a *záznamy* (riadky) [5].

*Relačná databáza* je typ databázy, ktorý uchováva a poskytuje prístup k dátovým bodom, ktoré spolu súvisia. Je založený na intuitívnom a priamom reprezentovaní dát v tabuľkách – teda na relačnom modele. Každý riadok v tabuľke má záznam s jedinečným identifikátorom nazývaným *kľúč*. Stĺpce obsahujú atribúty údajov a k tomu má zvyčajne každý záznam hodnotu pre každý atribút, čo uľahčí vytvorenie vzťahov medzi údajovými bodmi. *Relačný model* [11] značí oddelenie logickej dátovej štruktúry (dátové tabuľky, pohľady, indexy,...) od štruktúr fyzického úložiska. Táto skutočnosť umožní spravovať fyzické ukladanie údajov bez ovplyvnenia prístupu k daným údajom ako logickej štruktúre.

*Jazyk SQL* (Structured Query Language – štrukturovaný dotazovací jazyk) [5] je špecializovaný programovací jazyk slúžiaci na manipuláciu, správu a organizovanie dát uložených v databáze. Umožňuje získať odpovede na rôzne komplikované dotazy a zároveň slúži aj na definíciu dát, plnenie tabuľky údajmi, definovanie štruktúry tabuľky a vzťahov medzi dátami. Zaisťuje taktiež riadenie prístupu k dátam a zdielané využívanie dát pre viac užívateľov pristupujúcich k dátam súčasne. Je neprocedurálny a pracuje s relačnými databázami.

V dnešnej dobe existuje mnoho databázových systémov vykonávajúcich operácie spomenuté vyššie. K najznámejším a najpoužívanejším patria [11] `Oracle, MS SQL, MS Access, My SQL,` alebo `PostgreSQL`. V tejto práci sa používa spomínaný PostgreSQL.

*PostgreSQL* [11] je open source, objektovo-relačný databázový systém využívajúci a rozširujúci jazyk *SQL* v kombinácii s mnohými funkciami, ktoré bezpečne ukladajú a upravujú údaje. PostgreSQL zároveň umožňuje definovať vlastné dátové typy alebo vytvoriť vlastné funkcie. Snaží sa o pomoc pri vytváraní aplikácie, chránenie integrity údajov a vytvára prostredie odolné voči chybám.

### Peewee

*Peewee* [15]je jednoduchý ale expresívny ORM (angl. Object Relation Mapping) modul. ORM pracuje ako vrstva medzi relačnou databázou, ktorá dáta ukladá dáta ako relácie a objektovo-orientovanými jazykmi. Programy väčšinou pristupujú k dátam v databáze pomocou objektov, ktorých zmeny sú následne reflektované v dátach uložený v databáze. Výhodou je redukcia zbytočného kódu a jednoduchosť prístupu. Nevýhodou je niekedy nevhodný dizajn databáz kvôli spoliehaniu sa na framework pri vytváraní databázy. V ďalšom príklade je možné vidieť objekt mapovaný na tabuľku `user`:

```
class User(MyBaseModel):
   id = PrimaryKeyField()
   username = CharField(unique=True, null=False)
   password = CharField(null=False)
```

## 3.5 Raspberry Pi


Obr. 3.1: Raspberry Pi model 3B+

*Raspberry Pi* [22] je jednodoskový počítač, ktorého veľkosť zodpovedá približne veľkosti kreditnej karty. Vývoj a predaj tohto minipočítaču má na starosti nadácia Raspberry Pi sídliaca vo Veľkej Británii. Hlavným účelom Raspberry Pi je rozšírenie povedomia o počítačovej vede a uľahčenie jej výuky. Vývoj takýchto mini/mikropočítačov umožnilo zhusťovanie a zmenšovanie integrovaných obvodov. Dizajn spočívajúci v umiestnení celého systému na

jeden obvod ovplyvňuje hlavne jeho cenu a tým pádom aj dostupnosť na trhu. Poskytuje zaujímavé parametre čo sa týka výkonu, až 4 jadrový procesor, až 4GB operačnej pamäte alebo rôzne typy grafických čipov (napríklad čip VideoCore VI inštalovaný v modeli RPi 4 B+). Zariadenie síce nepodporuje pridávanie ďalších kariet (zvukových, grafických, a iných,..), za to však disponuje množstvom analógových/digitálnych vstupov/výstupov.

Pre pripojenie k internetu používa Gigabit Ethernet RJ45 rozhranie s rýchlosťou do 300 Mb/s z dôvodu použitého USB 2.0 spojenia so systémom. Zahrnuje tiež bezdrôtové pripojenie WiFi, vo verziách Pi Zero a Pi 3 je to modul 802.11n o rýchlosti 150 Mbit/b [18]. Verzia Pi 3B+, na obrázku 3.1, obsahuje WiFi modul 802.11b/g/n/ac. Zariadenia obsahujú aj Bluetooth modul pre bezdrôtové pripojenie k bezdrôtovým zariadeniam – model RPi 3B+ obsahuje Bluetooth 4.2, najnovší model RPi 5 už obsahuje Bluetooth 5.0.

Pre pripojenie výstupných video zariadení sú k dispozícii rôzne rozhrania, od DSI (MIPI), ktoré je dostupné na všetkých modeloch rovnako ako HDMI, až po rôzne konektory typu RCA alebo TRRS. Video rozhranie podporuje rozlíšenie až 4K pri rýchlosti 60 FPS pri použití HDMI výstupov[18]. Všetky modely disponujú jedným HDMI výstupom, okrem modelu Raspberry Pi 4B, ktorý obsahuje až 2.

# Kapitola 4

# Analýza existujúcich riešení a návrh riešenia

V tejto kapitole sa pojednáva o procese návrhu riešenia distribúcie videa v miestnostiach, napríklad v showroome FIT. V časti 4 je rozobratí postupný návrh riešenia, postupne je rozobratí spôsob uloženia videí, rola mikropočítačov Raspberry Pi v systéme ako celku, potom aj samotný spôsob distribúcie/transportu videí a use-case model. V časti 4.1 sú analyzované technológie a produkty dostupné na distribúciu videa a základné ciele aplikácie. Ďalej je popísaná architektúra aplikácie v časti 4.3, kde je aj schéma navrhnutého systému.

## 4.1 Analýza produktov a ciele aplikácie

Nasledujúca časť porovnáva a analyzuje technológie z kapitoly 2. Technológie sú analyzované ako po technickej tak aj po finančnej stránke. Nie všetky dostupné produkty a technológie sú popísané v tejto práci, kvôli obmedzeniu maximálneho rozsahu práce.

### Analýza produktov a technológií

Táto analýza vychádza z popisu technológií v kapitole 2. Všetky technológie popísané v spomínanej kapitole sú reálne dostupné na trhu od množstva firiem. Rôzne produkty od rôznych firiem sa môžu mierne líšiť v technických parametroch avšak majú jeden spoločný cieľ – priniesť divákovi/návštevníkovi/pozorovateľovi dokonalý zážitok z video prehrávania.

Prvá technológia je „obyčajné“ HDMI. Jeho výhodou je video signál prenášaný bez akejkoľvek kompresie a preto dokáže vždy priniesť ten najkvalitnejší obraz. Nevýhodou je len potreba vytvorenia novej infraštruktúry v podobe HDMI kabeláže rozvedenej na potrebné miesta. S týmto je spojené aj navýšenie celkovej ceny za dielo. Výhoda je, že všetky dátové toky fungujú v oddelených vodičoch, kde sa nenachádza nič, čo by mohlo znižovať prenosové pásmo. Ďalšia výhoda je priame zapojenie HDMI do monitora, nie je potreba žiadny prijímač.

Druhá technológia je HDMI cez IP. Ide o technológiu, ktorá z časti používa HDMI kabeláž a z časti využíva IP siete. Veľkou výhodou je možnosť využiť už existujúcu Cat6 sieť. Nevýhodou je ale potreba každé koncové zariadenie opatriť špeciálnym zariadením, ktoré bude dekódovať video posielané po sieti. Výhodou je možnosť použitia PoE (angl. *Power over Ethernet*) technológie, ktorá dokáže napájať zariadenia pomocou ethernetového vodiča. Ďalšou výhodou je jednoduchá rozšíriteľnosť o ďalšie zariadenia.

| Technológia | Cena/ zariadenie | Infraštruktúra | Kompresia | Koncové zariadenia |
|---|---|---|---|---|
| HDMI | - | nová | nie | nie |
| HDMI cez IP | od 600 eur | existujúca | áno | áno |
| HDBaseT | od 1500 eur | existujúca | nie | áno |

Tabuľka 4.1: Porovnanie niektorých parametrov popísaných technológií. *Infraštruktúra* značí pravdepodobnosť nutnosti budovať novú alebo možnosť použiť nejakú existujúcu. *Cena za zariadenie* vychádza z popísaných produktov z kapitole 2. *Koncové zariadenia* značia skutočnosť, či je treba ešte ku koncovým zariadeniam dokúpiť špeciálny HW alebo je možné pripojiť kabeláž priamo do zariadenia.

Tretia technológia je HDBaseT. Jedná sa tiež o technológiu, ktorá využíva v časti distribúcie HDMI kabeláž a v druhej časti ethernetovú kabeláž. HDBaseT má podobnú nevýhodu ako HDMI cez IP, je potreba ku koncovým zariadeniam špeciálne komponenty, ktoré budú dekódovať video signál posielaný po sieti. Výhodou oproti HDMI cez IP je fakt, že HDBaseT využíva vlastný formát paketov. Preto dokáže dosiahnuť vyššiu priepustnosť siete – nemusí sa spoliehať na kompresiu a môže dodávať video signál bez kompresie. Výhodou taktiež je využívanie už existujúcej Cat6 kabeláže.

