FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Vývoj video přehrávače pro cestovatelského průvodce americantravelshow.com
Student:	Petr Hasman
Vedoucí:	Ing. Tomáš Vondra, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

American Travel Show je nekomerční webová stránka, která zprostředkovává informace o cestovatelských destinacích v USA. Momentálně nabízí knihovnu videí, která není funkční a obsahuje pouze lokálně uložená videa.

Ve spolupráci s televizním producentem Johnem Honnerem analyzujte požadavky pro tvorbu nové interaktivní správy knihovny. Pomocí moderních webových technologií navrhněte a implementujte řešení v podobě nového pluginu do WordPressu.

Správa umožní upravovat jednotlivé pořady, jejich playlisty a video soubory jak uložené na serveru, tak převzaté z YouTube. Výstupem je přehrávač, který je možné vložit na webové stránky. Přehrávač musí pracovat v různých prohlížečích a na různých zařízeních. V prohlížeči Chrome využijte možnosti vysílat video na televize podporující ChromeCast. Aplikace by měla kontrolovat dostupnost externích videí a v případě jejich smazání upozornit správce.

Otestujte funkčnost pluginu a proveďte test použitelnosti na přispěvatelích Czech-American TV.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 9. ledna 2021 v Praze.

Bakalářská práce

VÝVOJ VIDEO PŘEHRÁVAČE PRO CESTOVATELSKÉHO PRŮVODCE AMERICANTRAVELSHOW.COM

Petr Hasman

Fakulta informačních technologií ČVUT v Praze
Katedra softwarového inženýrství
Vedoucí: Ing. Tomáš Vondra, Ph.D.
13. května 2021

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu práce panu Ing. Tomáši Vondrovi, Ph.D., za ochotu a pomoc při realizaci této práce. Další významnou osobou, které bych zde rád poděkoval je John Honner, díky kterému jsem získal vhled do praxe v oboru vývoje webových aplikací. Dále bych rád poděkoval své rodině a přítelkyni za pomoc a pochopení při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 10. května 2020 .....

Abstrakt

Tato bakalářská práce se zabývá tvorbou video přehrávače formou doplňku pro redakční systém WordPress. Doplněk je vyvinut pro americkou neziskovou organizaci CATV Herald a její webovou aplikaci American Travel Show. Přehrávač podporuje přehrávání ucelených seznamů videí, funkcionalitu Google Cast pro přehrávání video obsahu na vzdálené obrazovce, a kromě přehrávání videí ve formátu MP4 a WebM, také přehrávání videí z platformy YouTube.

V První části práce je popsán postup při volbě vhodného JavaScriptového přehrávače, a poté možnosti rozšíření zvoleného přehrávače o další funkce tak, aby plně vyhovoval požadavkům zadavatele.

V další části práce jsou popsány základní principy redakčního systému WordPress a především systém reprezentace a persistence dat.

Dále je v práci popsána implementace administrační části doplňku, a to zejména systému pro nahrávání videí, vkládání videí jakožto odkazů na platformu YouTube, tvorbu a úpravu seznamů videí a systému pro notifikace administrátora stránky na změnu dostupnosti video obsahu z platformy YouTube.

Dále je v práci detailněji přiblížena implementace rozšíření zvoleného přehrávače o funkce potřebné pro splnění požadavků zadavatele. Zejména se jedná o implementaci rozhraní pro přehrávání ucelených seznamů videí a implementaci funkcionality Google Cast.

V závěru práce je popsán průběh a výsledky testování přehrávače. Testování funkčnosti přehrávače proběhlo v rámci několika nejpoužívanějších prohlížečů. Uživatelské testování bylo provedeno za účelem otestování administračního rozhraní doplňku a proběhlo za pomoci několika přispěvatelů do projektů American Travel Show a Czech American TV USA.

Klíčová slova WordPress doplněk, notifikace, seznam videí, video přehrávač, Google Cast, YouTube, Plyr

Abstract

This bachelor's thesis describes the implementation of a video player as a plugin for the WordPress content management system. The plugin is developed for the American non-profit organization CATV Herald and its web application American Travel Show. The Player supports playback of video playlists, Google Cast functionality for remote video playback and in addition to MP4 and WebM videos playback, YouTube videos playback.

The first part of the thesis describes the procedure for selecting a suitable JavaScript player and then the possibility of extending the selected player with additional functions to fully meet the requirements of the client.

The next part of the thesis describes the basic principles of the WordPress content management system and especially the system of data representation and persistence.

Furthermore, the thesis describes the implementation of the administration part of the plugin, especially the system for uploading videos, inserting videos as links to the YouTube platform,

creating and editing video playlists and a system for notifying the site administrator to change of the availability of video content from YouTube.

Furthermore, the thesis describes in more detail the implementation of the extension of the selected player with the functions needed to meet the requirements of the client. In particular, it involves the implementation of an interface for playlists playback and the implementation of the Google Cast functionality.

At the end of the thesis, the process and results of testing the player are described. The functionality of the player was tested in several of the most used browsers. User testing was performed to test the plugin's administration interface with the help of several contributors to the American Travel Show and Czech American TV USA projects.

Keywords WordPress plugin, notifications, video playlist, video player, Google Cast, YouTube, Plyr

Shrnutí

Motivace

Práci jsem si vybral, jelikož mne zaujala práce Johna Honnera v rámci neziskové organizace CATV Herald a projektu American Travel Show. V rámci tohoto projektu je momentálně většina obsahu nedostupná a výsledkem mé práce je opětovné zpřístupnění tohoto obsahu veřejnosti.

Zároveň mne pro volbu tohoto tématu motivovala skutečnost, že přehrávač bude v budoucnu aktivně využíván.

Cíle práce

Prvním cílem práce je analýza a konkretizace požadavků zadavatele a analýza stávajících řešení a jejich nedostatků.

Dalším cílem práce je analýza dostupných technologií pro implementaci. Především se jedná o analýzu dostupných přehrávačů, redakčního systému WordPress a SDK pro implementaci funkcionality Google Cast.

Dalším cílem práce je implementace doplňku. Doplněk musí umožnit tvorbu a úpravu seznamů videí. Součástí těchto seznamů budou jak lokální videa, tak také videa z platformy YouTube. Dále doplněk musí implementovat systém pro notifikaci administrátora webových stránek, pokud se změní dostupnost některého z videí z platformy YouTube. V neposlední řadě musí přehrávač podporovat využití funkce Google Cast pro přehrávání videí na zařízeních podporujících přijímání obsahu pomocí této funkce.

Posledním cílem práce je konzultace a otestování výsledku práce se zadavatelem a s dalšími přispěvateli do projektů. V rámci testování je zhodnocena funkčnost doplňku a přívětivost jeho administračního rozhraní.

Postup

V první části analyzuji požadavky zadavatele, stávající řešení a dostupné technologie. Poté porovnám tři JavaScriptové přehrávače, které byly vybrány z nejvíce používaných přehrávačů. Z tohoto porovnání vzejde jeden, který využiji pro své řešení.

Dále analyzuji redakční systém WordPress. V rámci analýzy stručně shrnu historii redakčního systému, základní architekturu a myšlenky systému, které využiji při návrhu architektury a implementaci mého řešení.

Dále popíši implementaci doplňku. Zde shrnu případy užití doplňku, systém reprezentace a persistence dat a představím nejvýznamnější části mého řešení.

V poslední fázi shrnu průběh a výsledky testování vyvinutého doplňku.

Výsledky práce

Výsledkem práce je doplněk pro redakční systém WordPress. Tento doplněk podporuje vkládání videí a to jak nahraná na server pomocí knihovny médií, či pomocí protokolu FTP, tak také vložením videa formou odkazu na platformu YouTube.

Dále doplněk podporuje tvorbu, úpravu a přehrávání seznamů videí. Videa je možné přehrávat samostatně, v rámci seznamů videí, nebo formou přehrání všech dostupných videí. Přehrávač videí také podporuje přehrávání obsahu na vzdálené obrazovce pomocí technologie Google Cast.

Dále doplněk podporuje notifikace při změně dostupnosti videí v rámci platformy YouTube. Obsah je pravidelně kontrolován a při detekci nedostupného videa je odeslán e-mail správci aplikace. Dále doplněk umožňuje zobrazit seznam nedostupných videí a změnit URL adresu těchto videí, nebo je odstranit.

Závěr

Implementace doplňku proběhla úspěšně. V rámci testování se vyskytlo několik menších problémů, které byly na základě těchto zjištění opraveny.

Seznam zkratek

API	Aplication Programming Interface
CMS	Content Management System
ES6	ECMAScript 6
JS	JavaScript
PHP	PHP: Hypertext Preprocessor
RSS	Really Simple Syndication
SDK	Software Development Kit
URL	Uniform Resource Locator
W3C	World Wide Web Consortium

Kapitola 1
Úvod

Ve vyspělých zemích je dnes internet nedílnou součástí každodenního života. Počet uživatelů internetu stále roste a předpokládá se, že do roku 2023 bude internet využívat 66% světové populace. S rostoucím zájmem o internet se také snižuje cena za poskytování připojení a na druhé straně roste rychlost připojení k internetu. Díky těmto trendům zaujímá video obsah stále větší podíl celkového obsahu internetu.

Tohoto trendu si také všimla americká nezisková organizace CATV Herald, která produkuje video obsah již od roku 2003. V roce 2015 v rámci této organizace vznikl projekt American Travel Show. Cílem tohoto projektu je poskytovaní edukativního obsahu o amerických státech. Tohoto cíle dosahuje právě pomocí video obsahu ve formě spotů o zajímavých cestovatelských destinacích v rámci jednotlivých států USA.

Tato práce se zabývá tvorbou doplňku pro redakční systém WordPress sloužícího k tvorbě a přehrávání seznamů videí pro výše zmíněnou webovou aplikaci American Travel Show. Seznamy videí jsou členěny podle jednotlivých států, které jsou reprezentovány interaktivní mapou. Kliknutím na některý ze států je uživatel přesměrován na stránku věnující se vybrané oblasti. Stránka obsahuje stručné shrnutí informací o daném státu a především související seznam videí.

V současné době je nicméně přehrávání videí nedostupné, jelikož doplněk, který zajišťoval přehrávání videí v minulosti, již není funkční. Zároveň také původní doplněk nepodporoval některé funkce, jako je například přehrávání obsahu na vzdálené obrazovce.

Hlavními důvody pro vytvoření nového přehrávače je opětovné zpřístupnění video obsahu, poskytnutí možnosti využít dalších moderních technologií a v neposlední řadě také usnadnění administrace obsahu vytvořením přívětivějšího uživatelského rozhraní. Dále je nutné zajistit, aby doplněk co nejvíce vyhovoval způsobu využití v projektu American Travel Show, ale aby zároveň bylo možné přehrávač případně využít i na dalších sesterských projektech organizace CATV Herald. V neposlední řadě je potřeba minimalizovat riziko, že v budoucnu doplněk přestane plnit svůj účel.

Práce se zabývá rozesíláním emailových notifikací, výběrem nejvhodnějšího volně dostupného video přehrávače, jeho rozšířením pro splnění všech požadavků a popisem tvorby uživatelsky přívětivého rozhraní pro správu a přehrávání video obsahu z různých zdrojů.

Tato práce se naopak nebude zabývat kompletní implementací přizpůsobitelného HTML5 přehrávače. Dále se práce nezabývá využitím jiných služeb pro přehrávání obsahu na vzdálené obrazovce než je Google Cast. Je to z toho důvodu, že Google Cast a Apple Airplay jsou nejrozšířenějšími způsoby pro přehrávání obsahu na vzdálené obrazovce, a zároveň funkci Apple Airplay podporuje nativně většina přehrávačů a funguje automaticky na všech zařízeních podporujících Apple Airplay.

Téma práce jsem si zvolil, jelikož mne zaujal způsob, kterým bude výsledná práce využívána. Dále je pro mě důležité, že práce bude využita pro smysluplný účel. Výsledek práce totiž může

být využit kýmkoli, kdo uvažuje o vycestovaní do USA a není si jistý, do jakého státu vyrazit, nebo si není vědom všech zajímavostí, které stojí za to na své cestě vidět na vlastní oči.

Dále mne toto téma zaujalo díky využití moderních technologií jako je WordPress, či Google Cast, které zaujímají podstatnou část trhu a u kterých se dá předpokládat, že budou dále rozšiřovány a využívány stále větším počtem uživatelů.

Kapitola 2
Cíl práce

Hlavním cílem této práce je vytvoření doplňku pro redakční systém WordPress. Účelem tohoto doplňku je opětovné zpřístupnění video obsahu návštěvníkům webové aplikace American Travel Show. Hlavní částí doplňku je video přehrávač umožňující přehrávání videí lokálních videí a také videí z platformy YouTube. Tato videa jsou přehrávána především v rámci seznamů videí. Tyto seznamy je v rámci doplňku možné vytvářet a editovat. Dále doplněk poskytuje systém notifikací administrátora stránky při změně dostupnosti některého z videí v rámci platformy YouTube.

Prvním cílem práce je analýza a konkretizace požadavků zadavatele na funkce přehrávače a administračního rozhraní doplňku. V rámci plnění tohoto cíle proběhne seznámí se zadavatelem, projektem American Travel Show a stávajícím administračním prostředím tohoto projektu.

Druhým cílem práce je analýza stávajícího řešení a technologií dostupných pro tvorbu řešení nového. Zde se zaměřím především na analýzu nedostatků stávajícího řešení, výběr vhodného přehrávače a možnostmi jeho rozšíření.

Třetím cílem práce je návrh a implementace doplňku dle specifikovaných požadavků. Při implementaci je především důležité vytvořit intuitivní administrační rozhraní doplňku, aby doplněk mohli využívat i uživatelé s malými, či dokonce žádnými zkušenostmi s redakčním systémem WordPress

Posledním cílem práce je otestování implementace. V rámci tohoto bude otestována funkčnost přehrávače v různých prohlížečích a na různých zařízeních a také bude otestována přívětivost administračního rozhraní za pomoci několika přispěvatelů do projektu American Travel Show.

Kapitola 3

Analýza stávajících řešení a dostupných technologií

V první části této kapitoly se budu zabývat konkretizací požadavků zadavatele na vyvíjený přehrávač. V další části se stručně zaměřím na analýzu stávajících řešení. Z této analýzy vyplyne důvod, proč je nutné se tvorbou nového řešení zabývat.

V poslední části této kapitoly se zaměřím na analýzu dostupných JavaScriptových video přehrávačů. Z této analýzy poté vyplyne, který přehrávač je nejvhodnější pro realizaci zadaných požadavků a který ve svém řešení využiji.

3.1 Analýza požadavků zadavatele

V této části práce se budu zabývat podrobnější analýzou požadavků zadavatele. Konzultace se zadavatelem Johnem Honnerem probíhala prostřednictvím telefonních a video hovorů. Po představení projektu, který bude doplněk primárně využívat, jsme vyvodili následující požadavky.

Doplněk musí umožňovat tvorbu seznamů videí (*playlistů*). Dále musí doplněk umožňovat správu jednotlivých položek video seznamů. Tato správa musí zahrnovat možnost nahrávat videa přímo na úložiště webové aplikace. Dále musí správa umožňovat ukládat odkazy na video obsah, který je součástí platformy YouTube. Správa video obsahu musí také umožňovat editaci názvu videa a případně i popisu videa, který poté může být vypisován při přehrávání videa. V neposlední řadě musí administrace umožňovat odstranění video obsahu.

Součástí administrace doplňku musí být také systém umožňující notifikaci správce webového obsahu při změně dostupnosti videa uloženého formou odkazu na platformu YouTube. Notifikace musí proběhnout jak v rámci redakčního systému, tak také odesláním notifikace na e-mailovou adresu správce. Pokud dojde ke změně dostupnosti některého videa z platformy YouTube, musí být také možné v administraci video obsahu změnit URL adresu tak, aby odkazovala na jiné video.

Administrační rozhraní doplňku musí být také uživatelsky přívětivé a to především s ohledem na to, že velkou část přispěvatelů tvoří lidé, kteří přichází s redakčním systémem WordPress do kontaktu poprvé.

Doplněk musí také umožnit označovat videa jako reklamní spoty, a hlavní obsah. Videa označená jako reklamní spoty je pak nutné při přehrávání seznamů videí s různou frekvencí vkládat mezi hlavní přehrávaný obsah.

Požadavků na uživatelskou část bylo méně. Hlavní požadavek plyne z administrační části a to je, že doplněk musí umožňovat přehrávání video obsahu v prohlížeči návštěvníka stránky.

Video obsahem může být video nahrané přímo na úložiště aplikace nebo video, které je součástí platformy YouTube.

