**Česká zemědělská univerzita v Praze**

**Provozně ekonomická fakulta**

**Katedra informačního inženýrství**

# Diplomová práce

## Multimediální kiosek s centrální správou založený na Raspberry Pi

**Bc. Petr Kruntorád**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Petr Kruntorád

Informatika

Název práce

**Multimediální kiosek s centrální správou založený na Raspberry Pi**

Název anglicky

**Multimedia kiosk with central management based on Raspberry Pi**

---

**Cíle práce**

Cílem diplomové práce je vytvořit webové rozhraní určené pro správu multimediálních kiosků, které budou založeny na jednočipovém počítači Raspberry Pi a softwaru umožňujícím zobrazení multimédií z webového rozhraní.

Cílem teoretické části diplomové práce je vysvětlení teorie softwarového návrhu, pojmů týkajících se vytváření webových aplikací, multimediálních kiosků a typů multimédií, které bude zařízení mít možnost přehrávat. Dále je cílem provedení analýzy existujících řešení a technologií zaměřených na multimediální kiosky nebo jejich správu.

Cílem praktické části diplomové práce je vytvoření webové aplikace umožňující spravovat jednotlivé kiosky. Mezi správu kiosků je zahrnuto jejich přidávání, stahování potřebných skriptů pro jejich provoz, odebírání, zobrazování informací o nich a správa zobrazovaných multimédií. Toto webové rozhraní bude založeno na PHP frameworku Symfony. Kromě tvorby webového rozhraní pro správu kiosků je cílem i vytvoření softwaru, který bude sloužit pro zobrazení multimédií z webového rozhraní. Ke správné funkčnosti zařízení bude využit jazyk Python.

**Metodika**

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů týkajících se webového rozhraní a konfigurace Raspberry Pi. Dále bude provedena analýza stávajících řešení a technologií umožňujících zobrazení multimediálních informací. Při tvorbě této aplikace bude využita metodika softwarového návrhu na jejímž základě bude projekt rozdělen do několika fází. Nejprve budou analyzovány požadavky na systém, které budou následně využity při návrhu architektury a rozhraní. Na základě těchto návrhů bude poté vytvořena aplikace pro centrální správu multimediálních kiosků. V průběhu této tvorby bude v rámci kódu tvořena dokumentace. V závěrečné fázi bude celá aplikace otestována, aby bylo ověřeno, že aplikace odpovídá specifikovaným požadavkům a nenachází se v ní závady.

Na základě tohoto teoretického základu bude vytvořen vlastní návrh řešení webového rozhraní sloužícího pro centrální správu multimediálního kiosku založeného na zařízení Raspberry Pi.

**Doporučený rozsah práce**

60–80 stran

**Klíčová slova**

multimediální kiosek, webová aplikace, centrální správa zařízení, Raspberry Pi, Python, PHP framework Symfony

---

**Doporučené zdroje informací**

DEAN, John. Web programming with HTML5, CSS, and JavaScript [online]. Burlington: Jones & Bartlett Learning, 2019. ISBN 978-128-4091-793. Dostupné z: http://projanco.com/Library/Web%20Programming%20with%20HTML5,%20CSS,%20and%20JavaScript.pdf

HALFACREE, Gareth. THE OFFICIAL Raspberry Pi Beginner's Guide: How to use your new computer [online]. 4. Cambridge: Raspberry Pi Trading, 2020. ISBN 978-1-912047-73-4. Dostupné z: https://magazines--attachments.raspberrypi.org/books/full_pdfs/000/000/038/original/BeginnersGuide-4thEd--Eng_v2.pdf

LI, Ze-Nian, Mark S DREW a Jiangchuan LIU. Fundamentals of Multimedia [online]. 2nd ed. 2014. Cham: Springer International Publishing, 2014. Texts in Computer Science. ISBN 978-3-319-05290-8. Dostupné z: https://drive.uqu.edu.sa/_/mskhayat/files/MySubjects/20178FS%20Multimedia%20Systems/Fundamentals_of_

LUTZ, Mark. Learning Python [online]. 4th ed. Sebastopol: O'Reilly, c2009. ISBN 978-0-596-15806-4. Dostupné z: https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf

---

**Předběžný termín obhajoby**

2023/24 LS – PEF

**Vedoucí práce**

Ing. Dana Vynikarová, Ph.D.

**Garantující pracoviště**

Katedra informačního inženýrství

Elektronicky schváleno dne 28. 11. 2023

**Ing. Martin Pelikán, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 9. 2. 2024

**doc. Ing. Tomáš Šubrt, Ph.D.**

Děkan

V Praze dne 22. 03. 2024

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci „Multimediální kiosek s centrální správou založený na Raspberry Pi“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2024 _____

**Poděkování**

Rád bych touto cestou poděkoval vedoucí mé práce Ing. Daně Vynikarové, Ph.D. za odborné připomínky a rady, které byly cenné pro tvorbu této diplomové práce. Dále bych rád poděkoval Ing. Petře Pavlíčkové, Ph.D. za odbornou konzultaci ohledně tématu metodiky vývoje softwaru.

# Multimediální kiosek s centrální správou založený na Raspberry Pi


## Abstrakt

Tato diplomová práce se zabývá tvorbou multimédiálních kiosků založených na jednodeskovém počítači Raspberry Pi a jejich vzdálenou správou prostřednictvím webového rozhraní založeného na PHP frameworku Symfony.

Vytvořené webové rozhraní, které je založeno na PHP frameworku Symfony, umožňuje správu multimédií, zařízení, seznamů k přehrání, uživatelů s přístupem do systému. V systému je možné využívat hned několik typů multimédií (videa, YouTube videa, obrázky a webové stránky). Tato multimédia lze následně přiřadit do seznamu k přehrání, který umožňuje zobrazení jednoho multimédia celý den, nebo pouze v určitých časových úsecích. Takto nadefinovaný seznam k přehrání lze následně nastavit u jednotlivých zařízení, na kterých bude přehráván. U jednotlivých zařízení se také kromě seznamu k přehrání nastavuje název a umístění zařízení pro snadnější identifikaci.

Kromě webového rozhraní byl vytvořen i software, který umožňuje přehrávání obsahu z jednotlivých seznamů k přehrání na zařízení. Tento software má 2 hlavní části. První část se nachází ve webovém rozhraní, a to je samotný kód, který přehrává obsah pro dané zařízení na specifické URL adrese, kterou má každé zařízení vlastní. Druhá část se nachází na zařízení a je vytvořena v programovacím jazyce Python. Tato část řeší samotné načtení přehrávače v prohlížeči a kontrolu, že je prohlížeč spuštěn, aktualizaci informací o zařízení a udržování aktuálního konfiguračního souboru.



**Klíčová slova:** multimediální kiosek, multimédia, webová aplikace, centrální správa zařízení, Raspberry Pi, Python, PHP, PHP framework Symfony

# Multimedia kiosk with central management based on Raspberry Pi

**Abstract**

This thesis deals with the creation of multimedia kiosks based on the Raspberry Pi single board computer and their remote management through a web interface based on the PHP framework Symfony.

The created web interface, which is based on the PHP framework Symfony, allows the management of multimedia, devices, playlists, users with access to the system. Several types of multimedia (videos, YouTube videos, images and web pages) can be used in the system. These multimedia can then be assigned to a playlist, which allows one multimedia to be viewed all day or only at certain times. The playlist defined in this way can then be configured for the individual devices on which it will be played. For each device, in addition to the playlist, the name and location of the device is also set for easier identification.

In addition to the web interface, software has been developed to allow content from individual playlists to be played back on a device. This software has 2 main parts. The first part is in the web interface, and this is the code itself, which plays the content for a given device at a specific URL that each device has its own. The second part is located on the device and is created in the Python programming language. This part handles the actual loading of the player in the browser and checking that the browser is running, updating the device information and keeping the configuration file up to date.

**Keywords**: multimedia kiosk, multimedia, web app, central management of devices, Raspberry Pi, Python, PHP, PHP framework Symfony

# Obsah

# 1 Úvod

V současnosti, kdy je důležité neustále udržovat zobrazované informace aktuální a aktualizovat je co nejrychleji, se multimediální kiosky vyskytují nebo různé digitální obrazovky vyskytují stále na více místech. Avšak problémem je, že tyto kiosky jsou drahé, a ne vždy je potřeba zobrazovací zařízení, které je určeno pro venkovní použití nebo musí být umístěno ve stojanu, ale stačí pouze zařízení, jenž se připojí k televizi, dotykové obrazovce nebo jinému zobrazovacímu zařízení.

Přesně na řešení těchto situací je zaměřena tato diplomová práce, která zahrnuje vytvoření webové aplikace sloužící pro vzdálenou správu multimediálních kiosků založených na jednodeskovém počítači Raspberry Pi. Jelikož je toto zařízení založeno na Raspberry Pi, tak je možné jej využít i v případě, kdy není dostatek místa (například televize zavěšená na zdi.)

Práce je koncipovaná do několika hlavních kapitol: cíl práce a metodika, teoretická východiska, vlastní práce, výsledky a diskuse, závěr, seznam použitých zdrojů, seznam obrázků, tabulek, grafů a zkratek, přílohy. Hlavní obsah práce se nachází v kapitolách teoretická východiska a vlastní práce.

V kapitole teoretická východiska je čtenáři vysvětleno, co jsou to multimediální kiosky a jaké existují řešení. Dále jsou zde popsány metodiky vývoje softwaru, co jsou to multimédia, technologie a zařízení, které byly nezbytné pro vypracování této práce. To zahrnuje vysvětlení, co je to Raspberry Pi, HTML, kaskádové styly, PHP, Python a PHP framework Symfony.

Kapitola vlastní práce poté obsahuje výsledky analýzy provedené v kapitole teoretická východiska, popis výsledného řešení a jeho instalace.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Cílem diplomové práce je vytvořit webové rozhraní určené pro správu multimediálních kiosků, které budou založeny na jednočipovém počítači Raspberry Pi a softwaru umožňujícím zobrazení multimédií z webového rozhraní.

Cílem teoretické části diplomové práce je vysvětlení teorie softwarového návrhu, pojmů týkajících se vytváření webových aplikací, multimediálních kiosků a typů multimédií, které bude zařízení mít možnost přehrávat. Dále je cílem provedení analýzy existujících řešení a technologií zaměřených na multimediální kiosky nebo jejich správu.

Cílem praktické části diplomové práce je vytvoření webové aplikace umožňující spravovat jednotlivé kiosky. Mezi správu kiosků je zahrnuto jejich přidávání, stahování potřebných skriptů pro jejich provoz, odebírání, zobrazování informací o nich a správa zobrazovaných multimédií. Toto webové rozhrání bude založeno na PHP frameworku Symfony. Kromě tvorby webového rozhraní pro správu kiosků je cílem i vytvoření softwaru, který bude sloužit pro zobrazení multimédií z webového rozhraní. Ke správné funkčnosti zařízení bude využit jazyk Python.

## 2.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů týkajících se webového rozhraní a konfigurace Raspberry Pi. Dále bude provedena analýza stávajících řešení a technologií umožňujících zobrazení multimediálních informací. Při tvorbě této aplikace bude využita metodika softwarového návrhu, na jejímž základě bude projekt rozdělen do několika fází. Nejprve budou analyzovány požadavky na systém, které budou následně využity při návrhu architektury a rozhraní. Na základě těchto návrhů bude poté vytvořena aplikace pro centrální správu multimediálních kiosků. V průběhu této tvorby bude v rámci kódu tvořena dokumentace. V závěrečné fázi bude celá aplikace otestována, aby bylo ověřeno, že aplikace odpovídá specifikovaným požadavkům a nenachází se v ní závady.

# 3 Teoretická východiska

## 3.1 Multimediální kiosky

Multimediální informační kiosky nebo také pouze multimediální kiosky či informační kiosky jsou zařízením, která umožňují interaktivně (například skrze dotykovou obrazovku nebo průmyslové periferie) nebo neinteraktivně prezentovat informace. Takovým příkladem mohou být kiosky v knihovně, které umožňují procházet katalog knih. Neinteraktivním případem je kiosek v obchodě zobrazující momentálně nejprodávanější předměty na skladě. Hlavním rozdílem oproti běžnému počítači je to, že tyto kiosky jsou navrhnuty pouze pro realizaci a plnění pouze specifického množství úkonů pro které byly navrženy, a hlavně se mohou nacházet na různých veřejných prostranstvích. Z toho důvodu je vhodné, aby byla možnost je vzdáleně spravovat a upravovat nastavení bez potřeby fyzické přítomnosti v místě využití. [1], [2], [3]

Kiosky se často dělí na několik typů v závislosti na jejich místu využití (vnitřní nebo vnější) nebo způsobu montáže a velikosti (volně stojící, namontované na zdi nebo tablet umístěný na stojanu). [1]. [2], [3]

Z pohledu hardwaru je kiosek v základu většinou osazen minimálně displejem a hardwarem sloužícím pro zpracovávání informací. Z volitelného hardwaru, který je využit podle potřeb mohou být vybaveny klávesnicí, čtečkou čárových kódu nebo karet, tiskárnou nebo video kamerou. [1], [2], [3]

## 3.2 Existující řešení

### 3.2.1 Raspberry Pi režim kiosku

Toto řešení využívá nastavení Raspberry Pi OS, které umožňuje automatické spuštění Raspberry Pi v režimu kiosku. V případě, že uživatel zvolí toto řešení, tak musí mít dostupné Raspberry Pi, microSD kartu, napájecí adaptér, displej a kabel pro propojení s displejem. Nejprve se připraví microSD karta, na kterou se nainstaluje Raspberry Pi OS, poté se k zařízení připojí uživatel skrze SSH a provede dle návodu konfiguraci. [4]

Po dokončení konfigurace se nastaví URL adresa, která se má zobrazovat na zařízení, případně je možné vypnout některá nastavení pro větší zabezpečení. [4]

Nevýhodou tohoto řešení je, že nezahrnuje aplikaci, která by umožnovala nastavení obsahu, který se má zobrazovat a uživatel musí využít další řešení. Výhodou tohoto řešení je jeho jednoduchost a cena. [4]

### 3.2.2 piSignage

PiSignage je kompletní řešení obsahující jak webové řešení pro správu zařízení, tak software pro přehrávání obsahu z webového rozhraní. V rámci webového rozhraní lze spravovat zařízení, skupiny zařízení, seznamy k přehrání nebo nahrávat obsah (soubory, odkazy, zprávy, upozornění). Po registraci je možné používat maximálně 2 zařízení s tím, že zařízení je svázáno s licencí pomocí unikátního klíče a maximální kapacitou pro přehrávaný obsah 2048 MB. [5]

V případě, že chce uživatel využívat více zařízení, je každé další zařízení zpoplatněno 25$ v případě využití open-source serveru, 20$ pokud uživatel již má licenci pro zařízení, nebo 35$ v případě nového zařízení včetně předplatného pro dané zařízení. Základní verzi open-source serveru (založen na node.js) je možné si stáhnout zdarma, pokud by však uživatel chtěl využívat tento software pod svojí značkou se správou uživatelů nebo generováním reportů, je nezbytné zaplatit 1200$ ročně. [5]

### 3.2.3 Informační kiosky společnosti PROJEKTA

Jako možné řešení je volba komerčního řešení navrženého přímo za tímto účelem. To umožňuje vybírat z informačních kiosků přesně na míru podle potřeby a požadavků. [6]

Příkladem takovýchto kiosků mohou být kiosky od firmy PROJEKTA, která na svém internetovém obchodě nabízí celou škálu řešení. Nejlevnější takové řešení je Nástěnný informační kiosek 10,1" PA412 OP - Open Frame, které stojí 22 990 Kč včetně DPH. Toto řešení umožňuje dotykové ovládání a součástí ceny je dálkové ovládání, napájecí zdroj, klíče, nářadí a šrouby potřebné pro montáž. Naopak nejdražším zařízením z nabídky této firmy Je velký 86" displej LG86XE3FS s cenou 1 061 690 Kč včetně DPH, které umožňuje přehrávání informací ve venkovním prostoru. [6]

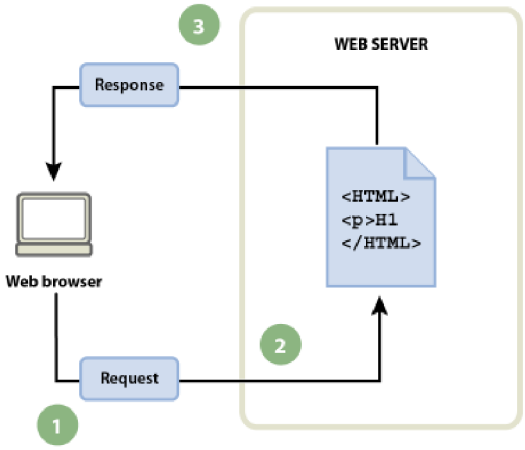
## 3.3 Webové aplikace

Pod pojmem webová aplikace se skrývá aplikace, která je přístupná prostřednictvím webového prohlížeče. Pokud se tato webová aplikace využívá na mobilním zařízení, tak může působit, že se jedná o běžnou mobilní aplikaci, ale ve skutečnosti tomu tak není.

Webové aplikace, jak již název napovídá, se nacházejí na webu a jejich obsah může být rozdílný. Účel těchto webových aplikací může být odlišný, protože jsou vytvářeny za účelem vyřešení různých úkolů nebo problémů. Mohou obsahovat stránky, které mají částečně nebo zcela předem nedefinovaný obsah, který je vygenerován až na základě požadavků a akcí návštěvníka, z tohoto důvodu se tento typ stránek označuje jako dynamické stránky. Existují také statické webové stránky, které mají již předem definovaný obsah. Tento obsah je určen tvůrcem webu a při každém požadavku o zobrazení je identický. [7], [8]

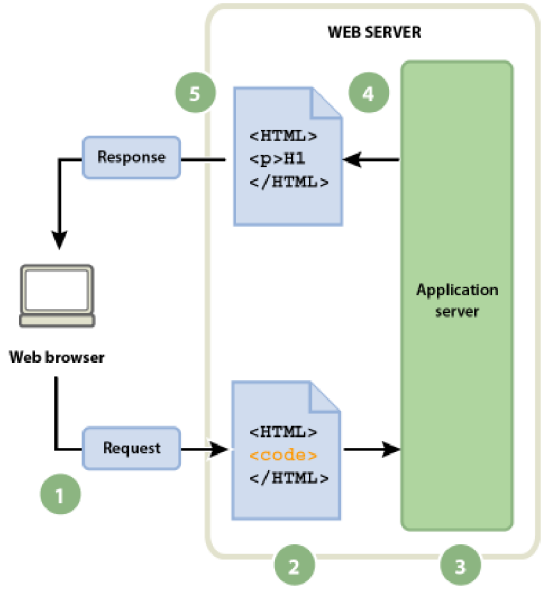
Nejběžnějším využitím webových aplikací může být poskytnutí vyhledávání informací pro návštěvníky, které mohou poté návštěvníci procházet. Dalším využitím může být shromažďování, ukládání a analýza dat, které byly návštěvníky poskytnuty (například data zadaná do formuláře). [7], [8]

### 3.3.1 Statické vs Dynamické webové stránky



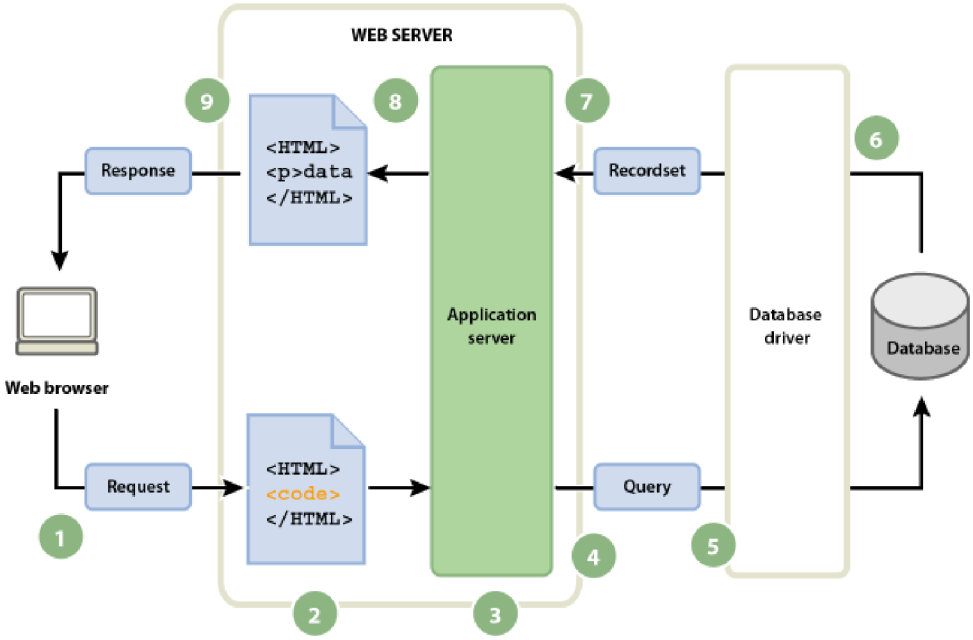

*Obrázek 1 – Zpracování statické webové stránky [8]*

Statické webové stránky fungují tak, že návštěvník požádá o obsah webový server. Jakmile webový server tento požadavek přijme, tak vyhledá požadovanou stránku a vrátí jí následně prohlížeči, ze kterého vznikl požadavek. [8]




*Obrázek 2 – Zpracování dynamických stránek [8]*

Oproti tomu v případě dynamických webových stránek uživatel odešle požadavek na webový server, ten ihned neodpovídá, ale vyhledá stránku a předá ji aplikačnímu serveru. Aplikační server vezme stránku, zpracuje její kód a nahradí jej podle instrukcí v kódu. Z této operace vznikne statická webová stránka, která je vrácena webovému serveru, který jí dále vrátí webovému prohlížeči, který obdrží již čisté HTML. [8]




*Obrázek 3 – Přistupování k databázi [8]*

V případě dynamických webových aplikaci lze využívat i další prostředky na straně serveru. Příkladem těchto prostředků může být databáze, ze které se poté mohou načíst data a vložit do příslušné šablony. Díky tomu je možné upravovat obsah centrálně a stačí mít pro

každý typ informací pouze jednu šablonu, do které je poté obsah dosazen na základě požadavku přímo z databáze. [8]