## 4.2 Ciele práce

Cieľom tejto práce je priniesť riešenie pre distribúciu videa použiteľné predovšetkým v priestoroch Creative IT ShowRoom na Fakulte Informačných Technológií VUT. V spomínanom priestore budú výsledky tejto práce demonštrované a zároveň aj aktívne využívané pri návštevách hostí alebo dňoch otvorených dverí. Hlavnými požiadavkami sú:

- **Jednoduchá inštalácia** – systém by nemal vyžadovať zdĺhavú inštaláciu množstva softvéru, napríklad pri rozširovaní systému.
- **Nenáročná obsluha** – obsluhu tohto systému by mal zvládnuť človek bez IT vzdelania a dlhého školenia o prevádzke.
- **Spoľahlivosť** – nemalo by dochádzať k výpadkom prehrávania videa, mrznutiu obrazu, atď.
- **Rýchle zapnutie** – celý systém by mal byť schopný prejsť zo stavu „vypnutý“ do stavu „prehrávanie“ čo najrýchlejšie, preto je dobré aby boli jednotlivé koncové stanice nezávislé.
- **Nízka cena** – cena súvisí hlavne s použitým HW, vybrať HW s vhodným pomerom cena/výkon, ideálne bežne dostupný.
- **Mobilita** – možnosť ľahko prekonfigurovať, prípadne premiestniť časti systému bez nutnosti zásahov do ostatných častí systému.
- **Dostupnosť komponentov** – použiť komponenty dostupné v obyčajných obchodoch, väčšinou skladom.
- **Prenos dát cez Wi-Fi** – všetky zriadenia budú spojené len pomocou Wi-Fi AP.

## 4.3 Návrh aplikácie

Pri návrhu architektúry aplikácie bolo vychádzané z analýzy existujúcich technológií a konkrétnych produktov na trhu. Následne sa do procesu návrhu začlenili aj možnosti hardvérového vybavenia miestnosti Creative IT ShowRoom na Fakulte Informačných Technológií VUT. V spomínanom priestore budú výsledky tejto práce demonštrované a zároveň aj aktívne využívané pri návštevách hostí alebo dňoch otvorených dverí. Navrhnutý systém by mal spĺňať všetky stanovené ciele.

V miestnosti sú panely, v ktorých sú vstavané Full HD monitory určené pre prehrávanie videí o exponátoch umiestnených pod nimi. Každý monitor má k dispozícii práve jedno zariadenie Raspberry Pi model 3B+, ktoré bude umiestnené na zadnej strane monitora aby nebolo voľným okom vidieť. Tieto vstavané zariadenia budú slúžiť ako video prehrávač spojený pomocou HDMI kábla s monitorom. V miestnosti sa nachádza PC zapojený do rovnakej siete ako všetky Raspberry Pi a ktorého úlohou bude spravovať video obsah dostupný pre prehrávače.

Na obrázku 4.1 je možné vidieť návrh vyjadrený ako blokovú schému. Jadrom celej aplikácie je web server, ktorý má na starosti príjem, ukladanie a poskytovanie informácií o celom systéme. Na web server sa ukladajú napríklad informácie o súboroch v hlavnom počítači a ich priradení k jednotlivým zariadeniam. Ďalej sa evidujú jednotlivé zariadenia, konkrétne ich meno, IP adresa a stav. Hlavný počítač prenáša dáta priamo do úložísk a web server použije ako zdroj informácií o zariadeniach.

### HTTP/S komunikácia

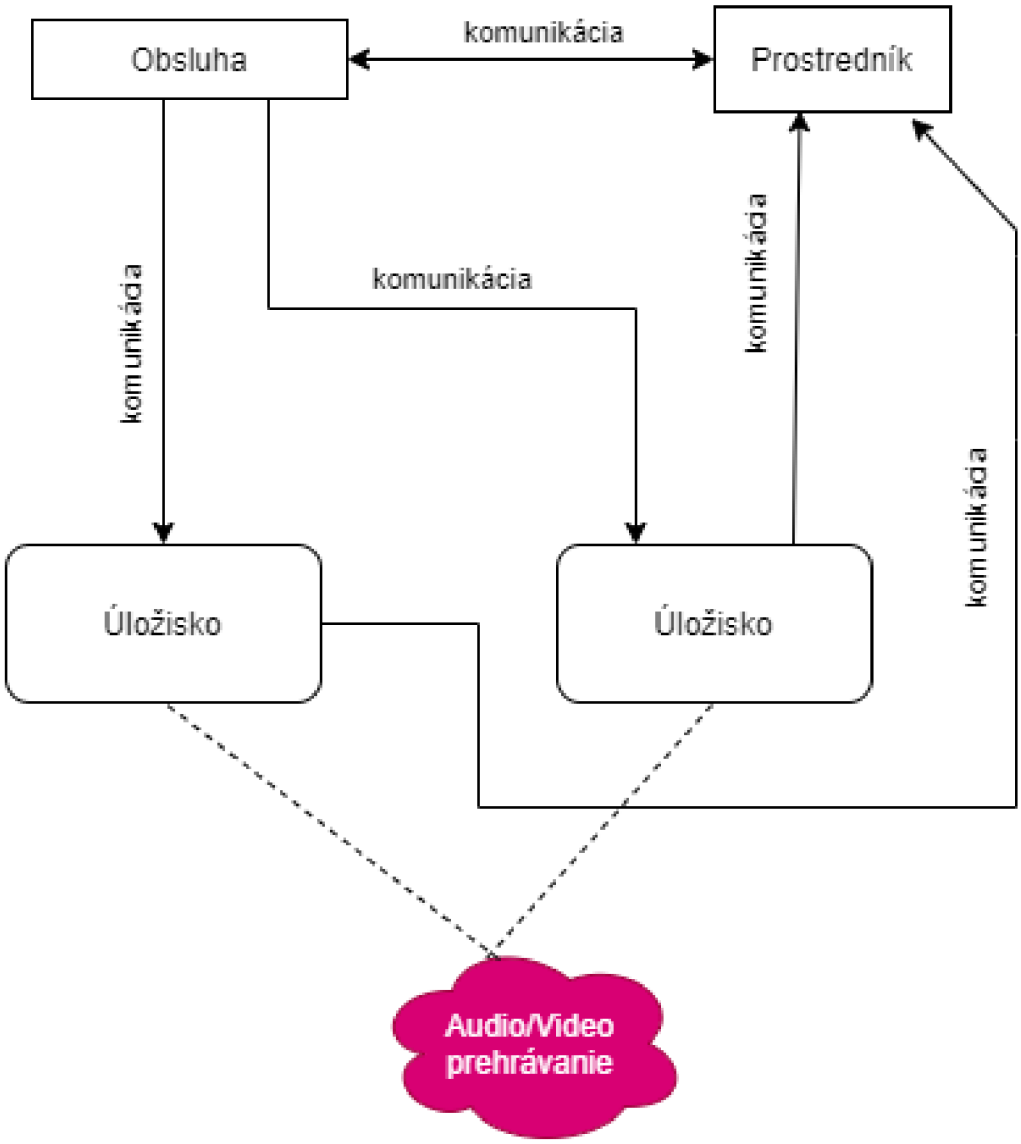
HTTP komunikácia, ktorá prebieha medzi web serverom a všetkými klientmi je šifrovaná pomocou SSL certifikátu. Použite HTTPS bolo potrebné z dôvodu zabezpečenia prístupu k web serveru. Všetky end-pointy web serveru vyžadujú od klientov, ktorý chcú pristupovať k web serveru, autentifikáciu pomocou bežného prístupového overenia (angl. Basic access authentication). Z dôvodu použitia nešifrovaného spôsobu prihlasovania bolo potrebné zabezpečiť celú komunikáciu pomocou HTTPS.

### Web server

Z obrázka 4.1 je možné vidieť, že web server je jadrom celej aplikácie, s ktorým komunikujú všetky súčasti. Hlavným cieľom tejto časti aplikácie je uchovávanie a sprostredkovávanie informácií o celom systéme (súboroch, zariadeniach, atď.).

Zariadenia umiestnené v sieti oznámia pri zapnutí serveru svoje meno a adresu. Aplikácie, ktoré potrebujú tieto adresy pre svoje fungovanie ich jednoducho jedným HTTP požiadavkom získajú zo servera a nemusia pracne zisťovať ich adresy prechádzaním siete. Server skladuje informácie o pridelených súboroch do jednotlivých zariadení. Ukladá jednotlivé cesty k súborom na všetkých zariadeniach kam boli pridané aby bolo možné ich lokalizovať. Server používa na ukladanie všetkých perzistentných dát relačnú databázu PostgreSQL.

Rozhranie servera je rozdelené do niekoľkých skupín podľa zdrojov, s ktorými pracujú. Toto rozdelenie je možné vidieť v tabuľke 4.2, podrobenejší popis navrhnutého rozhrania je v prílohe B.1.




Obr. 4.1: Znázornenie návrhu aplikácie ako bloková schéma. Všetko okrem farebne vyznačeného bloku je súčasťou riešenia tejto bakalárskej práce. Jadrom celého systému je prostredník, ktorý spracováva požiadavky od jednotlivých úložísk umiestnených v sieti, ktoré pri štarte posielajú informácie o sebe. Ďalej prostredník komunikuje s obsluhou, ktorá z neho získava údaje o zariadeniach v sieti a ukladá ďalšie informácie, ktoré následne môžu používať ďalšie spolupracujúce aplikácie. Obsluha v určitých prípadoch komunikuje aj priamo s úložiskom.

| Metóda | URI | Popis |
|---|---|---|
| Prihlásenie | /login | prihlásiť užívateľa |
| Súbory | /file | zobraziť súbory,<br>pridať/zmazať/modifikovať/zobraziť súbor |
| Raspberry Pi | /register | Zariadenia RPi pošlú v tele<br>požiadavky svoje<br>meno a sú pridané do systému |
| Zariadenia | /device | zobraziť zariadenia,<br>pridať/zmazať/modifikovať/zobraziť zariadenie |
| Stavy | /status | zobraziť stavy v ktorých<br>sa môže zariadenie nachádzať |
| Priradenie súborov | /relation | zobraziť všetky vzťahy medzi<br>súbormi a zariadeniami,<br>vytvoriť/zmazať/modifikovať vzťah |
| Používatelia | /user | zobraziť všetkých používateľov,<br>pridať/upraviť/zmazať používateľa |

Tabuľka 4.2: V tabuľke je možné vidieť rozhranie servera zlúčené do skupín podľa zdrojov, s ktorými manipulujú. Detailnejší popis ako sa nachádza v stĺpci *Popis* nájdete v prílohe B.1.