Dalším požadavkem adresovaným uživatelské části doplňku je umožnění přehrávání video obsahu na televizní obrazovce. Po další konzultaci byl tento požadavek zúžen na umožnění přehrávání video obsahu na televizní obrazovce pomocí funkce Google Cast.

Dále musí být možné doplněk využít pro přehrávání veškerého uloženého video obsahu bez ohledu na příslušnost k seznamu videí. Jinými slovy musí být možné doplněk využít takovým způsobem, aby bylo možné přehrát všechna uložená videa, jako by patřila všechna do jednoho výchozího seznamu videí bez nutnosti takový seznam vytvářet.

Na závěr konkretizace požadavků jsme vyvodili, že doplněk musí být dostatečně obecný, aby bylo možné jej použít nejen v rámci webové aplikaci American Travel Show, ale v budoucnu také pro účely internetového vysílání v rámci sesterského projektu Czech American TV USA.

Po sumarizaci dostáváme tedy následující seznamy funkčních a nefunkčních požadavků:

3.1.1 Funkční požadavky

F1 Nahrávání video obsahu

F2 Odstranění uloženého video obsahu

F3 Ukládání odkazů na obsah z platformy YouTube

F4 Úprava názvu a popisu video obsahu

F5 Označení videa jako reklamní spoty, a nebo jako hlavní obsah

F6 Tvorba seznamů videí

F7 Úprava seznamů videí

F8 Systém notifikací při odstranění videa z platformy YouTube

F9 Přehrávání vytvořených seznamů videí

F10 Přehrávání veškerých uložených videí v rámci výchozího seznamu videí

F11 Přehrávání videí označených jako reklamní spoty mezi hlavním obsahem

F12 Přehrávání videí na televizní obrazovce pomocí Google Cast

3.1.2 Nefunkční požadavky

N1 Intuitivní administrační rozhraní

N2 Přívětivé rozhraní přehrávače

N3 Možnost využití doplňku i pro další projekty využívající CMS WordPress

N4 Přehrávač bude fungovat v různých prohlížečích a na různých zařízeních

3.2 Analýza stávajících řešení

Nyní představím momentálně využívané řešení a podobné řešení. Zaměřím se především na nedostatky stávajícího řešení a možnost jejich zlepšení v rámci implementace nového řešení.

3.2.1 ATS video plugin

ATS Video plugin je řešení, které je momentálně využívané v rámci aplikace American Travel Show. Toto řešení je však momentálně nefunkční. Řešení využívá JS přehrávače JW Player. Pro tento přehrávač byla v minulosti vyjednána jednorázová licence. Jelikož však společnost, která přehrávač vyvíjí změnila svou cenovou politiku, platnost této licence byla ukončena a přehrávač přestal fungovat.

Přehrávač využitý doplňkem nepodporoval přehrávání obsahu na vzdálené obrazovce (televizi). Dále byl také v rámci doplňku problém s přehráváním videí z platformy YouTube na mobilních zařízeních. Tato funkcionalita nebyla podporována přehrávačem a v posledních letech vývojářská společnost přehrávače zcela odstranila možnost přehrávat videa z této platformy a to v rámci všech prohlížečů, nejen mobilních.

Dále tento doplněk potřeboval pro svou funkci další samostatný doplněk, který obsluhoval přímo samotný video přehrávač a licenci k tomuto přehrávači. Doplněk tedy neposkytoval jednotné administrační rozhraní. Při vývoji tohoto doplňku bylo pro ukládání informací o videích využito vlastní databázové schéma namísto využití schématu využívaného knihovnou médií, která je součástí redakčního systému. Toto může být problémem v budoucnu, kdy je vizí veškerý video obsah přesunout na Cloudové úložiště. Takový export by měl být jednoduchý, pokud bude video obsah uložen v rámci knihovny médií, jinak je ovšem tento proces značně komplikovanější.

Doplněk také nepodporoval nahrávání video obsahu na úložiště webové aplikace. S tímto problémem je spojený další nedostatek, a tím je aktualizace knihovny médií po nahrání videí pomocí FTP. Všechna videa musela být nahrána na platformu YouTube a veřejně přístupná. Tento přístup a politika vývojářů přehrávače ohledně videí z platformy YouTube by zapříčinily nefunkčnost doplňku, i pokud by nenastal problém s licencí k přehrávači

Tento doplněk tedy není vyhovující pro funkční požadavky zadavatele a je tedy nutné využít jiné řešení.

3.2.2 Radio Manager plugin

Tento doplněk je momentálně vyvíjen pro sesterský projekt Czech American TV. Jedná se o doplněk určený k tvorbě a správě jednotlivých rádiových stanic. Doplněk nepodporuje přehrávání video obsahu, ale je naopak přizpůsoben přehrávání pouze audio obsahu. Z toho také plyne, že nepodporuje přehrávání obsahu z platformy YouTube. Kvůli tomu také doplněk neimplementuje systém notifikací při změně dostupnosti obsahu.

Doplněk nicméně určitým způsobem podporuje tvorbu a přehrávání ucelených seznamů, nicméně tyto seznamy jsou tvořeny výhradně audio obsahem. Zároveň je administrační rozhraní doplňku přizpůsobeno právě pro tvorbu rádiových stanic a není možné využít ani toto administrační rozhraní.

Tento doplněk tedy také není možné využít pro řešení zadaného problému a jiné řešení zatím pro potřeby organizace nebylo vyvinuto. Další dostupné doplňky pro tvorbu seznamů videí neimplementují všechny požadované funkce pro splnění požadavků zadavatele. Je tedy nutné vyvinout zcela nové řešení.

3.3 Analýza video přehrávačů

V této části se budu zabývat analýzou dostupných video přehrávačů. Následně analýzu zhodnotím a vyberu nejvhodnější přehrávač pro využití při implementaci řešení. Důležitými aspekty pro výběr přehrávače jsou:

- Cena přehrávače
- Možnost přehrávání seznamů videí

- Možnost přehrávání obsahu z platformy YouTube
- Možnost využití funkce Google Cast
- Možnost rozšiřitelnosti a přizpůsobitelnosti
- Přehlednost a použitelnost API nabízeného přehrávačem

Rozšiřitelnost, přizpůsobitelnost a API přehrávače budu hodnotit na stupnici od 1 do 5, kde 1 značí velmi špatné zpracování daného parametru a 5 značí velmi dobré zpracování daného parametru. Při tomto hodnocení budu zohledňovat především přehlednost dokumentace k přehrávači, přehlednost a jednolitost API poskytovaného přehrávačem, a také komplikovanost rozšíření přehrávače o další ovládací prvky a funkce, či přizpůsobení vzhledu přehrávače.

3.3.1 Přehrávač JW Player

Pro využití tohoto přehrávače je třeba zakoupit licenci, kterou je potřeba pravidelně prodlužovat. V minulosti společnost poskytovala možnost zakoupit licenci jednorázově, od čehož však ustoupila a přehrávače s touto licencí přestaly fungovat. Cena za licenci začíná na 10 $/měsíc [1], což se nemusí zdát jako vysoká částka, nicméně pro neziskovou organizaci s omezenými finančními prostředky je to zbytný výdaj. Zvláštně v dnešní době, kdy existuje široké množství jiných přehrávačů, které je možné využít zdarma. Zároveň je v rámci této ceny omezené množství přehrání videí na 50 000 za měsíc, což by v budoucnu mohl být problém.

Přehrávač z dalších požadovaných funkcí nativně podporuje Google Cast. Dále podporuje tvorbu a přehrávání seznamů videí, avšak za využití doplňku. Přehrávač nepodporuje přehrávání videí z platformy YouTube. Pro starší verzi přehrávače existuje doplněk,který však není kompatibilní s funkcí Google Cast.

Přehrávač je možné rozšiřovat víceméně jen pomocí podporovaných doplňků. Další rozšíření, jako například přidání, či změna ovládacích prvků, momentálně možné není a rozšíření o další funkcionality by bylo možné pouze pomocí omezeného API pro ovládání přehrávače.

Co se týče přizpůsobitelnosti vzhledu, je zde velmi omezená možnost. Přehrávač je dostupný jakožto kompletní JavaScriptový balíček pro registrované a platící zákazníky. Ti si mohou přehrávač přizpůsobit pomocí grafického rozhraní v administraci svého účtu. Jedinou zbývající možností, jak provést zásadnější změny vzhledu přehrávače je vytvořením vlastních kaskádových stylů, které by přepisovaly styly dodané s balíčkem.

3.3.2 Přehrávač Video.js

Tento volně dostupný přehrávač nativně nenabízí přehrávání celých seznamů videí, videí z platformy YouTube, ani funkci Google Cast. Pro tento přehrávač sice existují doplňky, které tuto funkcionalitu zajišťují, nicméně tyto doplňky jsou vyvíjené komunitou a neprochází schvalovacím procesem a pravidelnou údržbou.

Další nevýhodou tohoto přístupu je částečná nekompatibilita doplňků. Pokud bych se rozhodl využít tento přehrávač s příslušnými doplňky, bylo by nutné implementovat také aplikaci pro zařízení podporující příjímání obsahu pomocí Google Cast. Jelikož je mým cílem zajistit co možná nejširší možnost využití této funkcionality, chtěl bych využít výchozí aplikace pro tato zařízení přímo od společnosti Google. Tím se také minimalizuje riziko ztráty této funkčnosti na určitých zařízeních v budoucnu. Dále pokud se změní rozhraní Google Cast SDK, bude potřeba upravit kód pro tuto funkcionalitu pouze na jednom místě.

Kvůli zpřístupnění některých funkcí přehrávače pomocí doplňků není vytvořená jednotná ucelená a přehledná dokumentace ke všem částem tohoto přehrávače. Velká část doplňků je také často málo udržovaná [2], čímž se zvyšuje riziko toho, že v budoucnu některé funkce přehrávače, či dokonce celý přehrávač přestane fungovat.

3.3.3 Přehrávač Plyr

Přehrávač Plyr je další volně dostupný přehrávač multimediálního obsahu. Tento přehrávač nativně nabízí přehrávání videí jak v klasických formátech, jako například MPEG4, tak také obsahu z platformy YouTube, nebo Vimeo.

Přehrávač je kompletně napsaný v jazyku JavaScript podle normy ES6, která má v dnešní době širokou podporu v moderních prohlížečích (97.87 % uživatelů internetu [3]). Dále používá pouze funkcionality HTML5 elementu video (podporuje 97.87 % uživatelů internetu [4]) a elementu iframe pro přehrávání videí z platformy YouTube (podporuje 88.13 % uživatelů internetu [5]).

Přehrávač však nepodporuje přehrávání seznamů videí a také nepodporuje funkcionalitu Google Cast. Na druhou stranu však přehrávač nabízí jednotné a přehledné API pro ovládání přehrávaného obsahu. Jeho rozšíření například o funkci přehrávání videí v rámci seznamů videí je poměrně jednoduché, jak se dozvíme dále.

Další výhodou tohoto přehrávače je také snadná přizpůsobitelnost. Přehrávač nabízí možnost nastavit, jaké se mají použít a vykreslit ovládací prvky. Této změny rozhraní je možné dosáhnout například vložením přímo HTML kódu, který se má na místo pro ovládací prvky vložit a vykreslit. Další možností je při inicializaci přehrávače zvolit, které z výchozích ovládacích prvků se mají použít. Přehrávač také nabízí jednoduchou možnost navázat vlastní funkce na určité události, jako je například ukončení videa. Přidat tedy další tlačítko a funkce pro podporu Google Cast je díky API tohoto přehrávače možné.

3.3.4 Volba přehrávače

Tabulka 3.1 Srovnání dostupných HTML5 přehrávačů podle potřebných funkcí

Parametr	JW Player	Video.js	Plyr
Rozšiřitelnost/přizpůsobitelnost	2/5	3/5	4/5
Přehlednost a jednolitost API	2/5	4/5	5/5
Cena	218 Kč / měsíc	0 Kč	0Kč
Youtube videa	Ne	Nutný plugin*	Ano
Plalisty	Nutný plugin	Nutný plugin	Ne
Google Cast	Ano	Nutný plugin*	Ne

**Pluginy nejsou kompletně vzájemně kompatibilní.*

Na základě předchozí analýzy jsem se rozhodl pro využití přehrávače Plyr. Tento přehrávač nativně podporuje přehrávání video obsahu jak pomocí HTML5 video přehrávače, tak také přehrávání videí z platformy YouTube pomocí embed funkcionality. Dále je přehrávač snadno rozšiřitelný a poskytuje přehledné ucelené API pro ovládání přehrávaného obsahu.

Při zhodnocení nevýhod tohoto přehrávače je možné vyvodit, že první nevýhodou je absence možnosti přehrávat ucelené video seznamy. Tuto překážku je však možné poměrně snadno vyřešit implementací vlastní nadstavby nad samotným přehrávačem, jelikož přehrávač poskytuje také možnost změnit přehrávaný obsah a také zachytávat problémy při načítaní a přehrávání obsahu.

Druhou nevýhodou je absence funkcionality Google Cast. Jelikož společnost Google nabízí SDK s přehledným API pro obsluhu a inicializaci této funkce a přehrávač nabízí možnost jednoduše přizpůsobit ovládací prvky přehrávače, bude možné rozšířit nadstavbu nad přehrávačem i o tuto funkcionalitu.

Protože zároveň budu mít plnou kontrolu nad odesíláním obsahu na vzdálenou obrazovku, bude také možné přehrávat na televizní obrazovce i videa z platformy YouTube bez nutnosti vytvoření další aplikace pro zařízení podporující přehrávání odesílaného obsahu z jiných zařízení.

Kapitola 4

Redakční systém WordPress

V první části této kapitoly se budu stručně věnovat historii redakčního systému WordPress. V další části analyzuji a osvětlím základní principy práce s daty v rámci tohoto systému a to zejména ve vztahu k ukládání a načítání dat souvisejících s video obsahem.

4.1 Historie redakčního systému

Redakční systém WordPress vznikl v roce 2003, kdy vznikla jeho první verze 0.70. Systém vznikl na základech platformy b2/cafelog. Tato platforma byla určená pro tvorbu tzv. *blogů*. Platforma b2/cafelog vznikla v roce 2001 a její vývoj pokračoval až do roku 2003, kdy ztratila převážnou většinu svých uživatelů. Iniciátory vzniku systému WordPress byli, v té době ještě aktivní uživatelé platformy b2/cafelog, Matt Wullenbeg a Mike Little.

O rok později, v roce 2004, byla vydána verze systému 1.2, která přinesla možnost systém rozšiřovat pomocí doplňků a také základní API pro usnadnění dalšího vývoje a rozšiřování systému. V tomto roce také narostl počet uživatelů platformy díky tomu, že konkurenční platforma *Movable Type* se rozhodla pro zpoplatnění své platformy pro většinu uživatelů. Velká část těchto uživatelů tedy přesunula své stránky právě na platformu WordPress.

Následně, v roce 2005, byla vydána verze 1.5. Tato verze přinesla rozšíření systému o další funkcionality podporující přizpůsobitelnost obsahu a to pomocí tzv. *themes*, neboli šablon, na základě kterých se dynamicky skládal obsah vytvářených stránek. Nová verze s sebou také přinesla systém správy jednotlivých stránek. Později v tomto roce byla představena verzi systému WordPress 2.0, která přinesla především přepsání hlavních funkcí systému, nový systém rolí uživatelů a také nové administrační rozhraní.

V následujících letech (do roku 2008 a verze 2.7) prošel WordPress mnoha zásadními změnami. Například se přidala podpora tzv. *widgetů*, byly představeny pojmy taxonomie, či štítek, a položily se základy hlavních konceptů, které se se systémem pojí dodnes [6, s. 2-4]. Od té doby prochází systém stálým vývojem a každý rok vychází přibližně jedna až tři nové verze [7]. Nové verze s sebou nepřináší pouze nové funkcionality, ale především bezpečnostní aktualizace. Tyto aktualizace jsou velmi důležité především proto, že je systém využíván velkým množství uživatelů, a proto je tento systém poměrně častým cílem hackerských útoků.

4.2 Základní architektura systému WordPress

Hlavní funkce, které jsou součástí jádra systému WordPress, je možné rozdělit do několika hlavních kategorií. [8, s. 62]

Příspěvky, stránky a vlastní obsah Tyto funkce slouží pro vytváření, ukládání a načítání většiny obsahu aplikace

Typy článků, taxonomie a metadata Funkce sloužící k vytváření vlastních typů článků, taxonomií a přidružených dat ukládaných s články

Šablony Funkce související s vývojem vlastních šablon pro vykreslování obsahu

Akce, filtry a doplňky Podpůrné funkce sloužící k rozšiřování systému pomocí doplňků a navazování určitých funkcí na různé fáze zpracování obsahu

Uživatelé a autoři obsahu Tvorba a správa uživatelů a jejich rolí pro údržbu správy přístupu

Formátování a komentáře Funkce související s vypisováním a ošetřováním vstupů a výstupů aplikace

4.2.1 Příspěvek, taxonomie, termín, štítek a metadata

Jelikož systém WordPress vznikal především jako platforma pro tvorbu blogů, většina principů a funkcí v systému je přizpůsobena právě pro tento webový obsah. S tím také souvisí hlavní principy ukládání a načítání dat.