## 3.4 Vývoj softwaru

Programové vybavení lze rozdělit na 2 hlavní kategorie: systémové a aplikační. Systémové programové vybavení je určeno pro umožnění práce s počítačem a poskytnutí rozhraní pro využívání aplikací, které jsou na něm provozovány. Oproti tomu aplikační software je určen pro vykonání specifických požadavků nebo úkolů. Dalšími kategoriemi jsou ovladače, Middleware a software určený k programování nebo také právě k vývoji. [9], [10]

Cílem je tedy vyvíjení softwaru, který funguje a má určitý význam pro splnění požadavků uživatele. Pro vývoj je důležité, aby nedocházelo k používání částí, návrhů či komponent z předešlých projektů bez nezbytného uvážení, protože ne vždy je to žádoucí pro zájmy projektu nebo zákazníka. Pro zájmy projektu nebo zákazníka to nemusí být žádoucího z toho důvodu, že již dříve použité části, návrhy nebo komponenty nemusí přesně vyhovovat konkrétnímu zákazníkovi a je potřeba je tvořit přesně na míru. [10]

### 3.4.1 Metodiky vývoje softwaru

Pro úspěšný vývoj softwaru je potřeba mít specifikovanou sadu procesů nebo také metodiky vývoje softwaru, které specifikují od začátku do konce jednotlivé činnosti a zároveň definují kdo a kdy má tyto jednotlivé činnosti vykonávat a případně umožňují zhodnotit proces a následně jej v případě potřeby vylepšovat. Díky této sadě procesů je možné do vývoje vnést systematičnost a metodiku, bez nichž i člověk s velkou odborností může selhat. Metodiky lze rozdělit na 2 typy: tradiční a agilní. [10]

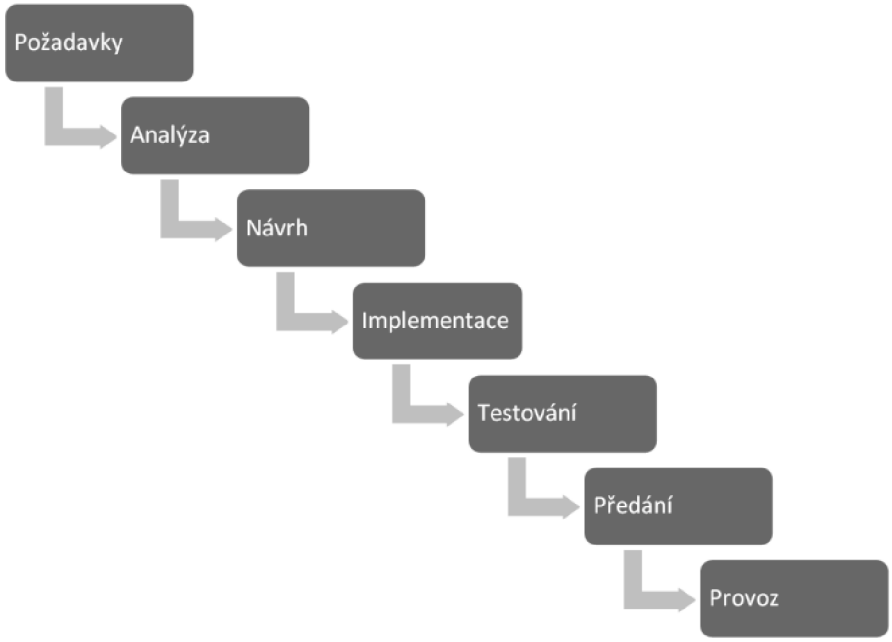
V případě tradičních metodik dochází k co nejdetailnějšímu vyspecifikování projektu, aby bylo možné co nejpřesněji určit jednotlivé požadavky a termíny. V rámci těchto metodik jsou role relativně přesně dány a lidé se tedy tolik nepodílejí na jiných fázích nebo činnostech v projektu, které jim nejsou přiděleny. Rolí existuje celá řada, příkladem může být: architekt, analytik, programátor, tester, projektový manažer, grafik a podobně. Výhodou těchto rolí je, že každý člen je specializován přesně na konkrétní činnosti podle jeho dovedností. Na druhou stranu je mezi jednotlivými členy nezbytná kooperace a komunikace, čímž dochází k prodloužení a zkomplikování procesu vývoje. [10]

V rámci tradičních metodik je také kladen důraz na dokumentaci všeho, nehledě na to, zda se jedná o komunikaci se zákazníkem, sběr požadavků, vývoj, testování nebo předávání a údržbu. Zároveň se s vývojem začíná většinou až tehdy, když je hotová analýza a návrh toho, jak by měl systém vypadat, což může být pro zákazníka frustrující. [10]

#### 3.4.1.1 Vodopádová metodika

Tato metodika se někdy nazývá vodopádový vývoj softwaru a jedná se o jednu z nejstarších metodik vývoje softwaru. Vznikla v 70. letech a dnes se již většinou nevyužívá, protože je nedostatečná i přes svou jednoduchost. Princip tohoto modelu je založen na úvaze, že proces vývoje softwaru by měl mít několik základních fází, které následují po sobě a tyto fáze lze plnit jednosměrně a nikdy se nevrací do předchozí fáze. [10]

**Fáze**




*Obrázek 4 – Schéma vodopádového modelu [10]*

Z obrázku výše je patrné, že se tato metodika skládá ze 7 fází. Těmito fázemi jsou požadavky, analýza, návrh, implementace, testování, předání a provoz. [10]

1. Požadavky – V této fázi se definují potencionální požadavky a termíny. Jsou zkoumány pokyny pro projekt, které jsou následně zaneseny do dokumentu obsahujícího formální požadavky, který se také nazývá funkční specifikace. V průběhu této fáze se prozatím nespecifikují konkrétní procesy, pouze dochází k definování a plánování projektu. [10], [11], [12]

2. Analýza – V rámci této fáze dochází k analýze specifikací systému za účelem vytvoření produktového modelu a obchodní logiky. Zároveň dochází ke kontrole finanční a technické části, aby se ověřilo, že je projekt proveditelný. [10], [11], [12]
3. Návrh – Dochází k vytvoření dokumentu obsahujícího specifikaci, která umožňuje vytvořit si náhled na technickou část návrhu (programovací jazyk, hardware, architektura, služby, zdroj dat). [10], [11], [12]
4. Implementace – Při implementaci dochází k vyvíjení samotného kódu softwaru na základě specifikací, logiky a modelů z předchozích kroků. Většinou se vytváří jednotlivé části nebo komponenty před tím, než je software zkompletován. [10], [11], [12]
5. Testování – Ve fázi testování je cílem odhalit a identifikovat problémy, které je nezbytné opravit nebo vyřešit. Toho je docíleno za využití různých druhů testování. Mezi tyto testy se řadí Alfa testování, které je realizováno vývojovým týmem. Po Alfa testování přichází na řadu Beta testování, ve kterém je produkt testován vybranou skupinou uživatelů. Jakmile je dokončeno Alfa i Beta testování, tak dochází k testování zákazníkem nebo také přijímací testování, které má za výsledek přijmutí nebo zamítnutí výsledku. [10], [11], [12]
6. Předání – V této fázi je software zcela dokončen a dochází k jeho nasazení do produkčního prostředí. [10], [11], [12]
7. Provoz – Fáze provoz je poslední fází ve vodopádovém modelu. Táto fáze nemá specifikovanou dobu trvání a při jejím trvání dochází ke korektivním, adaptivním nebo zdokonalujícím údržbám jejichž účelem je zajistit zlepšování, aktualizaci a zdokonalování produktu. Toho může být docíleno opravením chyb nebo špatného návrhu softwaru, přidáním funkce vyžádané zákazníkem nebo změnou systémového prostředí. [10], [11], [12]

**Výhody a nevýhody**

Výhodou této metodiky je, že je jednoduchá, protože všechny fáze jsou uspořádány za sebou. Lze zjistit v jaké fázi se aktuálně projekt nachází a nelze vstoupit do následující fáze, dokud není dokončena fáze aktuální. [10]

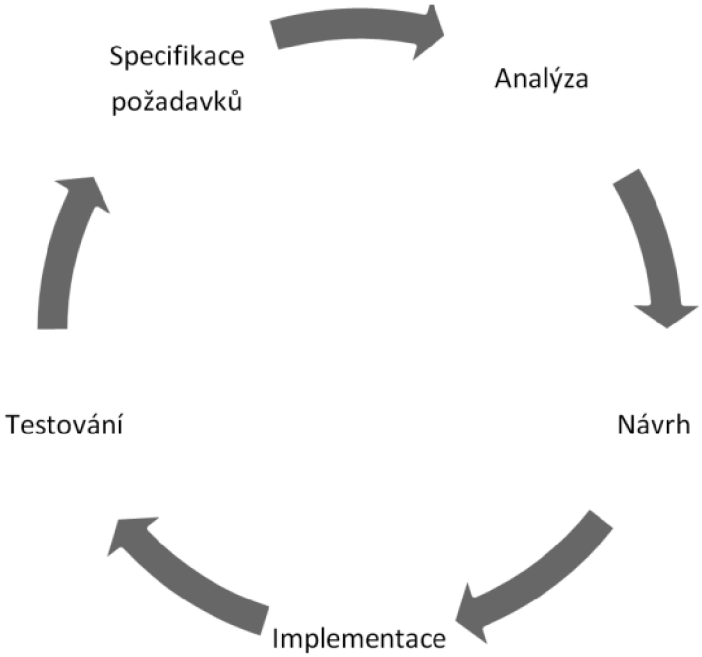
Nevýhody této metodiky jsou u větších projektů ovšem větší než výhody, protože se velmi obtížně detekují a opravují chyby. Důvodem je, že se na počátku vytváří specifikace, ale zákazník nemá často konkrétní představu či znalost možností a případných omezení a jelikož se v průběhu nepředpokládá zapojení zákazníka, který se znovu vyskytuje až ve

fázi předání a nevidí průběžné výsledky. Z tohoto důvodu je obtížné zapracovat případné připomínky. Dalším důvodem je, že pokud je ve fázi testování nalezena chyba, je nezbytné vrátit se na začátek a většinou i přepracovat celý projekt. Právě kvůli těmto problémům s flexibilitou se dnes již tolik tato metodika nepoužívá. [10]

#### 3.4.1.2 Iterativní metodiky

Dle [10] se tento typ metodik snaží využívat, jak již název napovídá iterativní přístup, který u vodopádové metodiky chybí. To je řešeno tím, že se provede určitý proces v následujícím pořadí:

1. Specifikace požadavků – Identifikuje se, co je potřeba.
2. Analýza – Provedou se potřebná měření.
3. Návrh – Dojde k návrhu postupu.
4. Implementace – Postup se realizuje.
5. Testování – Dojde k ověření toho, zda byl splněn cíl.
6. Pokud cíl nebyl splněn, dojde k vrácení do prvního bodu.




*Obrázek 5 – Iterační cyklus [10]*

Z obrázku výše je patrný takzvaný iterační cyklus, který je základem pro několik iterativních metodik (například Spirálová metodika nebo Rational unified process). Obě zmíněné metodiky mají společné to, že po každé iteraci si může zákazník dílčí část vyzkoušet a vrátit k této části připomínky. Zároveň je po každé iteraci provedeno testování. Případné opravy jsou součástí další iterace. Dále v každé iteraci dochází k testování aplikace

zákazníkem, který hledá nejen chyby, ale také posuzuje, zda systém splňuje jeho požadavky. [10]

Iterativní metodiky také využívají další nástroje, mezi které patří například nástroje typu CASE, tj. Computer Aided Software Engineering. Díky těmto nástrojům je možné využívat různé grafické modely, které usnadňují orientaci ve větších projektech a pochopení vazeb mezi individuálními strukturami v rámci projekt. [10]

Agilní metodiky jsou na rozdíl od tradičních metod zaměřeny na eliminaci zbytečné byrokracie, dokumentování každé události nebo aktivity a zjednodušení procesu změny. Nicméně je důležité podotknout, že cílem není eliminovat vše za každou cenu, ale jedná se o jiný a efektivnější přístup k pořádku. [10]

Základním principem je tedy tvorba systému, který je reálnou hodnotou vývoje a odsunutí dokumentace do pozadí, protože dokumentace se nevytváří z důvodu nezbytnosti její existence, ale aby bylo možné software lépe vyvíjet, udržovat a používat. [10]

#### 3.4.1.3 Agilní manifest

Agilní manifest je prohlášení sepsané skupinou jedinců zaměřených na vývoj softwaru s přesvědčením, že vývoj lze provádět lépe než do té doby. Tento manifest má 4 základní hodnoty a to:

1. Jednotlivci a interakce před procesy a nástroji
2. Fungující software před vyčerpávající dokumentací
3. Spolupráce se zákazníkem před vyjednáváním o smlouvě
4. Reagovaní na změny před dodržováním plánu [10]

První bod pojednává o tom, že v rámci projektu jsou sice procesy a nástroje důležité, ale pro agilní metodiky jsou důležitější jedinci včetně jejich potřeb, vkladu do projektu a interakce, protože nástroje a procesy slouží pouze pro uspokojení jejich potřeb a uskutečnění plánované interakce. [10]

Druhý bod popisuje to, co již bylo zmíněno výše a to, že cílem je funkční software a dokumentace je nástroj. Zároveň to ale neznamená, že by se dokumentace neměla vytvářet vůbec. [10]

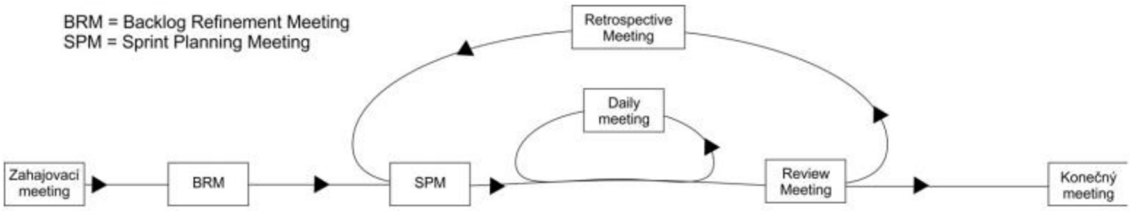
Třetí bod klade důraz na to, aby před tvorbou co nejpřesnějších smluv, které řeší právníci a vedoucí pracovníci, měla přednost spolupráce a komunikace, která vede k hotovému projektu na rozdíl od projednávání co nejdetailnějších smluv. [10]

Čtvrtý a poslední bod je zaměřen na to, že na rozdíl od tradičních metodik, kde se klade důraz na striktní dodržování plánu, což vede k tomu, že zákazník nemá v průběhu možnost zasahovat do výsledku. U agilních metodik je proto důrazem komunikace, aby bylo možné reagovat na změny ze strany zákazníka a tím docílit jeho spokojenosti. [10]

#### 3.4.1.4 SCRUM

SCRUM je framework sloužící pro usnadnění práce lidem, týmům nebo organizacím tak, aby bylo možné adaptivně řešit složité problémy. [13]

**Cyklus**




*Obrázek 6 – Průběh vývojového cyklu řízeného metodikou SCRUM [10]*

Cyklus řízený metodikou SCRUM je možné rozdělit do 3 etap (zahájení, samotný vývoj a ukončení). V etapě zahájení dochází k zahájení vývoje jako takového. Dochází k ujasnění způsobu spolupráce, cílů vývoje, dělí se role a kompetence a dochází k seznámení vývojového týmu. V samotném vývoji se poté vyvíjí daná aplikace formou iterací nebo také sprintů. Ve třetí a poslední etapě nazvané ukončení se produkt finálně testuje a po akceptování je následně nasazen. V okamžiku, kdy je projekt nasazen, dochází k oficiálnímu ukončení. [10]

Pro porozumění metodice SCRUM je nezbytné znát základní pojmy a artefakty, které jsou popsány níže. [10]

User story – uživatelské příběhy

Uživatelské příběhy jsou scénáře, kterými se definují akce pomocí jednoduchých vět. Tyto akce bude poté uživatel ze specifického pohledu v systému vykonávat. Příkladem může být, že po stisknutí určitého tlačítka dojde ke specifické akci. [10]

Sprint

Tento pojem označuje jednotlivé iterace z velkého množství cyklů, které se opakují při vývoji softwaru. Avšak je důležité zmínit, že ne všechny iterace musí být sprint, protože tento název se používá speciálně u metodiky SCRUM. Výsledkem

všech sprintů má být spustitelná aplikace, kterou lze ověřit a testovat. To v praxi znamená, že se vytvoří aplikace, která sice nemusí mít všechny funkce, ale je možné ji již využívat pro plnění dílčích úkolů. Jednotlivé sprinty by neměly být příliš krátké, ale ani příliš dlouhé, protože u krátkých není dostatek času na vývoj kvůli meetingům, a naopak u dlouhého je obtížné plánování. V ideálním případě by měl být sprint dlouhý dva až čtyři týdny, ale akceptovatelná délka je i od týdne do šesti týdnů. [10]

Backlog

Tímto pojmem lze označit jakýsi seznam nezpracovaných úkolů nebo také uživatelských příběhů, které je nezbytné dokončit. Backlog se dělí na product backlog a sprint backlog. V rámci product backlogu je obsažen seznam všech uživatelských příběhů, které je potřeba dokončit během vývoje celé aplikace. Oproti tomu sprint backlog obsahuje pouze úkoly, které se musí dokončit v rámci daného sprintu. Sprint backlog je tedy podmnožinou product backlogu. [10]

V rámci SCRUMu se vyskytují 3 role (Product Owner, SCRUM Master a SCRUM Team Member). Se všemi rolemi se v rámci tohoto frameworku počítá, není tedy možné, že by některá z těchto rolí byla vynechána. [10]

1. Product owner – Úkolem této role je dle [10] zastupování zákazníka, tedy rozhodování, jak bude produkt vypadat a fungovat. Není nezbytné, aby byl Product owner zaměstnancem zákazníka, ale může se jednat osobu, která byla pouze zákazníkem najata s předpokladem, že je schopna komunikovat s osobami z daného zaměření nebo okruhu. Jeho dalším účelem pro projekt je:
    a. Definice vize – Určuje spolu s dalšími osobami pracujícími pro zákazníka, jakým směrem se má projekt vyvíjet, definuje jeho cíl a čemu se má projekt vyvarovat. [10]
    b. Definice úkolů (tvorba Backlogu) – V rámci tohoto účelu tvoří Product owner seznam úkolů před zahájením vývoje a tím udává, jak bude výsledný software vypadat. Avšak, aby bylo možné tento seznam úkolů vytvořit, je nezbytná komunikace mezi Product ownerem, výkonnými pracovníky zákazníka a vývojového týmu. To je nezbytné pro to, aby úkoly odpovídaly tomu, co chce zákazník a co je možné realizovat. [10]

c. Definice priorit – V případě definice priorit dochází k definování priorit jednotlivých funkcionalit. Pro nastavení správné stupnice ohodnocení priorit je potřeba, aby se na konkrétní škále dohodl s týmem. Na základě této stupnice Product owner již bez týmu ohodnotí jednotlivé funkcionality. Toto hodnocení se může v průběhu měnit na základě rozhodnutí managementu firmy nebo různých vnějších změn (například změna legislativy). [10]
d. Respektovat tým – Product owner nemůže diktovat vývojovému týmu jak, v jakém pořadí nebo jakým způsobem mají úkoly plnit. Jeho účelem je starání se pouze o procesní stránku. V případě technické stránky musí respektovat vedení SCRUM mastera a technologické postupy. [10]
e. Zrušení sprintu – Tuto pravomoc má Product owner pro případ, že je jasné, že daný vývojový cyklus, nebo také sprint není možné dokončit s dobrým výsledkem a náklady na zrušení jsou přijatelnější než náklady potřebné na dokončení. [10]
f. Komunikace se zákazníkem – Jak již bylo zmíněno výše, úkolem Product ownera je komunikace se zákazníkem. Slouží tedy jako komunikační kanál mezi zákazníkem a vývojovým týmem. [10]

2. SCRUM Master – Jedná se manažerskou roli, která nemá za úkol fungovat jako běžný šéf, protože ten není ve SCRUMu vyžadován, ale má za účel umožnit týmu pracovat. V rámci projektu SCRUM Master pomáhá týmu dosahovat vytyčených cílů (koordinace meetingů, příprava dokumentace a další), eliminovat a řešit problémy, motivovat k lepším výsledkům a vytvářet ochranu před vnějšími vlivy, aby nedocházelo k rozptylování od definovaných cílů. [10]
3. SCRUM Team Member – Pod tuto roli spadají běžní členové týmu nebo také pracovníci, kteří vykonávají jednotlivé úkoly (uživatelské příběhy), ale zároveň se mohou vyjadřovat k věcem spojeným s projektem. [10]

#### 3.4.1.5 Extrémní programování

Tato metodika využívá krátkých délek interakcí oproti tradičním metodikám, kde se délka udává v měsících nebo v případě SCRUMu v týdnech, u extrémního programování se udává ve dnech (může se jednat i období kratší například hodin). Toto krátké období potřebné pro ukončení jednotlivých iterací je právě to, co dělá tuto metodiku extrémní. [10]

Dalšími zásadními rozdíly oproti tradičním metodikám jsou testování, spolupráce a změny. Zásadním rozdílem u testování v rámci této metodiky je skutečnost, že se nečeká s testováním týden či déle, ale testuje se neustále. Při každé iteraci se navrhne a naimplementuje změna, ta se poté otestuje a nasadí. Na tomto testování se může podílet i zákazník, protože se s ním spolupracuje, jako by byl součástí vývojového týmu a v podstatě se jí i stává. U změn je rozdíl takový, že jsou běžnou součástí vývoje, protože je možné měnit požadavky i několikrát denně nebo zahazovat různé části vytvořeného kódu. [10]

V rámci této metodiky existují 4 základní hodnoty, na kterých je založena. Jedná se o komunikaci, jednoduchost, zpětnou vazbu a odvahu. [10]

Komunikace je základní hodnotou, jelikož je v rámci extrémního programování nezbytné, aby spolu všichni komunikovali (manažeři, vývojáři, zákazník a další zúčastnění) v rámci celého vývojového cyklu. Z toho důvodu se zde téměř nevyskytují schůzky a v případě problému se ihned komunikuje bez čekání na jakoukoliv schůzku. K tomuto přispívá i fakt, že tým často sedí v jedné velké místnosti. To eliminuje situaci, která může nastat v metodikách tradičního vývoje, kdy analytik napíše analýzu a programátor tuto analýzu převezme bez jakékoliv komunikace mezi nimi. [10]

Jednoduchost je zde zařazena z toho důvodu, že se nedělá to, co není nezbytně nutné. Nepíše se dokumentace, která není potřeba a nepoužívají se různé formuláře. Nepíše se ani kód, který by mohl být potencionálně v budoucnu užitečný, protože v daný okamžik není potřeba, jelikož v budoucnu se může vše změnit. [10]

Zpětná vazba je pro extrémní programování potřeba, protože se vše neustále testuje a na základě výstupu z testování se upravuje další postup. Tím se zamezuje zbytečnému čekání, protože se vše okamžitě testuje a následně případně opravuje. Tento postup je nezbytný i ze strany zákazníka. [10]

Extrémní programování se také neobejde bez odvahy, protože je nezbytná pro zahájení okamžitého řešení problému bez odkládání, jelikož zde není čas čekat na schůzku, ale okamžitě komunikovat. Odvaha je zde potřeba i pro zahození již hotové práce a začátku od znova. [10]

#### 3.4.1.6 Vývoj řízený vlastnostmi (Feature Driven Development)

Při tomto typu vývoje se nejprve navrhne doménový model, který vystihuje celý model. Ten je poté transformován na seznam vlastností nebo také na elementární funkcionality, které mají určitou hodnotu pro uživatele. [10]

Vývoj se dělí na 5 fází, z nichž 3 jsou sekvenční a 2 iterativní. Cílem sekvenčních fází je zejména vytvořit doménový model. Jednotlivé iterativní fáze trvají zpravidla 2 týdny a v každé z nich se implementují specifické vlastnosti systému. Průběžné výsledky a nové verze produktu se dávají zákazníkovi podobně jako u SCRUMu. Na rozdíl od extrémního programování je však práce jednotlivým programátorům přidělena a nevybírají si ji sami. [10]

#### 3.4.1.7 Vývoj řízený testy (Test Driven Development)

V této metodice jsou prioritou testy, které se navrhují jako první a následně je napsáno jen takové množství kódu, které je potřeba pro úspěšný průchod testem. Není tedy naprogramováno to, co nedefinují zadané testy. Pokud projdou všechny testy, je vývoj považován za úspěšný. [10]

#### 3.4.1.8 Lean development

Lean development není až tolik metodika jako souhrn pravidel sloužící pro zefektivnění a zrychlení vývojového procesu. Tato pravidla říkají: eliminovat zbytečné, zdůraznit proces učení, rozhodovat se co nejrychleji, posílit odpovědnost týmu a mít náhled na systém jako na celek. Součástí jsou zároveň různé principy a nástroje, díky kterým je možné realizovat tato pravidla. [10]

## 3.5 Multimédia

Tato kapitola bude zaměřena na vymezení pojmu multimédia a formáty multimédií, které bude možno v rámci aplikace využít.