### Úložiská Raspberry Pi

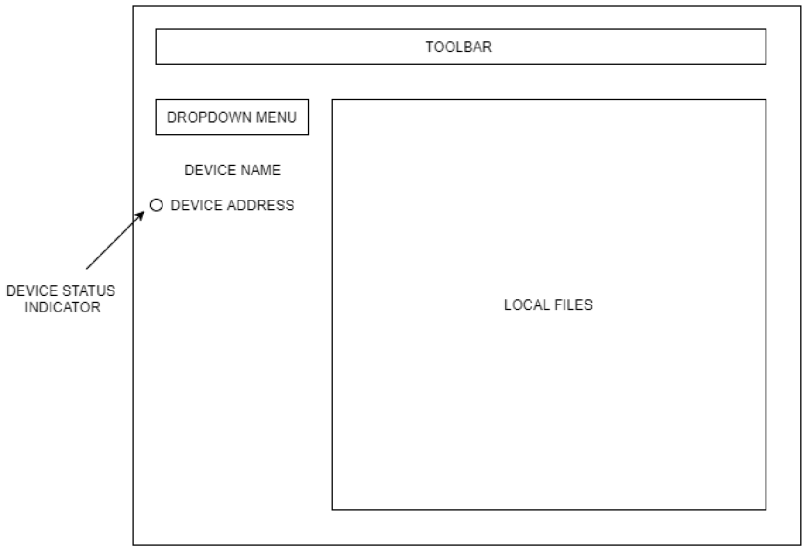
Zariadenia Raspberry Pi pri spustení spustia proces vykonávajúci skript, ktorý má dva módy. Prvý mód sa aktivuje ak sa skript nedokáže spojiť so serverom, alebo server neodpovedá kladne. V tomto móde sa bude častejšie snažiť spojiť so serverom. Po tom, ako server odpovie zariadeniu kladne, skript sa prepne do druhého módu. Ten má za cieľ občasné prihlásenie sa serveru, keby náhodou prišlo k poškodeniu alebo vymazaniu záznamu o zariadení. Časový úsek medzi kladne vybavenými požiadavkami je dlhší aby nebol server zbytočne zaťažovaný.

### TCP IP komunikácia

TCP IP komunikácia medzi hlavným počítačom a jednotlivými zariadeniami je využívaná na synchronizáciu súborov, sťahovanie dát zo zariadení, zisťovanie dostupnosti zariadenia a na pripojenie vzdialeného súborového systému. Synchronizácia súborov bude vykonávaná pomocou linuxových utilít ako je `scp` alebo `rsync`. Oba tieto programy budú pre autentifikáciu voči Raspberry Pi používať RSA kľúče a protokol SSH. Na zistenie dostupnosti budú využité nízkoúrovňové sockety. Vzdialený file systém sa bude pripájať pomocou programu `mount` alebo `sshfs` aby bol umožnený správcovi jednoduchý prístup k všetkým súborom na vzdialenom zariadení.

## 4.4 Ďalšie vlastnosti riešenia

Jednou z výziev pri riešení bol aj návrh rozhrania web serveru, tak aby bolo použiteľné, zrozumiteľné a jednoduché na používanie. Dáta z web serveru nepoužíva len moja práca ale aj iné práce, ktoré vznikajú pre Creative IT Showroom, preto niektorá funkcionalita serveru nebola použitá.




Obr. 4.2: Návrh GUI aplikácie na správu distribúcie súborov. Hlavným ovládacím prvkom je výber zariadenia na ľavej strane v dropdown menu. V tomto prvku je zoznam všetkých zariadení a po vybraní zariadenia je možné s ním vykonávať akcie pomocou klikania na ikony v lište toolbar. Pod rolovacím menu je zobrazené meno, adresa a stav zariadenia. Najväčší prvok s názvom local files slúži na prehliadanie súborov uložených na lokálnom disku.

### HTTP API serveru

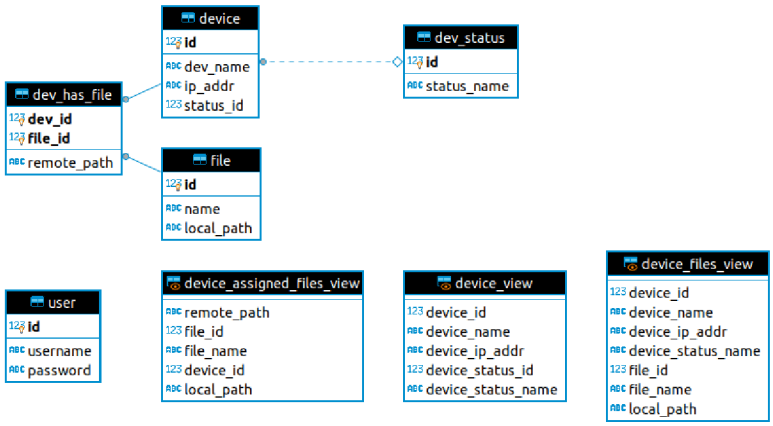
Pre návrh rozhrania bolo rozhodnuté použiť architektúru REST, ktorá je orientovaná na poskytovanie, editovanie, vytváranie a mazanie dát. Rozhranie web serveru sa skladá z rôznych URI a HTTP metód. Tieto kombinácie vytvárajú dvojice a tie sa nazývajú koncový bod (angl. *Endpoint*). Web server navrhnutý pre túto prácu má niekoľko koncových bodov, ktoré sa delia do skupín podľa toho s akými dátami pracujú. V tabuľke v prílohe B.1 je vypísaný zoznam všetkých URI a k ním priradené HTTP metódy spolu s textovým popisom použitia daného bodu.

### Návrh používateľského rozhrania

Grafické rozhranie umožňuje používateľovi spravovať jednotlivé časti systému. Pri návrhu používateľského rozhrania bolo dôležité dbať na jednoduchosť, prehľadnosť a jednotnosť jednotlivých prvkov. Celý návrh sa nachádza na obrázku 4.2 a má 4 hlavné časti: veľké okno zobrazujúce súbory v domovskom adresári používateľa, lištu obsahujúcu tlačidlá pre rôzne operácie a *dropdown* menu obsahujúce jednotlivé zariadenia. Pod týmto menu sú umiestnené informácie o aktuálne zvolenom zariadení. Informácie o zariadení pozostávajú z mena, adresy a stavu zariadenia.

### Schéma databázy PostgreSQL

Poslednou časťou, ktorá tvorí systém je relačná databáza uchovávajúca všetky potrebné údaje pre beh systému. Základom je schéma relačnej databázy, ktorú možno vidieť na obrázku 4.3. Tabuľka `device` je určená pre uchovávanie informácií o zariadeniach, ktoré sú k dispozícii ako úložiská. Minipočítače Raspberry Pi môžu mať rôzny stav. Tieto stavy sú de-




Obr. 4.3: Schéma relačnej databázy vytvorená pomocou programu DBeaver. V schéme sa nachádzajú tabuľky `file`, `device`, `dev_status`, `dev_has_file` a `user`. Ďalej sa v schéme nachádzajú aj databázové pohľady.

finované v tabuľke `dev_status` a odkazuje sa na ne pomocou cudzieho kľúča `status_id`. Informácie o súboroch, ktoré sú v systéme sú uložené v tabuľke `file`. Súčasťou záznamu o súbore je okrem identifikátoru aj meno a cesta k súboru na disku hlavného počítača. Záznamy z týchto dvoch sú pomocou svojich primárnych kľúčov spojené v tabuľke `dev_has_file` a vytvárajú vzťah súboru a úložiska. Tento vzťah vyjadruje, že zariadenie určené cudzím kľúčom `dev_id` obsahuje súbor určený cudzím kľúčom `file_id` a je v úložisku uložený v zložke s cestou zo stĺpca `remote_path`. Pre potreby autentizácie prístupu na web server bola vytvorená tabuľka `user`, kde je uložené meno používateľa a zahashované heslo.

# Kapitola 5

# Implementácia riešenia

V tejto kapitole sa pojednáva o implementácii navrhnutého riešenia z kapitoly 4. V časti 5.1 je prebraná implementácia web serveru podľa navrhnutého API (popísané ako tabuľka B.1 v prílohe) s využitím knižníc z kapitoly 3. V ďalšej časti 5.2 je popísaná implementácia front-end aplikácie pre správu súborov. V sekcii 5.3 je popísané, ako sa jednotlivé zariadenia v sieti registrujú na web server. V sekcii 5.4 je prebraný proces, ako sú súbory synchronizované medzi hlavným počítačom a jednotlivými zariadeniami. V poslednej časti 5.5 tejto kapitoly sú prebrané možnosti konfigurácie aplikácie na správu súborov tak aj skriptu zabezpečujúceho pripájanie zariadení do systému. Implementovaný systém je vyjadrený aj ako bloková schéma na obrázku 5.1.

## 5.1 Web server

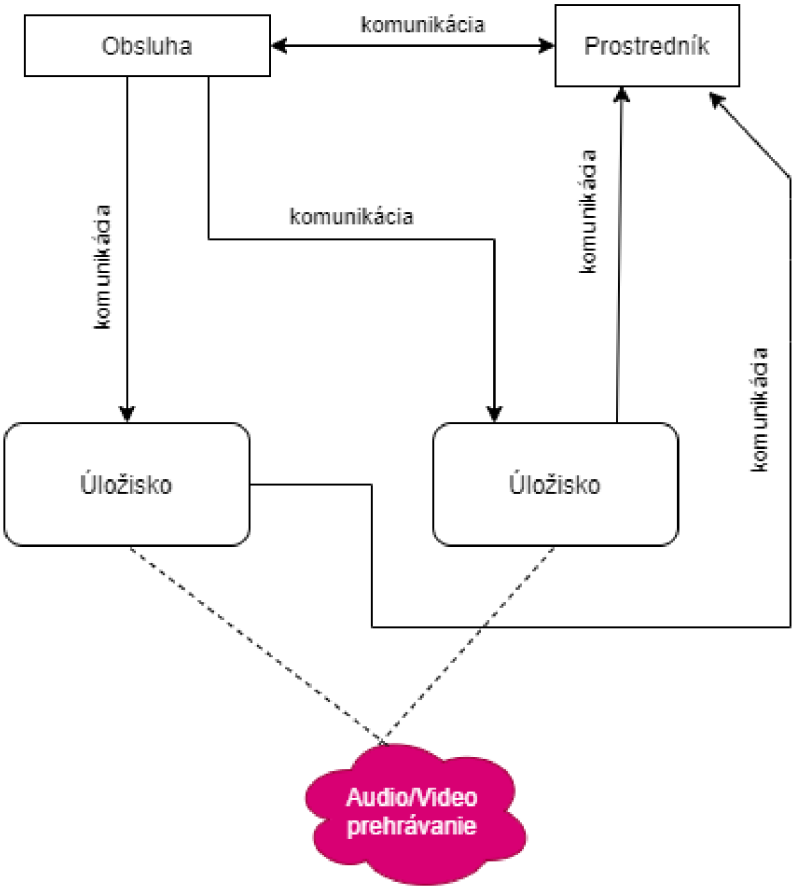
V predchádzajúcich kapitolách už bolo naznačené, že web server je jadrom celého systému a na jeho implementáciu je použitý jazyk Python spolu s knižnicami Flask a Peewee. V tejto sekcii je už popísaná samotná implementácia web serveru.