Nejprve představím koncept příspěvků (*post*). Příspěvek je hlavní formou reprezentace dat a většina obsahu v rámci systému je reprezentována právě jako nějaký typ příspěvku. Tuto datovou reprezentaci si je možné představit jako základní stavební jednotku obsahu stránek. Obecně mají všechny příspěvky nějaký typ, identifikátor, obsah, stav (publikovaný, koncept, v koši, ...), autora a další typická společná metadata jako je například datum vytvoření, či poslední úpravy.

Pojmem souvisejícím s příspěvky jsou taxonomie (*taxonomy*). Je to speciální druh metadat k příspěvkům, podle kterých je možné příspěvky třídit, či vyhledávat. Pro demonstraci významu tohoto pojmu využiji základní hierarchickou taxonomii, která je součástí systému - kategorii. Řekněme, že bychom si chtěli vytvořit blog, který bude obsahovat recepty na vaření. Jednotlivé recepty bychom reprezentovali právě pomocí příspěvků. Recepty bychom ale chtěli také někam zařadit (například recept z *České kuchyně*, *Italské kuchyně*, ...). K tomu bychom využili právě kategorie. Jednotlivé položky kategorií (například *Česká kuchyně*) se nazývají termíny (*term*).

Taxonomie je možné rozdělit podle možnosti přiřazení více termínů k jednomu příspěvku (*vícenásobná/jednonásobná*), nebo podle toho, zda jednotlivé termíny mohou mít své podružné termíny (například *Italská kuchyně* → *Těstoviny*). Pak mluvíme o tom že je daná taxonomie *hierarchická/nehierarchická*. Základní systémem nabízenou nehierarchickou taxonomií jsou štítky (*tagy*).

Pojmem metadata (*meta*) rozumíme další přidružená data vztahující se k příspěvku, uživateli, či komentáři. Pomocí metadat můžeme uložit informace, které se nevejdou přímo do příspěvku, a pro které není vhodné využít taxonomie. Pokud bychom si například chtěli ukládat poslední IP adresu uživatele, který upravoval nějaký příspěvek, využili bychom právě metadata. Metadata vždy obsahují námi zvolený klíč a nějakou hodnotu. Metadata by měla sloužit především pro ukládání dodatečných informací k příspěvkům, která se často nemění a podle kterých se také často nevyhledávají příspěvky. Naopak jsou přizpůsobena k načítání těchto informací na základě konkrétního příspěvku.

Na druhou stranu je samozřejmě možné také podle hodnoty metadat vyhledávat příspěvky. Pokud bychom ale podle některých metadat vyhledávali často a tato metadata bychom vybírali z nějaké nepříliš rozsáhlé množiny, stálo by za zvážení, zda raději nevyužít taxonomií a termínů. Taxonomie jsou totiž přizpůsobené a optimalizované právě pro vyhledávání a třídění jednotlivých příspěvků, které jsou k nim přiřazené.

Pokud metadata, která ukládáme, závisí na volbě uživatele, využijeme pro zobrazení možností v rámci administrace příspěvků prvek s názvem *meta box*. Tento prvek je vlastně formulářový

prvek, který se zobrazuje v rámci administrace při editaci příspěvku, kterému je tento meta box přiřazen. Při přidávání meta boxu je potřeba vytvořit dvě funkce.

První funkce je volána při vykreslování editační stránky příspěvku a slouží k vykreslení formuláře, se kterým poté pracuje správce obsahu. Tato funkce je předána jako třetí argument funkci *add_meta_box* [9].

Funkce *add_meta_box* přijímá až 7 argumentů a má následující rozhraní:

Výpis kódu 4.1 Rozhraní funkce add_meta_box

```
<?php
    add_meta_box(
    $id,
    $title,
    $callback,
    $screen,
    $context,
    $priority,
    $callback_args
    )
php>
```

Prvním argumentem funkce je identifikátor vytvářeného meta boxu. Tento identifikátor je také využit jakožto *id* atribut vytvořeného HTML elementu.

Druhým argumentem funkce je *title*, neboli název meta boxu. Tento název je poté využit jakožto nadpis pro vygenerovaný HTML element.

Třetím argumentem funkce je tzv. *callback* funkce. Tato funkce slouží pro vykreslení obsahu meta boxu, jako například formulářových prvků.

Čtvrtým a nepovinným argumentem funkce je identifikátor obrazovky, na které se má daný meta box zobrazit. Tímto argumentem může být vlastní typ příspěvku (pak se tento meta box zobrazí při editaci příspěvku daného typu), nebo některý nativní typ příspěvku jako například *comment* nebo *link*. Tímto argumentem může být také pole typů příspěvků.

Pátým a nepovinným argumentem funkce je kontext vykreslení meta boxu. Tento argument určuje, v jaké části obrazovky má být meta box vykreslen. Možnými hodnotami tohoto argumentu, pokud je meta box přiřazen k příspěvku, jsou *normal* (v hlavní části obrazovky), *side* (po straně obrazovky) a *advanced* (v hlavní části obrazovky po meta boxech *normal*). Kontext může být také vytvořen vlastní.

Šestým a nepovinným argumentem funkce je priorita. Tento argument určuje, v jaké části specifikovaného kontextu má být meta box vykreslen. Možnými hodnotami pro tento argument (seřazeny od nejvyšší priority po nejnižší) jsou *high*, *core*, *default*, a *low*.

Posledním a nepovinným argumentem, který funkce přijímá jsou data, která mají být předána *callback* funkci. Data v tomto argumentu jsou předána jakožto vlastnost *$args* druhého argumentu předaného *callback* funkci.

Druhá funkce, kterou je nutné vytvořit při vytváření meta boxu, je funkce, která se volá při ukládání celého příspěvku. Tato funkce by měla obsluhovat ukládání metadat, která jsou získána z vykresleného formuláře v rámci meta boxu. Tuto funkci je nutné zaregistrovat pomocí funkce *add_action* [10] na událost *save_post*. Tím je zaručeno, že tato vytvořená funkce je zavolána právě ve chvíli, kdy se ukládá příspěvek. V rámci této funkce je nutné vyhnout se volání funkce *wp_update_post* [11], nebo před voláním této funkce odstranit a po volání této funkce opět přidat naši funkci pro uložení metadat. Pokud je funkce *wp_update_post* [11] zavolána bez ošetření, pak se systém dostane do nekonečné rekurze, kde se stále vzájemně volají vytvořená funkce pro uložení metadata a funkce *wp_update_post* [11].

4.2.2 Typy příspěvků

Typy příspěvků blíže specifikují o jaký druh obsahu se jedná. Určité typy příspěvků mají svá typická metadata, která se společně s nimi ukládají. Systém WordPress nabízí několik základních typů příspěvků. [12]

Článek (*Post*) Tento typ příspěvku je nejvíce využívaný a typický pro blogy. Vetšinou jsou zobrazovány od nejnovějších po starší a slouží také k vytváření RSS kanálů.

Stránky (*Pages*) Tyto příspěvky reprezentují jednotlivé podstránky v rámci webové aplikace. Hlavním rozdílem oproti článkům je to, že stránka může mít další podružnou stránku, což vytváří hierarchický web. Další rozdíl je ten, že se většinou zobrazují naopak od nejstarších po nejnovější. Dále typicky stránky nevyužívají taxonomií a štítků.

Přílohy (*Attachments*) Tento typ je specifický pro přílohy nahrané do aplikace, jako například obrázky či videa. Většinou obsahují přidružená metadata týkající se nahraných souborů jako je například typ souboru, délka trvání videa, nebo velikost souboru.

Revize (*Revisions*) Revize jsou speciálním typem příspěvku, jelikož slouží k ukládání historie změn ostatních příspěvků. Například pokud autor udělá při editaci příspěvku chybu a chtěl by se vrátit k jeho předchozí verzi.

Navigace (*Menus*) Díky tomuto příspěvku je možné vytvářet seznamy odkazů na stránky, které jsou součástí webové aplikace. Tyto seznamy je poté možné využít na různých místech jako navigaci. Tyto seznamy se editují v rámci správy šablon, tedy na jiném místě, než typicky ostatní příspěvky.

Vlastní kaskádové styly (*Custom CSS*) Tento typ příspěvku je specifický pro šablony a používá se pro ukládání stylů vytvořených v administraci aplikace.

Množiny změn (*Changesets*) Tento typ je podobný revizím, ale slouží speciálně pro ukládání změn provedených právě v rámci vlastních kaskádových stylů.

Kromě využívání zmíněných typů příspěvků, umožňuje systém také vytvářet typy vlastní. To stejné platí také pro taxonomie. Díky tomu je možné postihnout poměrně velké množství případů využití redakčního systému bez nutnosti zasahovat do databázového modelu. Pokud by však reprezentace dat pomocí různých typů příspěvků nebyla dostatečně vyhovující, je možné pomocí rozhraní *WP_Query* [13] změnit databázové schéma a v rámci schématu vyhledávat, či vkládat data.

Před zásahem do databázového modelu je však dobré si rozmyslet, zda opravdu není možné reprezentaci našich dat realizovat pomocí výchozího rozhraní případně vlastních typů příspěvků a taxonomií. Prvním důvodem, proč využít právě výchozí schéma, je ten, že k tomuto schématu jsou dostupné připravené funkce k vyhledávání či ukládání dat. Dalším důvodem, proč využít právě tyto výchozí funkce, je ten, že tyto funkce jsou přehledné a dobře se používají. Například není nutné vytvářet vlastní SQL dotazy a také za nás často tyto funkce řeší ošetření vstupů od uživatelů (například kvůli ochraně proti *SQL injection*). V neposlední řadě je dobré zmínit, že dotazy, které se generují pomocí výchozích funkcí pro získávání dat z databáze, jsou optimalizované k zamýšlenému účelu.

4.3 Hooks

Systém tzv. *Hooks* je jednou z nejdůležitějších součástí redakčního systému WordPress. V podstatě se jedná o způsob jak napojit vlastní funkce na různé fáze zpracování požadavků v rámci redakčního systému. Díky těmto funkcím je možné například ukládat přidružená metadata, či změnit výslednou podobu zpracovaných stránek.

Hooky se dají rozdělit do dvou kategorií - akce a filtry. Akce jsou vyvolávány událostmi při zpracování požadavků redakčním systémem. Filtry jsou použity pro úpravu dat před uložením do databáze, nebo před zobrazením uživateli.

Přiřazení vlastních funkcí k různým fázím zpracování požadavku se provádí pomocí dvou funkcí - *add_action* a *add_filter*. Tyto funkce mají obě stejné následující rozhraní:

Výpis kódu 4.2 Rozhraní funkce add_action souborů

```
<?php
    add_action( $tag, $function_to_add, $priority, $accepted_args );
php>
```

Obě funkce přebírají 4 argumenty. Prvním argumentem je název *hooku*, ke kterému se má funkce přiřadit. Druhým argumentem je samotná funkce, která se má k události přiřadit. Třetím nepovinným argumentem je priorita přiřazené funkce. Funkce přiřazené s nižší hodnotou tohoto argumentu jsou spuštěny dříve než funkce, které mají hodnotu tohoto argumentu vyšší. Posledním nepovinným argumentem funkce je počet argumentů, které přijímá přiřazovaná funkce. [8, s. 166]

4.4 Ošetření vstupů a výstupů

V rámci ochrany webové aplikace a také uživatelů je v rámci redakčního systému WordPress kladen vysoký důraz na ošetření vstupů a výstupů. K tomuto účelu nabízí systém poměrně široké množství předdefinovaných funkcí. Tyto funkce je možné dělit na dvě kategorie podle způsobu využití. Těmito kategoriemi jsou - funkce určené k ošetření uživatelských vstupů a funkce určené k ošetření výstupu v rámci šablon.

4.4.1 Přehled funkcí k ošetření vstupů

- **sanitize_email()** - Ošetření vstupu při očekávání e-mailové adresy
- **sanitize_file_name()** - Ošetření vstupu při očekávání názvu souboru
- **sanitize_html_class()** - Ošetření vstupu při očekávání názvu HTML třídy
- **sanitize_key()** - Ošetření vstupu při očekávání klíče (identifikátoru pro key/value data)
- **sanitize_meta()** - Ošetření vstupu při očekávání meta hodnoty
- **sanitize_mime_type()** - Ošetření vstupu při očekávání *mime type* řetězce
- **sanitize_option()** - Ošetření vstupu při očekávání *option* hodnoty na základě názvu *option*
- **sanitize_sql_orderby()** - Ošetření vstupu při očekávání *order by* hodnoty
- **sanitize_text_field()** - Ošetření vstupu při očekávání textového řetězce
- **sanitize_title()** - Přetvoření vstupu ve formátu názvu na část URL identifikátoru
- **sanitize_title_for_query()** - Ošetření vstupu při očekávání hodnoty z URL pro vyhledávání v databázi
- **sanitize_title_with_dashes()** - Ošetření vstupu při očekávání hodnoty pro využití v URL s nahrazením mezer za podtržítka
- **sanitize_user()** - Ošetření vstupu při očekávání uživatelského jména
- **esc_url_raw()** - Ošetření vstupu při očekávání URL adresy pro využití v rámci databáze

- **wp_filter_post_kses()** - Ošetření vstupu při očekávání obsahu příspěvku při zanechání pouze povolené HTML značky
- **wp_filter_nohtml_kses()** - Ošetření vstupu při očekávání obsahu příspěvku při odstranění všech HTML značek

4.4.2 Přehled funkcí k ošetření výstupů

- **esc_html()** - Ošetření výstupu obsahujícího HTML element
- **esc_url()** - Ošetření výstupu ve formě URL adresy
- **esc_js()** - Ošetření výstupu obsahujícího JavaScriptový kód
- **esc_attr()** - Ošetření výstupu obsahujícího hodnotu HTML atributu
- **esc_textarea()** - Ošetření výstupu předaného jako hodnotu pro *textarea* element

[14]

4.5 Jmenné konvence

V rámci redakčního systému WordPress je dbáno na jmenné konvence. Hlavním pravidlem pojmenovávání jednotlivých částí, funkcí a proměnných je využití tzv. *snake_case* stylu. V rámci tohoto stylu by u víceslovných názvů měly být mezery nahrazeny podtržítkem a jednotlivá slova začínat malým písmenem. V rámci redakčního systému je však jedna výjimka a tou je pojmenovávání tříd. Pravidlo s podtržítky zde platí také, ale jednotlivá slova by měla začínat naopak velkými písmeny.

V rámci redakčního systému jsou také pravidla pro pojmenovávání jednotlivých souborů. Názvy všech souborů by měly obsahovat pouze malá písmena. Dále by v názvu souboru měl být zřetelný jeho obsah. U souborů, které obsahují jednu třídy je pravidlem, že název souboru by měl začínat předponou *class-*. Dále by v názvu měl následovat název dané třídy obsažené v souboru. V rámci tohoto názvu by měla být nahrazena podtržítka pomlčkami. U rozhraní je konvence stejná jako u tříd s tím rozdílem, že názvy rozhraní by měly začínat předponou *interface-*.[15]

WordPress nabízí několik balíčků pro kontrolu programovacích standardů v rámci různých IDE. Nejširší balíček těchto standardů je balíček s názvem *WordPress*. V rámci tohoto balíčku jsou kontrolovány jmenné konvence, pravidla pro psaní mezer, komentářů, a spousty dalších kontrol, které jsou užitečné v rámci vývoje. Já jsem doplněk vyvíjel v editoru Atom, který má pro kontrolu kódu doplněk, a díky tomu jsem mohl využít právě tohoto balíčku pro kontrolu mého kódu.

Kapitola 5
Implementace

V této části nastíním postup implementace a návrh architektury. V první části se zaměřím především na administrační část doplňku. Poté nastíním postup pro rozšíření přehrávače o požadované funkce na uživatelské straně.