Pojmem multimédia se označuje zobrazení informací přitažlivou a interaktivní formou za použití kombinace textu, zvuku, videa, grafiky nebo animací. Lze tak tedy označit prezentování informací za využiti počítačů. Pod multimédia spadá e-mail, videokonference, multimediální zprávy MMS a další. [14]

### 3.5.1 Součásti

#### 3.5.1.1 Text

Text je součástí většiny multimédií a může mít různé velikosti a typy fontů. [14]

#### 3.5.1.2 Grafika

S využitím grafiky lze docílit atraktivnějšího multimédia, protože lidé nemají často rádi dlouhé čtení textů. Z toho důvodu se využívá grafika, která může pomoct s vysvětlením konceptu. Grafika se dělí na 2 typy: bitmapové obrázky, vektorová grafika. Bitmapové obrázky jsou běžné obrazy, které lze pořídit například vyfotografováním nebo naskenováním. Oproti tomu vektorová grafika se nepořizuje, ale je tvořena v počítači a je tedy i editovatelná. [14]

#### 3.5.1.3 Zvuk

V případě potřeby je možné využívat zvuk. Pod zvuk spadá mluvené slovo, hudba nebo zvukové efekty. [14]

#### 3.5.1.4 Video

Video je další část, která se používá v případě multimédií. Pod tímto pojmem se skrývá pohybující se obraz. Kromě pohybujícího se obrazu se může video vyskytovat v kombinaci se zvukem. Video má sice vyšší nároky na výpočetní výkon, ale za to je možné skrze něj prezentovat velké množství informací v krátkém časovém úseku. [14]

#### 3.5.1.5 Animace

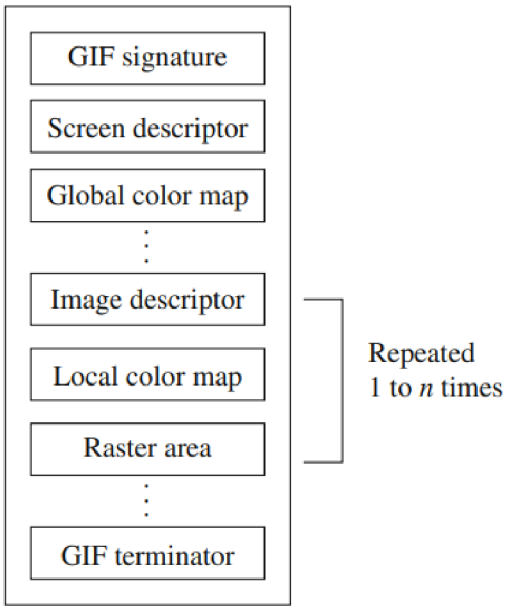
Pojmem animace se označuje proces rozpohybování statických obrazů tak, že způsobují dojem pohybu. Ve skutečnosti, se ale jedná o sekvenci obrazů, které jsou po sobě rychle zobrazeny. [14]

### 3.5.2 Použité formáty

#### 3.5.2.1 Graphics Interchange Format (GIF)

Jedná se o formát vyvinutý společnostmi UNISYS Corporation a CompuServe, který byl originálně určen pro přenos grafických obrázků s využitím telefonní linky skrze modemy. V rámci tohoto standardu je využíván Lempel-Ziv-Welch kompresní algoritmus, který byl lehce upraven tak, aby využíval seskupení řádků. Limitem tohoto formátu je to, že jej lze použít pro 8bitové obrázky, tedy maximálně takové, které využívají do 256 barev. Zajímavou vlastností formátu GIF je to, že umožňuje prokládání, neboli postupné vykreslování pixelů v řádcích vzdálených od sebe s využitím 4 průchodů. [15]

Tento formát má dvě verze. První verze se nazývá GIF87a a jedná se o relativně jednoduchý formát. [15]




*Obrázek 7 – Formát GIF87a [15]*

Na obrázku výše je zobrazena struktura formátu GIF87a. První část je podpis a skládá se ze 6 bytů. Následuje deskriptor, který je 7 bytový a obsahuje informace, které přísluší všem obrázkům v souboru jako je výška a šířka obrazu, pozadí, jehož barva je na základě indexu definována v globální barevné mapě, nebo také v tabulce barev. [15]

| Bits 7 6 5 4 3 2 1 0 | Byte # | |
|---|---|---|
| Red intensity | 1 | Red value for color index 0 |
| Green intensity | 2 | Green value for color index 0 |
| Blue intensity | 3 | Blue value for color index 0 |
| Red intensity | 4 | Red value for color index 1 |
| Green intensity | 5 | Green value for color index 1 |
| Blue intensity | 6 | Blue value for color index 1 |
| ⋮ | | (continues for remaining colors) |

*Obrázek 8 – Tabulka barev formátu GIF [15]*

V globální barevné mapě se může nacházet více informací o barvách, kdy každá barva má svůj index, který se poté dále využívá. Poté se zde nachází definice obrázku, kterých může být i více podle potřeby. [15]

Důležitou změnou ve verzi GIF89a je přidání nové řady definic rozšiřujících bloků, které umožnily podporu jednoduchých animací, průhlednosti nebo prodlevy mezi obrázky. Zároveň bylo v této verzi přidáno seřazení barev v tabulce barev, což umožňuje přednostní zobrazení nejdůležitějších barev pro případ, že dekodér nemá k dispozici velké množství barev. [15]

#### 3.5.2.2 Joint Photographic Experts Group (JPEG)

Standard pro kompresi JPEG byl vytvořen Mezinárodní organizací pro normalizaci, která se také označuje zkratkou ISO nebo neformálně Joint Photographic Experts Group na základě jejíž názvu byl tento standard také pojmenován. Principem JPEGu je využití nedokonalosti lidského zraku, díky čemuž lze docílit vysoké míry komprese. Lidské oko v kombinaci s mozkem není schopné vidět extrémně jemné detaily. V případě, že se objevuje spousta změn v rámci několika pixelů, označuje se to jako obrazový segment s vysokou prostorovou frekvencí (dochází k velkému množství změn v (x, y) prostoru). V případě barevného vidění je tento jev ještě viditelnější než ve stupních šedi. Proto jsou barevné informace u formátu JPEG vypuštěny (můžou být vypuštěny částečně nebo zprůměrovány) a následně jsou reprezentovány ve frekvenčním prostoru (u, v) místo (x, y). To v praxi znamená, že dochází k ohodnocení rychlosti změn v osách x a y od nízké k vysoké rychlosti a na základě toho je vytvořen nový obraz seskupením koeficientů nebo vah daných rychlostí. [15]

Výsledné váhy, které jsou použity pro ohodnocení změn s nízkými rychlostmi, jsou preferovány pomocí dělení nějakým velkým celým číslem a zkráceny. Po dokončení jsou nízké hodnoty vynulovány. Jakmile dojde k vynulování nízkých hodnot, použije se schéma pro efektivní reprezentování dlouhých sérií nul a obrázek je tímto způsobem zkomprimován. Protože však dochází k zahazování informací, tak se tento formát označuje za ztrátový i přes to, že existuje bezeztrátový režim. V případě JPEGu je možné si zvolit úroveň komprimace, (výchozí faktor kvality je 75) čímž bude ovlivněna velikost, ale také kvalita. [15]

#### 3.5.2.3 Portable Network Graphics (PNG)

Tento formát je určen pro tvorbu obrázku nezávisle na platformě. Zároveň je jeho cílem nahradit standard GIF, oproti kterému dokáže PNG využít na 16 bitů na pixel v každém barevném kanálu, což odpovídá 48bitové barvě, a přidat další důležité funkce.

Částečným důvodem pro vytvoření tohoto standardu bylo to, že společnosti UNISYS a CompuServe vlastnily patent na kompresní metodu Lempel-Ziv-Welch. [15]

Mezi významné změny dále patří to, že soubory jsou schopné obsahovat informace o korekci gama sloužící pro korektní zobrazení barevných obrázků a informace o alfa kanálu pro potřeby řízení průhlednosti. Oproti GIFu, který pro zobrazování pixelů používá progresivní zobrazení, které je postaveno na řádkovém prokládání se u PNG využívá postupné zobrazení ve dvourozměrném prokládání s využitím 7 průchodů skrze každý blok 8x8 obrázku. [15]

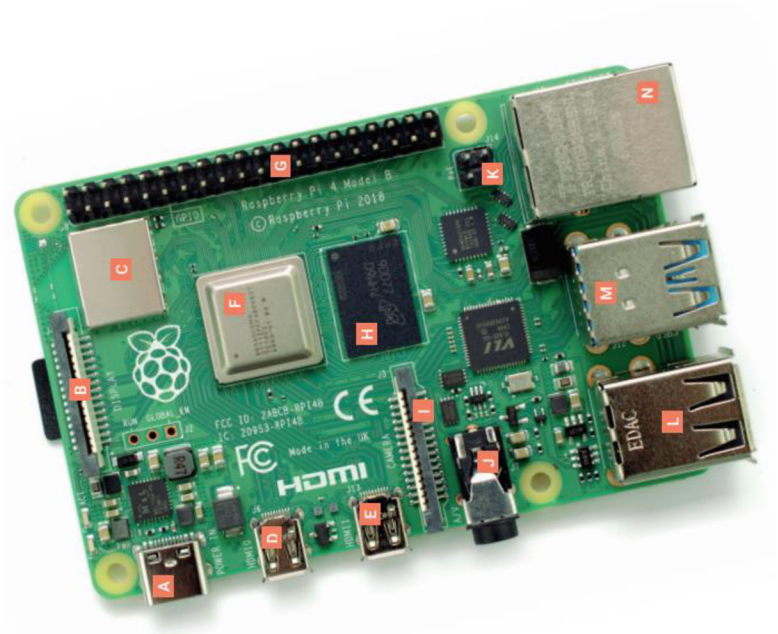
#### 3.5.2.4 MP4 (MPEG-4 část 14)

MP4 je kontejnerový formát určený pro stahování, streamování a sdílení video obsahu. Tento formát je schopný obsahovat obrázky zvuk nebo i titulky a vychází z QuckTime File Format. Důvodem, proč je tento formát označován jako kontejnerový je to, že může obsahovat velké množství typů zvuků a videí. Jedinou podmínkou je, že tato videa a zvuky musí být zakódované kodekem, který je kompatibilní s tímto formátem. [16]

V rámci formátu MP4 se využívá ztrátová komprese, ale i přes to si video zachovává vysokou kvalitu, díky čemuž je tento formát ideální pro streamování nebo stahování videí na internetu. Do videa v tomto formátu lze přidat nejrůznější typy medií od zvuku, přes obrázky, až po 3D obrázky s komplexními metadaty. Díky tomu, je možné poskytnout divákovi různé interaktivní prvky, jako jsou například navigace. [16]

U formátu MP4 není přesně dána struktura a lze ho rozšířit. Díky tomu lze vytvořit vlastní struktury a hierarchii pro data. Soubor ve formátu MP4 lze rozdělit na 2 části. První jsou data, která souvisí s médii (zvuk, obraz) a druhá část jsou metadata, která obsahují příznaky a časová razítka. Individuální části tohoto souboru nesou označení atomy a jejich minimální velikost je 8 bytů (První 2 označují jeho velikost a další 4 nesou informaci o jeho typu). [16]

## 3.6 Raspberry Pi




*Obrázek 9 – Raspberry Pi 4 Model B [17]*

Raspberry Pi je zařízení podobné počítači, telefonu nebo notebooku s tím rozdílem, že se vše potřebné nachází na jediné desce tištěných spojů přibližně s rozměrem platební karty, proto se také tento počítač označuje jednodeskový. Je možné jej využít pro podobné úkony jako běžný počítač i když tyto úkony na něm mohou trvat o něco déle. [17]

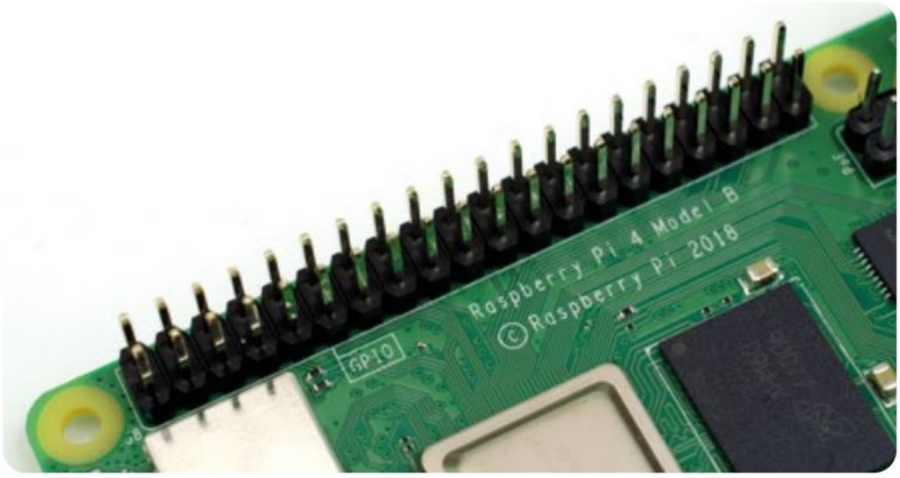
Toto zařízení vzniklo za účelem podpory praktického využívání počítačů při výuce. Zařízení z rodiny Raspberry Pi se používají v třídách, domácnostech, kancelářích, továrnách, datových centrech nebo dokonce i v bezpilotních lodích. Od původního vydání se vydalo hned několik různých modelů s různými rozdíly. Mezi takové rozdíly patří vylepšené specifikace, konfigurace pro specifické využití nebo s jinými rozměry (například Raspberry Pi Zero). Všechny tyto verze, ale mají společné to, že je možné spouštět různý software napříč všemi verzemi. To znamená, že pokud lze pustit software na jedné z verzí Raspberry Pi, je možné jej pustit i na ostatních verzích. A to se týká i operačního systému. Avšak tento software nemusí fungovat stejně rychle, jako na verzi pro kterou je určen, nicméně fungovat bude. [17]

### 3.6.1 Komponenty

Na desce tohoto jednodeskového počítače se může nacházet hned několik důležitých komponent, kdy každá z nich má svůj specifický účel. Mezi tyto komponenty patří systém na čipu (SoC), operační paměť (RAM), rádiový modul, USB řadič a integrovaný obvod pro

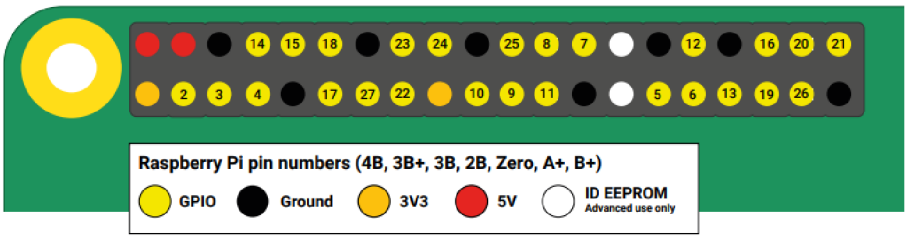
správu napájení (PMIC). V rámci SoC je zahrnuta centrální procesorová jednotka (CPU), která se stará o výpočetní operace a grafický procesor (GPU), jehož účelem jsou operace spojené s grafikou. RAM je určena pro uchovávání dat, se kterými se v daný okamžik pracuje. Tato data se v RAM nachází pouze, pokud je napájena, pokud dojde ke ztrátě napájení, nebo k vypnutí zařízení, tak jsou data RAM ztracena. V okamžik, kdy dojde k uložení, tak dochází k uložení dat na microSD kartu, která si tato data uchová oproti RAM i po vypnutí. Účelem rádiového modulu je umožnění bezdrátové komunikace. Tento modul zahrnuje jak WiFi rádio, tak Bluetooth rádio, díky čemuž je možné komunikovat s okolními zařízeními nebo senzory. USB řadič je nezbytný pro správu a provoz USB portů na Raspberry Pi. Poslední komponenta, a to integrovaný obvod pro správu napájení, má za úkol řídit napájení. To je dodáváno skrze USB port určený pro napájení Raspberry Pi. [17]

### 3.6.2 Univerzální vstupně/výstupní (GPIO) piny




*Obrázek 10 – GPIO piny [17]*

GPIO piny se na Raspberry Pi nachází pro umožnění komunikace s dalším hardwarem. Příkladem můžou být různé LED diody, tlačítka nebo i teplotní čidla, joysticky a senzory sloužící pro monitorování pulzní frekvence. [17]




*Obrázek 11 – Význam jednotlivých GPIO pinů [17]*

GPIO header na Raspberry Pi 4B, 3B+, 3B, 2B, Zero, A+ a B+ je složen celkem ze 40 pinů. Zatímco některé piny jsou určené pro poskytování napájení, jiné jsou určeny pro připojení, vyhrazeny pro hardware rozšiřující funkce nebo pro fyzické výpočetní projekty. Tyto piny se dále dělí do typových kategorií 3V3, 5V, Zem, GPIO, ID EEPROM. První 3 kategorie (3V3, 5V a zem) slouží pro napájení, kdy 3V3 poskytuje napětí 3,3 V, které odpovídá internímu napětí Raspberry Pi, 5V, jak již název napovídá poskytuje napětí 5 V. Zem slouží pro uzavření obvodu. Piny z kategorie GPIO jsou univerzální vstupně výstupní piny označené čísly od 2 do 27. Piny z poslední kategorie ID EEPROM jsou vyhrazeny pro využití takzvaných HAT (Hardware Attached on Top) nebo jiného příslušenství. [17]

## 3.7 Hypertext Markup Language

Hypertext Markup Language nebo také zkráceně HTML je značkovací sloužící pro určení struktury webový stránek, sám o sobě neumožňuje úpravu vzhledu stránky nebo přidání interaktivity. Pro přidání interaktivity a úpravy vzhledu se HTML často využívá v kombinaci s Kaskádovými styly nebo skriptovacím jazykem Javascript. [18], [19]

Struktura HTML se vytváří pomocí takzvaných tagů, kdy každý má svůj specifický účel. Většina tagů se zapisuje ve formě 2 tagů (počáteční a uzavírací – <tag>obsah</tag>), ty se nazývají párové. Poté jsou tagy, které uzavírací tag nemají a ty se nazývají nepárové (například tag <img>, který slouží pro vykreslení obrázku). Kromě tagů existují také atributy, které se zapisují v rámci tagů (například  obsah ), kde class je atribut a text je hodnota tohoto atributu. Atributy jsou určeny pro uchovávání informací o daném tagu nebo označení tohoto tagu, které lze využít pro stylování nebo jinou manipulaci. [18], [19]