Server je rozdelený do niekoľkých súborov, skladá sa zo spúšťacieho skriptu `run_server.py` a súboru `server.py`, kde sa nachádza implementácia všetkých koncových bodov. Pri manipulácii s databázou server využíva triedy mapované na relačnú databázu pomocou knižnice Peewee, zmienenej v časti 3.4. Triedy objektov sa nachádzajú v súbore `database_models.py` a reprezentujú jednotlivé tabuľky a databázové pohľady.

Pri obsluhe jednotlivých požiadaviek sa pred zavolaním funkcie vykonávajúcej obsluhu vytvorí databázové spojenie. Toto sa deje vo funkcii `before_request` a následne je volaná príslušná obslužná funkcia. Funkcie sú k obsluhe toho ktorého bodu priradené pomocou python anotácií `@app.route(cesta,HTTP_metody)`. Parameter `cesta` je reťazec znakov a parameter `HTTP_metody` je pole reťazcov, kde každý reťazec vyjadruje jednu HTTP metódu. Príklad takejto anotácie môže byť:

```
@app.route('/file', methods=['GET'])
```

tento konkrétny end-point slúži na získanie všetkých súborov v systéme a jediná povolená HTTP metóda je GET. Všetky dostupné end-pointy sú zhrnuté v tabuľke B.1 v prílohách. Väčšina funkcií má navyše ešte anotáciu `@auth.login_requested`, ktorá vyžaduje aby používateľ/klient komunikujúci so serverom priložil v hlavičkách požiadavky prihlasovacie údaje.




Obr. 5.1: Znázornenie implementovanej aplikácie ako bloková schéma. Všetko okrem farebne vyznačeného bloku je súčasťou riešenia tejto bakalárskej práce. Jadrom celého systému je web server, ktorý spracováva HTTP požiadavky od jednotlivých Raspberry Pi umiestnených v sieti, ktoré pri štarte posielajú informácie o sebe. Ďalej web server komunikuje s hlavným počítačom, ktorý z neho získava údaje o zariadeniach v sieti a ukladá ďalšie informácie, ktoré následne môžu používať ďalšie spolupracujúce aplikácie. Hlavný počítač komunikuje aj priamo s úložiskom a to hlavne keď prenáša súbory.

Po týchto úvodných operáciách nastane obsluha requestu. Vo väčšine prípadov server vykonáva operácie s dátami v databáze (napríklad vyberá záznamy z tabuľky `file`,..). Do databáze sa server dostane pomocou tried, ktoré sú mapované na tabuľky relačnej databázy. Príkladom takéhoto mapovania je trieda `File` zo súboru `database_models.py`.

```
class File(MyBaseModel):
    id = PrimaryKeyField()
    name = TextField(null=False)
    local_path = TextField(null=False)
```

Táto trieda má atribúty `id`, `name` a `local_path`, ktoré sa zhodujú presne s názvami stĺpcov v tabuľke, ktorú je možno vidieť v schéme databázy na obrázku 4.3.

Po tom, ako sa ukončí práca s dátami v databázy a je známy výsledok spracovania požiadavky, tak obslužná funkcia vráti ako svoju návratovú hodnotu volanie funkcie `response(body,status,statusCode,errmsg)`. Spomenutá funkcia vytvorí Flask objekt `Response`, do ktorého vloží dáta zapísané do JSON notácie, číselný kód `statusCode`, ktorý reprezentuje HTTP status kód. Ak pri obsluhe požiadavky nastala chyba, tak parameter `errmsg` bude obsahovať stručný popis chyby a bude pridaný do odpovede. Po skončení obsluhy požiadavky sa zavolá funkcia `after_request`, ktorá len zavrie spojenie s databázou.

## 5.2 Aplikácia pre správu súborov

V tejto sekcii je popísaná jediná grafická časť tejto práce. Aplikácia je front-end pre celý systém a interakciou s rozhraním môže používateľ spravovať distribúciu dát – pridávať, odoberať alebo modifikovať videá, sledovať stav zariadení, odstraňovať zariadenia, atď.

Aplikácia sa skladá z viacerých súborov. Hlavným súborom je skript `app.py`, ktorým sa spúšťa celá aplikácia. Súbory `device.py`, `file.py`, `user.py` obsahujú triedy so statickými metódami, pomocou ktorých aplikácia komunikuje so serverom.

Ďalšou skupinou súborov sú súbory `filemissingdialog.py`, `editlabel.py`, `customtreeview.py`, `tablemodel.py`, `filemodelfilter.py` a `deviceswindow.py`. V týchto súboroch sa nachádzajú triedy objektov definujúce rôzne interaktívne prvky aplikácie, ako napríklad okno zobrazujúce zoznam zariadení alebo modálne okno vyvolané pri synchronizácii neexistujúceho súboru. Dôvodom rozdelenia do viacerých súborov bolo zvýšenie čitateľnosti kódu.

Aplikácia využíva návrhového vzoru Model-Pohľad-Radič (angl. Model-View-Controller). Použitá GUI knižnica PyQt5 zo sekcie 3.1 plne podporuje tento návrhový vzor už existujúcimi triedami objektov. Pre ilustráciu pri implementácii zoznamu zariadení bol ako *Model* použitá vlastná implementácia abstraktnej triedy `QAbstractTableModel`, ktorá obsahuje dáta o zariadeniach. V roli *Pohľadu* bola použitá trieda `QTableView`, ktorá zobrazuje dáta v tabuľke. Ako *Radič* slúži kontextové menu vyvolané kliknutím pravého tlačidla myši nad zariadením.

V hlavnom skripte `app.py` sa nachádza trieda `App`, ktorá definuje hlavné okno aplikácie. V konštruktore tejto triedy je možné nájsť inicializáciu všetkých potrebných atribútov, ktoré tvoria rozhranie aplikácie. Po inicializácii všetkých atribútov je vyvolaná metóda `initUI`, ktorá prepojí všetky objekty vytvorené v konštruktore tak aby spolu vytvorili požadované rozhranie.

Po zapnutí sa používateľovi zobrazí hlavné okno aplikácie, ktorého návrh je možné vidieť na obrázku 4.2 a výslednú podobu na obrázku v prílohe C. Rolovacím menu na ľavej strane je

možné vyberať zo zoznamu všetkých zariadení, ktoré sú registrované na serveri. Po vybraní zariadenia je pod rolovacím menu zobrazený názov zariadenia, adresa a stav.

V hornej lište hlavného okna na ľavej strane sú operácie pracujúce s aktuálne vybraným zariadením. Je možné zobraziť súbory priradené k zariadeniu v stromovej hierarchii, pripojiť pomocou `mount` utility vzdialený adresár zariadenia a zobraziť jeho obsah. Ďalej je možné zistiť, či je zariadenie dostupné a synchronizovať priradené súbory do zariadenia.

V pravej časti hornej lišty v hlavnom okne sú operácie, ktoré sa neviažu so zvoleným zariadením. Je možné si zobraziť zariadenia v systéme, používateľov, súbory v systéme, skontrolovať dostupnosť všetkých zariadení a synchronizovať všetky zariadenia s naviazanými súbormi. Používatelia a zariadenia sa po kliknutí na ikonu pre zobrazenie načítajú zo servera a zobrazia v novom okne v tabuľke. Zariadenia je možné zo systému mazať, aby keď sa v systéme vyskytne zariadenie, ktoré už fyzicky nie je v sieti, bolo odstrániteľné. Používateľov je možné mazať, upravovať a pridávať do systému. Zisťovanie dostupnosti zariadení si zo servera stiahne zoznam všetkých zariadení a postupne skontroluje ich dostupnosť a používateľa o jednotlivých výsledkoch informuje.

## 5.3 Pripájanie zariadení do systému

V aplikácii si môže používateľ zobraziť okrem mena úložiska aj jeho adresu. Adresa je uložená v databáze a poskytuje ju web server na koncovom bode `/device`, detailnejšie popísaný v tabuľke B.1 v prílohe B. Problém s IP adresami nastáva, ak nie sú pridelené staticky ale dynamicky pomocou protokolu DHCP a pod.

Pri statickom pridelení IP adresy bude mať pri pripojení do siete rovnakú adresu. V tomto prípade by stačilo pri pridávaní zariadenia do systému priradiť k jeho záznamu v databáze adresu. Pri dynamickom získavaní adresy je potreba takúto zmenu zapísať do databázy. V tejto práci je táto záležitosť riešená jednoduchým Python skriptom s názvom `connect.py` uloženým na každom jednom zariadení.

Skript má na starosti pri štarte operačného systému zariadenia poslať na adresu web serveru jednoduchú HTTP požiadavku, ktorá v tele obsahuje názov zariadenia a jeho aktuálny stav. Na odosielanie HTTP požiadaviek je využitá knižnica `requests`. Skript obsahuje nekonečný cyklus v ktorom sa požiadavky posielajú na server a podľa návratového kódu odpovede sa zvolí oneskorenie medzi požiadavkami. Toto oneskorenie je definované v súbore `device_config.json`. V časti 4.3 sú spomínané dva módy skriptu. Prvý posiela požiadavky s krátkymi oneskoreniami, pretože nastala výnimka pri posielaní requestu, najčastejšie keď je server nedostupný. Druhý mód je aktívny ak nenastane žiadna výnimka pri odosielaní dát na server. V tomto móde sú dve možné oneskorenia, ak server odpovie návratovým kódom 200, tak sa uplatní iné oneskorenie ako v prípade, že kód je iný ako 200. Aj tieto oneskorenia sa dajú konfigurovať v súbore `device_config.json`.

## 5.4 Synchronizácia súborov medzi RPi a PC

V nasledujúcej časti je prebraná podstata tejto práce, a to prenos alebo synchronizácia súborov medzi hlavným počítačom a úložiskami v sieti. Synchronizácia je vo svojej podstate jednoduchá, ale predchádza ju niekoľko krokov, ktoré musí aplikácia alebo používateľ aplikácie vykonať.