5.1 Návrh

V rámci popisu návrhu aplikace nejprve shrnu případy užití. S doplňkem bude nejvíce pracovat administrátor obsahu, či přispěvatel. Pro tohoto účastníka je nutné zajistit možnost přidávat videa, jejich úpravy a odstranění. Kromě videí samotných bude administrátor pracovat také se seznamy videí. Další účastník, který s doplňkem bude pracovat, je z druhé strany uživatel, který navštíví stránku pro přehrání připraveného obsahu. Předpokládanými případy užití doplňku je přehrání jednoho videa, přehrání seznamu videí a přehrání všech videí dostupných v rámci celé aplikace. Při přehrávání videí se před každým a za posledním videem přehraje nějaké video vybrané z videí označených v administraci jako reklamní spot. Uživatel má také možnost přehrávat videa na vzdálené obrazovce pomocí funkcionality Google Cast. Pokud se tuto funkci uživatel rozhodne využít, bude se obsah videa přehrávat na vzdálené obrazovce a v zařízení, ze kterého se video vysílá bude zobrazen náhledový obrázek a ovládací prvky pro ovládání přehrávače na vzdálené obrazovce.

Obrázek 5.1 Seznam účastníků

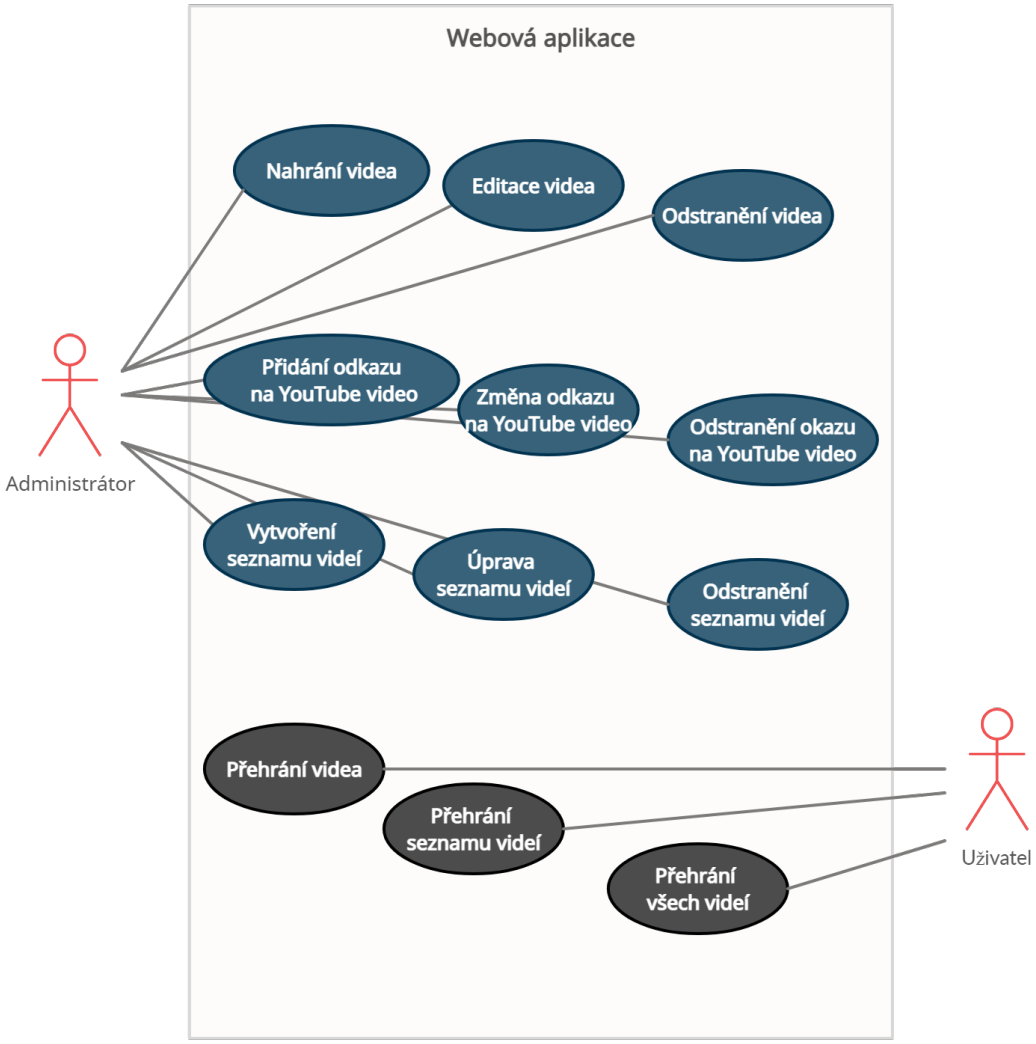
5.1.1 Seznam účastníků

- **Administrátor** - spravuje obsah v rámci aplikace. Přidává, edituje a maže videa. Zároveň se pod tímto pojmem skrývá účastník, který obdrží notifikaci, pokud se změní dostupnost některého videa z platformy YouTube. Tento účastník také reprezentuje běžného přispěvatele, který bude hledat, či vytvářet obsah a vkládat jej do aplikace.

- **Uživatel** - běžný návštěvník stránky. Přehrává videa a konzumuje obsah vytvořený administrátory obsahu.

5.1.2 Případy užití

Nyní detailněji představím jednotlivé případy užití aplikace a možné scénáře pojící se s těmito případy užití.



Obrázek 5.2 Případy užití

Administrátor

1. **Nahrání videa (1)** - Video, které chce administrátor přidat, je dostatečně malé pro nahrání pomocí PHP. Pro nahrání videa je využita knihovna médií. Administrátor vstoupí do správy obsahu a z navigace vybere položku *Media*. V podmenu vybere možnost *Add new*, čímž se dostane na stránku určenou k nahrání videa. Zde vybere příslušné video ze svého počítače a nahraje jej. Poté je přesměrován na stránku určenou k zobrazení přehledu nahraných příloh.
2. **Nahrání videa (2)** - Video, které chce administrátor přidat, má větší velikost, než podporuje server pro nahrání pomocí PHP. V tomto případě nahraje uživatel video pomocí

FTP do složky *wp-content/uploads/**. Po nahrání videa tímto způsobem však video není automaticky přidané do knihovny médií. Je tedy nutné přejít do administračního rozhraní a z navigace v části *Media* zvolit položku z podmenu *Update Video Library*. Po aktualizaci jsou všechna nahraná videa pomocí FTP přidána do knihovny médií a je možné s nimi dále pracovat. Administrátor je přesměrován na stránku s výpisem obsahu knihovny médií.

3. **Editace videa** - Tato část popisuje případ, kdy administrátor obsahu chce upravit informace o videu. Administrátor vstoupí do administračního prostředí aplikace. V navigaci zvolí položku *Media*. Po přesměrování na stránku s výpisem obsahu knihovny médií zvolí video, které chce upravit. Poté je přesměrován na stránku určenou k editaci zvoleného videa. Zde je možné upravit text, který se bude zobrazovat při přehrávání videa, zvolit, resp. změnit, zda je video reklamním spotem, nebo hlavním obsahem. Případně je také možné video odstranit.
4. **Odstranění videa** - Případ se týká situace, kdy administrátor chce odstranit nahrané video. Administrátor vstoupí do administračního rozhraní aplikace. V navigaci zvolí položku *Media*. Po přesměrování najde ve výpisu videí to, které chce momentálně odstranit. Nyní je možné na video najet myší a zvolit možnost Odstranit video. Poté vyskočí hláška s varováním, že po odstranění již video nebude dostupné a nebude jej možné obnovit z koše. Po odsouhlasení zobrazené hlášky je video odstraněno. Druhá možnost po nalezení videa je kliknout na video. Poté je administrátor přesměrován na stránku určenou k administraci konkrétního videa. V pravé části obrazovky je opět možnost odstranit video. Po kliknutí na tuto možnost se opět zobrazí varovná hláška. Po odsouhlasení hlášky je video odstraněno.
5. **Přidání odkazu na YouTube video** - Případ se týká situace, kdy administrátor chce do aplikace přidat video, které nalezl na platformě YouTube. Administrátor zkopíruje URL adresu videa. Poté vstoupí do administračního rozhraní aplikace. Nyní administrátor v podmenu položky navigace *Media* zvolí možnost *Add YouTube video*. Poté je administrátor přesměrován na stránku, kde má možnost vložit odkaz a uložit video. Po uložení videa je administrátor přesměrován na stránku sloužící k úpravě uloženého videa. Zde má administrátor stejné možnosti, jako při úpravě nahraného videa, pouze s tím rozdílem, že název videa a krátký popisek se automaticky uložili při nahrávání videa.
6. **Změna odkazu na YouTube video** - Změna odkazu probíhá na administrační stránce pro úpravu videí. Zde administrátor vidí meta box, ve kterém je aktuální URL adresa videa. Tu je možné změnit a video uložit.
7. **Odstranění odkazu na YouTube video** - Tento případ probíhá stejně, jako odstranění nahraného videa jen s tím rozdílem, že v knihovně médií vybereme video, které je ve skutečnosti pouze odkazem na YouTube video.
8. **Vytvoření seznamu videí** - Administrátor chce z videí, která jsou uložená v knihovně médií vytvořit seznam videí. Administrátor vstoupí do administračního rozhraní aplikace a v navigaci vybere položku *Video Playlists*. Na nové obrazovce stiskne tlačítko *Add New*. Poté je přesměrován na stránku určenou k vytvoření a úpravě seznamu videí. Zde vyplní název a popis seznamu. V další části vybere z rozbalovacího menu, které obsahuje videa, která nejsou označena jako reklamní spoty, video, které chce do seznamu přidat a stiskne tlačítko *Add Video*. Tento postup opakuje, dokud nejsou v seznamu všechna videa, která mají v seznamu být. Poté klikne na tlačítko v pravé části obrazovky *Publish*, čímž se seznam uloží.
9. **Úprava seznamu videí** - Administrátor vstoupí do administračního rozhraní aplikace a v navigaci vybere položku *Video Playlists*. Poté z přehledu uložených seznamů videí vybere ten seznam, který chce upravit. Po kliknutí na konkrétní seznam videí je administrátor přesměrován na stránku určenou k úpravě zvoleného seznamu videí. Zde může administrátor upravit název či textový popis seznamu. Dále má možnost přidat do seznamu další videa postupem, který byl popsán v předchozím bodě, nebo videa ze seznamu

odebrat pomocí kliknutí na odkaz *Remove Video* u konkrétního videa. Po provedení všech potřebných úprav administrátor klikne na tlačítko *Update* v pravé části obrazovky pro uložení provedených změn.

10. **Odstranění seznamu videí** - Administrátor vstoupí do administračního rozhraní aplikace, v navigaci zvolí položku *Video Playlists*. Poté je přesměrován na stránku, kde jsou zobrazeny všechny uložené seznamy videí. Po nalezení příslušného seznamu má administrátor dvě možnosti. První možností je najet na zvolený seznam videí a kliknout na odkaz *Remove*. Druhou možností je kliknout přímo na zvolený seznam a po přesměrování na stránku určenou k editaci seznamu videí, kliknout na odkaz v pravé části obrazovky s popisem *Remove*.

- **Uživatel**

1. **Přehrání videa** - Uživatel zvolí video, které chce přehrát. Po kliknutí na video je uživatel přesměrován na stránku s konkrétním videem. Po kliknutí na tlačítko *play* se video začne přehrávat. Po přehrání hlavního videa se přehraje video, které je vybráno z videí označených jako reklamní spot.
2. **Přehrání seznamu videí** - Uživatel zvolí na hlavní stránce stát, k němuž si přeje přehrát seznam videí. Po kliknutí na příslušný stát je přesměrován na stránku s příslušným seznamem videí. Po kliknutí na tlačítko *play* se začne seznam přehrávat. Před každým videem a za posledním videem se přehrávají videa, která jsou vybrána z videí označených jako reklamní spot.
3. **Přehrání všech videí** Uživatel na hlavní stránce nebo v menu zvolí možnost přehrát celý program. Po kliknutí je přesměrován na stránku s přehrávačem obsahujícím všechna videa. Po kliknutí na tlačítko *play* se začnou videa přehrávat. Před každým videem se přehrávají videa, která byla v administraci označená jako reklamní spot. Po přehrání určitého počtu videí se přehrávání pozastaví a na obrazovce se zobrazí hláška *Are you still watching?* Pokud uživatel klikne na tlačítko *play*, seznam se začne přehrávat dále.

5.2 Persistence dat

Pro ukládání dat spojených s videi a jejich seznamy jsou k dispozici dva způsoby, které je možné zvolit. První možností je vytvořit databázové schéma pro persistenci vlastních dat a pracovat s ním pomocí rozhraní *WP_Query* [13].

Druhá možnost je co nejvíce využít připravené rozhraní a případně vytvořit vlastní typ příspěvku či taxonomii. V rámci doplňku je potřeba ukládat videa, seznamy videí a označit videa jako reklamní spot, nebo hlavní obsah.

V rámci mého řešení jsem se pro ukládání videí a odkazů na YouTube rozhodl využít připravené knihovny médií. V rámci připraveného řešení je možné nahrávat, upravovat a odstraňovat nahraná videa. Chybí zde však možnost pracovat stejně s odkazy na YouTube a aktualizovat knihovnu médií po nahrání videí pomocí FTP.

Knihovnu médií je však možné rozšířit o požadovanou funkčnost pomocí připravených funkcí. První funkcí, kterou využiji je funkce *add_media_page()* [16]. Pomocí této funkce je možné do hlavní navigace administračního rozhraní přidat stránku pro přidávání odkazů na videa z platformy YouTube. Dále je nutné přidat meta box pro změnu odkazu na YouTube videa.

Další informací o videích, kterou je nutné v administraci nastavit je typ video obsahu (reklamní spot / hlavní obsah). K tomu využiji vlastní taxonomii. Při vytváření taxonomie zakáži její úpravu a přiřadím k ní dva vlastní termíny - *ATS Ad* a *Video*. Při vytváření taxonomie také zakáži možnost taxonomii upravovat, aby se omylem nestalo, že by se v administraci objevily nové termíny pro označení videa. Po vytvoření taxonomie ji zbývá jen přiřadit k příspěvkům typu příloha (*Attachment*). Tím zajistím, že se v administraci při editaci video obsahu bude

zobrazovat pole pro správné zařazení videa. Podle této taxonomie bude také možné vyhledávat videa nabízená pro přidání do seznamů videí.

Dalším typem dat, která je potřeba ukládat, jsou seznamy videí. V tomto případě jsem se rozhodl pro vytvoření vlastního typu příspěvku, který jsem pojmenoval *video_playlist*. Díky tomu bude možné se seznamy videí zacházet stejně jako se všemi ostatními příspěvky. Administrační rozhraní automaticky připraví stránku pro vytvoření či zobrazení seznamu uložených seznamů videí a automaticky bude zajištěno, že seznamy videí bude možné smazat.

Po vytvoření vlastního typu příspěvku však chybí možnost ukládat informaci o tom, která videa jsou spojena s jednotlivými seznamy videí. K uložení této informace využiji *post_meta*. Pro správu přidružených videí k jednotlivým seznamům, vytvořím k novému typu příspěvku meta box. Při ukládání seznamu videí zjistím ID videí, která mají být přiřazena ke konkrétnímu seznamu, uložím je do pole a toto pole po serializaci uložím jako *post_meta* s klíčem *vp_videos*.

Posledními daty, která je potřeba ukládat jsou informace o dostupnosti videí z platformy YouTube. Tyto informace jsem se rozhodl ukládat pomocí metadat. Jako klíč pro tuto hodnotu jsem zvolil *not-accessible*. Hodnotou, kterou budu s tímto klíčem ukládat je boolean hodnota (true/false). Vzhledem k návrhu databáze je však potřeba, aby i hodnotou metadat byl řetězec. Hodnoty tedy nebudu ukládat jako typ boolean, ale jako řetězec obsahující *true*, pokud je video nedostupné, či *false* pokud je video naopak dostupné.

5.3 Architektura aplikace

V této části se budu zabývat návrhem aplikace z hlediska architektury. Nejprve představím roztřídění jednotlivých souborů v rámci doplňku. Poté stručně shrnu popis jednotlivých tříd a jejich funkci v rámci doplňku.

5.3.1 Hierarchie souborů

Redakční systém WordPress má za sebou poměrně dlouhou historii a základní myšlenka návrhové architektury se se systémem nese vlastně od doby vzniku. Tento fakt s sebou nese i jistá úskalí. Poměrně rozšířeným návrhovým vzorem je v dnešní době vzor *Model, View, Controller* (MVC). Jedná se o vzor, ve kterém se využívají tři základní typy tříd.

Prvním typem tříd jsou tzv. *Modelové třídy*. Tyto třídy představují modelový návrh objektů z reálného světa, které chceme v naší aplikaci simulovat. Takovým objektem může být například příspěvek, kniha, video a další. Třída reprezentující takový objekt by měla popisovat především jeho vlastnosti a může případně obsahovat také funkce určené k realizaci interakcí ostatních tříd s daným objektem (především tzv. *gettery* a *settery*).