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Definition of HTML</title>
  </head>
  <body>
    <!--Page content such as text and images goes in here-->
  </body>
</html>
```

*Obrázek 12 – základní struktura html stránky [18]*

Na obrázku výše je vyobrazeno základní uspořádání HTML stránky. Počáteční tag <!DOCTYPE html> slouží pro specifikaci toho, že soubor je ve formátu HTML5, pod ním

následuje párový tag html, který se musí nacházet ve všech HTML dokumentech. Obsah tagu html se dělí na 2 důležité tagy head a body. V tagu head se nachází informace o stránce, stylech. Oproti tomu tag body obsahuje obsah, který je prezentován uživateli. [18], [19]

## 3.8 Kaskádové styly

Kaskádové styly nebo zkráceně CSS (Cascading Style Sheets) se používá definování vzhledu stránky. V rámci CSS se používají pro změny stylů takzvané vlastnosti, kdy každá vlastnost slouží pro úpravu specifického formátu jednotlivých prvků. Jednotlivé styly se mohou buď nacházet v hlavičce jednotlivých souborů nebo jsou definovány v separátním souboru, který se napojí na danou webovou stránku. [20]

### 3.8.1 Princip

V rámci CSS se využívají pravidla, která se aplikují na prvky vyskytující se na webové stránce. Tyto pravidla aplikuje prohlížeč na jednotlivé prvky s využitím takzvaných selektorů. Tyto selektory se dělí na několik typů a mezi základní patří typové a univerzální. [20]

Pod typové selektory lze zařadit pravidla, která vybírají typ prvku (například hr). Pokud je vytvořeno pravidlo pro typový selektor, tak jsou pravidla aplikována na všechny instance tohoto typu. Například pravidlo: „hr {width: 50%;}“ nastaví šířku na 50% šířky kontejneru, který daný element obklopuje pro všechny elementy typu hr. [20]

Oproti tomu univerzální selektor, který je označen * slouží pro nastavení pravidla pro všechny elementy na stránce. [20]

Existují však i další selektory, mezi které patří selektory tříd a selektory ID. Tyto selektory fungují na podobném principu s tím rozdílem, že se liší lehce způsob zápisu a aplikují se pouze na elementy, které mají definovanou specifickou hodnotu v rámci atributu *class* nebo *id*. Selektor tříd se zapisuje s tečkou (například .nazev-tridy) na začátku a za ní následuje název třídy. V případě selektoru ID se na začátku vyskytuje hashtag (například #nazev-id). V případě ID je nezbytné dodržet v rámci jedné stránky unikátnost. Nesmí se tedy na jedné stránce vyskytovat více elementů se stejným ID. [20]

Jednotlivá pravidla mohou být sdílené napříč všemi selektory. Lze toho docílit využitím čárky, která jednotlivé selektory oddělí. Následující pravidlo: „.red, .conspicuous, h1 {background-color: tomato;}“ je tedy aplikováno na element s třídou .red, .conspicuous, a instance elementu typu h1. [20]

## 3.9 Hypertext Preprocessor

Hypertext Preprocessor neboli PHP je skriptovací jazyk určený zejména pro tvorbu webových stránek s tím, že jej lze vložit přímo do HTML. To je možné z toho důvodu, že kód je proveden na straně serveru a výsledek je vrácen do prohlížeče uživatele. [21], [22]

Aby bylo možné využívat PHP, tak je nezbytné do souboru, kde bude kód umístěn, vložit značku <?php, která označuje začátek skriptu a ?>, která označuje konec. Cokoliv mezi těmito 2 značkami se zpracuje jako PHP skript. Kromě toho je potřeba, aby měl soubor i požadovanou koncovku tedy .php, bez toho se kód neprovede. Soubory s koncovkou .php však nejsou vyhrazeny pouze pro PHP, ale mohou obsahovat veškerý kód jako běžné html soubory. [22], [23]

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

*Obrázek 13 – Soubor .php ve spojení s HTML [22]*

Na obrázku výše lze vidět způsob zápisu HTML a PHP dohromady. Na tomto konkrétním obrázku je zapsán kód, který na stránce vypíše text „Hello World!“. [22]

### 3.9.1 Základní syntax

Aby bylo možné v PHP vytvářet funkční kód, tak se musí dodržet pravidla udávající správnou syntaxi. Nejzákladnější z těchto pravidel je, že jednotlivé příkazy musí být ukončeny středníkem. V rámci PHP není potřeba dbát na velikost písmen u podmínek, názvů tříd, funkcí nebo funkcí definovaných uživatelem, protože PHP není v těchto případech citlivé na velikost písmen. Naopak názvy proměnných jsou na velikost písmen citlivé a musí být velikost jednotlivých písmen dodržena. [22]

Komentování se v PHP může realizovat několika způsoby na základě počtu řádků. Komentáře na jeden řádek lze vytvářet buď pomocí značky # nebo //. Pro víceřádkové

komentáře se využívají značky /* pro začátek komentování a */ pro konec komentované sekce. [23]

PHP není senzitivní na mezery v kódu, to znamená, že bez ohledu na to, kolik je za sebou v kódu mezer, tak bude vykreslení a zpracování stejné. [23]

```
if (3 == 2 + 1)
   print("Good - I haven't totally lost my mind.<br>");

if (3 == 2 + 1) {
   print("Good - I haven't totally");
   print("lost my mind.<br>");
}
```

*Obrázek 14 – Bloky kódu v PHP [23]*

V PHP je možné zapisovat bloky kódu několika způsoby, jak je vyobrazeno na obrázku výše. Prvním z nich je s využitím složených závorek (druhá varianta na obrázku výše) a pomocí odsazení (první varianta na obrázku výše) podobně jako v programovacím jazyce Python s tím rozdílem, že se na konci předchozího řádku nenachází dvojtečka. [23]

## 3.10 Python

Python je vyšší programovací jazyk, který je interpretovaný a objektově orientovaný. Označení vyšší programovací jazyk udává, že není potřeba se starat o činnosti, které se jinak v nižších programovacích jazycích musí řešit (například správa paměti). Zároveň se jedná o interpretovaný programovací jazyk, takže není potřeba jej kompilovat jako v případě kompilovaných jazyků (například c a c++). To v praxi znamená, že se program spouští přímo se zdrojového kódu, který se za běhu převádí do nativního jazyka, kterému dané zařízení rozumí. Mezi hlavní výhody jazyka Python patří to, že je nezávislý na platformě, a tudíž je možné daný kód spustit jak na Windows, tak Linuxu nebo Mac OS. [24], [25], [26]

### 3.10.1 Základní syntax

V programovacím jazyce Python se pro jednořádkové komentování používá symbol #, ale u víceřádkového je už oproti jiným jazykům rozdíl. Na rozdíl od běžného znaku /* pro zahájení komentování a */ pro ukončení komentování se využívají pro zahájení znaky """, které slouží rovněž i pro ukončení. [27]

Pro ukončení příkazu se v programovacím jazyku Python běžně nepoužívá středník, ale ukončuje se koncem řádku. Pokud má být příkaz přes větší množství řádku, tak se musí na konec řádku umístit zpětné lomítko, které se v programovacím jazyce Python nachází přesně pro tento účel. Avšak to, že se středník běžně nevyužívá, neznamená, že se v jazyku

Python nenachází nebo jej nelze využít. Středník lze využít, pokud se definuje větší množství příkazů na jednom řádku. [27]

```
lower = []; upper = []
```

*Obrázek 15 – Zápis 2 příkazů na jeden řádek v jazyce Python [27]*

Důležitým rozdílem je způsob zápisu jednotlivých bloků v tomto programovacím jazyce. Oproti jiným se pro oddělení bloků kódu nevyužívají složené závorky ({}), ale odsazení a umístění dvojtečky na konec předchozího řádku jako je znázorněno na obrázku níže. [27]

```
for i in range(10):
    if i < midpoint:
        lower.append(i)
    else:
        upper.append(i)
```

*Obrázek 16 – Oddělování bloků kódu v jazyce Python [27]*

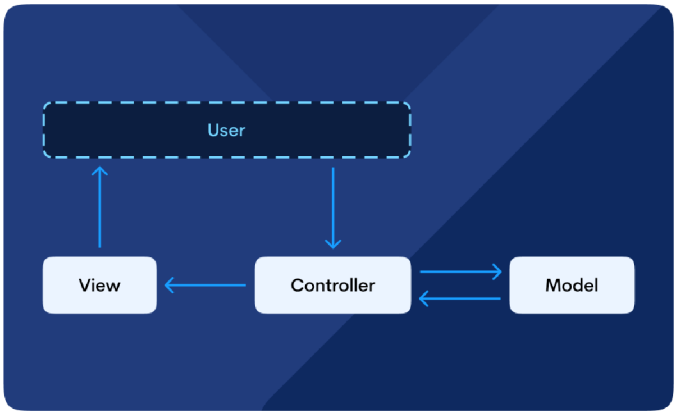
Odsazení na začátku řádku má význam pro funkci kódu, ale lze jej využít v rámci řádku, tedy po začátku. To nemá sice vliv na funkcionalitu, ale lze využít pro zlepšení čitelnosti v kódu. Na obrázku níže je patrné odsazení v rámci řádku a jeho vliv na zlepšení čitelnosti, ale je potřeba brát ohled na to, aby se toto odsazení nevyužívalo nadměrně, protože v ten okamžik bude kód vypadat jako poslední řádek v obrázku níže. [27]

```
x=1+2
x = 1 + 2
x            =        1    +              2
```

*Obrázek 17 – Odsazování v řádku u jazyka Python [27]*

## 3.11 PHP Framework Symfony

Symfony je webový open-source framework založený na skriptovacím jazyce PHP. Tento framework založil Fabien Potencier koncem roku 2004 a o jeho provoz se stará společnost SensioLabs. Jeho hlavní výhodou je, že obsahuje nástroje na správu chyb a ladění kódu. Kromě toho dále obsahuje komponenty a balíčky, které se často používají opakovaně a tím předchází potřebě psát kód, který byl již dříve vytvořen. [28]




*Obrázek 18 – Schéma architektury MVC [28]*

Jeho návrh je založen na architektuře Model-view-controller (MVC), která se, jak již název napovídá, skládá ze tří částí. [28], [29]

První částí je model, jehož účelem je starat se o uchovávání a načítání dat, také může obsahovat validaci či jiné operace spojené s daty. [28], [29]

Druhou částí je view nebo také pohled, který je určen k poskytování rozhraní pro aplikaci, díky čemuž je možné s aplikací provádět interakci. [28], [29]

Poslední částí je controller, který obsahuje aplikační logiku a slouží jako komunikační rozhraní mezi vrstvami view a model. [28], [29]

```
symfony_project/
├─ assets/
├─ bin/
│  └─ console
├─ config/
│  ├─ packages/
│  └─ services.yaml
├─ migrations/
├─ public/
│  ├─ build/
│  └─ index.php
├─ src/
│  ├─ Command/
│  ├─ Controller/
│  ├─ DataFixtures/
│  ├─ Entity/
│  ├─ EventSubscriber/
│  ├─ Form/
│  ├─ Repository/
│  ├─ Security/
│  └─ Twig/
├─ templates/
├─ tests/
├─ translations/
├─ var/
│  ├─ cache/
│  └─ log/
└─ vendor/
```

*Obrázek 19 – Struktura složky frameworku Symfony projektu [29]*

Na Obrázek 19 – Struktura složky frameworku Symfony projektu [29] je znázorněna základní struktura projektu vytvořeného v PHP frameworku Symfony. Ve složce *assets* se nachází, jak již název napovídá assets, tedy různé obrázky, styly a další, které se v případě, že se využívá webpack po zkompilování, budou nacházet ve složce *public/build*. Ve složce *var* se nachází mezipaměť a záznamy aktivity nebo chyb. Ve složce *src* se poté nachází další podsložky, soubory a zejména složka *controller*, ve které se nachází jednotlivé řadiče. [29]

### 3.11.1 Doctrine ORM

Doctrine ORM je nástroj sloužící pro mapování objektů na tabulky z databáze bez využití SQL. Konfiguraci entit lze realizovat prostřednictvím anotací, XML nebo YAML. Jednotlivé entity se nachází ve složce *src/Entity* a repozitáře, ve kterých lze vytvářet vlastní dotazy, se nachází ve složce *src/Repository*. [29]

### 3.11.2 Twig

Jako šablonovací systém se využívá Twig, který dokáže využívat v šablonách dědění nebo různé filtry, díky čemuž je možné vytvářet HTML kód generovaný na straně serveru. [29]

```
{% extends 'base.html.twig' %}

{% block stylesheets %}
    {{ parent() }}
    {{ encore_entry_link_tags('index_index') }}
{% endblock %}

{% block main %}

    <section>
        <div class="header-color header-resize" style="...">
            <div class="hero-text">
                <h1>{% trans %}page.index.s1.h.ulinkyprojects{% endtrans %}</h1>
                {% trans %}page.index.s1.t.subtitle{% endtrans %}
            </div>
        </div>
    </section>

    <section class="section">
        <h2>{% trans %}page.index.s2.h.welcome{% endtrans %}</h2>
    </section>

{% endblock %}
```

*Obrázek 20 – Příklad kódu s využitím Twig [29]*

### 3.11.3 Překlady

V Symfony lze vytvářet překlady textů bez jakýchkoliv obtíží v několika formátech (YAML, XLF, JSON, INI, PO a PHP). Překlady se nachází ve složce *translations* a Symfony obsahuje i doplňující nástroje, které mohou například s využitím příkazu pomoci s dohledáním chybějících překladů. [29]

### 3.11.4 Profiler Toolbar


*Obrázek 21 – Symfony Debugging toolbar [29]*

Pro pomoc s testováním a laděním se v Symfony nachází ladící nástroj nazývaný Profile, skrze který lze sledovat jednotlivé akce, obsah posílaných hlaviček, detaily o mezipaměti nebo jaké byly provedeny databázové dotazy a další. [29]

### 3.11.5 Příkazy

V rámci tohoto frameworku se nachází velké množství příkazů sloužící pro usnadnění správy webové aplikace. Mezi takové příkazy patří čištění mezipaměti, migrace databáze nebo již výše zmíněné hledání chybějících překladů. Pokud to aplikace vyžaduje, je možné vytvořit i vlastní příkazy. [29]