Na to, aby mohli byť do úložiska prenášané dáta, musí byť zaznamenané v databáze. Tento proces bol popísaný v sekcii 5.3. Prvým krokom pre prenos súboru je priradenie

súboru k zariadeniu. K zariadeniam je možné priraďovať súbory z viacerých častí aplikácie. Prvý spôsob je použiť v hlavnom okne aplikácie vstavaný prehliadač súborov, kde po kliknutí pravým tlačidlom myši sa zobrazí kontextové menu. V ponuke je priradenie súboru do aktuálneho zariadenia, do špecifického zariadenia, ktoré si používateľ vyberie alebo priradenie do všetkých zariadení v systéme. Po výbere akcie na asociáciu súboru s úložiskom aplikácia požiada obsluhu, aby zadala cestu súboru vo vzdialenom úložisku na ktoré bude súbor premiestnený. Následne aplikácia odošle pomocou knižnice `requests` HTTP požiadavku na koncový bod `/relation`. O výsledku požiadavky je užívateľ informovaný. Ak sa podarí súbor spárovať so zariadením, je možné ho ihneď preniesť do úložiska, avšak za predpokladu, že je toto zariadenie dostupné.

Synchronizácia sa môže vykonávať dvoma spôsobmi – hromadná synchronizácia, kedy sú stlačením tlačidla sekvenčne synchronizované všetky zariadenia. Druhým spôsobom je synchronizácia aktuálne zvoleného zariadenia. Na prenos dát sa využíva linuxová utilita `rsync` pomocou ktorej je možné porovnať súbor uložený lokálne so súborom uloženým na vzdialenom disku. Rsync má nespornú výhodu v tom, že neprenáša zväčša všetky súbory ale len tie, ktoré sa zmenili.

Pre potreby aplikácie je vytvorený krátky linuxový skript `upload.sh`, ktorý má päť vstupných parametrov. Prvým je privátny kľúč slúžiaci na prihlásenie k vzdialenému systému. Druhý parameter je cesta k súboru, ktorý má byť poslaný. Tretí parameter je meno používateľa na vzdialenom operačnom systéme pomocou ktorého sa nahrávajú súbory. Štvrtý parameter je IP adresa zariadenia. Piaty a zároveň posledný parameter je adresár nachádzajúci sa vo vzdialenom operačnom systéme do ktorého bude súbor nahraný.

## 5.5 Možnosti konfigurácie

Počas vývoja používateľskej aplikácie vyvstala nutnosť uložiť prehľadným spôsobom niektoré údaje, napríklad adresu web serveru, privátny kľúč alebo adresár určený pre pripájanie vzdialených súborových systémov. Tieto informácie je potrebné nastaviť počas inštalácie, aby bol zabezpečený správny chod systému.

Konfigurácia je uložená v textovom súbore vo formáte JSON. Pre pohodlnú prácu s konfiguračným súborom je vytvorená trieda `Config`. Táto trieda ukladá informácie do asociatívneho poľa ako kľúč a hodnotu, kde kľúč je názov konfiguračnej premennej. V aplikácii na správu súborov je možné nastaviť:

- **server_url** – adresa servera, s ktorým má aplikácia komunikovať,
- **https_cert** – cesta k HTTPS certifikátu na overenie HTTPS spojenia so serverom,
- **private_key** – cesta k privátnemu kľúču potrebnému na overenie spojenia so vzdialeným zariadením,
- **removable_media_dir** – adresár, v ktorom sa zobrazia pripojené vymeniteľné médiá napríklad USB kľúč,
- **mount_dir** – adresár, do ktorého môže aplikácia pripájať vzdialené súborové systémy,
- **username** – používateľské meno, ktoré má byť použité pri zasielané requestov na server,

- **password** – heslo, ktoré má byť použité pri zasielaní requestov na server.

V konektore na zariadeniach je taktiež niekoľko nastavení:

- **server_url** – adresa servera, s ktorým má konektor komunikovať,
- **dev_name** – unikátny názov zariadenia,
- **username** – používateľské meno, ktoré má byť použité pri zasielaní requestov na server,
- **password** – heslo, ktoré má byť použité pri zasielaní requestov na server,
- **https_cert** – cesta k HTTPS certifikátu na overenie HTTPS spojenia so serverom,
- **exception_delay** – oneskorenie, ktoré má byť medzi dvoma požiadavkami končiacimi výnimkou (napríklad nedostupný server),
- **success_delay** – dĺžka oneskorenia, ak sa konektor úspešne prihlási na server,
- **fail_delay** – dĺžka oneskorenia, ak server odpovie iným kódom ako je 200.

Všetky tieto nastavenia ovplyvňujú fungovanie systému ako celku. Niektoré sú kritické, napríklad adresa servera na ktorý sa pripájajú zariadenia alebo aplikácia a preto je potreba dbať na správne nastavenie. Po zmene konfigurácie je potrebné reštartovať aplikáciu, ktorej konfiguračný súbor sa zmenil.

# Kapitola 6

# Testovanie a zostavenie systému

Každý projekt potrebuje na svojom konci postúpiť testovaciu fázu, kedy sa dokazuje funkčnosť, správny dizajn, odolnosť proti chybe používateľa atď. V tejto kapitole sa najprv pojednáva o testovaní web serveru. Server poskytuje informácie o systéme nie len pre aplikáciu pre správu súborov ale aj pre ovládanie videa, ktoré je súčasťou inej práce realizovanej v Creative IT Showroom. Preto je potrebné overiť, či sa informácie správne ukladajú, zobrazujú atď. Automatické testy uľahčujú a zefektívňujú celý proces testovania. V druhej časti je prebrané postupné testovanie aplikácie pre správu súborov v rôznych fázach. Aplikácia musí byť testovaná z dôvodu, že s ňou do kontaktu používatelia a pre to musí fungovať s čo najmenej chybami. Testovanie bolo dokončené s ohľadom na časové možnosti práce. Dlhodobé testovanie by si vyžadovalo viac času. Taktiež by sa mohla testovať prenositeľnosť medzi rôznymi HW platformami prípadne medzi rôznymi OS. Avšak k dispozícii bol len obmedzený HW.

## 6.1 Testovanie web serveru

Testovanie webových služieb je možné robiť niekoľkými spôsobmi, hlavné delenie je na manuálne a automatické. Pri manuálnom testovaní sa využijú nástroje pomocou ktorých môže tester alebo aj programátor posielať HTTP dotazy na server a sledovať výsledky operácie. Pri automatizovanom testovaní sa využije software, ktorý poskytuje okrem zasielania dotazov aj kontrolu odpovede, napríklad pomocou XPath výrazov, regulárnych výrazov atď.

### Manuálne testovanie

V tejto práci bol pre manuálne testovanie použitý program Postman do Postman Inc. Program Postman ponúka okrem posielania dotazov aj možnosť ukladať jednotlivé dotazy, organizovať tieto dotazy to tzv. kolekcií a tieto kolekcie následne aj exportovať. Ďalšou výhodou tohto softvéru je pripojenie napríklad s Google účtom a ukladanie dát na cloud. Pri vývoji tejto práce bol Postman použitý v počiatočnej fáze programovania web serveru. Manuálne testovanie naráža na svoje hranice ak je treba posielať veľa requestov alebo ak je treba vykonať dlhšie testovacie scenáre.

### Automatické testovanie

Hlavným prínosom automatického testovania je úspora času testera alebo možnosť programátora skontrolovať, či vykonané zmeny neovplyvnili iné časti systému. Automatické testy

sú v tejto práci vykonávané pomocou softvéru Apache JMeter[1] od Apache Software Foundation. Tento nástroj má široké možnosti testovania – od jednoduchého testovania posielania dotazov cez realizáciu komplexných scenárov po záťažové testovanie. Nástroj ponúka možnosť inštalovať rôzne zásuvné moduly, ktoré ešte rozšíria funkcionalitu.

Testy je možné spustiť po načítaní súboru `HTTPAPItest.jmx` do programu Apache JMeter. V tejto práci bolo automatické testovanie použité na overenie základnej funkcionality, rôznych chybových situácií a jednoduchého záťažového testu. Overenie základnej funkcionality a chybových kódov sa nachádza pod názvom *Basic test*. V tejto sekcii sa nachádza 6 častí:

- **Prepare data** – táto časť vyčistí databázu zavolaním inštalačného skriptu `installDB.py` a vytvorí základné testovacie dáta,
- **Device group** – v tejto časi sa testujú operácie so zariadeniami, vytvorenie zariadenia s už existujúcim menom, modifikácia neexistujúceho zariadenia, zobrazenie neexistujúceho zariadenia, atď.,
- **User group** – v tejto časti sa preveria operácie s užívateľmi, pridanie, zmazanie, pridanie používateľa bez povinného atribútu v tele správy, atď.,
- **File group** – táto časť preveruje operácie so súbormi, pridávanie, modifikácia, modifikácia s neplatnými dátami, atď.,
- **Relation group** – táto pod sekcia pracuje so vzťahmi medzi súbormi a zariadeniami, vytvorenie vzťahu, vytvorenie duplicitného vzťahu, modifikácia vzťahu, zmazanie, atď.,
- **Remove data** – táto časť znova spustí skript na inštaláciu databázy a tým vyčistí všetky dáta.

Záťažový test sa nachádza pod sekciou *Small Load test*, ktorý spočíva v jednoduchom poslaní 30 dotazov na server. Toto správanie má za účelom simuláciu počiatočného náporu na server pri spustení všetkých zariadení naraz v jeden moment. Toto číslo môže byť zvýšené po kliknutí na *Small Load test* a zmene parametru `Number of Threads(users)`.

## 6.2 Testovanie užívateľskej aplikácie

Na začiatku vývoja celej aplikácie boli použité statické dáta uložené v databáze. Tie sa nijak nemenili – zariadenia mali vymyslené adresy po celú dobu rovnaké, cesty k súborom na rôznych diskoch boli vymyslené. Tieto dáta boli vytvorené umelo pomocou jednoduchého scenára v Apache JMeter. Nad takýmito dátami sa dali testovať hlavne situácie, kedy sa zobrazujú dáta ale nevykonávajú sa s nimi žiadne operácie – napríklad otestovanie správneho zobrazenie dát v rôznych situáciách ako je: v systéme sa nenachádzajú žiadne zariadenia, neobsahuje súbory atď.