Dalším typem tříd v rámci architektury MVC je *View*. Třídy spadající do tohoto typu zajišťují vykreslování jednotlivých prvků, či celých stránek návštěvníkům. Typicky tyto třídy obsahují nějakou část HTML kódu s dalšími řídícími prvky například pro vykreslení určitých prvků vícekrát, či vykreslení pouze za splnění určité podmínky. Tyto třídy se poté vzájemně skládají do sebe a ve výsledku tvoří jednotlivé stránky, které se poté odesílají návštěvníkovi stránek jako celek.

Posledním typem tříd v rámci této architektury je typ *Controller*. Třídy spadající do tohoto typu zajišťují komunikaci typu „*požadavek - odpověď*“. V rámci těchto tříd probíhá načítání dat z databáze, vytváření modelových instancí, předávání dat do tříd typu *View* a odesílání sestavených HTML stránek a případně dalších souborů na stranu uživatele.

Tato architektura však není zcela ideální pro vývoj doplňku v rámci systému WordPress. Zde se nabízí spíše využití jednotlivých komponent, které zajišťují navěšování různých funkcí na tzv. *hooky*. Každá z komponent pak řeší svou jednotlivou část odpovědnosti.

Já jsem se ve svém řešení snažil využít právě návrhu s pomocí komponent, nicméně jsem se rozhodl oddělit od sebe funkčnost jednotlivých komponent a jejich vykreslování. Výsledná

adresářová struktura je tedy následující:

```
video-playlists-manager.php ..... hlavní soubor doplňku
includes ..... veškeré PHP třídy obsluhující doplněk
    controls ..... jednotlivé vykreslitelné prvky
    metaboxes ..... třídy obsluhující vykreslení a uložení meta boxů
admin
    js ..... JavaScriptové soubory obsluhující prvky administrace
    css ..... kaskádové styly pro prvky administrace
public
    js ..... JavaScriptové soubory obsluhující přehrávač
    css ..... kaskádové styly přehrávače
```

5.4 Administrace videí

V této části shrnu popis implementace rozhraní pro správu jednotlivých videí. V rámci návrhu administračního rozhraní jsem v maximální možné míře využil výchozích funkcí, které redakční systém nabízí. Tyto funkce jsem dále přizpůsoboval tak, aby postihly všechny potřebné případy užití doplňku.

5.4.1 Videa

Administrace videí využívá připravené knihovny médií. Pro nahrávání video souborů o velikosti podporované serverem využívá doplněk připravené rozhraní. Toto rozhraní však nepodporuje vkládání video obsahu z platformy YouTube a také není připravené na nahrávání větších souborů pomocí FTP přístupu k souborům. Dále také ve výchozím stavu knihovny médií není možné video soubory označit jako reklamní spot či hlavní obsah. Právě o tyto funkce jsem administrační rozhraní rozšířil.

5.4.2 Velké soubory

Prvním rozšířením administračního rozhraní je nahrávání video souborů větších, než podporuje server, na kterém běží aplikace. WordPress bohužel sám neumí aktualizovat knihovnu médií na základě nově nahraných souborů na webové úložiště. Rozšířil jsem proto administrační rozhraní o možnost aktualizace knihovny po nahrání souborů na prostory úložiště webové aplikace. K tomuto účelu jsem vytvořil v podmenu položky navigace *Media* položku pro aktualizaci knihovny médií.

Nejprve provedu vyhledání podporovaných video formátů, kterými jsou MP4 a WebM, kvůli zajištění podpory přehrání veškerého video obsahu při využití funkce Google Cast. Toto vyhledávání probíhá pomocí funkce *getAllVideosInDirectory* (5.1), kterou volám s parametry *wp-uploads*, což je složka do které se ukládají soubory nahrané přes výchozí systém WordPress. Druhým parametrem je prázdné pole, které funkci předávám referencí a které poté obsahuje všechny nalezené video soubory.

Funkce využívá především funkce *glob* [17], která slouží pro vyhledání všech souborů odpovídajících specifikovanému regulárnímu výrazu. V rámci funkce nejprve zkontroluji, zda adresář, ve kterém se funkce momentálně nachází, není prázdný. Poté vložím do předaného pole nalezené soubory s příponou *.mp4* a také *.webm*. Po přidání nalezených souborů se funkce rekurzivně zanořuje do všech podadresářů.

Po nalezení všech videí získané pole namapuji tak, aby každá položka obsahovala čas vytvoření, název souboru a cestu k souboru ze složky *wp-content*. Poté vzniklé pole procházím cyk-

Výpis kódu 5.1 Rekurzivní vyhledání všech podporovaných video souborů

```
public static function getAllVideosInDirectory( $dir, &$array ) {
    $contents = \glob( $dir );
    if ( count( $contents ) === 0 ) {
        return;
    } else {
        $array = array_merge( $array, glob( $dir . '*.mp4' ) );
        $array = array_merge( $array, glob( $dir . '*.webm' ) );
        self::getAllVideosInDirectory( $dir . '*/', $array );
    }
}
```

Obrázek 5.3 Meta box pro změnu URL adresy YouTube videa

lem. V tomto cyklu zkontroluji, že video ještě v knihovně médií není pomocí WordPress funkce *attachment_url_to_postid* [18]. Tato funkce vrátí identifikátor příspěvku s připojeným souborem s danou cestou, či 0, pokud takový příspěvek neexistuje. Pokud tedy zmíněná funkce vrátí 0, vložím soubor do databáze pomocí WordPress funkcí *wp_insert_attachment* [19], *wp_generate_attachment_metadata* [20] a *wp_update_attachment_metadata* [21]. První z těchto funkcí vloží nový soubor do databáze. Druhá ze zmíněných funkcí vygeneruje metadata o souboru, se kterými poté pracuje knihovna médií. Třetí zmíněná funkce uloží vygenerovaná metadata do databáze.

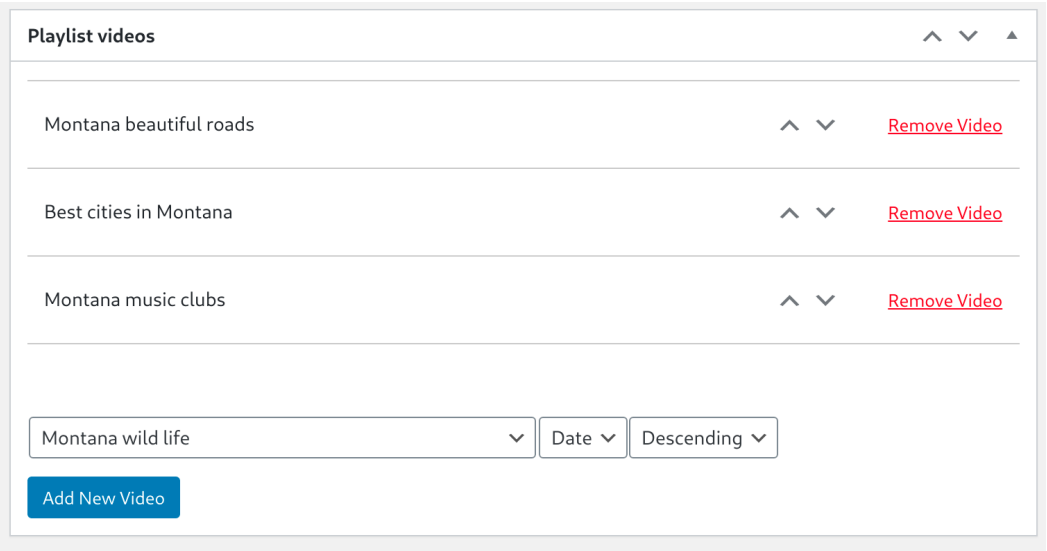
5.4.3 Ukládání YouTube videí

Další rozšíření administračního rozhraní pro správu médií, které jsem implementoval je možnost přidávání videí formou odkazu na platformu YouTube. Pro tuto funkcionalitu jsem opět v menu určenému správě videí vytvořil položku pro přidání videa z platformy YouTube.

K uložení videa formou odkazu na platformu YouTube jsem opět využil WordPress funkce *wp_insert_attachment* [19]. Této funkci je totiž možné předat i URL adresu, která odkazuje na jiný web. Po vložení odkazu stahuji informace o videu spojeném s daným odkazem. K tomu využívám API poskytovaného platformou YouTube. Díky tomuto API je možné získat informaci o stavu videa (zda je dostupné, či nikoli), název videa a popis videa. Tyto informace ukládám společně s odkazem do databáze a jako *mime-type* videí z platformy YouTube jsem pro své potřeby zvolil typ video/youtube.

Pokud se při získávání informací o videu vyskytne problém, uložím do databáze místo popisku a názvu videa pouze informaci o tom, že se informace o videu nepodařilo načíst, a také uložím jako metadata k takovému videu informaci o tom, že je toto video nedostupné.

Úprava videí poté probíhá prostřednictvím administračního rozhraní poskytovaného systémem WordPress, které slouží k úpravě všech multimediálních souborů. Pro videa z platformy YouTube jsem však vytvořil ještě další meta box 5.3, díky kterému je možné změnit URL adresu videa. Toto je pro případ, že by bylo video odstraněno, ale administrátor by našel video podobné. V takovém případě je zachován název i popis videa a pouze je nahrazená URL adresa v rámci seznamů videí, kterých bylo takové video součástí.



Obrázek 5.4 Meta box sloužící ke správě videí seznamu videí

5.4.4 Označení videa

Pro potřeby zadavatele je nutné, aby bylo možné videa označit jako reklamní spot, a nebo jako hlavní obsah. Při přehrávání jednotlivých seznamů videí se pak přehrávají primárně videa označená jako hlavní obsah. Mezi tato videa se frekvencí, kterou je možné nastavit v rámci hlavní administrace doplňku, vkládají videa označená jako reklamní spoty.

Pro tyto účely jsem využil možnosti vytvořit vlastní taxonomii. V rámci této taxonomie jsem vytvořil dva termíny - *ATS Ad* (reklamní spot) a *Video* (Hlavní obsah). Taxonomii jsem zvolil jakožto hierarchickou, jelikož v budoucnu může například vyvstat potřeba označit některá videa jako reklamní spot, který je vlastní, či reklamní spot, který je převzatý od jiného poskytovatele obsahu.

K vytvoření taxonomie jsem využil vestavěné funkce *register_taxonomy* [22]. Jako první parametr, kterým je název (identifikátor) taxonomie, jsem zvolil *video_category*. Dalším parametrem této funkce je typ příspěvku, ke kterému se má tato taxonomie přiřadit (zobrazovat při úpravě v administraci). Zde jsem zvolil typ *attachment*, kvůli čemuž se meta box pro tuto taxonomii bude zobrazovat i u dalšího multimediálního obsahu, jako je například obrázek. Nicméně při přístupu k datům specifikuji, že se má jednat pouze o příspěvky typu *attachment*, které mají hodnotu *mime_type* začínající řetězcem *video/*. Při výběru videí se tedy nestane, že by z databáze byly získány i obrázky či audio soubory. U těchto souborů tedy tato taxonomie nebude mít žádný vliv.

5.5 Seznamy videí

V této části popíši implementaci administračního rozhraní sloužícího ke správě seznamů videí. Pro vytváření seznamů videí jsem se rozhodl vytvořit vlastní typ příspěvku. K samotné registraci vlastního typu příspěvku slouží v rámci systému WordPress funkce *register_post_type* [23]. Tato funkce přebírá dva argumenty. Prvním argumentem je název typu příspěvku, který chceme zaregistrovat. S tímto názvem se poté také pracuje při filtraci či vyhledávání jednotlivých příspěvků. Já jsem jako název příspěvku zvolil *vp_video_playlist*.

Druhým argumentem, který je možné funkci předat, je pole s dalšími argumenty. V rámci tohoto pole je možné specifikovat jednotlivé popisky, které se v rámci administrace mají zobra-

zovat, jaká metadata daný typ příspěvku podporuje (např. náhledový obrázek, název, popis, ...), a další nastavení pro systém WordPress.

Pro ukládání videí, která jsou připojená ke konkrétním seznamům videí, jsem se rozhodl vytvořit další vlastní meta box 5.4. Tento meta box obsahuje seznam videí, která jsou k seznamu přiřazena. U každé položky představující jedno video je odkaz pro odstranění videa z konkrétního seznamu, a také ovládací šipky, kterými je možné videa v seznamu libovolně posouvat, čímž se změní pořadí přehrávání jednotlivých videí. Na konci tohoto meta boxu jsou poté prvky pro přidávání videí do seznamu. Prvním prvkem je menu, které obsahuje videa neoznačená jako reklamní spoty a které ještě nejsou přiřazená ke konkrétnímu seznamu videí. Dále jsou součástí těchto ovládacích prvku dvě menu, která slouží k seřazení jednotlivých položek v menu pro výběr videa. Tato videa je možné řadit podle data přidání, či podle názvu a to vzestupně či sestupně. Posledním prvkem je tlačítko, které po kliknutí přidá zvolené video do seznamu.

Při ukládání příspěvku jsou z meta boxu získány identifikátory přidružených videí. Takto získané identifikátory poté ošetřuji pomocí funkcí *wp_unslash* a *sanitize_text_field*. Po ošetření pole s identifikátory provedu serializaci tohoto pole pomocí funkce *wp_json_encode*. Po serializaci identifikátory uložím pomocí funkce *update_post_meta*, které předávám jako argumenty ID ukládaného příspěvku klíč, pod kterým jsou identifikátory videí uloženy *playlist-items* a posledním argumentem je řetězec obsahující serializované pole s identifikátory jednotlivých videí.

Po uložení metadat získaných z meta boxu ještě provádím kontrolu, zda textový obsah seznamu videí obsahuje značku *shortcode* pro vykreslení přehrávače při zobrazení stránky se seznamem videí. Pokud tuto značku text neobsahuje, je značka připojena na začátek textového obsahu.

5.6 Notifikace

Součástí administrace musí být také notifikace administrátora stránky pokud, je nějaké video odstraněno z platformy YouTube. Tyto notifikace musí být součástí administrace a zároveň se při zjištění změny dostupnosti videa v rámci platformy YouTube musí odeslat e-mailová notifikace administrátoru stránky. V rámci administračního rozhraní systému jsem vytvořil stránku, která obsahuje seznam jednotlivých videí, která jsou nedostupná. V rámci výpisu jednoho nedostupného videa je výpis seznamů videí, které obsahují dané video, vstupní *input* element pro zadání nové URL adresy videa a odkaz pro smazání videa. Kontrola dostupnosti videí probíhá ve čtyřech případech, které jsou popsány dále.

Přidání videa Prvním případem je vkládání odkazu na video do knihovny médií. Zde se pokusím načíst informace o videu pomocí API, které platforma YouTube nabízí. Součástí odpovědi v rámci tohoto API je i stav vyhodnocení požadavku. Pokud nebude stav v pořádku, rovnou video označím jako nedostupné. V opačném případě načtu informace o videu a uložím je.

Problém při přehrávání videa Další případ, kdy budu kontrolovat dostupnost videa je při přehrávání jednotlivých videí. Přehrávač Plyr dokáže vyhodnotit, že došlo při načítání videa k chybě. Na tuto událost jsem navázal funkci, která v takovém případě odešle požadavek na server. V těle tohoto požadavku je identifikátor daného videa, u kterého došlo k chybě při načítání. Na serveru se pokusím znovu načíst informace o videu. Toto provádím pro kontrolu, že je video opravdu momentálně nedostupné. U klienta totiž mohlo dojít například k dočasnému výpadku internetového připojení. Pokud stav odpovědi od YouTube není v pořádku, uložím o tom informaci k danému videu. V opačném případě považuji video za dostupné.

Cron Úloha Třetím případem kontroly dostupnosti videa je tzv. *Cron úloha*. Jedná se o script, který se spouští v určitém časovém intervalu (například každý týden). V tomto scriptu budu

procházet všechna videa, která zatím byla dostupná a kontrolovat jejich aktuální dostupnost. Pokud u nějakého videa zjistím, že je nově nedostupné, označím jej příslušně v databázi.

Vstup do administračního přehledu nedostupných videí Posledním případem, kdy dojde ke kontrole dostupnosti videí, je vstup do části administrace, kde je přehled nedostupných videí. Před vstupem do této části proběhne ještě jednou kontrola stavu videí, která jsou označená jako nedostupná. Tento krok zařazuji pro postihnutí případů, kdy služba YouTube bude mít pouze dočasný výpadek. V takovém případě předpokládám, že než se administrátor dostane ke správě videí, dojde k obnovení funkčnosti, a tudíž by tento krok měl zamezit falešným upozorněním na nedostupný obsah.