```
www-data@03e2c9817b93:~/html$ php bin/console cache:clear

 // Clearing the cache for the dev environment with debug true

[OK] Cache for the "dev" environment (debug=true) was successfully cleared.
```

*Obrázek 22 – Příklad příkazu pro vymazání mezipaměti v Symfony [29]*

### 3.11.6 Router

Router se stará o mapování jednotlivých URL adres na správný controller. S jeho využitím lze nastavit pro každou URL adresu individuální konfiguraci, která se zapisuje pomocí anotací, YAML nebo XML. Mezi takovou individuální konfiguraci patří parametry v adresách (například jazyk) nebo správa přístupu na danou adresu. [29]

### 3.11.7 Framework na vytváření testů PHPUnit

Aby bylo možné ověřit správnou funkcionalitu, tak je v Symfony možné vytvářet Unit testy a pro potřeby rozsáhlejšího testování je možné využít i předpřipravené komponenty. [29]

### 3.11.8 Konfigurace

Aby bylo možné snadno vytvářet webové aplikace, je důležitá i možnost konfigurace. V PHP frameworku Symfony je konfigurace řešena skrze konfigurační soubory nacházející se ve složce *config*, kde se nachází nastavení pro překlady, databázi, šablony, mezipaměť, směrování a jiné. [29]

### 3.11.9 Systém balíčků

Důležitou částí je systém pro správu balíčků, díky kterému lze vytvářet a instalovat různé pluginy (balíčky) do aplikace skrze správce balíčků určeného pro PHP nazývaného composer. Veškeré nainstalované balíčky se konfigurují ve složce *config*. [29]

# 4 Vlastní práce

## 4.1 Vyhodnocení analýzy

### 4.1.1 Existující řešení

Z analýzy existujících řešení vyplynulo, že již existují řešení, která jsou založena na Raspberry Pi, nicméně buď nejsou kompletním řešením (Raspberry Pi režim kiosku) nebo jsou zpoplatněna za použitá zařízení a přehrávače nebo v případě jejich open-source verzí jsou založené na jazycích, které by nemusely všem vyhovovat. [4], [5], [6]

V případě profesionálních řešení od společnosti PROJEKTA, nelze dohledat, zda je lze centrálně spravovat, ale na rozdíl od ostatních analyzovaných řešení je možné je využívat i ve venkovním prostředí na základě modelu. [6]

Aby tedy byl výsledek této práce použitelnou alternativou, měl by být veřejně dostupný s tím, že si jej může kdokoliv bezúplatně upravit podle svých potřeb. Řešení by zároveň mělo umožňovat využití více zařízení, seznamů k přehrání a nahrávání obsahu s tím, že limitem je kapacita hostingu.

### 4.1.2 Metodiky vývoje softwaru

Na základě analýzy metodik vývoje softwaru vyplynulo, že pro efektivní a úspěšný vývoj softwaru je dobré využívat některou z metodik vývoje. Každá z metodik má své pro a proti, které je nezbytné vzít při výběru v potaz. Zároveň jsou jednotlivé metodiky vhodné pro jiné typy projektů, nicméně vzhledem k povaze tohoto projektu se jeví jako ideální Vodopádová metodika. [10]

### 4.1.3 Vybrané jazyky, framework a hardware

Tato diplomová práce využívá hned několik jazyků, jako jsou PHP, Python, PHP framework Symfony a hardware v podobě Raspberry Pi. Nicméně je vhodné zmínit, proč byly tyto jazyky, framework a Raspberry Pi vlastně zvoleny.

Hlavním důvodem pro zvolení Raspberry Pi byly rozměry, které odpovídají přibližně velikosti platební karty. Zařízení lze tedy využít i v prostorech, kde by jinak nebylo možné využít plnohodnotný počítač. [17]

V případě programovacího jazyka Python byly klíčovými faktory nezávislost na platformě, jednoduchá syntaxe a jelikož se jedná o vyšší programovací jazyk, který ulehčuje vývoj z důvodu absence potřeby pro manuální správu paměti. [24], [25], [26], [27]

Pro webové rozhrání byl vybrán jazyk PHP z důvodu, že je jednoduchý na naučení, je stabilní, rychlý, k dubnu 2022 využívalo až 77,6 % všech webových stránek na internetu a je zejména nenáročný na vývoj. Teoreticky se jej tedy může naučit kdokoliv. A aby se vývoj tohoto projektu ulehčil ještě více, došlo k využití PHP frameworku Symfony. Ten obsahuje další nástroje ulehčující vývoj a zejména komponenty a balíčky eliminující potřebu opětovného psaní kódu, který řeší již dříve vytvořenému kódu. [28], [29], [30]

## 4.2 Zvolená metodika

V rámci kapitoly 3.4.1 byly analyzovány metodiky vývoje softwaru. Z analyzovaných metodik byla vzhledem způsobu tvorby vybraná jako vhodná vodopádová metodika. [10], [11], [12]

Nejprve byly navrhnuty požadavky na to, co by měla daná aplikace řešit, poté na základě těchto požadavku byla analyzována existující řešení a jejich přístup k problematice. [10], [11], [12]

Z výstupu této analýzy byl poté zhotoven návrh, ze kterého se vycházelo při implementaci. Následně bylo vše otestováno a nasazeno do produkčního prostředí.

## 4.3 Funkční a nefunkční požadavky

### 4.3.1 Funkční požadavky

Na základě výše vypracované analýzy by mělo řešení dodržovat následující funkční požadavky:

- Správa kiosků
    - Výpis kiosků
    - Přidávání a editace
    - Mazání
    - Přiřazení seznamu k přehrání
    - Stažení softwaru pro přehrávání
- Správa multimédií
    - Výpis multimédií
    - Detail
    - Přidávání a editace
    - Mazání
- Správa seznamů k přehrání

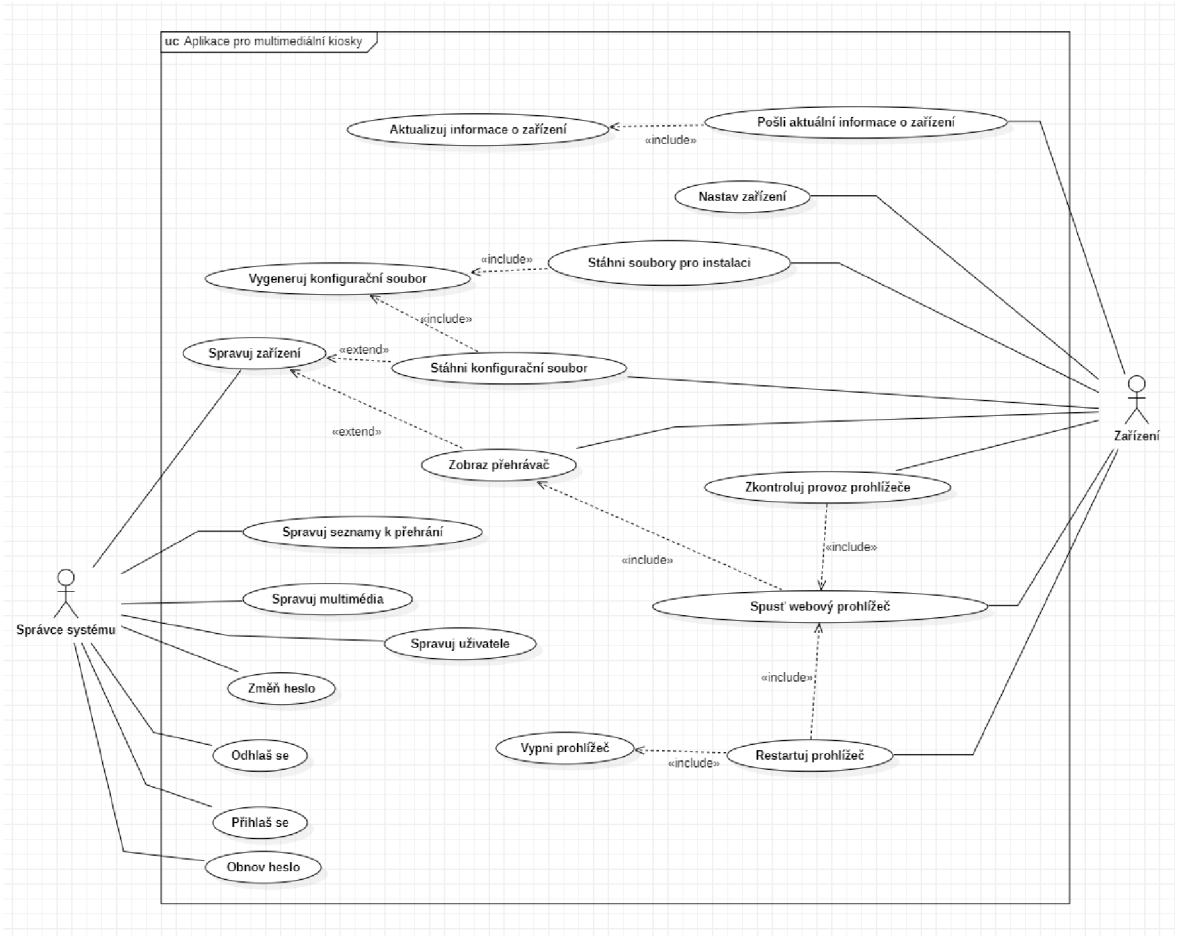
    - Výpis seznamů k přehrání
    - Přidávání a editace
    - Mazání
    - Přidávání a plánování přehrávaných multimédií do seznamu
- Správa uživatelů
    - Přidání a editace
    - Mazání
- Změna hesla
- Přihlášení
- Odhlášení
- Obnovení hesla

### 4.3.2 Nefunkční požadavky

Na základě výše vypracované analýzy by mělo řešení splňovat níže zmíněno nefunkční požadavky:

- Zabezpečená administrace
- Obnovení přehrávání v případě, že vypadne elektrický proud
- Jednoduché a přívětivě uživatelské rozhraní
- Snadná konfigurace zařízení
- Nízké náklady

## 4.4 Use Case diagram




*Obrázek 23 – Use Case diagram aplikace*

Na obrázku výše je znázorněn Use Case diagram pro webovou administraci. V tomto diagramu se vyskytují dva aktéři. Prvním z nich je správce systému, tedy uživatel, který přistupuje do administrace a spravuje v ní všechny nezbytné položky. Druhým aktérem je zařízení nebo také kiosek, který přehrává to, co je nastaveno v administraci.

## 4.5 Hardware

Jako hardware pro jednotlivé kiosky slouží Raspberry Pi. Není třeba využívat pouze konkrétní model, ale lze si vybrat téměř jakékoliv Raspberry Pi z nabídky internetových obchodů. Nicméně je potřeba brát ohled na to, aby pro správný a plynulý chod mělo zařízení alespoň 1 GB paměti RAM. Lze využít i zařízení, které má paměti méně, ale chod prezentace a odezva nemusí být ideální.

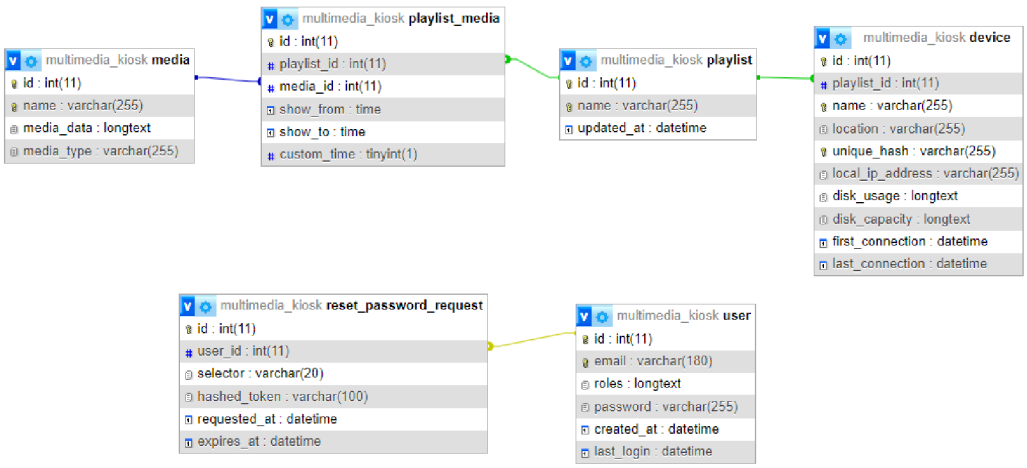
V rámci práce bylo vyzkoušeno Raspberry Pi 2W a Raspberry Pi 3B+ s 1 GB paměti RAM. Z testu bylo patrné, že Raspberry Pi 2W vzhledem k pomalé odezvě není ideálním

řešením, protože nelze zaručit správnou funkcionalitu a plnění role jako multimediálního kiosku.

Oproti tomu Raspberry Pi 3B+ se jeví jako ideální doporučené minimum. Na základě těchto testů lze pro kiosek využít jakékoliv Raspberry Pi s alespoň 1 GB paměti RAM.

## 4.6 Databáze

Pro potřeby webového rozhraní a zařízení sloužících pro přehrávání bylo potřeba navrhnout databázovou strukturu, která by byla schopna uchovávat veškeré potřebné informace o zařízeních, uživatelích, multimédiích a seznamech k přehrání. Z toho důvodu byla vytvořena následující struktura tabulek, která je schopna uchovávat všechny tyto informace.




*Obrázek 24 – Grafické znázornění tabulek v databázi a jejich propojení*

### 4.6.1 Tabulka device

Tato tabulka slouží pro uchovávání informací o jednotlivých zařízeních, která se v systému nachází. V tabulce se nachází následující sloupce:

*Tabulka 1 – tabulka device*

| Název sloupce | Typ sloupce | Popis |
|---|---|---|
| Id | Int | Obsahuje unikátní celočíselný Identifikátor záznamu. |
| Playlist_id | Int | Obsahuje cizí klíč z tabulky playlist, který jednoznačně identifikuje seznam |

| | | |
|---|---|---|
| | | k přehrání přehrávaný na daném zařízení. |
| Name | Varchar | Obsahuje název zařízení, který musí být unikátní. |
| Location | Varchar | Slouží pro uchování krátkého popisu o tom, kde se zařízení nachází. |
| Unique_hash | Varchar | V tomto sloupci se nachází unikátní hash, který se využívá pro komunikaci mezi serverem a zařízením. |
| Local_ip_address | Varchar | Tento sloupec slouží pro ukládání informace o IP adrese, kterou má zařízení přidělenou v dané lokální síti. |
| Disk_usage | Longtext | V tomto sloupci se uchovávají informace o využití disku v procentech. |
| Disk_capacity | Longtext | Tento sloupec obsahuje informaci o kapacitě uložiště na zařízení. Tato informace je uchovávána v bytech. |
| First_connection | Datetime | Obsahuje datum a čas prvního připojení zařízení k serveru. |
| Last_connection | Datetime | Obsahuje datum a čas posledního připojení zařízení k serveru. |

### 4.6.2 Tabulka media

Tabulka media slouží pro uchovávání dat o jednotlivých médiích jako je název, typ a data pro dané médium. Na základě toho obsahuje následující sloupce:

*Tabulka 2 – tabulka media*

| **Název sloupce** | **Typ sloupce** | **Popis** |
|---|---|---|
| Id | Int | Obsahuje unikátní celočíselný Identifikátor záznamu. |
| Name | Varchar | V tomto sloupci se nachází název jednotlivých multimédií. Pro jednotlivá multimédia musí být název unikátní. |
| Media_data | longtext | V tomto sloupci se nachází data pro zobrazení multimédia. Obsah závisí na hodnotě ve sloupci media_type, který určuje, zda se zde nachází pouze název nebo URL adresa. |
| Media_type | varchar | Tento sloupec obsahuje typ multimédia. Možné hodnoty jsou: IMAGE, WEBSITE, VIDEO a YOUTUBE. |

### 4.6.3 Tabulka playlist

V tabulce playlist se nachází jednotlivé seznamy k přehrání a nezbytné informace o nich. Pro tento účel obsahuje tabulka následující sloupce:

*Tabulka 3 – tabulka playlist*

| Název sloupce | Typ sloupce | Popis |
|---|---|---|
| Id | Int | Obsahuje unikátní celočíselný Identifikátor záznamu. |
| name | Varchar | Obsahuje unikátní název seznamu k přehrání. |
| Updated_at | Datetime | V tomto sloupci se nachází datum a čas, kdy byl seznam k přehrání naposledy upraven. |

### 4.6.4 Tabulka playlist_media

Tabulka playlist_media slouží pro propojení jednotlivých multimédií a seznamů k přehrání. V rámci této tabulky jsou rovněž obsaženy parametry přehrávání daných multimédií v daném seznamu.

*Tabulka 4 – tabulka playlist_media*

| Název sloupce | Typ sloupce | Popis |
|---|---|---|
| Id | Int | Obsahuje unikátní celočíselný Identifikátor záznamu. |
| Playlist_id | Int | Obsahuje cizí klíč z tabulky playlist, který jednoznačně identifikuje seznam k přehrání, ke kterému je přiřazené dané multimédium. |
| Media_id | Int | Obsahuje cizí klíč z tabulky media, který identifikuje |

| | | |
|---|---|---|
| | | multimédium, které je přiřazeno k danému záznamu. |
| Show_from | Time | Obsahuje časovou informaci o tom, od kolika hodin a minut se má dané multimédium v definovaném seznamu k přehrání přehrávat. |
| Show_to | Time | Obsahuje časovou informaci o tom, do kolika hodin a minut se má dané multimédium v definovaném seznamu k přehrání přehrávat. |
| Custom_time | Tinyint | Obsahuje hodnoty 0 (false) nebo 1 (true) identifikující záznam, který se v případě 1 přehrává celý den, nebo v případě 0 pouze ve specifikovaném časovém úseku. |

### 4.6.5 Tabulka reset_password_request

Tato tabulka obsahuje informace k žádostem o obnovení hesla do administrace pro jednotlivé uživatele.

*Tabulka 5 – tabulka reset_password_request*

| **Název sloupce** | **Typ sloupce** | **Popis** |
|---|---|---|
| Id | Int | Obsahuje unikátní celočíselný Identifikátor záznamu. |

| | | |
|---|---|---|
| User_id | Int | Obsahuje cizí klíč odkazující na uživatele, který si zažádal o obnovení hesla. |
| Selector | Varchar | Obsahuje náhodný text. |
| Hashed_token | Varchar | Obsahuje zahashovaný token využívaný pro ověření žádosti. |
| Requested_at | Datetime | Obsahuje datum a čas zažádání o obnovení hesla. |
| Expires_at | Datetime | Obsahuje datum a čas, kdy dojde k expiraci odkazu pro obnovení hesla. |

### 4.6.6 Tabulka user

Tabulka user obsahuje informace o jednotlivých uživatelích, kteří mají přístup do systému a mohou spravovat jednotlivá zařízení, seznamy k přehrání, multimédia a uživatele.

*Tabulka 6 – tabulka user*

| **Název sloupce** | **Typ sloupce** | **Popis** |
|---|---|---|
| Id | Int | Obsahuje unikátní celočíselný Identifikátor záznamu. |
| Email | Varchar | Obsahuje e-mail jednotlivých uživatelů. Tento email musí být v tabulce unikátní. |
| Roles | Longtext | V tomto sloupci se nachází pole rolí. |
| Password | Varchar | Obsahuje zahashované heslo uživatele. |

| | | |
|---|---|---|
| Created_at | Datetime | Tento sloupec obsahuje datum a čas vytvoření uživatelského účtu. |
| Last_login | Datetime | V rámci tohoto sloupce se uchovává datum a čas posledního přihlášení uživatele. |

## 4.7 Webové rozhraní

Webové rozhraní je založeno na PHP frameworku Symfony a využívá šablonu AdminLTE [31], která obsahuje velké množství možností a integrovaných knihoven. Samotná administrace je přístupná po napsání domény a */admin*. V případě, že uživatel není přihlášen, je přesměrován na přihlašovací formulář. V opačném případě je přesměrován na záložku se zařízeními.

Samotná administrace je rozdělena do několika základních kategorií. Mezi tyto sekce patří zařízení, seznamy k přehrání, multimédia, uživatelé a změna hesla. Tyto sekce byly navrženy tak, aby co v nejpřehlednější podobě bylo možné prezentovat informace o jednotlivých zařízeních, seznamech k přehrání, multimédiích a uživatelích.

K administraci je možné za účelem testování přistupovat na adrese https://dp.petrkruntorad.cz/admin s uživatelským účtem „no-reply@petrkruntorad.cz“ a heslem „Abcdef0/“. A celý projekt lze poté stáhnout z platformy GitHub na této adrese: https://github.com/petrkruntorad/dp-thesis.

### 4.7.1 Návrh webového rozhraní

Při návrhu webového rozhrání bylo dbáno na to, aby bylo webové rozhraní co nejpřehlednější a uživatelsky nejpřívětivější. Responzivitu, drobečkovou navigaci a některé další aspekty řeší zvolená šablona AdminLTE [31], a proto nebylo potřeba vše řešit od základu. Zároveň nebylo potřeba řešit SEO optimalizaci, protože webová administrace není určena pro veřejnost, ale pouze pro soukromé využití. Z toho důvodu je přístup řešen skrze uživatelské účty a není možné se přímo do systému registrovat. Je možné se přihlásit pouze na pozvání z administrace uživatelem, který do ní má přístup. Kromě toho veškerá

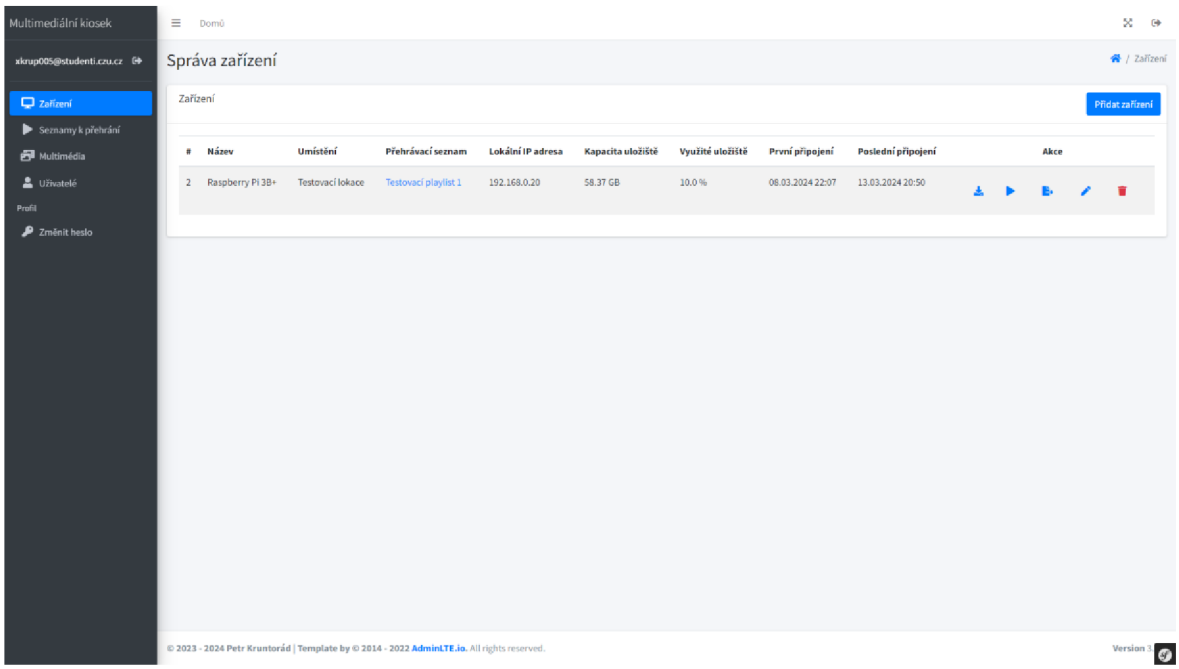
komunikace probíhá vynuceně skrze protokol https, kdy je komunikace šifrovaná pro zamezení jejího odposlechu.

Pro přehlednost jsou veškerá data vypsána v tabulkách, které můžou mít maximálně 10 položek na stránku a v případě, že je více jak 10 záznamů, tak je zobrazeno stránkování. Aby řádky v tabulce nesplývaly, tak je nastaveno rozdílné pozadí pro liché řádky. Správa jednotlivých záznamů je řešena skrze tlačítka s ikonami, kde každé má vlastní popisek po najetí kurzorem. Kromě toho tlačítka sloužící pro mazání mají nastavená vyskakovací okna, která musí uživatel pro smazání potvrdit, to předchází nechtěnému vymazání záznamů.

### 4.7.2 Požadavky na provoz

Aby bylo možné využívat webové rozhraní, je potřeba aby hosting splňoval několik parametrů. Takovými parametry je dostupnost MySQL nebo MariaDB databáze a přítomnost PHP ve verzi >=8.1.

### 4.7.3 Zařízení

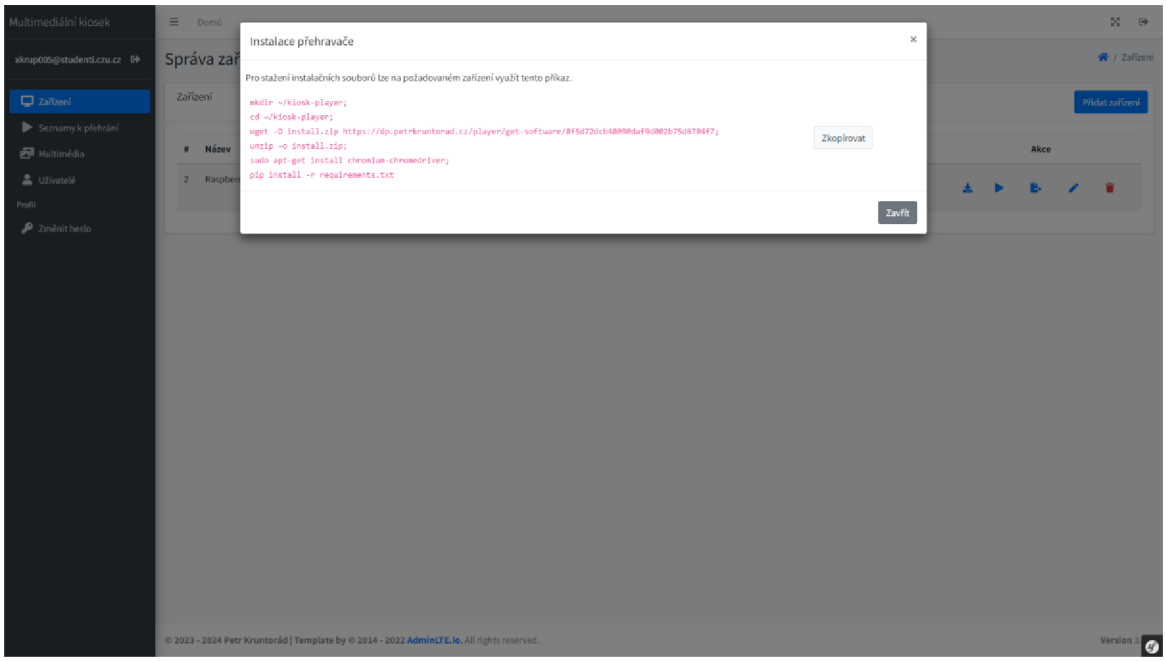



*Obrázek 25 – Stránka zařízení*

Tato kategorie obsahuje výpis a správu jednotlivých zařízení nebo také kiosků, které jsou propojeny s danou administrací.

Kromě správy jednotlivých kiosků je možné u jednotlivých kiosků pomocí doplňujících tlačítek realizovat i další operace.

Prvním těchto tlačítek využívá tuto ikonu . Jejím účelem je otevření vyskakovacího okna, které obsahuje příkazy sloužící pro stažení potřebných skriptů do zařízení a instalaci všech nezbytných závislostí.



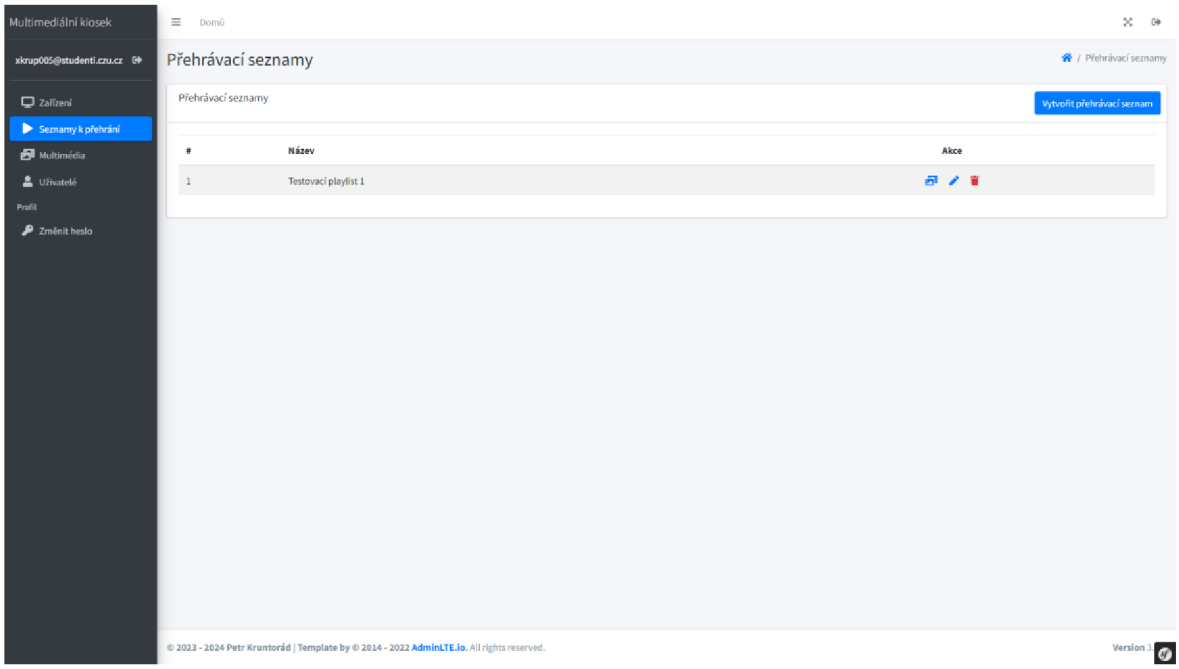

*Obrázek 26 – Stránka zařízení – vyskakovací okno*

Příkazy jsou zde vypsány pro potřeby ručního přepsání nebo je možné je zkopírovat pomocí tlačítka „Zkopírovat“. Při použití těchto příkazů na zařízení dojde nejprve k vytvoření složky s názvem kiosk-player v uživatelově domovském adresáři, do které poté příkazový řádek vstoupí, stáhne z administrace skripty v archivu zip, který je následně rozbalen. Poté se nainstaluje balíček chromium-chromedriver, který je nezbytný pro spuštění přehrávače a veškeré nezbytné závislosti specifikované v souboru requirements.txt.

Dalším tlačítkem je tlačítko s ikonou přehrát ( ), které otevře v aktuálním prohlížeči, ze kterého se přistupuje do administrace nové okno s náhledem obsahu, který se aktuálně přehrává na zařízení.

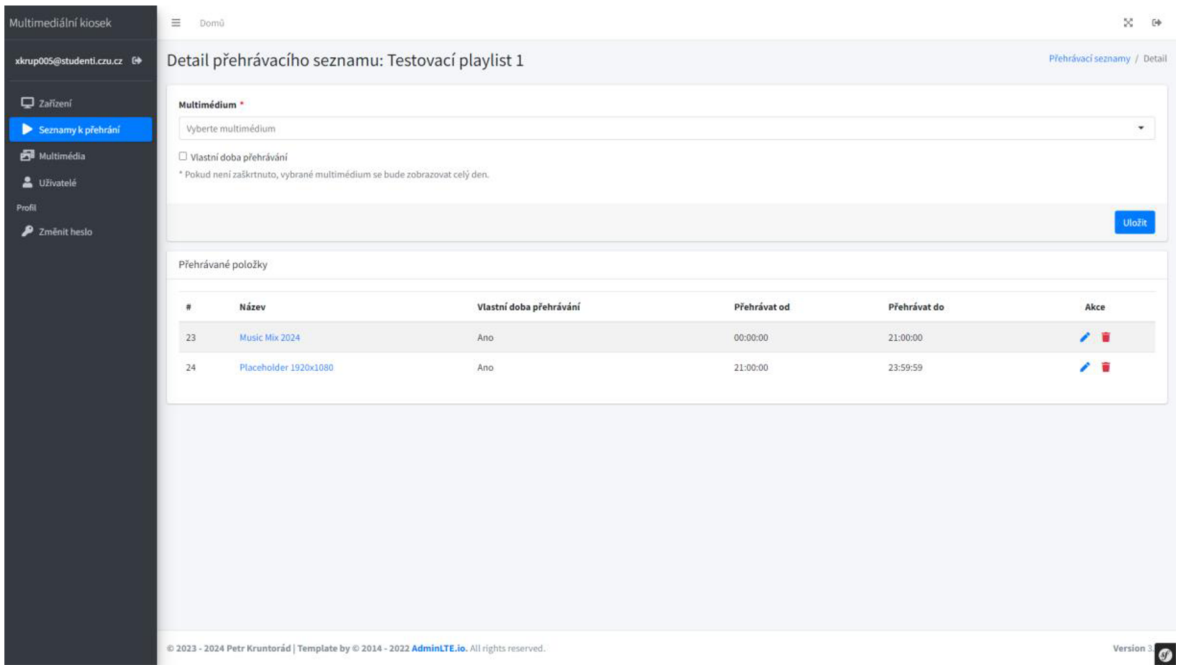
Posledním důležitým tlačítkem je tlačítko s ikonou export ( ), které umožňuje stažení aktuální konfigurace pro vybrané zařízení ve formátu json.

### 4.7.4 Seznamy k přehrání




*Obrázek 27 – Stránka seznamy k přehrání*

Kategorie seznamy k přehrání slouží, jak již název napovídá, ke správě seznamů k přehrání. Pod správu patří jejich přidávání, editace, mazání, nebo správa multimédií (ikona obrázku), které se v daném seznamu mají přehrávat.



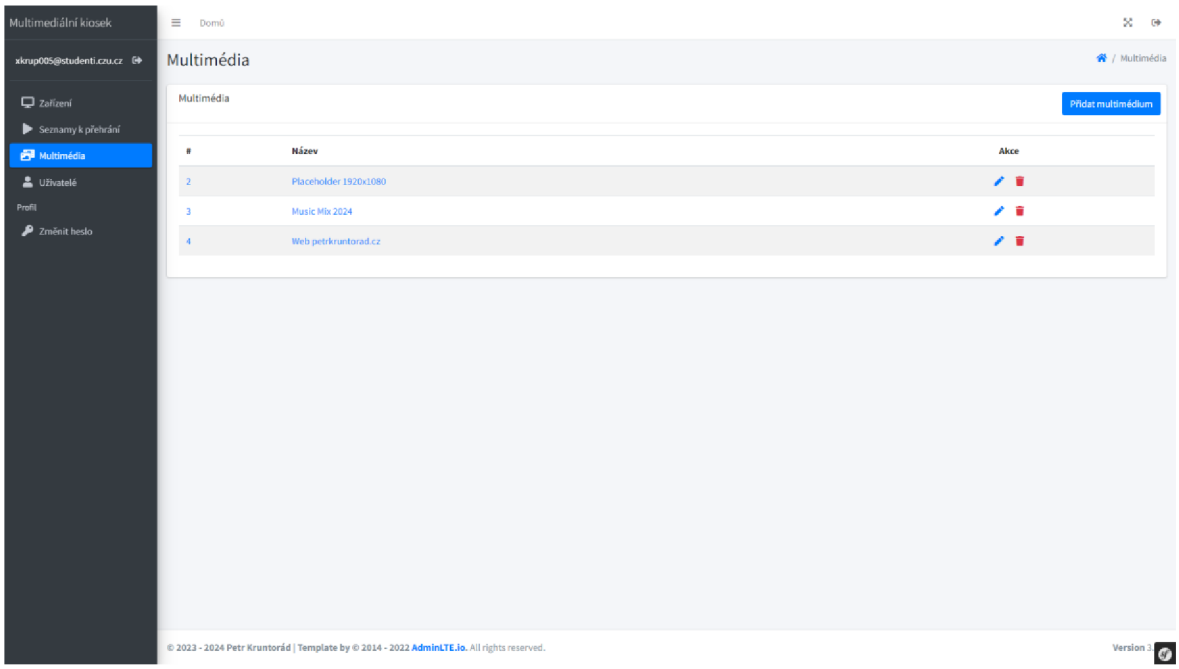

*Obrázek 28 – Stránka pro správu multimédií v přehrávacím seznamu*

Po kliknutí na tlačítko sloužící pro správu multimédií u přehrávacího seznamu je uživatel přesměrován na stránku zobrazenou na obrázku výše. Zde může uživatel skrze formulář přidávat do seznamu dříve vytvořená multimédia s možností nastavit vlastní dobu

přehrávání. Možnost nastavení vlastní doby přehrávání je určena pro nastavení specifického času, od kdy do kdy se má dané multimédium přehrávat. Pokud není zaškrtnuto, bude se multimédium zobrazovat celý den.

Díky tomu je možné rozvrhnout přehrávání multimédií pouze v zamýšlených časech. Pokud však seznam obsahuje položku, která se má zobrazovat celý den nebo by nová položka kolidovala s již existující položkou, tak nelze naplánovat další multimédium pro daný čas.

### 4.7.5 Multimédia




*Obrázek 29 – Stránka multimédia*

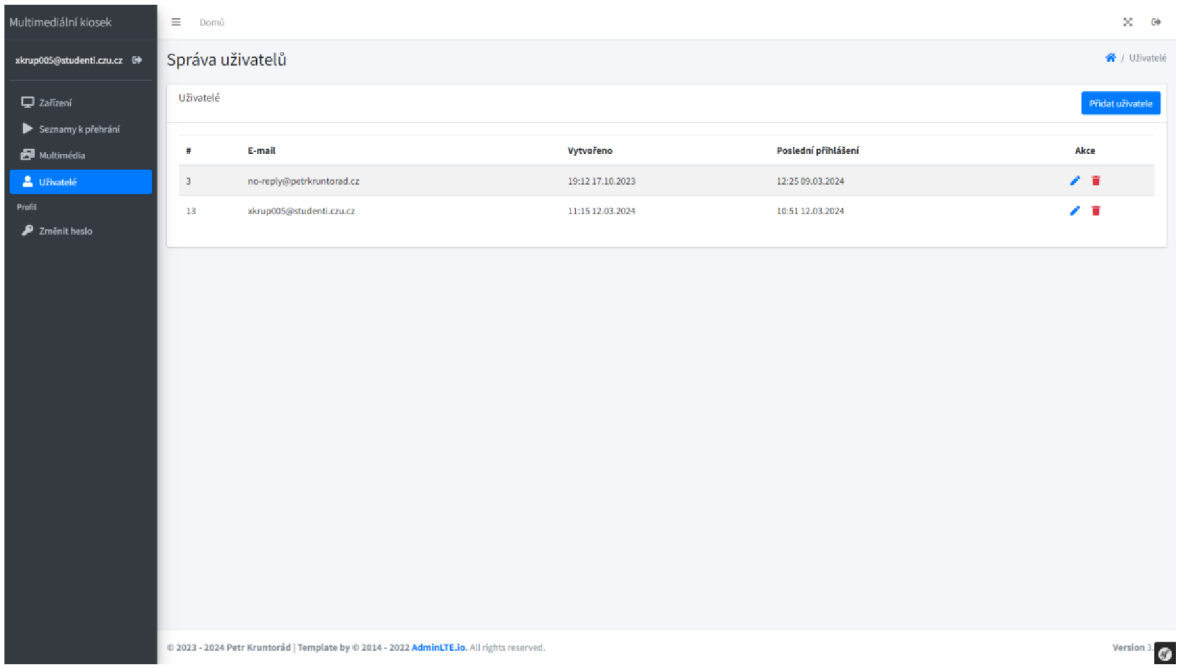
V rámci kategorie multimédia je možné spravovat multimédia. Lze zde přidávat multimédia typu obrázek, webová stránka, video nebo YouTube video.

Multimédia typu obrázek jsou určena pro vykreslení prostého obrázku nebo GIFu na kiosku podle potřeby.

V případě webové stránky je důležité, aby webová stránka umožňovala zobrazení v elementu html iframe, pokud se nachází na jiné doméně. Formulář toto validuje a v případě, že není možné danou webovou stránku vykreslit, tak dojde k zobrazení upozornění, že daný web nelze použít. V případě využití domény webové stránky, která umožňuje zobrazení a využití dotykové obrazovky, lze umožnit uživatelům kiosku interaktivitu s kioskem.

Multimédia typu video a YouTube jsou podobné, nicméně pokud není na hostingu, který je využit pro provoz administrace, dostatek uložiště, je lepší využít platformu YouTube, protože v opačném případě se bude video nahrávat na daný hosting. Dále je v případě využití multimédia typu video omezena velikost souboru na 1024 MB, aby se předešlo nahrávání příliš velkých souborů.

### 4.7.6 Uživatelé




*Obrázek 30 – Stránka uživatelé*

Tato kategorie slouží pro správu uživatelů, kteří mají přístup do administrace kiosků. Lze zde přidávat nové uživatele, změnit email pro jednotlivé uživatele nebo smazat existujícího uživatele.

### 4.7.7 Kód přehrávače

Kód přehrávače je rozdělen na 2 hlavní části, na takzvanou Live component a stimulus controller. Live component se stará o vykreslení samotného obsahu, který se aktuálně přehrává. Kód pro tuto část se nachází na 2 místech, PHP část se nachází ve složce */src/Twig/Components* v souboru s názvem *MediaPlayerComponent.php* a šablona se nachází ve složce */templates/components* v souboru s názvem *MediaPlayerComponent.html.twig*.

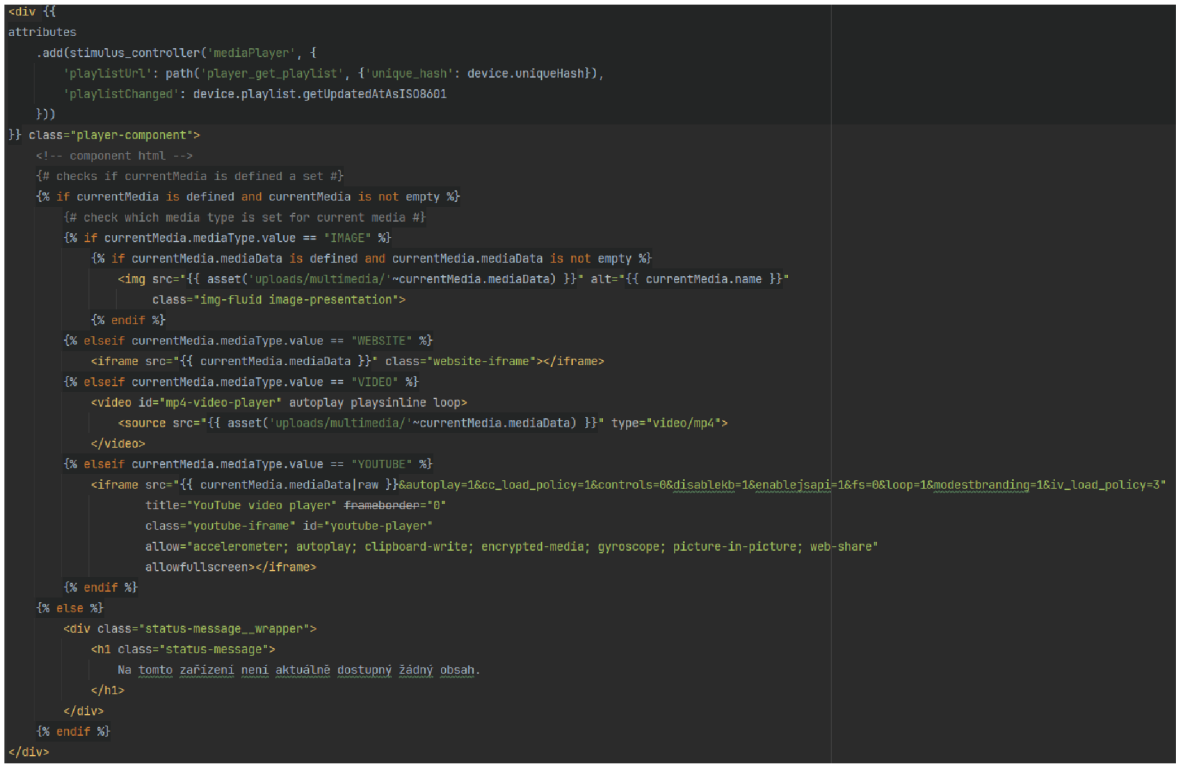
```php
#[AsLiveComponent]
final class MediaPlayerComponent
{
    use DefaultActionTrait;