Po tom, ako boli otestované tieto jednoduché situácie bolo pristúpené k simulácii showroomu a to vytvorením siete virtuálnych počítačov. Tieto počítače boli vytvorené pomocou virtualizačného softvéru VirtualBox od Oracle. Pre jednoduchšie vytváranie bol použitý dopredu nainštalovaný obraz virtuálneho linuxového zariadenia. Tento vzor systému bol následne trikrát klonovaný, aby simuloval 3 zariadenia pripojené do siete. Jedna virtuálna

[1] https://jmeter.apache.org/

stanica bola v roli hlavného PC a ostatné stanice mali za úlohu simulovať úložiská. S takýmito prostriedkami už sa dalo testovať správanie aplikácie pri práci s dynamickými dátami (napríklad zmena stavu zariadení). Rovnako bolo otestované aj synchronizovanie súborov medzi jednotlivými virtuálnymi stanicami. Spolu s testovaním grafickej aplikácie bol otestovaný aj krátky skript na pripájanie vzdialených staníc na server. Pri testovaní aplikácie s živými dátami sa prišlo na niektoré chyby v reprezentácii dát smerom z API web-serveru, takže bolo nutné pridať niektoré údaje do odpovedí. Taktiež vyvstala potreba doplniť niektoré koncové body s ktorými sa nepočítalo pri počiatočnom návrhu webového rozhrania. Jedným z príkladov je zmazanie súboru pomocou cesty k súboru alebo získanie unikátnych ciest použitých v súboroch priradených ku konkrétnemu zariadeniu.

Po otestovaní aplikácie vo virtuálnom prostredí bolo vykonané testovanie s improvizovaným showroomom, ktorý obsahoval niekoľko reálnych Raspberry Pi 3B+ s operačným systémom LibreELEC pripojených do domácej Wi-Fi siete. Týmto testovaním sa overilo správne chovanie systému v podmienkach mimo virtuálneho HW.

## 6.3 Zostavenie systému

Celý systém sa skladá z hardvérovej časti, ktorá musí byť správne zapojené a nakonfigurovaná, aby bola možná komunikácia. Druhá časť je softvérová a priamo závisí na správnej konfigurácii hardvéru. Hardvér je možné pre testovacie účely možné nahradiť aj virtuálnymi počítačmi, ktoré sú spojené sieťou. Jednotlivé časti a návod na ich zostavenie je uvedený nižšie.

### Hardvér

Potrebný hardvér pre spustenie je jeden počítač s linuxovou distribúciou ako operačným systémom (testované na Ubuntu 20.04), aspoň jeden minipočítač Raspberry Pi s linuxovým operačným systémom (testované s LibreELEC) a interným úložiskom. Tieto všetky zariadenia musia byť spojené do jednej LAN siete.

### Softvér

Pre správne fungovanie softvérovej časti je potrebné mať na všetkých počítačoch/minipočítačoch dostupný jazyk Python verzie 3.x, linuxové utility mount a rsync. Pre spustenie systému je potreba stiahnuť všetky potrebné knižnice, ktoré sú spolu s ich verziami uvedené v súbore `requirements.txt`. Tento súbor bol automaticky vygenerovaný pomocou softvéru na správu virtuálnych prostredí pre jazyk Python *Miniconda*. Je teda zároveň použiteľný pre jednoduché vytvorenie virtuálneho prostredia so všetkými knižnicami.

Po nainštalovaní všetkých potrebných knižníc je potreba na všetkých počítačoch/minipočítačoch povoliť komunikáciu na porte 5000, cez ktorý komunikuje server a port 22, cez ktorý sa prenášajú dáta. Ak je server umiestnený na inom počítači, ako bude spúšťaná aplikácia na správu súborov, nie je potrebné povoľovať port 22.

Po povolení komunikácie v sieti je potrebné nainštalovať databázu PostgreSQL verzie 12. Databázové pripojenie servera je viazané na databázu s názvom `devbp` a používateľa `admin` s heslom `admin`. Po tom, ako je správne nainštalovaná databáza, je treba spustiť skript `installDB.py` pre vytvorenie databázovej schémy a databázových pohľadov. Teraz je databáza pripravená na použitie.

Po nainštalovaní databázy je možné spustiť server, ktorý sa spúšťa pomocou skriptu `run_server.py`. Tento server čaká na dotazy na všetkých IP adresách zariadenia, na ktorom je spustený.

Do minipočítačov v roli úložísk je treba vložiť súbory `connect.py`, `config.py` a `device_config.json`, ktoré budú pridávať zariadenie do systému. Ďalej je treba nastaviť automatické spúšťanie skriptu `connect.py` pri štarte zariadenia. Toto sa dá v operačnom systéme LibreELEC nastaviť v súbore `/storage/.config/autostart.sh` a to pridaním príkazu spustenia skriptu tak, aby bežal na pozadí ako proces. Pripájací skript potrebuje poznať adresu serveru, prístupové meno a heslo a HTTPS certifikát pre správne naviazanie šifrovanej komunikácie.

Po úspešnom splnení týchto krokov je možné spustiť samotnú aplikáciu pre správu súborov. Aplikácia potrebuje poznať prístupové údaje na web server a jeho adresu, aby bolo zabezpečená správna funkčnosť. Význam jednotlivých konfiguračných parametrov je možné nájsť v časti 5.5.

# Kapitola 7

# Záver

Cieľom práce bolo popísať aktuálne dostupné technológie a riešenia na distribúciu videa v miestnosti a vytvoriť systém, ktorý bude schopný distribuovať video v miestnosti. Ďalej bolo cieľom vytvoriť k tomuto systému aj grafickú aplikáciu, v ktorej bude môcť používateľ pohodlne presúvať súbory do zariadení a spravovať ich. Táto práca zo zadania pokrýva distribúciu videa.

Pred začatím tvorby návrhu systému bolo potrebné preštudovať technológie využívané na distribúciu videa v rôznych podmienkach s rôznym zázemím. Následne bolo zhodnotené zázemie dostupné v Creative IT Showroom a jeho požiadavky. Po preskúmaní bolo rozhodnuté, že distribúcia nebude prebiehať na požiadanie, ale súbory s video obsahom budú uložené na každom stanovišti.

Ako hardvér pre ukladanie bol vybraný minipočítač Raspberry Pi 3B+, ktorý je zároveň vhodný aj na prehrávanie tohoto obsahu. Následne bolo treba vymyslieť spôsob, ako spravovať tento obsah s prihliadnutím na potrebu zdielať informácie o obsahu s inými aplikáciami, napríklad s ovládačom prehrávania na jednotlivých zariadeniach. Toto bolo docielené vytvorením web serveru, na ktorom sú spomínané informácie ukladané. Výzvou bolo nájsť vhodný nástroj pre bezpečný prenos súborov. Zároveň bolo potrebné, aby bol prenášaný čo najmenší objem dát. Preto bolo rozhodnuté použiť Rsync.

Systém bol najprv testovaný len vo virtuálnom prostredí s použitím virtualizačného softvéru. Takto bola overená základná funkčnosť aplikácie bez potreby zložitého zapájania niekoľkých Raspberry Pi a monitorov. Po otestovaní systému na virtuálnom prototype showroomu sa pristúpilo k testovaniu aplikácie na niekoľkých Raspberry Pi zapojených do mini showroomu. Týmto boli vyskúšané potrebné vlastnosti OS LibreELEC pre Raspberry Pi.

Cieľ práce bol splnený a výsledná práca má dokopy viac ako 2800 riadkov a je rozdelená do 21 zdrojových súborov. Práca používa aj 6 obrázkov, z toho 4 slúžia na zobrazenie stavu zariadení a dva slúžia ako ikony v nástrojovej lište.

Do budúcna by bolo možné zapracovať na väčšej integrácii tejto práce s ostatnými prácami, ktoré vznikli v Creative IT Showroom. Prepojenie súborov uložených v databáze s playlistami prehrávačov na jednotlivých zariadeniach. Vykonať rozsiahlejšie testy s rôznymi minipočítačmi, rôznymi platformami, atď.

# Literatúra


[1] *KD-HD8X8Lite* [online]. KeyDigital, 2017 [cit. 2020-8-13]. Dostupné z: http://keydigital.be/items.asp?ItemCode=KDHD8x8Lite&Company=KEY.

[2] *About Bluestream* [online]. Australia: Blustream, 2020 [cit. 2020-8-13]. Dostupné z: https://www.blustream.co.uk/about-blustream.

[3] *About the Company* [online]. KeyDigital, 2020 [cit. 2020-8-13]. Dostupné z: https://keydigital.org/about/company.

[4] Barrett, D. J. *Unix in 24 Hours, Sams Teach Yourself: Covers OS X, Linux, and Solaris*. 2. vyd. O'Reilly Media, 2005. ISBN 978-0596008956.

[5] Beaulieu, A. *Learning SQL: Generate, Manipulate, and Retrieve Data*. 3. vyd. O'Reilly Media, 2020. ISBN 978-1492057611.

[6] Bodnar, J. *Introduction to PyQt5* [online]. zetcode.com, júl 2020 [cit. 2020-8-13]. Dostupné z: http://zetcode.com/gui/pyqt5/introduction/.

[7] Dolák, J. a Šobáňová, P. *Museum presentation*. Palacký University Olomouc, 2018. ISBN 978-80-244-5522-8.

[8] Eby, P. *PEP 3333 – Python Web Server Gateway Interface v1.0.1* [online]. Wilmington: Python Software Foundation, september 2010 [cit. 2020-8-13]. Dostupné z: https://www.python.org/dev/peps/pep-3333/.

[9] GASPAR, D. a. J. S. *Mastering Flask web development : build enterprise-grade, scalable Python web applications*. 2. vyd. Packt Publishing, 2018. ISBN 978-1-78899-540-5.

[10] Gourley, D. et al. *HTTP: The Definitive Guide: The Definitive Guide*. 1. vyd. O'Reilly Media, 2002. ISBN 978-1565925090.

[11] Hernandez, M. J. *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*. 3. vyd. Addison-Wesley Professional, 2013. ISBN 978-0321884497.

[12] Hillier, L. *How museums are using immersive digital experiences* [online]. London: econsultancy.com, júl 2018 [cit. 2020-8-13]. Dostupné z: https://econsultancy.com/how-museums-are-using-immersive-digital-experiences/.

[13] Howard, J. *Case Study - O'PTASIA Super Yacht* [online]. Blustream, 2019 [cit. 2020-8-13]. Dostupné z: https://www.blustream.co.uk/blustream-news/2019/5/23/3khiej78kjnkfxcci51hor8d3hqr5m.

[14] Inc., O. *HDMI Over IP vs Circuit Switched Video Distribution- tutorial* [online]. Peachtree Corners: octavainc.com [cit. 2020-8-13]. Dostupné z: https://octavainc.com/hdmi-video-over-ip-vs-circuit-switched-video-distribution-tutorial/.