5.7 Uživatelská část

V této části se budu věnovat popisu implementace uživatelské části doplňku. Nejprve popíši rozšíření přehrávače o nutné funkce. V další části se budu věnovat rozšíření přehrávače o funkcionalitu Google Cast.

5.7.1 Přehrávač

Pro implementaci uživatelské části přehrávače jsem se rozhodl využít přehrávače Plyr. Tento přehrávač nabízí API pro usnadnění přehrávání pomocí HTML elementu *video*, a také videí z platformy YouTube. Přehrávač také umožňuje přehrávání videí z platformy Vimeo.

K inicializaci a ovládání přehrávače slouží třída *PlayerManager*, která je umístěna v souboru *public/js/playerManager.js*. Tato třída obsahuje 9 funkcí. První funkcí je konstruktor, v rámci kterého se nastavují proměnné využívané přehrávačem. Na konci konstruktoru se volá funkce *initPlayer*. Tato funkce vytvoří potřebné elementy a především samotný objekt přehrávače Plyr. Na konci této funkce se naváží připravené funkce na důležité události v rámci přehrávače Plyr.

Další funkcí je funkce *handleError*. Tato funkce je volána ve chvíli, kdy při načítání, či přehrávání videa nastane chyba. Součástí této funkce je také volání API pro označení YouTube videa jako nedostupné při chybě při načítání videa z platformy YouTube. V této funkci se také volá funkce *nextVideo* která slouží k načtení dalšího videa v pořadí v rámci seznamu videí.

Zmíněná funkce *nextVideo* slouží k načtení dalšího videa v pořadí. Tato funkce je také navázána na událost *ended*, která je vyvolána při dokončení přehrávání videa. Funkce přebírá dva argumenty. Prvním argumentem je objekt události. Druhým argumentem je argument *play*. Tento argument slouží k určení, zda po načtení dalšího videa má přehrávač automaticky začít video přehrávat a nebo nikoli. Tento argument může nabývat hodnot *true* nebo *false*. Součástí této funkce je také ověření, zda přehrávač přehrál poslední video seznamu. Funkce také určuje, zda dalším videem, které se přehraje, bude přehrán reklamní spot, a nebo video z hlavního obsahu. V rámci této funkce se také ověřuje, zda prohlížeč podporuje přehrávání formátu, kterého je následující video v seznamu. Dále je také v rámci této funkce zobrazována hláška *Are you still watching?* pokud se přehrál dostatečný počet videí.

Další funkcí, kterou třída implementuje je funkce *showVideoDescription*. Tato funkce je volána po načtení nového videa ze seznamu a slouží k zobrazení popisu aktuálně přehrávaného videa.

Dalšími funkcemi v rámci této třídy jsou funkce *areYouStillWatching* a *stillWatchingDissmiss*. Tyto funkce slouží k zobrazení, resp. skrytí hlášky *Are you still watching?* po přehrání nastaveného počtu videí.

Posledními funkcemi, které jsou součástí této třídy jsou funkce *getVideo* a *getPlayerControls*. První funkce slouží k získání zdroje videa které se má momentálně načíst, ve formátu, který vyžaduje přehrávač Plyr. Druhá zmíněná funkce slouží k získání řetězce obsahujícího HTML kód, na základě kterého se mají vykreslit ovládací prvky přehrávače.

5.7.2 Funkce Google Cast

Dalším rozšířením, které jsem implementoval je funkcionalita Google Cast. Ovládání této funkcionality zajišťuje třída *CastManager*, která je uložena v souboru *public/js/castManager.js*. Tato třída zajišťuje inicializaci rozhraní pro podporu této funkcionality, vytvoření instance třídy *PlayerManager* a následné ovládání přehrávaného obsahu. Tato třída obsahuje 22 funkcí.

Zásadní funkcí pro Google Cast SDK je funkce *initializeCastApi*. Tato funkce je volána po načtení stránky a slouží k inicializaci API pro sdílení obsahu. V rámci této funkce je zjištěno, zda prohlížeč, ve kterém je kód spuštěn, podporuje funkcionalitu Google Cast. Pokud ano je vytvořena instance třídy *CastManager* a nastaveny základní parametry pro SDK. Pokud prohlížeč, ve kterém je kód spuštěn, tuto funkcionalitu nepodporuje, je vytvořena pouze instance třídy *PlayerManager*.

Důležité je ještě podotknout, že pokud webová aplikace nepodporuje spojení přes protokol HTTPS, nebo má neplatný SSL certifikát, pak funkcionalita Google Cast nebude podporována. Tato funkce musí být přiřazena globálnímu objektu *window* a jeho vlastnosti *__onGCastApiAvailable*.

Další funkce v rámci této třídy slouží především k ovládání přehrávaného obsahu. Komunikace je navázána obousměrně. Pokud se tedy změní čas videa přehrávaného na zařízení, které obsah přijímá, je také aktualizován čas v rámci přehrávače na zařízení, které obsah vysílá, a aktualizován stav posuvníku signalizující postup v přehrávání videa. Pokud se ale změní čas přehrávaného obsahu na zařízení, které obsah vysílá (například posunutím jezdce na přehrávači v prohlížeči vysílajícího zařízení), je také příslušně posunut čas videa na zařízení, které obsah přijímá.

Ovládání přehrávače v prohlížeči vysílajícího zařízení probíhá především interakcí s instancí třídy *PlayerManager*. Funkce, které obsluhují lokální obsah, jsou navázány na události, které vyvolává rozhraní Google Cast SDK. Toto rozhraní nabízí například události změny času na přijímajícím zařízení, změny načteného obsahu, změny stavu vzdáleného přehrávače a další události.

5.8 Popis tříd a jejich funkcí

V této části shrnu jednotlivé implementované třídy, jejich účel a stručně popíši jejich rozhraní. V první části popíši třídy zaměřené na administraci doplňku, v druhé pak třídy obsluhující uživatelskou část doplňku.

5.8.1 Třídy administrace

Digram závislostí tříd je připojen v rámci přílohy C.1

VP_Manager Hlavní třída administrace. Slouží k definovaní konstant doplňku, jeho inicializaci, obsluhuje hlavní nastavení a nápovědu k doplňku. V rámci této třídy jsou také inicializované, deaktivované a odinstalované jednotlivé komponenty. Jednotlivé funkce této třídy jsou detailněji popsány v příloze A.1.

VP_Cron_Scheduler Třída obsluhující *Cron úlohy*. Třída registruje vlastní úlohy při inicializaci doplňku. Při deaktivaci, či odstranění doplňku pak zajišťuje jejich odstranění. Jednotlivé funkce této třídy jsou detailněji popsány v příloze A.2.

VP_Menu_Creator Třída sloužící k vytvoření vlastních položek navigace administračního rozhraní. Jednotlivé funkce této třídy jsou detailněji popsány v příloze A.3.

VP_Notifications Třída obsluhující notifikace ohledně nedostupných videí z platformy YouTube. Třída slouží k vytvoření administrační stránky, odesílání notifikačního e-mailu a aktualizaci stavu videí. Tato třída také registruje REST API pro přidání nedostupného videa při

chybě při načítání v rámci přehrávače. Jednotlivé funkce této třídy jsou detailněji popsány v příloze A.4.

VP_Post_Type_Creator Třída obsluhující registraci vlastního typu příspěvku. Tato třída také kromě vlastního příspěvku registruje taxonomii *video_category*, která slouží k označování jednotlivých videí jako reklamní spot, či hlavní obsah a přidává její termíny. Jednotlivé funkce této třídy jsou detailněji popsány v příloze A.5.

VP_Short_Code_Creator Třída zajišťující registraci vlastní krátké značky (*shortcode*) pro projení serverové strany aplikace s klientskou stranou. Třída kromě vytvoření této značky také připraví data pro klientský přehrávač. Jednotlivé funkce této třídy jsou detailněji popsány v příloze A.6.

VP_Videos Třída obsluhující především aktualizaci knihovny po nahrání videí pomocí FTP. V rámci této třídy jsou definovány funkce pro získání názvů nahraných souborů, vykreslení stránky pro aktualizaci a přidání jednotlivých videí do knihovny médií. Jednotlivé funkce této třídy jsou detailněji popsány v příloze A.7.

VP_Youtube_Videos_Manager Třída obsluhující videa z platformy YouTube. Třída přidává REST API pro získání URL odkazu na video ve formátu MP4, implementuje funkce pro získání informací o videu, získání nedostupných videí z databáze a přidávání videí. Jednotlivé funkce jsou detailněji popsány v příloze A.8.

VP_Add_Playlist_To_Post_Meta_Box Třída obsluhující meta box pro přidání seznamu videí do příspěvku. Jednotlivé funkce jsou detailněji popsány v příloze A.9.

VP_Edit_Youtube_Video_Url_Meta_Box Třída obsluhující meta box pro změnu URL adresy YouTube videa. Jednotlivé funkce jsou detailněji popsány v příloze A.10.

VP_Playlist_Videos_Meta_Box Třída obsluhující meta box pro editaci videí v rámci seznamu videí. Jednotlivé funkce jsou detailněji popsány v příloze A.11.

5.8.2 Třídy přehrávače

PlayerManager Hlavní třída přehrávače. Tato třída zajišťuje ovládání přehrávače Plyr, doplňuje funkčnost o přehrávání seznamů videí a odchytává chyby při přehrávání. Jednotlivé funkce této třídy jsou detailněji popsány v příloze B.1.

CastManager Třída obsluhující rozhraní Google Cast. Třída zajišťuje propojení lokálního a vzdáleného přehrávání, načítání obsahu na vzdálenou obrazovku a zobrazuje a skrývá ovládací prvky pro vzdálené přehrávání. Jednotlivé funkce této třídy jsou detailněji popsány v příloze B.2.

Kapitola 6
Testování

6.1 Testování prohlížečů

V této části testování se budu zabývat testováním přehrávače v deseti nejpoužívanějších prohlížečích. Testování probíhá za použití JavaScriptu pomocí nástroje browserstack. Prohlížeče a jejich verze jsou vybrány na základě nástroje W3Counter [24]. Údaje jsou platné k 30. 4. 2021.

V rámci testování prohlížečů jiných, než je prohlížeč Google Chrome, nebude testována podpora funkce Google Cast, jelikož tato funkcionalita není podporována jinými prohlížeči.

Tabulka 6.1 Nejpoužívanější prohlížeče včetně verzí k 30. 4. 2021

Pořadí	Prohlížeč, verze	procentuální zastoupení
1	Chrome, 89	32.51 %
2	Chrome, 90	13.79 %
3	Safari, 14	13.76 %
4	Chrome, 87	3.37 %
5	Chrome, 88	2.43 %
6	Edge, 89	2.22 %
7	Firefox, 87	2.02 %
8	Samsung, 13	1.80 %
9	Chrome, 83	1.48 %
10	Safari, 13	1.35 %

6.1.1 Chrome, verze 89

Testování přehrávače v prohlížeči Google Chrome verze 89 proběhlo úspěšně. Přehrávač byl schopný přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i hláška *Are you still watching?* se zobrazovaly v nastaveném intervalu. Po přehrání celého seznamu videí se přehrávač vrátil k prvnímu videu seznamu a přehrávání se pozastavilo. Všechny ovládací prvky přehrávače fungovaly podle očekávání.

6.1.2 Chrome, verze 90

Testování přehrávače v prohlížeči Google Chrome verze 90 proběhlo úspěšně. Přehrávač byl schopný přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i

hláška *Are you still watching?* se zobrazovaly v nastaveném intervalu. Po skončení přehrávání seznamu se přehrávač vrátil k prvnímu videu seznamu a přehrávání se pozastavilo. Všechny ovládací prvky přehrávače fungovaly podle očekávání.

6.1.3 Safari, verze 14

Testování v prohlížeči Safari verze 14 proběhlo téměř úspěšně. Přehrávač je schopný přehrávat videa ve formátu MP4 a také videa z platformy YouTube. Při testování se však vyskytl problém s videi formátu WebM. Tato videa nebylo v rámci prohlížeče možné přehrát. Zobrazování reklamních spotů a hlášky *Are you still watching?* proběhlo v pořádku. Stejně tak proběhlo v pořádku ukončení přehrávání a návrat k prvnímu videu po skončení přehrávání seznamu videí.

6.1.4 Chrome, verze 87

Testování přehrávače v prohlížeči Google Chrome verze 87 proběhlo úspěšně. Přehrávač byl schopný v tomto prohlížeči přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i hláška *Are you still watching?* se zobrazovaly dle nastavených intervalů. Po ukončení přehrávání celého seznamu se přehrávač vrátil k prvnímu videu seznamu a přehrávání se pozastavilo. Všechny ovládací prvky přehrávače fungovaly podle očekávání.

6.1.5 Chrome, verze 88

Testování přehrávač v prohlížeči Google Chrome verze 88 proběhlo úspěšně. Přehrávač byl schopný přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i hláška *Are you still watching?* se zobrazovaly v nastavených intervalech. Po přehrání celého seznamu videí se přehrávač vrátil k prvnímu videu seznamu a přehrávání se pozastavilo. Všechny ovládací prvky přehrávače fungovaly dle očekávání.

6.1.6 Edge, verze 89

Testování přehrávače v rámci prohlížeče Microsoft Edge proběhlo úspěšně. Přehrávač byl schopný přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i hláška *Are you still watching?* se zobrazovaly podle nastavených intervalů. Po skončení přehrávání seznamu videí se přehrávač vrátil k prvnímu videu seznamu a přehrávání bylo pozastaveno. Všechny ovládací prvky přehrávače fungovaly dle očekávání.

6.1.7 Firefox, verze 87

Testování přehrávače v rámci prohlížeče Firefox verze 87 proběhlo v úspěšně. Přehrávač byl schopný přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i hláška *Are you still watching?* se zobrazovaly podle nastavených intervalů. Po přehrání celého seznamu se přehrávač opět vrátil k prvnímu videu ze seznamu a přehrávání se pozastavilo. Všechny ovládací prvky přehrávače fungovaly dle očekávání.

6.1.8 Samsung, verze 13

Testování přehrávače v rámci prohlížeče Samsung verze 13 proběhlo úspěšně. Testování proběhlo na zařízení Samsung Galaxy S10. Přehrávač byl schopný přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i hláška *Are you still watching?* se zobrazovaly podle nastavených intervalů. Po přehrání celého seznamu se přehrávač opět vrátil k prvnímu

videu ze seznamu a přehrávání se pozastavilo. Všechny ovládací prvky přehrávače fungovaly dle očekávání. Při testování v rámci mobilního telefonu se však ukázalo, že je potřeba ještě upravit styly přehrávače pro menší obrazovky, a to především z důvodu příliš malého posuvníku videa.

6.1.9 Chrome, verze 83

Testování prohlížeče Google Chrome ve verzi 83 proběhlo úspěšně.Přehrávač byl schopný přehrát videa ve formátu MP4, WebM i videa z platformy YouTube. Reklamní spoty i hláška *Are you still watching?* se zobrazovaly podle nastavených intervalů. Po přehrání celého seznamu se přehrávač opět vrátil k prvnímu videu ze seznamu a přehrávání se pozastavilo. Všechny ovládací prvky přehrávače fungovaly dle očekávání.

6.1.10 Safari, verze 13

Testování v prohlížeči Safari verze 13.1 proběhlo téměř úspěšně. Přehrávač je schopný přehrávat videa ve formátu MP4 a také videa z platformy YouTube. Při testování se však vyskytl problém s videi formátu WebM. Tato videa nebylo v rámci prohlížeče možné přehrát. Zobrazování reklamních spotů a hlášky *Are you still watching?* proběhlo v pořádku. Stejně tak proběhlo v pořádku ukončení přehrávání a návrat k prvnímu videu po skončení přehrávání seznamu videí.

6.1.11 Závěr

V rámci testování v jednotlivých prohlížečích nebyl objeven zásadní problém, který by znemožnil funkčnost přehrávače. Díky tomuto testování je možné zhodnotit, že přehrávač funguje správně ve většině moderních prohlížečů. V rámci některých prohlížečů však byly objeveny menší chyby, které byly opraveny.

Nejzásadnější objevenou chybou bylo objevení neschopnosti přehrát videa ve formátu WebM. Pro vyřešení tohoto problému jsem se rozhodl nepodporovaná videa nepřehrávat. Ke kontrole podpory konkrétních formátů je využita funkce *supports* v rámci API přehrávače Plyr. Tato funkce, kromě vedlejších testů týkajících se zejména nesprávného předání argumentu, využívá ke kontrole funkce *HTMLMediaElement.canPlayType* [25]. Díky tomu lze předpokládat, že pokud bude podpora zmíněného formátu v rámci přehrávače Safari v budoucnu doplněna, budou se taková videa přehrávat i v rámci tohoto prohlížeče.