    #[LiveProp(writable: true)]
    public ?Device $device = null;

    2 usages
    #[LiveProp(writable: true)]
    public ?PlaylistMedia $currentPlaylistMedia = null;

    2 usages
    #[LiveProp(writable: true)]
    public ?Media $currentMedia = null;

    Petr Kruntorád
    public function __construct(
        private readonly PlaylistMediaRepository $playlistMediaRepository,
        private readonly DeviceService $deviceService
    )
    {
    }

    Petr Kruntorád
    public function mount(Device $device): void
    {
        $this->device = $device;
        $this->currentPlaylistMedia = $device->getCurrentlyPlayedMedia();
        $this->currentMedia = $device->getCurrentlyPlayedMedia()?->getMedia();
    }

    no usages  Petr Kruntorád
    #[LiveListener('media:update')]
    public function refreshMedia(): void
    {
        if($this->device->getPlaylist())
        {
            $currentPlaylistMedia = $this->playlistMediaRepository->getCurrentMediaForPlaylist($this->device->getPlaylist());
            if($currentPlaylistMedia)
            {
                $this->currentPlaylistMedia = $currentPlaylistMedia;
                $this->currentMedia = $currentPlaylistMedia->getMedia();
            }

        }
    }
}
```

*Obrázek 31 – Live component – MediaPlayerComponent.php*

Stimulus controller je určen pro udržování aktuálnosti přehrávaného souboru. Kód je psán v programovacím jazyce JavaScript. Tato část se nachází ve složce */assets/controllers* v souboru *mediaPlayer_controller.js*. Propojení obou těchto částí se děje v souboru *MediaPlayerComponent.html.twig*, kde je napojen stimulus controller na hlavní element.


*Obrázek 32 – Šablona MediaPlayerComponent.html.twig – napojení stimulus controlleru*

Při napojení se předávají 2 parametry. První je adresa pro získání dat pro dané zařízení o aktuálním seznamu k přehrání. Druhý parametr je o tom, kdy byl naposledy přehrávací seznam upraven. To je nezbytné pro reakci na případné změny. V této šabloně už se pak nachází pouze podmínka, která vyhodnocuje typ přehrávaného multimédia.




*Obrázek 33 – Šablona MediaPlayerComponent.html.twig*

Samotný soubor pro stimulus controller je okomentován v kódu, nicméně je důležité zmínit část kódu, která se stará o obnovení obsahu v případě změny. O toto se stará funkce *refreshCurrentMedia*, která má jediný parametr a to čas, za jak dlouho se má obsah obnovit.

```
async refreshCurrentMedia(timeToRefresh) : Promise<void>
{
    setTimeout( callback: async () : Promise<void>  => {
        console.log('refreshing');
        //sets current media to null to prevent showing of old media
        this.component.set( model: 'currentMedia', value: null);
        //emits event to update media
        this.component.emit( name: 'media:update');
        //loads playlist again
        await this.loadPlaylist()
        //re-renders component
        this.component.render();
        console.log('refreshed');
    }, timeToRefresh);
}
```

*Obrázek 34 – Stimulus controller - mediaPlayer_controller.js – funkce refreshCurrentMedia*

Tato funkce obsahuje funkci *setTimeout*, která provede obnovení obsahu pro Live component po čase specifikovaném v parametru *timeToRefresh*. Kód obsažený v této funkci nejprve nastaví aktuální multimédium jako prázdnou hodnotu, poté zavolá událost s názvem: „*media:update*“, které naslouchá funkce *refreshMedia* na Obrázek 31 – Live component – MediaPlayerComponent.php. To způsobí opětovné načtení celé komponenty a získání aktuálně přehrávaného multimédia.

## 4.8 Software zařízení pro přehrávání

Software pro přehrávání je naprogramován pomocí programovacího jazyka Python. Díky tomu je možné využít důležité balíčky, které umožňují spouštění prohlížeče a automatickou konfiguraci pravidelného spouštění skriptů ve specifikovaných časech bez zásahu uživatele. Prvním z nich je balíček selenium [32], který se používá pro automatizaci webových aplikací. Druhým důležitým je balíček crontab [33], který přidává do jazyka Python možnost práce s nástrojem cron, díky kterému je možné nastavit skripty, které se mají provést v určitých časech.

Samotný software je rozdělen do několika souborů, z nichž hlavním je functions.py, který obsahuje všechny nezbytné funkce a procedury pro správné fungování. Dalšími soubory jsou *check_player_status.py*, *init_device.py*, *init_player.py*, *refresh_browser.py*,

*update_config.py*, *update_device_details.py*. Všechny tyto soubory ale slouží pouze pro volání funkcí ze souboru *functions.py*.

### 4.8.1 Soubor functions.py

Tento soubor obsahuje celkem 24 funkcí a procedur, které jsou nezbytné pro funkci přehrávače. Všechny procedury a funkce jsou důležité, nicméně díky komentářům v kódu a tomu, že některé plní pouze jednoduché operace, zde budou rozebrány jen ty nejpodstatnější.

První z nich je procedura *start_browser*, která se stará o spouštění prohlížeče na specifické adrese. Tato procedura obsahuje jeden parametry s názvem *url*, který je typu string. V tom musí být při zavolání nastavena adresa na které má být otevřen prohlížeč.

```
# starts browser with specified url
1 usage   Petr Kruntorád
def start_browser(url: str):
    options = webdriver.ChromeOptions()
    # sets chrome to maximized mode on start
    options.add_argument("--start-maximized")
    options.add_experimental_option( name: 'excludeSwitches', value: ['enable-automation'])
    # starts chrome in kiosk mode
    options.add_argument("--kiosk")
    # allows to use autoplay with video that uses sound
    options.add_argument('--autoplay-policy=no-user-gesture-required')
    # starts chrome as separate process otherwise chrom will close when script finished
    options.add_experimental_option( name: "detach", value: True)

    # sets path for chrome driver on raspbian
    chrome_service = ChromeService(executable_path='/usr/lib/chromium-browser/chromedriver')
    driver = webdriver.Chrome(service=chrome_service, options=options)

    # starts browser at specific url
    driver.get(url)

    process_pids = []
    # possible names of used browser
    valid_process_names = ['chrome.exe', 'chromium-browser']

    # iterates through processes and looks for processes with required name and parameters
    for process in psutil.process_iter():
        if process.name() in valid_process_names and '--test-type=webdriver' in process.cmdline():
            with suppress(psutil.NoSuchProcess):
                process_pids.append(process.pid)

    # check if there are process pids and saves them to temp json file
    if len(process_pids) > 0:
        # saves browser pids to temp file
        store_browser_process_id(process_pids)
```

*Obrázek 35 – Přehrávač – procedura start_browser*

Z obrázku výše je patrné, že nejprve dochází k nastavení vybraných argumentů pro prohlížeč. Důležitými argumenty jsou *--start-maximized*, který se stará o to, že se prohlížeč spustí v maximalizovaném režimu, poté argument *--kiosk*, který vypne ovládací prvky, díky čemuž u dotykových obrazovek není schopen běžný uživatel zavřít prohlížeč. Dále je zde argument *--autoplay-policy=no-user-gesture-required*, který umožňuje kiosku automaticky přehrávat videa se zvukem, protože v opačném případě prohlížeč blokuje automatické

spuštění videí se zvukem. Důležité je zapnutí možnosti *detach* na *true*. To zajišťuje, že prohlížeč není zavřen po dokončení Python skriptu.

Poté se už jen nastaví cesta pro ovladač a dojde k otevření prohlížeče s požadovanou URL adresou. Tato procedura zároveň uloží pomocí procedury *store_browser_process_id* identifikátory procesů, aby bylo možné průběžně kontrolovat, zda je prohlížeč spuštěn a případně ho opětovně spustit.