[15] Leifer, C. *Peewee Documentation* [online]. charles leifer, 2017 [cit. 2020-8-13]. Dostupné z: http://docs.peewee-orm.com/_/downloads/en/latest/pdf/.

[16] Ltd., B. B. N. S. U. *What Is HDMI over Ethernet and How Does it Work* [online]. Berkshire: blackbox.co.uk [cit. 2020-8-13]. Dostupné z: https://www.blackbox.co.uk/gb-gb/page/44064/Resources/Technical-Resources/Black-Box-Explains/Multimedia/What-Is-HDMI-over-Ethernet-and-How-Does-it-Work.

[17] Lutz, M. *Programming Python*. 4. vyd. O'Reilly Media, 2011. ISBN 978-0-596-15810-1.

[18] Monk, S. *Raspberry Pi Cookbook: Software and Hardware Problems and Solutions*. 3. vyd. O'Reilly Media, 2019. ISBN 978-1492043225.

[19] Monoprice, I. *What are the differences between an HDMI Splitter, a Switch, and a Matrix* [online]. Rancho Cucamonga: kayako.com [cit. 2020-8-13]. Dostupné z: https://monopricesupport.kayako.com/article/259-what-are-the-differences-between-an-hdmi-splitter-a-switch-and-a-matrix.

[20] Preston, W. C. *Backup & Recovery: Inexpensive Backup Solutions for Open Systems*. 1. vyd. O'Reilly Media, 2007. ISBN 978-0596102463.

[21] Reese, G. *The REST API Design Handbook* [e-book]. 1. vyd. Amazon.com Services LLC, 2012.

[22] Sean McManus, M. C. *Raspberry Pi For Dummies (For Dummies (Computers))*. 3. vyd. For Dummies, 2017. ISBN 978-1119412007.

[23] Shiflett, C. *HTTP Developer's Handbook*. 1. vyd. Sams Publishing, 2003. ISBN 978-0672324543.

[24] Shriki, G. a Wacks, K. *High-Quality Multimedia Distribution in Commercial Buildings* [online]. Ottawa: Caba.org, november 2014 [cit. 2020-8-13]. Dostupné z: https://www.caba.org/wp-content/uploads/2020/04/IS-2015-02.pdf.

[25] Taylor, D. *Unix in 24 Hours, Sams Teach Yourself: Covers OS X, Linux, and Solaris*. 5. vyd. Sams Publishing, 2015. ISBN 978-0672337307.

[26] Technology, C. *HTML QuickStart Guide: The Simplified Beginner's Guide To HTML*. CreateSpace Independent Publishing Platform, 2015. ISBN 978-1511617994.

[27] Willman, J. M. *Beginning PyQt: A Hands-on Approach to GUI Programming*. 1. vyd. Apress, 2020. ISBN 978-1484258569.

# Príloha A

# Obsah priloženého CD

V priloženom CD sa nachádza:

- `xbukov15/src/aplikacia` Zdrojové súbory k aplikácii pre správu súborov.
- `xbukov15/src/connectorv2` Zdrojové súbory ku konektoru spustiteľnému pomocou Python 2.x
- `xbukov15/src/connectorv3` Zdrojové súbory ku konektoru spustiteľnému pomocou Python 2.x
- `xbukov15/src/server` Zdrojové súbory k web serveru a inštalácii DB PostgreSQL
- `xbukov15/src/HTTPAPItest.jmx` Zdrojový súbor k automatickým testom
- `xbukov15/src/requirements.txt` Textový súbor popisujúci potrebné knižnice pre beh serveru a aplikácie na správu súborov
- `xbukov15/src/README.txt` Súbor README s užitočnými informáciami
- `xbukov15/doc/technicka_sprava.pdf` Text práce v PDF formáte
- `xbukov15/doc_src` Zdrojové súbory k textu práce

# Príloha B

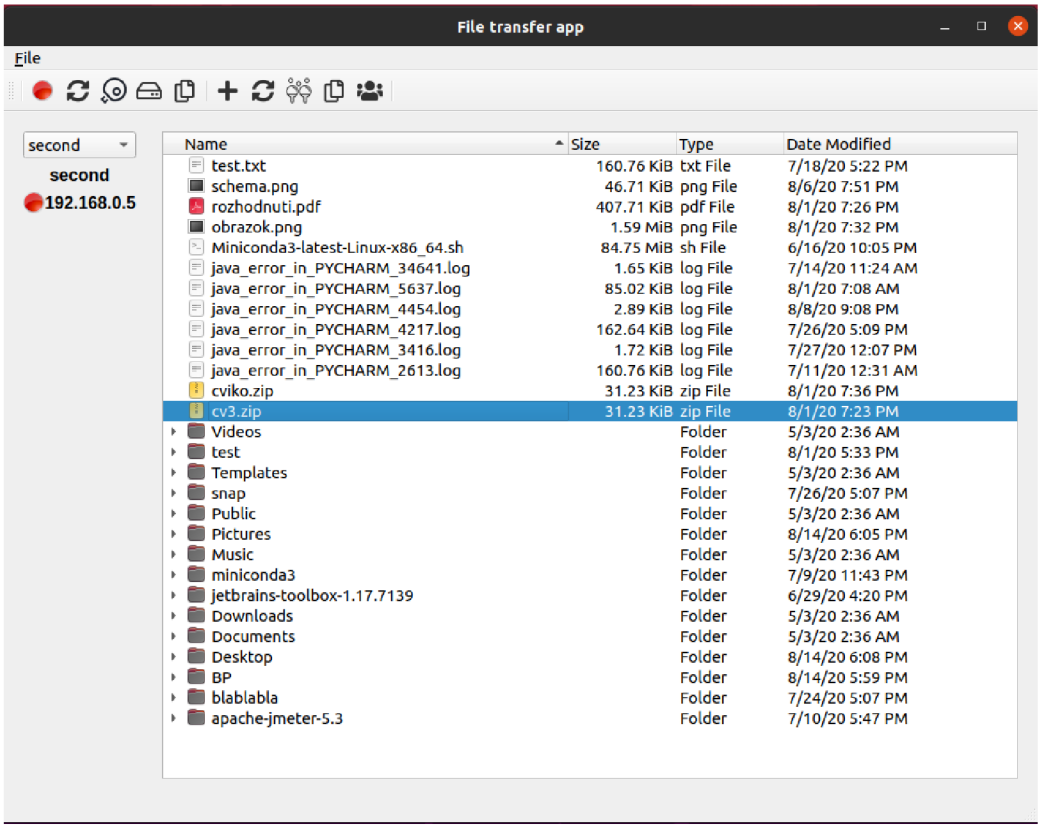
# Popis API webserveru

| Metóda | URI | Popis |
|---|---|---|
| POST | /login | Prihlási užívateľa |
| GET | /file | V odpovedi vráti zoznam<br>všetkých súborov, ktoré sú v systéme<br>pridané so všetkými informáciami |
| PUT | /file | Slúži na pridávanie súborov do systému.<br>V tele požiadavky sa odošle meno<br>súboru a jeho lokálnu cesta |
| DELETE | /file/<int:fileID> | Slúži na zmazanie súboru s fileID zo systému |
| POST | /file/<int:fileID> | Slúžia na mofikáciu informácii o súbore s fileID |
| GET | /file/<int:fileID> | Slúži pre získanie informácii o súbore s fileID |
| PUT | /register | Zariadenia RPi pošlú v tele požiadavky<br>svoje meno a sú priadné do systému |
| PUT | /device | Slúži pre pridanie nového RPi zariadenia do systému |
| DELETE | /device/<int:devID> | Zmazanie zariadenia s devID zo systému |
| POST | /device/<int:devID> | Umožnuje pozmeniť informácie o zariadení s devID |
| GET | /device/<int:devID> | Získanie inrofmácii o zadiadení s devID |
| GET | /device | Získanie inrofmácii o všetkých zariadeniach v systéme |
| GET | /device/<int:devID>/path | Slúži na získanie všetkých<br>unikátnych ciest použitých vo vzdialenom zariadení |
| GET | /status | Vráti zoznam všetkých možných stavov,<br>v ktorých sa môže zariadenie nachádzať |
| PUT | /relation/<int:fileID>/<int:devID> | Priradí súbor s fileID do zariadenia s devID,<br>avšak je ešte treba vykonať synchronizáciu súborov |
| POST | /relation/<int:fileID>/<int:devID> | Umožnuje modifikovať vzdialenú<br>cestu k súboru uloženú v systéme |
| DELETE | /relation/<int:fileID>/<int:devID> | Zruší priradenie súboru s fileID k zariadeniu s devID |
| GET | /realtion?<br>device=<int:devID>,file=<intfileID> | Poskytuje informácie o všetkých priadeniach<br>všetkých súborov k zadiadeniam,<br>je možné filtrovať pomocou url parametrov |