V rámci testování různých verzí prohlížeče Google Chrome byla otestována i funkčnost rozhraní Google Cast. Toto testování probíhalo lokálně stažením starších verzí prohlížeče Chromium, který je postavený na stejném jádru jako je prohlížeč Google Chrome. Toto jádro má v sobě již podporu funkce Google Cast a proto bylo možné ověřit, že tato funkcionalita je funkční i na všech zmíněných verzích prohlížeče Google Chrome.

Na závěr proběhlo otestování přehrávače v rámci prohlížeče Internet Explorer. V tomto prohlížeči nebylo možné přehrávač spustit. Dle zjištění je to proto, že tento prohlížeč má pouze velmi omezenou podporu JavaScriptového standardu ES6, dle kterého je právě přehrávač Plyr naprogramovaný.

Možným řešením by bylo provést transpilaci zdrojových kódů do předchozí verze standardu ES5. Z průzkumu však vyplývá, že prohlížeč Internet Explorer používá méně než 1 % uživatelů internetu. Jelikož podpora tohoto prohlížeče již byla ukončena, je v budoucnu předpokládán klesající trend počtu uživatelů tohoto prohlížeče. Rozhodl jsem se proto tento prohlížeč vynechat a nepodporovat jej.

6.2 Uživatelské testování

V této části popíši, jak probíhalo uživatelské testování a také závěry z tohoto testování.

Uživatelské testování probíhalo formou video hovorů. Touto formou bylo provedeno testování za přítomnosti 3 přispěvatelů do projektu Czech American TV USA a potenciálních budoucích přispěvatelů do projektu American Travel Show. V rámci této části jsem se zaměřil především na otestování administračního rozhraní doplňku a jednoduchost a intuitivnost ovládání tohoto rozhraní. Dále byla hodnoceno, jak je doplněk přizpůsobený konkrétním potřebám zadavatele.

Další část testování proběhla s pomocí dalších třech uživatelů. Tito uživatelé nebyli seznámeni s redakčním systémem WordPress, ani s projekty American Travel Show a Czech American TV USA. S těmito testery jsem se zaměřil opět na použitelnost administračního rozhraní doplňku. Dále pak byla zhodnocena použitelnost doplňku pro využití v rámci jiných projektů a univerzálnost doplňku.

V závěru testování s jednotlivými testery bylo také otestováno prostředí přehrávače v uživatelské části webové aplikace.

6.2.1 Průběh uživatelského testování

1. Seznámení s testerem
2. Představení projektu American Travel Show
3. Osvětlení doplňku a jeho funkce
4. Stručné představení administrační a uživatelské části doplňku
5. Přidání videa z platformy YouTube
6. Úprava popisu a názvu videa
7. Označení videa správnou kategorií
8. Nahrání videa pomocí systému WordPress
9. Úprava informací o videu
10. Nahrání videa pomocí FTP a aktualizace knihovny
11. Úprava informací o videu
12. Vytvoření seznamu videí
13. Úprava názvu a popisu seznamu videí
14. Přidání videí do seznamu
15. Změna pořadí videí v rámci seznamu videí
16. Přidání nedostupného videa z platformy YouTube
17. Změna URL adresy videa v prostředí notifikací
18. Přidání seznamu videí do článku
19. Úprava frekvence reklamních spotů a hlášky *Are you still watching?* a opětovné přehrání vytvořeného seznamu videí
20. Přehrání vytvořeného seznamu videí v rámci článku
21. Odstranění vytvořeného seznamu videí
22. Odstranění přidaných videí
23. Poděkování za účast na testování, zhodnocení výsledků testování, připomínek a návrhů na zlepšení doplňku

6.2.2 Závěr

Z uživatelského testování vyplynulo, že administrační rozhraní doplňku je dostatečně intuitivní. Testeři zvládli všechny scénáře popsané v kapitole 5.1.2. Testeři, kteří neměli zkušenosti s redakčním systémem WordPress vyžadovali více času na orientaci v prostředí tohoto redakčního systému. Těmto testerům však stačilo několik nápověd a přiblížení principů tohoto systému a poté již testeři neměli problém scénáře realizovat.

Naopak testeři, kteří měli alespoň nějaké zkušenosti s tímto redakčním systémem, neměli problém jednotlivé části administrace najít. Menší problém byl s hledáním stránky pro administraci seznamů videí, kterou někteří z testerů nejprve hledali pod položkou navigace *Media*. Když však odkaz do administrace seznamů videí nenalezli zde, nebylo třeba další nápovědy a testeři sami našli odkaz na tuto stránku ve správné části navigace.

Administrační rozhraní pro správu chybějících videí z platformy YouTube bylo testerům také srozumitelné. Bez nápovědy dokázali změnit URL adresu odstraněného videa a také takové video odstranit.

Dále bylo testováno přidávání seznamů videí do jednotlivých článků. Zde se objevil pouze jeden problém, a to přidání seznamu videí pro přehrání všech dostupných videí do článku. Tuto funkci nebyli testeři schopní odhalit bez nápovědy. Na základě tohoto zjištění byla do menu pro přidání seznamu videí do příspěvku přidána možnost *All Videos*.

Z připomínek testerů vyplynulo, že někteří testeři by uvítali více kategorií videí kromě reklamního spotu a hlavního videa. Dalším návrhem bylo v rámci hlášky *Are you still watching?* přidat možnosti Ano/Ne a při zvolení možnosti Ne přesměrovat uživatele na domovskou stránku aplikace.

Kapitola 7
Závěr

Cíle mé práce byly následující:

1. Analýza a konkretizace požadavků zadavatele a analýza stávajících řešení a jejich nedostatků
2. Analýza dostupných technologií pro implementaci
3. Implementace doplňku
4. Otestování výsledku implementace

Prvním cílem práce bylo provést detailní analýzu požadavků zadavatele na funkčnost doplňku. Tento cíl se mi v rámci několika telefonátů podařilo splnit. Na základě těchto požadavků jsem dále připravil příklady užití doplňku, které byly základním podkladem pro testování doplňku.

Dalším cílem práce bylo provést analýzu stávajících řešení a dostupných technologií. Tento cíl se mi také podařilo splnit. Podstatnou částí analýzy stávajících řešení byla analýza doplňku ATS Video Plugin. V rámci této analýzy jsem s pomocí zadavatele identifikoval nedostatky tohoto řešení, kterým jsem se při své implementaci vyvaroval. V rámci analýzy dostupných technologií jsem zvolil video přehrávač, který částečně splňoval požadavky a poskytoval dostatečné API pro rozšíření přehrávače o chybějící funkce potřebné ke splnění požadavků nepokrytých.

Dalším cílem práce byla samotná implementace doplňku. V rámci implementace vyvstalo několik problémů, které jsem byl schopný vyřešit. Především se jednalo o problém s přehráváním YouTube videí na vzdálené obrazovce pomocí Google Cast. Výchozí aplikace pro přijímací zařízení totiž nepodporuje přehrávání videí pomocí elementu *iframe*, pomocí kterého se běžně přehrávají videa z platformy YouTube v rámci internetových prohlížečů. Tento problém se podařilo vyřešit využitím interního API platformy YouTube, které umožňuje vytvoření odkazu na video ve formátu MP4.

Implementace doplňku proběhla úspěšně. Podařilo se mi splnit všechny požadavky zadavatele. Pro správu knihovny videí jsem využil interních mechanismů systému WordPress, které jsem rozšířil o další požadované funkce. Tím se mi podařilo zajistit intuitivnost rozhraní pro přispěvatele s částečnou zkušeností s redakčním systémem.

Posledním cílem bylo testování implementace. Toto testování proběhlo ve dvou fázích.

V první fázi jsem otestoval přehrávač v deseti nejvyužívanějších internetových prohlížečích. Z tohoto testování vyplynuly určité nedostatky řešení, které jsem na základě tohoto zjištění upravil. Nejzásadnějším problémem se v rámci testování ukázala být nekompatibilita prohlížeče Safari a video formátu WebM. Přehrávač jsem se na základě zjištění tohoto problému rozhodl rozšířit o ověření schopnosti prohlížeče přehrát jednotlivé video formáty. Pokud se ukáže, že přehrávač některý video formát nepodporuje, jsou videa tohoto formátu přeskočena.

Další částí testování bylo testování uživatelské. S pomocí zadavatele bylo domluveno a realizováno několik video hovorů s testery. Tato část testování proběhla úspěšně a z tohoto testování vzešlo několik návrhů na další rozšíření a zlepšení doplňku.

Příloha A

Popis funkcí tříd administrace

A.1 Třída VP_Manager

activate() Funkce volaná při aktivaci doplňku. Funkce provádí výchozí nastavení doplňku a registruje vlastní typ příspěvku.

deactivate() Funkce volaná při deaktivaci doplňku. Funkce deaktivuje jednotlivé komponenty.

init() Funkce volaná při inicializaci doplňku. V rámci této funkce jsou registrovány aktivační a deaktivační hooky a dále jsou inicializovány jednotlivé komponenty.

uninstall() Funkce volaná při odstranění doplňku. Provádí odinstalaci jednotlivých komponent a vyčištění dat doplňku, jako je například hlavní nastavení.

render_general_settings_admin_page() Funkce sloužící k vykreslení administrační stránky obsahující prvky ke změně hlavních nastavení doplňku.

render_help_admin_page() Funkce sloužící k vykreslení administrační stránky s nápovědou k administračnímu rozhraní doplňku.

save_general_settings() Funkce sloužící uložení hlavních nastavení doplňku. Po uložení nastavení je vymazána cache aplikace, pokud je tento systém využíván.

A.2 Třída VP_Cron_Scheduler

init() Funkce volaná při inicializaci doplňku. Tato funkce přidává vlastní *Cron hook*, v rámci kterého registruje vlastní úlohy.

deactivate() Funkce volaná při deaktivaci doplňku. Funkce ověří, zda jsou registrované vlastní *cron* úlohy a pokud ano, odebere je.

uninstall() Funkce volaná při odstranění doplňku. Tato funkce ve svém těle volá funkci *deactivate()*.

cron_exec() Funkce spouštějící úlohy v rámci *cron* úloh. Funkce je registrována v rámci *cron* úloh s týdenní frekvencí.

A.3 Třída VP_Menu_Creator

init() Funkce volaná při inicializaci doplňku. Registruje funkci *create_menu* k hooku, který vytváří navigaci administračního rozhraní.

deactivate() Funkce volaná při deaktivaci doplňku. V rámci této funkce se odstraňují položky navigace vytvořené funkcí *create_menu*.

uninstall() Funkce volaná při odstranění doplňku. Tato funkce ve svém těle volá funkci *deactivate()*.

create_menu() Funkce zajišťující přidání položek do hlavní navigace administračního rozhraní. Funkce přidává položky do podmenu položky *Media* a také vytváří menu doplňku a jeho podmenu.

A.4 Třída VP_Notifications

init() Funkce volaná při inicializaci doplňku. Funkce registruje REST API cestu pro přidání nedostupného videa při chybě při načítání v rámci přehrávače.

deactivate() Funkce volaná při deaktivaci doplňku.

uninstall() Funkce volaná při odstranění doplňku. Tato funkce ve svém těle volá funkci *deactivate()*.

render_admin_page() Funkce pro vykreslení stránky administračního rozhraní věnující se notifikacím. Tato stránka slouží k aktualizaci URL adresy nedostupných videí, či odstranění těchto videí.

update_missing_videos_urls() Funkce sloužící k uložení nových URL adres nedostupných videí při jejich změně v rámci administračního rozhraní věnujícího se notifikacím.

send_notification_email() Funkce sloužící k odeslání notifikačního e-mailu administrátoru aplikace. Tato funkce přebírá jeden argument, kterým je pole *attachemnt* příspěvků, které představují nově nalezená nedostupná videa.

add_missing_video() Funkce sloužící k uložení informace o nedostupnosti videa z platformy YouTube v rámci REST API. Funkce přebírá jeden argument, kterým je POST požadavek odeslaný z přehrávače. V těle tohoto požadavku je očekáván jeden parametr - *VideoId*. V tomto parametru je očekáván identifikátor nedostupného videa. Funkce dále kontroluje, zda je video opravdu nedostupné. Pokud ano, je informace uložena. Pokud ne, je pouze odeslána informace o tom, že video je dostupné.

A.5 Třída VP_Post_Type_Creator

init() Funkce volaná při inicializaci doplňku. Funkce registruje vlastní typ příspěvku a taxonomii pro označení jednotlivých videí jako reklamní spot, nebo hlavní obsah.

deactivate() Funkce volaná při deaktivaci doplňku.

uninstall() Funkce volaná při odstranění doplňku. Tato funkce ve svém těle volá funkci *deactivate()*. Dále funkce odebírá vlastní typ příspěvku, taxonomii pro označování videí jako reklamní spot, nebo hlavní obsah a termíny této taxonomie.

register_video_playlist_post_type() Funkce obsluhující registraci vlastního typu příspěvku a taxonomie pro označení videí jako reklamní spot, nebo hlavní obsah. V rámci funkce jsou definovány popisky vlastního typu příspěvku. Dále je vlastní typ příspěvku zaregistrován. Poté je zaregistrována taxonomie a na závěr jsou vloženy termíny této taxonomie.

A.6 Třída VP_Short_Code_Creator

init() Funkce volaná při inicializaci doplňku. V rámci této funkce je zaregistrována krátká značka.

deactivate() Funkce volaná při deaktivaci doplňku. Funkce odstraní vlastní krátkou značku.

uninstall() Funkce volaná při odstranění doplňku. Tato funkce ve svém těle volá funkci *deactivate()*.

create_shortcodes() Funkce vytváří vlastní krátkou značku. Tato značka má následující formát [vp_video_playlist id=”ID”].

get_playlist_by_id() Tato funkce slouží k získání seznamu videí na základě ID tohoto seznamu předaného jako argument této funkci. Pokud ID není specifikováno, je navrácen výchozí seznam videí obsahující všechna uložená videa. Pokud ID specifikováno je, funkce nalezne seznam vidí s tímto ID a vrátí tento seznam seřazený dle pořadí určeného při editaci daného seznamu.

get_spots() Funkce, která vrací pole obsahující všechna videa označená jako reklamní spoty.

create_video_playlist_shortcode_html() Tato funkce vytváří HTML kód pro nahrazení vlastní krátké značky při vykreslování stránky uživateli. Funkce lokalizuje JS soubory, čímž jsou přehrávači předána všechna potřebná data včetně videí a reklamních spotů.

A.7 Třída VP_Videos

get_all_videos_in_directory() Tato funkce slouží k nalezení všech video souborů v podporovaných formátech v rámci specifikovaného adresáře (a jeho podadresářů). Funkce přijímá dva argumenty. Prvním je adresář, který se má prohledat. Druhým argumentem je pole předané pomocí reference, ve kterém budou navrácena nalezená videa. Toto pole zároveň slouží jako akumulátor, jelikož je tato funkce rekurzivní.

update_video_library() Tato funkce slouží pro vykreslení obrazovky při ukládání videí a také po dokončení procesu ukládání. Dále tato funkce obsluhuje volání funkce pro přidání videí do knihovny médií.

add_ftp_uploaded_videos_to_library() Tato funkce slouží k uložení videí nahraných pomocí FTP do knihovny médií. Funkce nejprve vyhledá všechna videa v adresáři *wp-content/uploads* a jeho podadresářích. Dále jsou nalezená videa namapována pro získání všech potřebných informací k uložení do knihovny médií. Poté jsou nalezená videa po jednom procházena a pokud je nalezeno video, které ještě v knihovně médií není, je takové video přidáno.

A.8 Třída VP_Youtube_Videos_Manager

init() Funkce volaná při inicializaci doplňku. V rámci této funkce je zaregistrována REST API cesta pro získání URL adresy na video ve formátu MP4.

deactivate() Funkce volaná při deaktivaci doplňku. Funkce odstraní vlastní krátkou značku.

uninstall() Funkce volaná při odstranění doplňku. Tato funkce ve svém těle volá funkci *deactivate()*.

api_get_youtube_video_info() Funkce použitá pro registraci REST API cesty pro získání URL odkazu na YouTube video ve formátu MP4. Funkce poskytuje rozhraní nad funkcí *get_youtube_video_info*.

get_youtube_video_info() Funkce sloužící k získání informací o videu z platformy YouTube. Funkce využívá interního API YouTube *get_video_info*. Tato funkce vrací asociativní pole obsahující informace o názvu, popisu, statusu videa a také URL adresu odkazující na video ve formátu MP4.

get_missing_videos() Funkce vracející chybějící YouTube videa. Tato funkce přijímá jeden argument, určující zda se má zkontrolovat stav videí před navrácením.

render_add_youtube_video_page() Funkce pro vykreslení administrační stránky pro přidání YouTube videa.

add_youtube_video() Funkce pro uložení YouTube videa po přidání pomocí administrační stránky z předchozí funkce.

check_all_youtube_videos() Funkce sloužící k ověření stavu všech YouTube videí a odeslání e-mailové notifikace při nalezení nových nedostupných videí.

update_youtube_video_url() Funkce sloužící k uložení změny URL adresy YouTube videa.