```
# saves process pid to temp file
1 usage   Petr Kruntorád
def store_browser_process_id(process_ids: list):
    # create temp dir if not exists
    create_temp_dir()

    # Data to be written
    data = {
        "process_ids": process_ids,
    }

    # writes data to file
    with open(browser_temp_file, 'w') as file:
        json.dump(data, file)
```

*Obrázek 36 – Přehrávač – procedura store_browser_process_id*

Mezi důležité procedury patří, již výše zmíněná procedura *store_browser_process_id*, jejímž úkolem je ukládat id procesů do dočasného souboru. Tato procedura má pouze jeden parametr typu list, který obsahuje id daných procesů. Po zavolání se nejprve zavolá procedura *create_temp_dir*, která vytvoří složku s názvem *temp* pro dočasné soubory, pokud neexistuje. Následně jsou data z listu zapsaná do formátu json a ten je následně uložen do souboru *browser_temp.json* v dříve vytvořené složce temp.

## 4.9 Instalace

### 4.9.1 Administrace

Prvotní instalace administrace probíhá stažením administrace z platformy GitHub na adrese https://github.com/petrkruntorad/dp-thesis. Zde jsou popsány kroky nezbytné pro instalaci.

Aby bylo možné instalaci provést je nezbytné nainstalovat software XAMPP [34], Composer a Node.js. Software XAMPP [34] je potřeba kvůli podpoře jazyka PHP v příkazové řádce, které bude nezbytné v následujících krocích (případně lze nainstalovat pouze samotné PHP [35]). Dále je potřeba nainstalovat Composer [36] pro správu závislostí

v PHP a Node.js [37], který zahrnuje npm, jenž je určeno pro správu balíčků jazyka JavaScript a přidání podpory pro další příkazy.

Po nainstalování obou nástrojů stačí již jen otevřít umístění, ve kterém se nachází stažený projekt v příkazové řádce či editoru, který obsahuje příkazovou řádku a spustit nejprve příkaz *composer install*, poté *npm install* a následně *npm run build*. To nainstaluje veškeré nezbytné závislosti a vygeneruje nezbytné *assets* potřebné pro chod webového rozhraní.

Jakmile jsou všechny závislosti nainstalovány, tak je ještě nezbytné dokončit základní konfiguraci. Nejprve je potřeba vytvořit soubor *.env* v domovském adresáři, zkopírováním a přejmenováním souboru *.env.example*.

```
# In all environments, the following files are loaded if they exist,
# the latter taking precedence over the former:
#
#  * .env                contains default values for the environment variables needed by the app
#  * .env.local          uncommitted file with local overrides
#  * .env.$APP_ENV       committed environment-specific defaults
#  * .env.$APP_ENV.local uncommitted environment-specific overrides
#
# Real environment variables win over .env files.
#
# DO NOT DEFINE PRODUCTION SECRETS IN THIS FILE NOR IN ANY OTHER COMMITTED FILES.
# https://symfony.com/doc/current/configuration/secrets.html
#
# Run "composer dump-env prod" to compile .env files for production use (requires symfony/flex >=1.2).
# https://symfony.com/doc/current/best_practices.html#use-environment-variables-for-infrastructure-configuration

###> symfony/framework-bundle ###
# options: dev, prod
APP_ENV=prod
APP_SECRET=<app_secret>
###< symfony/framework-bundle ###

###> doctrine/doctrine-bundle ###
# Format described at https://www.doctrine-project.org/projects/doctrine-dbal/en/latest/reference/configuration.html#connecting-using-a-url
# IMPORTANT: You MUST configure your server version, either here or in config/packages/doctrine.yaml
#
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
# DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8.0.32&charset=utf8mb4"
# DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=10.11.2-MariaDB&charset=utf8mb4"
#DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
DATABASE_URL="mysql://<db_user>:<db_user_password>@127.0.0.1:3306/<db_name>?serverVersion=15&charset=utf8"
###< doctrine/doctrine-bundle ###

###> symfony/mailer ###
MAILER_FROM=<smtp-email>
MAILER_DSN=smtp://<smtp-email>:<smtp-email-password>@<smtp-email-server>:<smtp-email-server-port>
###< symfony/mailer ###
```

*Obrázek 37 – Instalace administrace – soubor .env.example*

V něm se nachází hned několik hodnot, které se musí vyplnit. Tyto hodnoty lze poznat tak, že jsou označeny jako <nazev_hodnoty>. První hodnota, která se musí změnit je hodnota pro položku APP_SECRET. Ta se získá zavoláním příkazu *php bin/console app:generate-app-secret*.

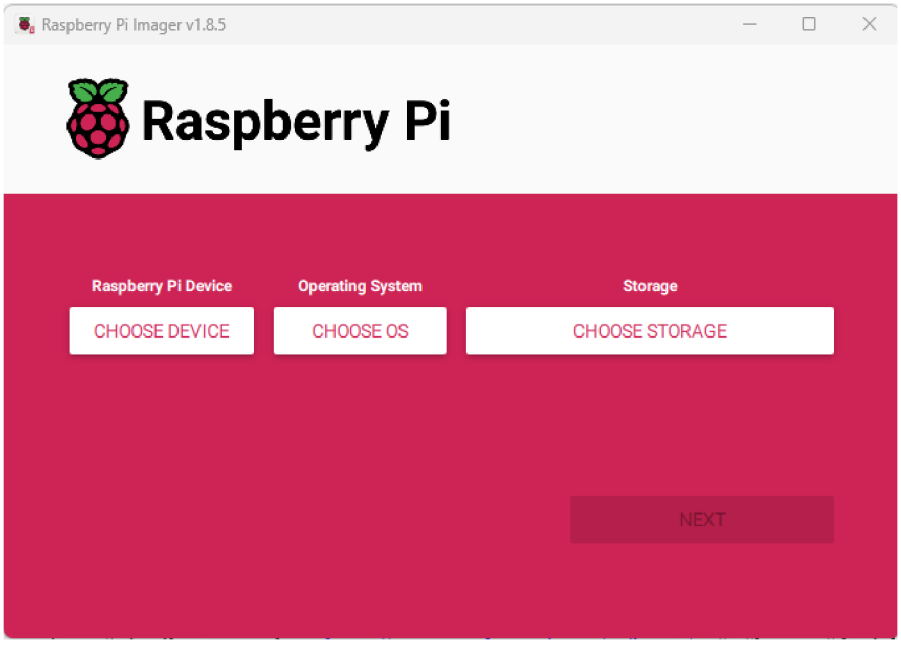

*Obrázek 38 – Instalace administrace – Vygenerování app_secret*

Zbývající hodnoty se doplňují na základě vlastních údajů poskytnutých provozovatelem hostingu nebo jinak vygenerovaných. Následně stačí už jen nahrát vzorovou databázi nacházející se v domovském adresáři projektu s názvem *multimedia_kiosk.sql* spolu s projektem na požadovaný hosting a lze začít využívat administraci. Po instalaci je doporučeno změnit údaje pro výchozího uživatele „dev@test.com“ s heslem „Abcdef0/“.

### 4.9.2 Kiosek

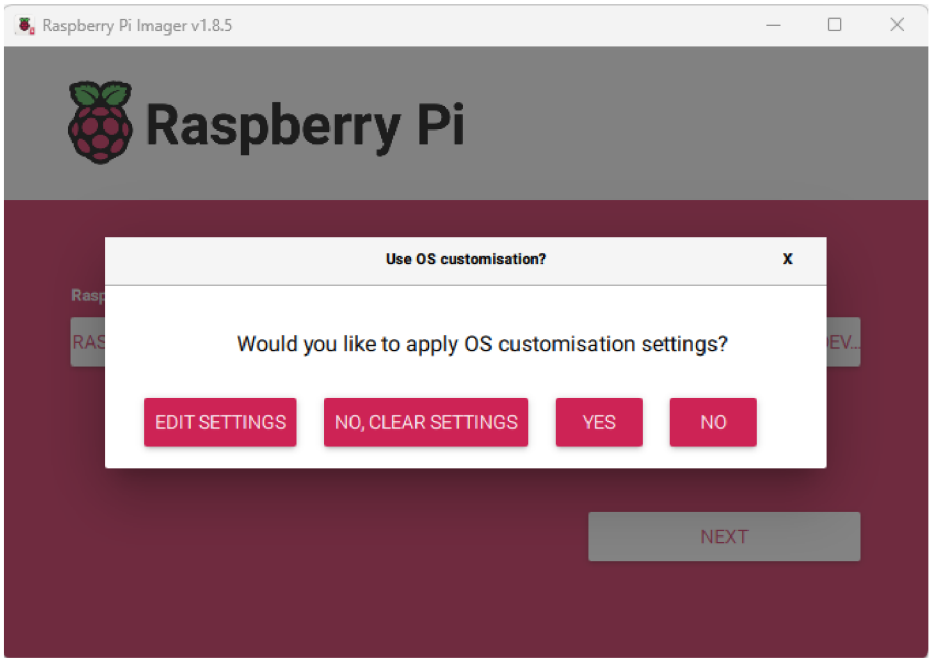
Instalace kiosku je rozdělena do několika základních kroků. Tyto kroky zahrnují instalaci operačního systému na paměťovou kartu formát microSD, změny systému nezbytné pro optimální chod a instalace skriptů pro přehrávač. Některé kroky je možné realizovat prostřednictvím protokolu SSH, jiné je možné realizovat fyzicky na zařízení.

Pro instalaci je nezbytné si připravit paměťovou kartu ve formátu microSD s minimální kapacitou 8 GB a počítač s nainstalovanou aplikací Raspberry Pi Imager [38] a čtečku paměťových karet podporující formát microSD.



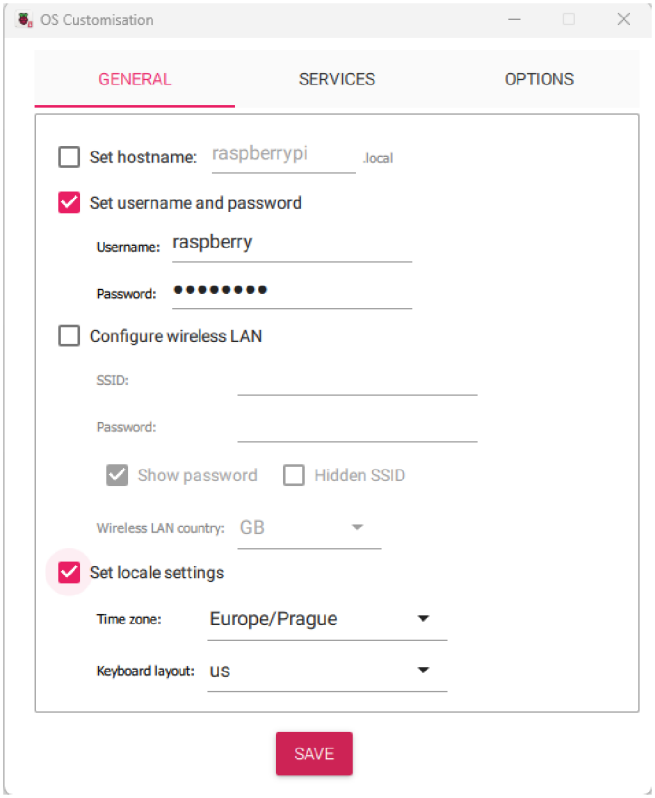

*Obrázek 39 – Raspberry Pi Image – Úvodní okno [38]*

Po nainstalování a spuštění aplikace Raspberry Pi Image se zobrazí okno, které je vyobrazeno na Obrázek 39 – Raspberry Pi Image – Úvodní okno [38]. Zde je potřeba zvolit zařízení, které bylo vybráno pro kiosek, operační systém a microSD kartu, na kterou bude systém nainstalován.

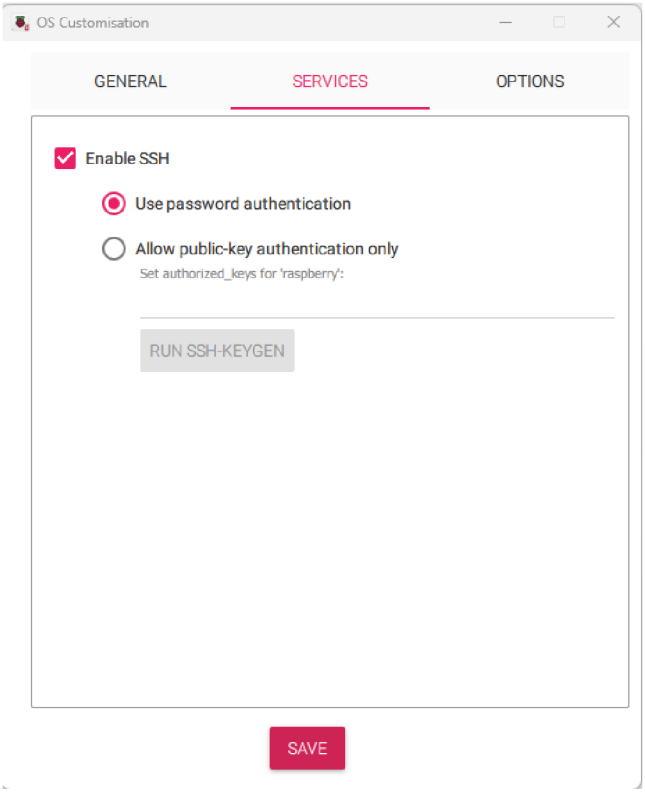



*Obrázek 40 – Raspberry Pi Image – Vyskakovací okno [38]*

Po zvolení všech požadovaných položek a kliknutí na tlačítko *Next* se zobrazí okno nabízející nastavení některých hodnot. Zde je dobré kliknout na tlačítko *Edit settings* a nastavit nastavení dle Obrázek 41 – Raspberry Pi Image – Nastavení – záložka General [38] s tím, že hodnoty pro *username* a *password* je doporučeno si nastavit vlastní. To stejné platí i pro položku *hostname*, díky čemuž bude možné přistupovat k zařízení i bez použití IP adresy.




*Obrázek 41 – Raspberry Pi Image – Nastavení – záložka General [38]*



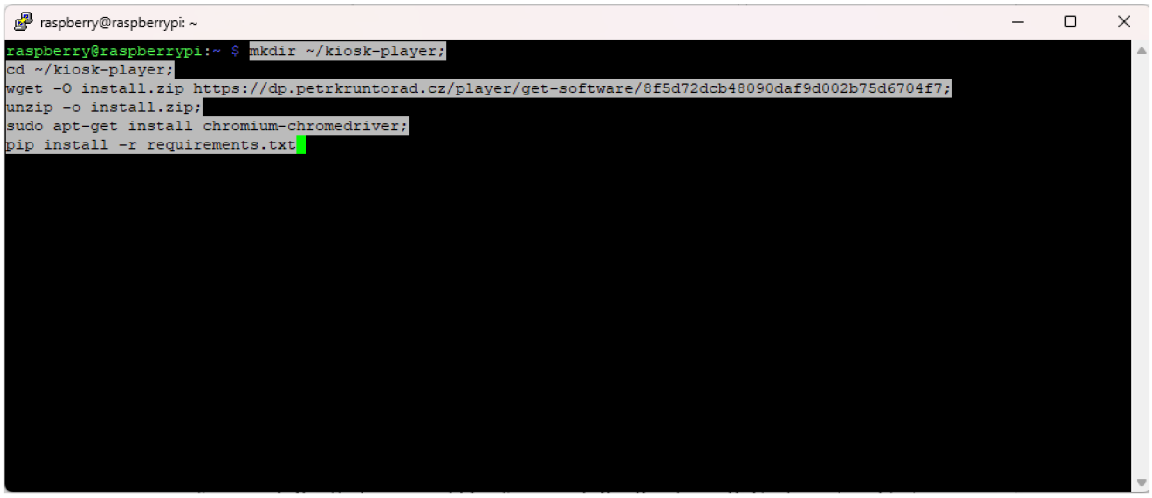

*Obrázek 42 – Raspberry Pi Image – Nastavení – záložka Services [38]*

Po dokončení všech změn stačí kliknout na tlačítko *save*, následně ve vyskakovacím okně zobrazeném na Obrázek 40 – Raspberry Pi Image – Vyskakovací okno [38] kliknout

na tlačítko *yes* a následně potvrdit zformátování microSD karty, která byla zvolena pro instalaci. Poté bude na paměťovou kartu nainstalován operační systém a po dokončení lze pokračovat k dalšímu kroku.

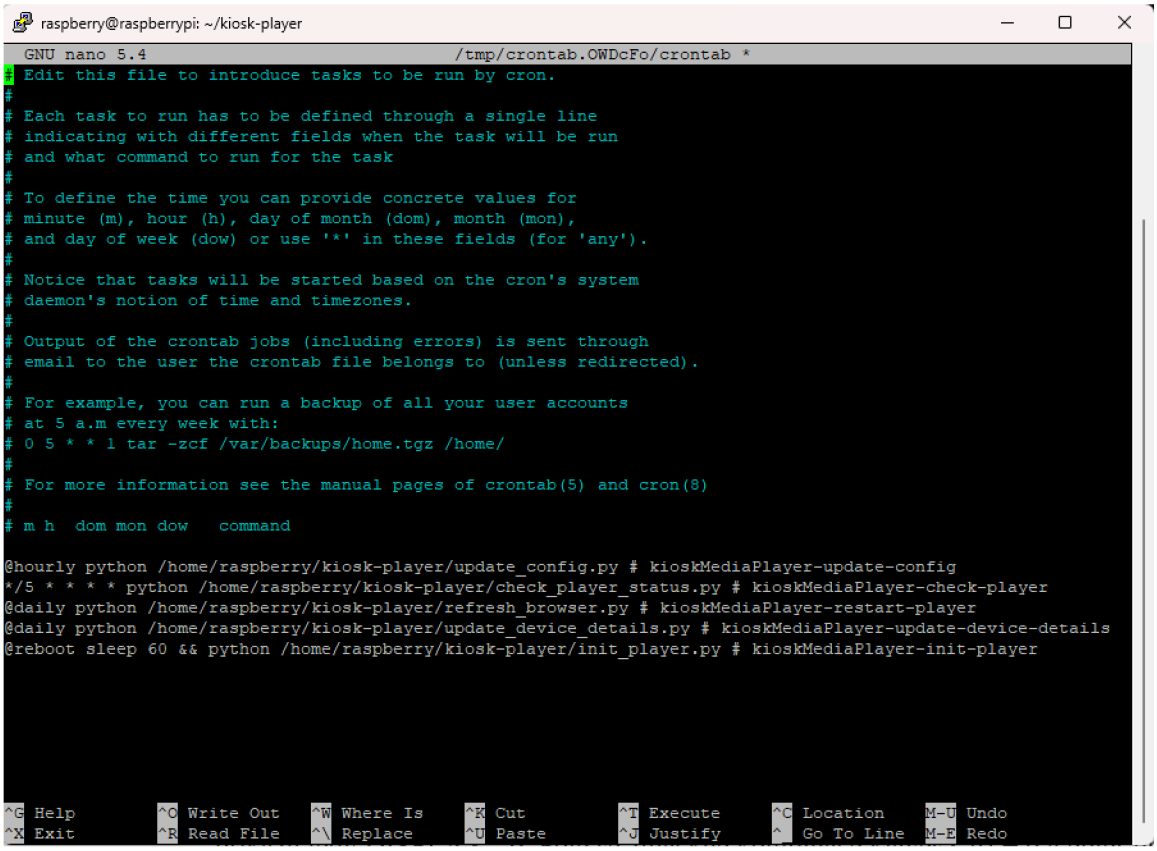
V dalším kroku se nejprve vyjme microSD karta ze čtečky paměťových karet a následně se vloží do Raspberry Pi, které odpovídá nainstalovanému operačnímu systému. Poté je možné Raspberry Pi spustit a počkat na naběhnutí operačního systému. Jakmile naběhne operační systém, je potřeba nastavit několik věcí. První z nich je deaktivace vypínání displeje. Toho se docílí kliknutím v levém horním rohu na ikonu maliny, poté najetím na položku *Preferences* a kliknutím na položku *Raspberry Pi Configuration*, která se ukáže v rozbalené nabídce. Poté, co se otevře okno s nastavením, je potřeba přejít na záložku *Display* a vypnout položku *Screen Blanking*.

Následující krok lze provést buď přímo na zařízení nebo vzdáleného připojení s využitím protokolu SSH, protože se jedná o instalaci skriptů. V tomto příkladě je zvoleno vzdálené připojení.



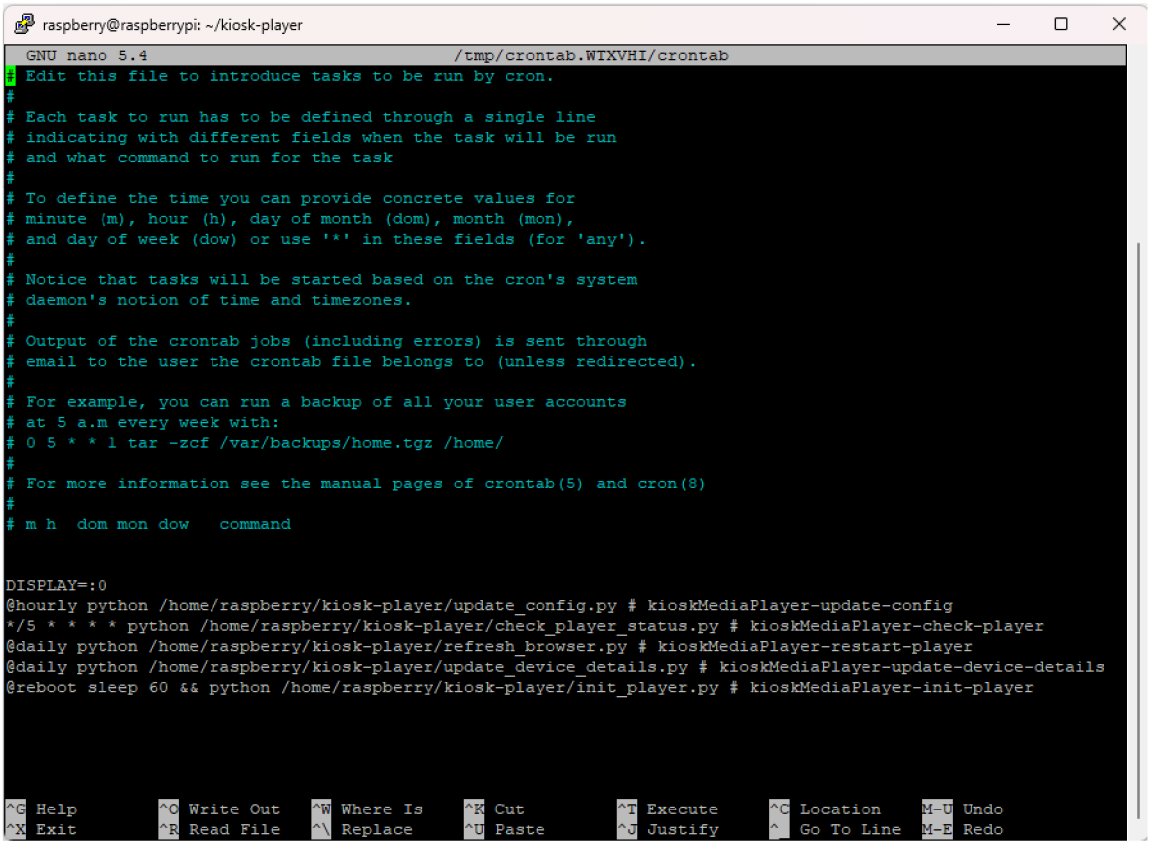

*Obrázek 43 – Instalace skriptů na zařízení prostřednictvím SSH*

Do příkazové řádky se vloží příkazy zobrazené ve vyskakovacím okně vyobrazeného na Obrázek 26 – Stránka zařízení – vyskakovací okno, které nainstalují vše potřebné na zařízení. Jakmile doběhne instalace skriptů je potřeba přidat záznamy do cron, toho se docílí zavoláním příkazu *python init_device.py* ve složce, do které byl nainstalován přehrávač. Následně je doporučeno zkontrolovat, že se záznamy úspěšně nastavily zavoláním příkazu *crontab -e*, který by měl obsahovat položky jako na Obrázek 44 - Crontab po přidání nezbytných skriptů.




*Obrázek 44 - Crontab po přidání nezbytných skriptů*

Posledním povinným krokem je nastavení displeje, na kterém se bude přehrávat výstup ze skriptu spuštěného pomocí nástroje cron. To lze nastavit otevřením terminálu, kde se zadá příkaz *echo $DISPLAY,* jehož výstupem bude hodnota ve formátu :0 nebo :1 a podobně. Tuto hodnotu je nezbytné si zapamatovat, protože se s ní dále pracuje. Poté se v terminálu zavolá příkaz *crontab -e*, který zobrazí nabídku s výběrem editoru, pokud nebyl vybrán dříve (v tomto příkladě se pracuje s editorem nano). Následně se otevře cron pro aktuálního uživatele. Zde se vytvoří nový řádek s hodnotou DISPLAY= a hodnotou z předchozího příkazu tedy DISPLAY=:0. Výsledný obsah cron by měl vypadat podobně jako na Obrázek 45 – Crontab po přidání displeje pro výstup.



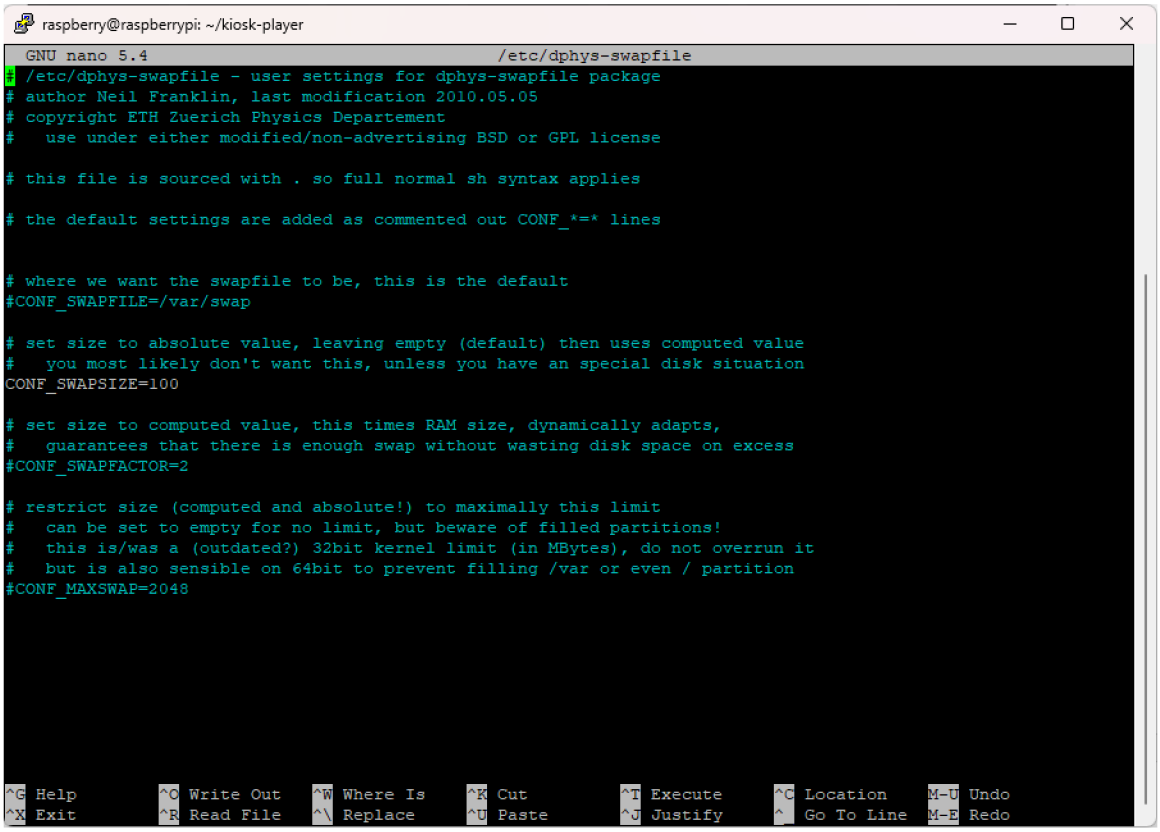

*Obrázek 45 – Crontab po přidání displeje pro výstup*

Na obrázku výše je vidět hned pod řádkem *DISPLAY=:0* celkem pět naplánovaných událostí, které jsou nezbytné pro provoz přehrávače. První skript se volá každou hodinu a má na starost to, aby byl zachován aktuální konfigurační soubor vzhledem k nastavení v administraci. Dále je zde příkaz, který volá skript *check_player.py* každou 5. minutu a dohlíží na to, že přehrávač je stále v provozu. Následují 2 skripty, které se volají každý den o půlnoci. První skript *refresh_browser.py* restartuje prohlížeč, tento skript je přítomen, aby bylo zařízení v případě nějakého problému schopno problém vyřešit. Druhý skript *update_device_details.py* aktualizuje informace o zařízení (kapacita uložiště, využití uložiště a lokální IP adresu). Poslední skript se volá 60 vteřin pro zapnutí zařízení a slouží k automatickému spuštění přehrávače.

Po dokončení změn a ověření obsahu se musí zmáčknout kombinace kláves Ctrl + o a enter pro uložení změn a Ctrl + x pro zavření. Následně by mělo dojít k automatickému spuštění přehrávače do několika minut, nebo lze případně restartovat operační systém a přehrávač bude spuštěn do 60 vteřin po naběhnutí operačního systému.

Kromě povinných kroků existuje ještě jeden nepovinný, který je možný provést ke zlepšení plynulosti zařízení. Tímto krokem je změna velikosti souboru pro odkládání, pokud

není dostupné dostatečné množství operační paměti. Toho lze docílit prostřednictvím několika příkazů. Prvním z nich je příkaz *sudo dphys-swapfile swapoff*, který vypne odkládání. Poté se musí z příkazové řádky zavolat příkaz *sudo nano /etc/dphys-swapfile*, který otevře konfiguraci odkládání pro editaci.




*Obrázek 46 – Soubor /etc/dphys-swapfile*

Na obrázku výše se nachází konfigurace odkládání, zde je potřeba změnit hodnotu u řádku s textem *CONF_SWAPSIZE* z hodnoty 100 například na 2048, to zvětší velikost ze 100 MB na 2048 MB. Poté už stačí jen zmáčknout kombinaci kláves Ctrl + O a enter pro uložení a Ctrl + X pro zavření. Následně je nezbytné zavolat příkaz *sudo dphys-swapfile setup* pro opětovnou inicializaci a *sudo dphys-swapfile swapon* pro opětovné zapnutí odkládání.

## 4.10 Výsledky testovaní

Kromě průběžného testování, aby se ověřila funkcionalita psaného kódu, bylo provedeno otestování funkcionalit před nasazením na testovací hosting a po nasazení na testovací hosting a finálním otestování na zařízení.

Na základě testování funckionality webového rozhraní před nasazením nebyly nalezeny žádné problémy a software pro zařízení byl v této fázi testován pouze na základní funkcionality, jako jsou spuštění prohlížeče a uložení id procesů prohlížeče kvůli rozdílné platformě vývojového prostředí. Nicméně po nasazení webového rozhraní na testovací hosting byly objeveny problémy s přehráváním obsahu, které byly nedetekovatelné na vývojovém prostředí. Po rozsáhlém zkoumání kódu byla objevena příčina nacházející se v dotazu na databázi, který slouží pro získávání přehrávaného multimédia pro daný čas, který poté nevracel správná data pro daný čas. Tato chyba se projevovala pouze na webovém serveru využívajícím software NGINX a z toho důvodu na vývojovém prostředí využívající webový server založený na softwaru Apache nebylo možné tuto chybu detekovat. Celá chyba spočívala v tom, že se v dotazu dával celý čas včetně datumu (formát datetime), ale na serveru, kde je provozován NGINX, byl vyžadován čas v textovém formátu jako hodina a minuta. Poté, co byl tento problém vyřešen, bylo otestováno přehrávání obsahu na Raspberry Pi, kde po úpravách kódu závislého na platformě bylo již možné software úspěšně spustit.

# 5 Výsledky a diskuse

Software pro správu multimediálních kiosků a následné přehrávání obsahu splňuje požadavky, které si autor vytyčil, nicméně existují části, které by šly vytvořit jinak. Jednou z těchto částí může být přehrávání obsahu v režimu, kdy není dostupné internetové připojení, ale s tím se pojí problém s uložištěm. Jelikož SD karty mohou snáze selhat při opětovném přepisování obsahu, bylo by vhodné, aby řešení obsahovalo uložiště s využitím například SSD disku. Tím by bylo možné docílit delší provozuschopnosti zařízení. Výhodou řešení, které by stahovalo multimédia do zařízení, je, že není potřeba neustálého připojení k internetu pro přehrávání obsahu. To může být užitečné v některých případech, kdy není potřeba zobrazovat webovou stránku nebo jiný obsah vyžadující připojení k internetu. Avšak existuje i nevýhoda v podobě distribuce obsahu. Jelikož není obsah nahráván na server, odkud je přímo přehráván, ale nejprve na server, odkud probíhá distribuce na zařízení, které jej má přehrávat, tak je nezbytné brát v potaz, že reakce na změnu obsahu nebude okamžitá, ale bude záviset na rychlosti internetového připojení zařízení i uživatele nahrávajícího obsahu a rychlosti uložiště na zařízení sloužícímu pro přehrávání.

Oproti tomu vytvořené řešení umožňuje zobrazovat webové stránky, se kterými může uživatel v případě potřeby realizovat interakci. Obě tyto řešení mají své pro i proti, nicméně nelze říct přesně, které řešení je lepší a které horší, protože obě jsou vhodná pro rozdílné využití. Pokud by však provozovatel chtěl využít čistě přehrávání bez připojení k internetu, tak využití centrální správy skrze webové rozhraní postrádá význam. Je zde sice případ, při kterém by bylo webového rozhraní provozováno v lokální síti a nebylo by tedy nezbytné přistupovat k internetu, ale výskyt takového přístupu nebude nejspíše příliš častý.

Nicméně v obou případech lze využít výše zmíněný SSD disk, jehož instalace záleží na osobě instalující software, a nikoliv na vybrané variantě. Tento přístup je tedy doporučen jako vhodná volba pro zařízení, které bude provozováno v dlouhodobém časovém horizontu.

Kompletní software, jenž je výstupem této práce je možné stáhnout z platformy GitHub na adrese https://github.com/petrkruntorad/dp-thesis, nebo případně vyzkoušet na adrese https://dp.petrkruntorad.cz/admin s údaji zmíněnými v kapitole 4.7.

# 6 Závěr

Cílem této diplomové práce bylo vytvoření webové aplikace umožňující vzdálené spravování multimediálních kiosků založených na jednodeskovém počítači Raspberry Pi a přehrávání obsahu na nich.

V teoretické části byly vymezeny pojmy nezbytné pro dosažení vytyčených cílů, použité technologie a zařízení, analyzovány metodiky vývoje a existující řešení. Na základě analýzy vyplynulo, že existující řešení jsou buď drahá, nabízejí jen část řešení anebo jsou zpoplatněna a omezena na prostředky. Dále byla na základě analýzy metodik vývoje vybrána jako vhodná metodika vzhledem k rozsahu a povaze projektu vodopádová metodika.

V praktické části byly vyhodnoceny výsledky analýzy z teoretické části, na základě kterých bylo navrženo a vytvořeno výsledné webové rozhraní pro správu multimediálních kiosků založené na PHP frameworku Symfony a software starající se o přehrávání obsahu nastaveného v administraci, který byl vytvořen v programovacím jazyce Python. Následně byla popsána instalace webového rozhraní i softwaru na zařízení. Výsledný kód byl poté publikován na platformě GitHub, kde je možné jej stáhnout a libovolně si ho upravit dle vlastních potřeb.

I přesto, že výsledek této práce splnil cíle, které si vytyčil, tak to neznamená, že se projekt již dále nemůže vyvíjet. Do budoucna by šlo řešení rozšířit o kameru, která by mohla například snímat tváře a tím umožňovat zobrazení statistiky, kolik lidí vidělo obsah nebo umožnit dříve zmíněný režim bez připojení k internetu. To by umožnilo pro jednotlivé kiosky nastavit, které budou neustále připojeny do administrace a které naopak nikoliv. Jelikož software nacházející se na zařízení je založen na programovacím jazyce Python, tak jedním z možných směrů je i zahrnutí podpory pro jiné operační systémy. To by sice vyžadovalo úpravy kódu, ale k přehrávání by bylo možné využít i běžný počítač využívající operační systém Windows

# 7 Seznam použitých zdrojů


1. CZECH MULTIMEDIA INTERACTIVE S.R.O. Multimediální informační kiosky, terminály a samoobslužné platební automaty. CZ MULTIMEDIA INTERACTIVE [online]. c2006-2024 [cit. 2024-03-21]. Dostupné z: https://czmi.cz/sluzby/multimedialni-prezentace-systemy-hry-fun-content/multimedialni-informacni-kiosky-terminaly-a-panely/
2. REDYREF INTERACTIVE KIOSKS. What is an Information Kiosk? Different Types and Benefits. REDYREF [online]. 2019 [cit. 2024-01-15]. Dostupné z: https://redyref.com/information-kiosk/
3. SAKOVICH, Natallia. What Is an Information Kiosk? Definition, Functioning, Types, Benefits. SAM SOLUTIONS. SaM Solutions [online]. 2023 [cit. 2024-01-15]. Dostupné z: https://www.sam-solutions.com/blog/information-kiosk/
4. RASPBERRY PI FOUNDATION. How to use a Raspberry Pi in kiosk mode. Raspberry Pi [online]. 2022 [cit. 2024-01-15]. Dostupné z: https://www.raspberrypi.com/tutorials/how-to-use-a-raspberry-pi-in-kiosk-mode/
5. COLLOQI CONSULTING. PiSignage [online]. 2014, 2024 [cit. 2024-01-15]. Dostupné z: https://www.pisignage.com/
6. PROJEKTA CORP S.R.O. INFORMAČNÍ KIOSKY. PROJEKTA [online]. 2023 [cit. 2024-01-15]. Dostupné z: https://eshop.projekta.cz/59-informacni-kiosky
7. FITZGERALD, Anna. HUBSPOT, INC. What Is a Web App? A Beginner's Guide. HUBSPOT, INC. HubSpot [online]. 2021 [cit. 2024-03-21]. Dostupné z: https://blog.hubspot.com/website/what-is-web-app
8. ADOBE. Informace o webových aplikacích. Adobe [online]. 2015, 19.5.2021 [cit. 2023-12-27]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html
9. ROSENCRANCE, Linda. TECHTARGET. What is Software? Definition, Types and Examples. TECHTARGET. TechTarget [online]. 2021 [cit. 2024-02-15]. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/software
10. MYSLÍN, Josef. Scrum: průvodce agilním vývojem softwaru. Brno: Computer Press, 2016. ISBN 978-80-251-4650-7.
11. LEWIS, Sarah, LUTKEVICH, Ben, ed. TECHTARGET. What is the Waterfall Model? - Definition and Guide. TECHTARGET. TechTarget [online]. 2022 [cit. 2024-03-21]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model
12. PEDAMKAR, Priya. Waterfall Model. EDUCBA [online]. 2023 [cit. 2023-12-17]. Dostupné z: https://www.educba.com/waterfall-model/
13. SCHWABER, Ken a Jeff SUTHERLAND. The 2020 Scrum Guide. Srum Guides [online]. 2020 [cit. 2024-03-21]. Dostupné z: https://scrumguides.org/scrum-guide.html
14. TUTORIALS POINT. Multimedia Introduction. Tutorials Point [online]. c2024 [cit. 2024-03-21]. Dostupné z: https://www.tutorialspoint.com/multimedia/multimedia_introduction.htm
15. LI, Ze-Nian, Mark S DREW a Jiangchuan LIU. Fundamentals of Multimedia. 2nd ed. 2014. Cham: Springer International Publishing, 2014. Texts in Computer Science. ISBN 978-3-319-05290-8.
16. CLOUDINARY. MP4 Format (MPEG-4 Part 14): How It Works, Pros and Cons. Cloudinary [online]. 2023 [cit. 2024-03-22]. Dostupné z:

https://cloudinary.com/guides/video-formats/mp4-format-mpeg-4-part-14-how-it-works-pros-and-cons
17. HALFACREE, Gareth. THE OFFICIAL Raspberry Pi Beginner's Guide: How to use your new computer [online]. 4. Cambridge: Raspberry Pi Trading, 2020 [cit. 2023-04-28]. ISBN 978-1-912047-73-4. Dostupné z: https://magazines-attachments.raspberrypi.org/books/full_pdfs/000/000/038/original/BeginnersGuide-4thEd-Eng_v2.pdf
18. KOLADE, Chris. What is HTML – Definition and Meaning of Hypertext Markup Language. FreeCodeCamp [online]. 2021 [cit. 2024-02-21]. Dostupné z: https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/
19. KAĎOUSKOVÁ, Barbora. HTML pro začátečníky aneb jak začít psát web. RASCASONE, S.R.O. Rascasone [online]. 2021 [cit. 2024-02-21]. Dostupné z: https://www.rascasone.com/cs/blog/html-pro-zacatecniky-jak-psat-web
20. DEAN, John. Web programming with HTML5, CSS, and JavaScript [online]. Burlington: Jones & Bartlett Learning, 2019 [cit. 2023-04-28]. ISBN 978-128-4091-793. Dostupné z: http://projanco.com/Library/Web%20Programming%20with%20HTML5,%20CSS,%20and%20JavaScript.pdf
21. PHP GROUP. What is PHP? Php [online]. c2001-2024 [cit. 2024-01-10]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php
22. REFSNES DATA. PHP Syntax. W3Schools [online]. c1999-2024 [cit. 2024-01-22]. Dostupné z: https://www.w3schools.com/php/php_syntax.asp
23. TUTORIALS POINT. PHP – Syntax. Tutorials Point [online]. c2024 [cit. 2024-01-22]. Dostupné z: https://www.tutorialspoint.com/php/php_syntax_overview.htm
24. LUTZ, Mark. Learning Python [online]. 4th ed. Sebastopol: O'Reilly, c2009 [cit. 2024-02-10]. ISBN 978-0-596-15806-4. Dostupné z: https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf
25. SANCHHAYA EDUCATION PVT. LTD. Introduction To Python. GeeksforGeeks [online]. 2023 [cit. 2024-02-10]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-python/
26. PYTHON SOFTWARE FOUNDATION. What is Python? Executive Summary. Python [online]. c2001-2024 [cit. 2024-02-10]. Dostupné z: https://www.python.org/doc/essays/blurb/
27. VANDERPLAS, Jake. A Whirlwind Tour of Python [online]. O'Reilly Media, 2016 [cit. 2024-02-10]. ISBN 978-1-491-96465-1. Dostupné z: https://github.com/jakevdp/WhirlwindTourOfPython
28. KROLL, Nicolas a Damian MURAWSKI. What makes Symfony framework a great choice for PHP web development? BITBAG. BitBag [online]. 2022 [cit. 2024-02-14]. Dostupné z: https://bitbag.io/blog/symfony-framework-a-great-choice-for-php-web-development
29. KIENLEIN, Manuel. PHP Symfony Is a great Framework to Create a Web Application. Medium [online]. 2021 [cit. 2024-03-21]. Dostupné z: https://medium.com/codex/php-symfony-is-a-great-framework-to-create-a-web-application-882fb98c7a2
30. AHMED, Shahzeb. Why Developers Prefer PHP Programming Language For Web Development. CLOUDWAYS LTD. Cloudways [online]. 2022 [cit. 2024-03-27]. Dostupné z: https://www.cloudways.com/blog/php-developer/
31. AdminLTE Bootstrap Admin Dashboard Template. AdminLte [online]. c2014-2024 [cit. 2024-03-23]. Dostupné z: https://adminlte.io/

32. SOFTWARE FREEDOM CONSERVANCY INC. Selenium 4.18.1. PYTHON SOFTWARE FOUNDATION. PyPI [online]. c2024 [cit. 2024-03-24]. Dostupné z: https://pypi.org/project/selenium/
33. OWENS, Martin. Python-crontab 3.0.0. PYTHON SOFTWARE FOUNDATION. PyPI [online]. c2024 [cit. 2024-03-24]. Dostupné z: https://pypi.org/project/python-crontab/
34. APACHE FRIENDS. XAMPP Installers and Downloads for Apache Friends [online]. c2023 [cit. 2024-03-23]. Dostupné z: https://www.apachefriends.org/
35. PHP GROUP. PHP: Hypertext Preprocessor [online]. c2001-2024 [cit. 2024-03-23]. Dostupné z: https://windows.php.net/download
36. ADERMANN, Nils a Jordi BOGGIANO. Composer: A Dependency Manager for PHP [online]. 2024 [cit. 2024-03-23]. Dostupné z: https://getcomposer.org/
37. OPENJS FOUNDATION. Run JavaScript Everywhere [online]. 2024 [cit. 2024-03-23]. Dostupné z: https://nodejs.org/
38. RASPBERRY PI FOUNDATION. Raspberry Pi [online]. 2024 [cit. 2024-03-23]. Dostupné z: https://www.raspberrypi.com/software/

# 8 Seznam obrázků, tabulek, grafů a zkratek

## 8.1 Seznam obrázků

## 8.2 Seznam tabulek

# Přílohy

Příloha 1 – CD se zdrojovými kódy
Příloha 2 – Odkaz na stažení zdrojových kódů https://github.com/petrkruntorad/dp-thesis