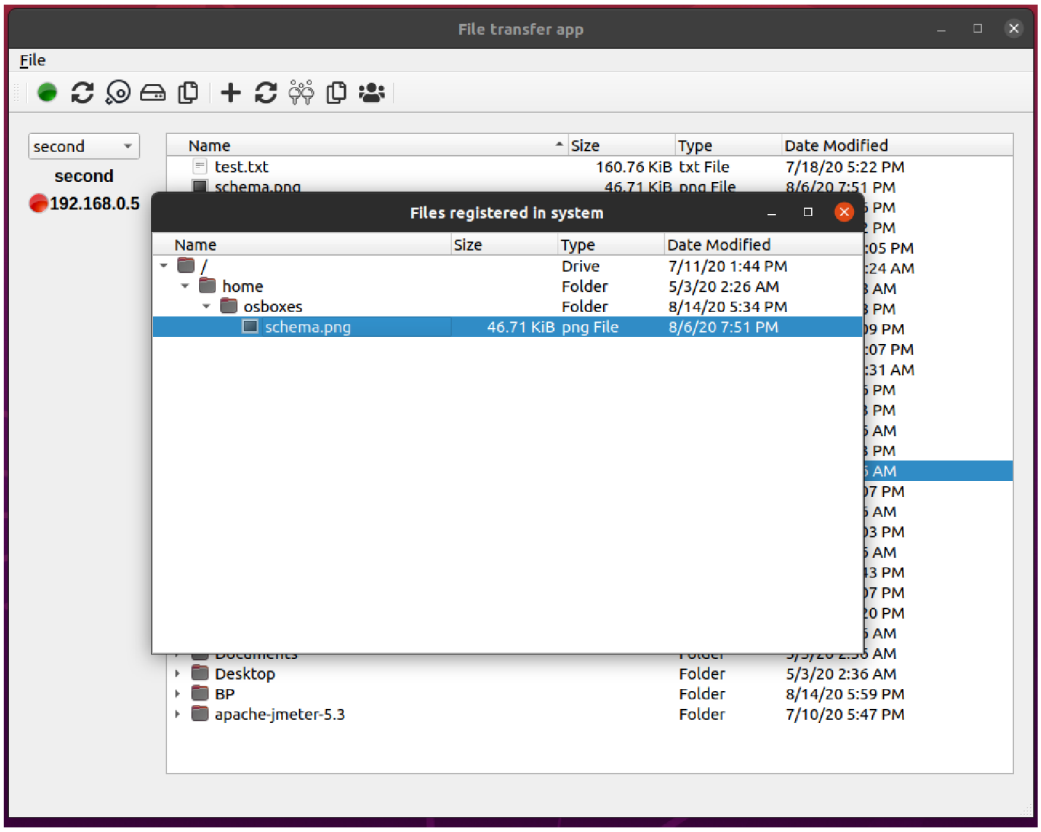
Tabuľka B.1: Popis API web serveru

# Príloha C

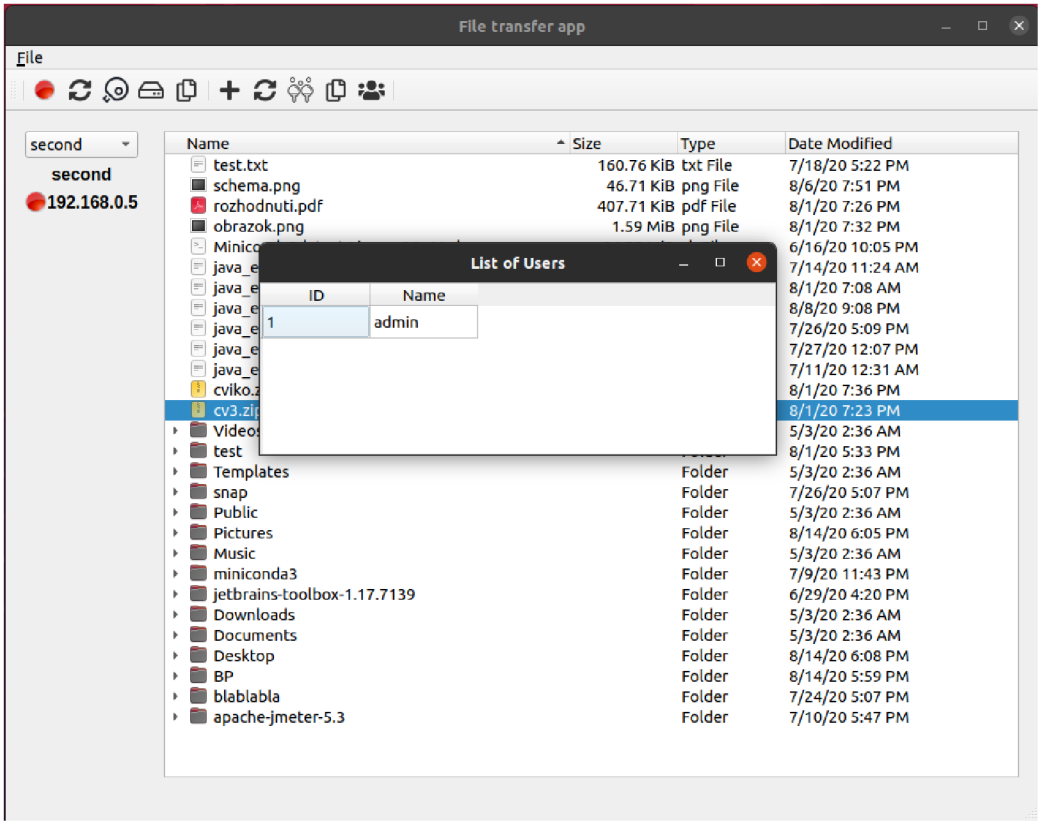
# Grafické užívateľské rozhranie aplikácie



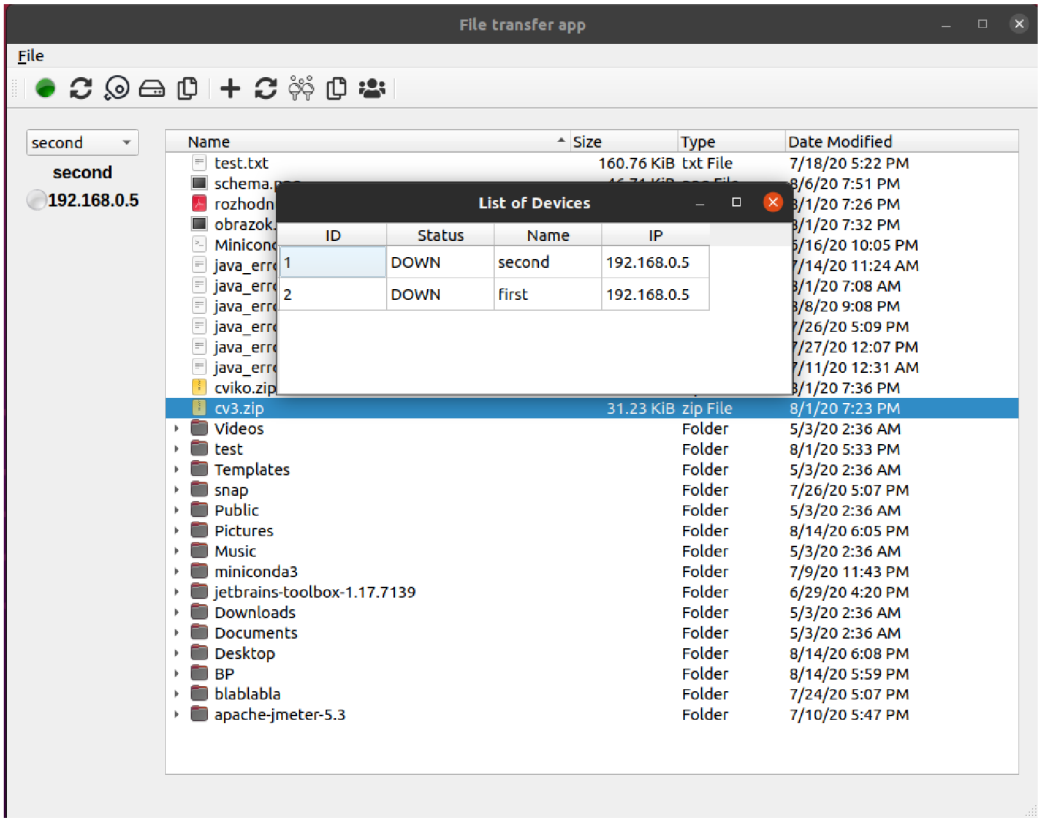

Obr. C.1: Hlavné okno aplikácie, ktoré je možné vidieť pri jej spustení



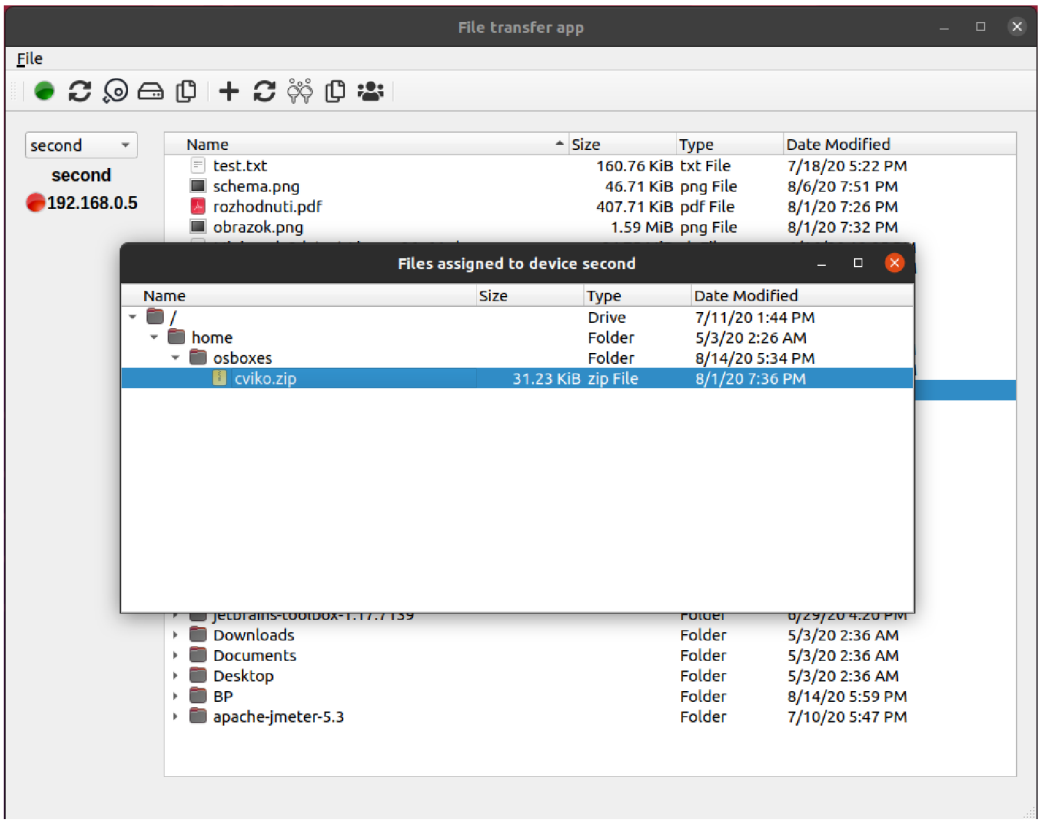

Obr. C.2: Okno, ktoré sa zobrazí nad aplikáciou ukazuje, aké súbory sú registrované s celom systéme




Obr. C.3: Na tomto obrázku je ukázané okno so zoznamom používateľov




Obr. C.4: Tiež sa dá v aplikácii zobraziť v samostatnom okne zoznam všetkých zariadení, ktoré sú aktuálne s systéme




Obr. C.5: Okno zobrazuje, ktoré súbory z lokálneho file systému sú priradené k aktuálnemu zariadeniu