A.9 Třída VP_Add_Playlist_To_Post_Meta_Box

init() Funkce volaná při inicializaci doplňku. V rámci této funkce jsou volány funkce pro přidání meta boxu a uložení metadat z tohoto meta boxu.

create_custom_fields() Funkce zajištující registraci meta boxu.

display_custom_fields() Funkce zajištující vykreslení obsahu meta boxu.

save_custom_fields() Funkce zajišťující uložení metadat získaných z meta boxu.

A.10 Třída VP_Edit_Youtube_Video_Url_Meta_Box

init() Funkce volaná při inicializaci doplňku. V rámci této funkce jsou volány funkce pro přidání meta boxu a uložení metadat z tohoto meta boxu.

create_custom_fields() Funkce zajištující registraci meta boxu.

display_custom_fields() Funkce zajištující vykreslení obsahu meta boxu.

save_custom_fields() Funkce zajišťující uložení metadat získaných z meta boxu.

A.11 VP_Playlist_Videos_Meta_Box

init() Funkce volaná při inicializaci doplňku. V rámci této funkce jsou volány funkce pro přidání meta boxu a uložení metadat z tohoto meta boxu.

create_custom_fields() Funkce zajištující registraci meta boxu.

display_custom_fields() Funkce zajištující vykreslení obsahu meta boxu.

save_custom_fields() Funkce zajišťující uložení metadat získaných z meta boxu.

Příloha B

Popis funkcí tříd Přehrávače

B.1 Třída PlayerManager

constructor() Konstruktor třídy. Zajišťuje připravení HTML elementů pro přehrávač Plyr a nastavení třídních proměnných instance.

handleError() Funkce volaná pří výskytu chyby při přehrávání nebo načítání videa. Spouští další video a pokud se jedná o YouTube video, je zavoláno REST API rozhraní pro uložení informace o nedostupnosti videa.

areYouStillWatching() Funkce zobrazující hlášku *Are you still watching?*.

stillWatchingDissmiss() Funkce skrývající hlášku *Are you still watching?*.

nextVideo() Funkce pro načtení dalšího videa. Funkce přebírá dva argumenty. Prvním je událost, pokud je funkce zavolána při nějaké události. Druhým je parametr určující, zda se má nové video po načtení začít přehrávat. Tato funkce rozhoduje, zda bude přehráno hlavní video, nebo reklamní spot a také zda bude zobrazena hláška *Are you still watching?*. Funkce také zobrazuje popis aktuálně načteného videa.

initPlayer() Funkce sloužící k inicializaci přehrávače Plyr.

getVideo() Tato funkce vrátí zdroj videa ve formátu vyžadovaném přehrávačem Plyr. Argumentem je video objekt získaný z redakčního systému WordPress.

getPlayerControls() Tato funkce vrací HTML kód, který je vykreslen na místě určeném pro ovládací prvky přehrávače Plyr.

showVideoDescritpion() Tato funkce slouží pro vykreslení popisu aktuálně přehrávaného videa.

B.2 Třída CastManager

constructor() Konstruktor Třídy. V rámci konstruktoru jsou inicializovány proměnné instance a navázány tzv. *listenery*

initializeCastApi() Tato třída slouží k inicializaci rozhraní Google Cast SDK. V rámci této funkce je vytvořena instance třídy CastManager, pokud je Google Cast dostupné, nebo instance třídy PlayerManager, pokud Google Cast dostupné není.

addListeners() V rámci této třídy jsou navázány funkce pro načtení video obsahunavzdálenou obrazovku na událost ivzdáleného přehrávače.

bindArrowPress() Funkce navazující funkci pro posunutí času přehrávaného obsahu při stisku šipky vlevo, nebo vpravo na klávesnici.

playPauseRemote() Tato funkce slouží ke spuštění přehrávání obsahu na vzdálené obrazovce, pokud je toto přehrávání pozastaveno, a nebo k pozastavení přehrávání obsahu na vzdálené obrazovce, pokud je přehrávání spuštěno.

playRemote() Tato funkce slouží ke spuštění přehrávání obsahu na vzdálené obrazovce.

pauseRemote() Tato funkce slouží k pozastavení přehrávání obsahu na vzdálené obrazovce.

showRemoteControls() Tato funkce slouží k zobrazení ovládacích prvků specifických pro vzdálené přehrávání.

hideRemoteControls() Tato funkce slouží ke skrytí ovládacích prvků specifických pro vzdálené přehrávání.

jump() Tato funkce slouží k posunutí času přehrávaného obsahu na vzdálené obrazovce. Funkce přijímá jeden argument, kterým je počet sekund, o které se má obsah posunout. Tento argument může být i záporný.

updateLocalProgress() Tato funkce slouží k aktualizaci stavu posuvníku v rámci lokálního přehrávače při změně v rámci přehrávače vzdáleného.

updateRemoteProgress() Tato funkce slouží k aktualizaci stavu posuvníku v rámci vzdáleného přehrávače při změně v rámci přehrávače lokálního.

bindProgressBar() Tato funkce slouží k navázání lokálního posuvníku postupu videa na událost změny času přehrávání ve vzdáleném přehrávači.

bindJumpButtons() Tato funkce slouží k navázání ovládacích prvků pro posun videa.

unbindJumpButtons() Tato funkce slouží ke zrušení navázání ovládacích prvků pro posun videa.

unbindProgressBar() Tato funkce slouží ke zrušení navázání lokálního posuvníku postupu videa na událost změny času přehrávání ve vzdáleném přehrávači.

switchScreen() Tato funkce slouží ke změně obrazovky (lokální/vzdálená). V rámci této funkce je načteno video na vzdálenou obrazovku a navázány ovládací všechny prvky lokálního přehrávače, pokud se jedná o změnu z lokální obrazovky na vzdálenou. V opačném případě jsou zvláštní ovládací prvky skryty a je obnoveno přehrávání v rámci lokálního přehrávače.

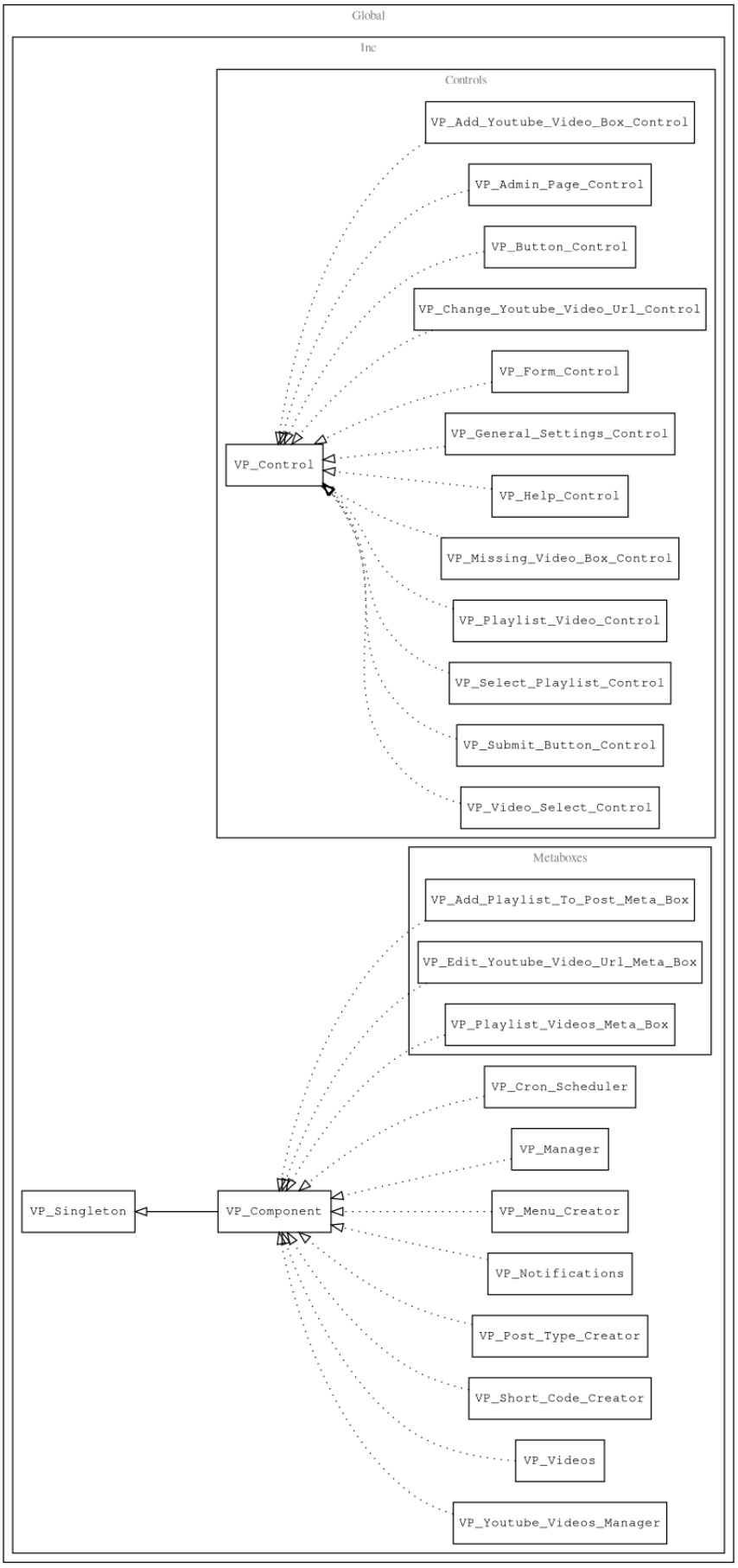
loadMedia() Tato funkce slouží k načtení video obsahu na vzdálenou obrazovku.

getYoutubeUrl() Tato funkce slouží k získání URL adresy odkazující na video ve formátu MP4, pokud načítané video je z platformy YouTube.

loadNewMedia() Tato funkce poskytuje rozhraní nad funkcí *loadMedia*.

Příloha C

Diagram tříd



Obrázek C.1 Diagram tříd

Bibliografie

1. LONGTAIL AD SOLUTIONS, Inc. JW player pricing, *JW Player* [online]. New York, ©2007-2021 Longtail Ad Solutions, Inc. [Cit. 2021-04-07]. Dostupné z: `https://www.jwplayer.com/pricing/`.
2. TREMBLAY, Benoit. YouTube Playback Technology for Video.js, *GitHub* [online]. 2013, Aktualizováno: 2020-01-16 [cit. 2021-04-07]. Dostupné z: `https://github.com/videojs/videojs-youtube`.
3. DEVERIA, Alexis. Can I use ECMAScript 6, *Can I use* [online]. Kalifornie, 2009, Aktualizováno: Březen 2021 [cit. 2021-04-07]. Dostupné z: `https://caniuse.com/es6`.
4. DEVERIA, Alexis. Can I use Video element, *Can I use* [online]. Kalifornie, 2009, Aktualizováno: Březen 2021 [cit. 2021-04-07]. Dostupné z: `https://caniuse.com/video`.
5. DEVERIA, Alexis. Can I use HTML element: iframe, *Can I use* [online]. Kalifornie, 2009, Aktualizováno: Březen 2021 [cit. 2021-04-07]. Dostupné z: `https://caniuse.com/mdn-html_elements_iframe`.
6. BRAZELL, Aaron; JAQUITH, Mark. *Wordpress bible*. 1. vyd. Indianapolis, Indiana: John Wiley & Sons, 2010. ISBN 978-0-470-56813-2.
7. WORDPRESS. *History* [online]. 2018, Aktualizováno: 2021-03-09 [cit. 2021-04-15]. Dostupné z: `https://wordpress.org/about/history/`.
8. WILLIAMS, Brad; DAMSTRA, David; STERN, Hal. *Professional WordPress: Design and Development*. 3. vyd. Indianapolis, Indiana: John Wiley & Sons, 2015. ISBN 978-1-118-98724-7. Dostupné také z: `https://ebookcentral.proquest.com/lib/cvut/detail.action?docID=1895867`.
9. WORDPRESS. *Code Reference: add_meta_box* [online]. [N.d.] [cit. 2021-04-30]. Dostupné z: `https://developer.wordpress.org/reference/functions/add_meta_box/`.
10. WORDPRESS. *Code Reference: add_action* [online]. [N.d.] [cit. 2021-04-30]. Dostupné z: `https://developer.wordpress.org/reference/functions/add_action/`.
11. WORDPRESS. *Code Reference: wp_update_post* [online]. [N.d.] [cit. 2021-04-30]. Dostupné z: `https://developer.wordpress.org/reference/functions/wp_update_post/`.
12. WORDPRESS. *Post types* [online]. 2018, Aktualizováno: 2019-12-06 [cit. 2021-04-15]. Dostupné z: `https://wordpress.org/support/article/post-types/`.
13. WORDPRESS. *Code Reference: WP_Query* [online]. [N.d.] [cit. 2021-04-20]. Dostupné z: `https://developer.wordpress.org/reference/classes/wp_query/`.
14. WORDPRESS. *Theme Handbook: Data Sanitization/Escaping* [online]. [N.d.] [cit. 2021-05-05]. Dostupné z: `https://developer.wordpress.org/themes/theme-security/data-sanitization-escaping/`.

15. MCFARLIN, Tom. *WordPress Coding Standards: Naming Conventions & Function Arguments* [online]. [N.d.] [cit. 2021-05-08]. Dostupné z: https://code.tutsplus.com/articles/wordpress-coding-standards-naming-conventions-function-arguments--wp-31683.
16. WORDPRESS. *Code Reference: add_media_page* [online]. [N.d.] [cit. 2021-04-15]. Dostupné z: https://developer.wordpress.org/reference/functions/add_media_page/.
17. GROUP, PHP. glob, *PHP Manual* [online]. ©2001-2021 The PHP Group [cit. 2021-04-30]. Dostupné z: https://www.php.net/manual/en/function.glob.php.
18. WORDPRESS. *Code Reference: attachment_url_to_postid* [online]. [N.d.] [cit. 2021-04-25]. Dostupné z: https://developer.wordpress.org/reference/functions/attachment_url_to_postid/.
19. WORDPRESS. *Code Reference: wp_insert_attachment* [online]. [N.d.] [cit. 2021-04-25]. Dostupné z: https://developer.wordpress.org/reference/functions/wp_insert_attachment/.
20. WORDPRESS. *Code Reference: wp_generate_attachment_metadata* [online]. [N.d.] [cit. 2021-04-25]. Dostupné z: https://developer.wordpress.org/reference/functions/wp_generate_attachment_metadata/.
21. WORDPRESS. *Code Reference: wp_update_attachment_metadata* [online]. [N.d.] [cit. 2021-04-25]. Dostupné z: https://developer.wordpress.org/reference/functions/wp_update_attachment_metadata/.
22. WORDPRESS. *Code Reference: register_taxonomy* [online]. [N.d.] [cit. 2021-04-30]. Dostupné z: https://developer.wordpress.org/reference/functions/register_taxonomy/.
23. WORDPRESS. *Code Reference: register_post_type* [online]. [N.d.] [cit. 2021-04-30]. Dostupné z: https://developer.wordpress.org/reference/functions/register_post_type/.
24. LLC, Awio Web Services. *W3Counter: Global Web Stats* [online]. ©2004-2021 Awio Web Services LLC [cit. 2021-04-30]. Dostupné z: https://www.w3counter.com/globalstats.php?year=2021&month=4.
25. MOZILLA; CONTRIBUTORS, individual. *MDN Web Docs: HTMLMediaElement.canPlayType()* [online]. ©2005-2021 Mozilla and individual contributors [cit. 2021-05-01]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/HTMLMediaElement/canPlayType.

Obsah přiloženého média

- `src`
  - `video-playlists-manager` ..... zdrojové kódy implementace
    - `admin` ..... JS a CSS soubory administrační části doplňku
    - `includes` ..... PHP soubory administrační části doplňku
    - `public` ..... JS a CSS soubory uživatelské části doplňku
    - `video-playlists-manager.php` ..... Hlavní soubor doplňku
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF