Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Wordpress plugin pro snadné přehrávání videa na webu

Plzeň 2020 Jakub Váverka

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 21. května 2020

Jakub Váverka

Abstract

This thesis focuses on the creation of a plugin for a content management system called Wordpress. It is going to be a video player plugin, that can be fully customized and allows user to make video playlists. Goal of this plugin is the simulation of television broadcast. Part of this work also focuses on exploring available options for video playback on the web and discusses the results of the final plugin testing.

Abstrakt

Tato práce se zabývá vytvořením pluginu pro redakční systém Wordpress. Půjde o přehrávač videa, který bude plně konfigurovatelný a umožní provádět skládání videí dle požadavků uživatele. Cílem je simulace vysílání televizního pořadu, kdy se před video vkládá úvodní znělka a za video zase upoutávky na další videa. Součástí práce je také průzkum problematiky přehrávání videa na webu a diskuze ohledně testování pluginu.

Obsah

1 Úvod **7**

2 Použité technologie **8**
2.1 Javascript 8
2.1.1 ECMAScript 9
2.2 Plyr 11
2.3 Wordpress 15
2.4 Sass 19

3 Analýza **20**
3.1 Flash 20
3.1.1 Boj společnosti Apple proti přehrávači Flash Player . 20
3.2 Požadavky na výkonnost nástroje Wordpress 21
3.3 Funkční požadavky vytvářeného pluginu 22
3.4 Komponenty pro přehrávání videa 23

4 Návrh **25**
4.1 Vytvoření videa 25
4.1.1 Wordpress databáze 25
4.1.2 Výběr tabulky 28
4.2 Spárování s příspěvkem 28
4.3 Zobrazení videa 29

5 Implementace **30**
5.1 Obalový projekt 30
5.1.1 Gutenberg 30
5.1.2 Video editor 32
5.1.3 Zástupné kódy 37
5.2 Přehrávač 41
5.3 Souborová struktura 42
5.4 Balíčkovací systém 46

6 Testování **48**
6.1 Jednotkové testy 48
6.2 Testování funkčnosti 49
6.3 Testování použitelnosti 50

6.4 Testování rozhraní 51
6.5 Testování výkonu 51
6.6 Testování kompatibility 52
6.7 Vzdálené ukládání chybových hlášení 53

7 Nasazení **54**
7.1 Web hosting 54
7.2 Migrace dat 55

8 Závěr **62**

Literatura **63**

A Schéma **66**

B Uživatelská příručka **67**

C Programátorská příručka **73**
C.1 Vnitřní projekt 74

1 Úvod

Cílem této práce je vytvoření pluginu v redakčním systému Wordpress. Hlavním popudem pro tento projekt se stal zájem ze strany nezávislé vysílací stanice CzechAmericanTV. Tato stanice vysílá své pořady jak v televizi, tak online na svých webových stránkách. Díky dřívějším školním projektům jsem přišel do styku s redakčním systémem, který tato stránka používá pro publikování článků a videí. V minulosti byly webové stránky této televizní stanice celé naprogramovány v jazyce PHP, následně přešly na redakční systém Wordpress. Některé části této stránky stále nedodržují programovací konvence systému Wordpress. Úzké spojení funkčnosti a vzhledu snižuje modularitu a budoucí vývoj. Jednou z těchto úzce spojených částí je přehrávač videí. Stránka CzechAmericanTV plánuje v nejbližší době přejít na novou vzhledovou šablonu a to znamená opuštění funkcí, které jsou takto úzce spjaté. Přestože je tedy hlavním cílem této práce vytvoření obecného pluginu, je třeba ho přizpůsobit primárnímu uživateli a tím bude právě CzechAmericanTV.

V kapitole Použité technologie se práce věnuje vysvětlení všech nástrojů, které bude třeba pro konstrukci pluginu využít. Čtenář je v ní seznámen s kontextem, který tyto technologie mají v oboru softwarového inženýrství. Následující kapitola se pak věnuje analýze požadavků, které vznikají při vývoji Wordpress pluginů. Také je zde rozebrána problematika přehrávačů fungujících na technologii Flash a historické důvody pro přechod na novou technologii. Práce se následně věnuje navržení způsobu, kterým bude možné vytvořit, uložit a přehrát video playlist. K tomu je třeba využít databázový model a architekturu systému Wordpress. V kapitole Implementace jsou pak zužitkovány všechny informace z předešlých kapitol pro vytvoření zmíněného pluginu. Je zde detailně popsán způsob, kterým jsou realizovány funkční části z kapitoly Návrh. Část této kapitoly se také věnuje uložení naprogramovaného přehrávače do balíčkovacího systému NPM. Kapitola Testování se pak věnuje oblastem ve kterých byl výsledný plugin testován a jejich výsledkům. V práci je dále popsán proces nasazení pluginu na servery CzechAmericanTV. Výsledky, klady a zápory jsou pak zhodnoceny v poslední kapitole s názvem Závěr.

2 Použité technologie

Redakční systém Wordpress je naprogramován v jazyce PHP a tak i vyvíjené pluginy vždy obsahují alespoň z malé části tento programovací jazyk. PHP je skriptovací jazyk, který provádí své operace na straně serveru. Pokud má plugin obsahovat i funkce vykonávané ve webovém prohlížeči na straně klienta je třeba využít nějaký sekundární programovací jazyk, jako je například JavaScript. Tato kapitola se věnuje technologiím, které byly použity při vývoji výsledného pluginu.

2.1 Javascript

JavaScript patří mezi nejdůležitější programovací jazyky dnešní doby. Obrovský nárůst popularity internetu z JavaScriptu udělal všestranný všudypřítomný nástroj. Počátky programovacího jazyka JavaScript se odehrály v roce 1995. Společnost Netscape byla v té době jedním z největších hráčů na poli nově se rozmáhající technologie internetu. Netscape svoji pozici získal díky tomu, že vytvořil velice kvalitní webový prohlížeč, který díky malé konkurenci rychle vzrostl na popularitě [22].
Společnost Netscape byla založena lidmi, kteří vytvořili první první široce používaný prohlížeč Mosaic. Po svém prvním úspěchu s tímto prohlížečem, tito lidé hledali způsob jak dále rozšířit a obohatit uživatelský požitek z procházení webu. Vize vývojového týmu společnosti Netscape byla taková, že současný statický web musí přejít do více dynamické formy. Různé animace, uživatelské interakce a malé automatizace by se měly stát budoucností webových stránek. Internet potřeboval jednoduchý malý skriptovací jazyk, který bude interagovat se stromovou strukturou tvořící webové stránky. Zároveň bylo důležité, aby tento jazyk nebyl orientovaný pouze na zkušené vývojáře, protože web teprve začínal a byl dostatečně jednoduchý na to, aby ho mohli používat a programovat i nezkušení uživatelé.
Zrodil se tedy nápad vytvořit programovací jazyk Mocha. V té samé době Netscape spolupracoval se společností Sun na přivedení jejich programovacího jazyka Java na web. Přestože by si jazyky mohli konkurovat, prvotní myšlenka byla taková, že Java bude určena pro podnikatele a firmy a Mocha bude spíše určena pro amatéry a neprogramátory. V podstatě by mohl být vztah programovacích jazyků Mocha a Java podobný vztahu programovacích jazyků C a Visual Basic.

Tento vztah k programovacímu jazyku vytvořeným společností Sun zapříčinil to, že Mocha adoptuje podobnou syntaxi a sémantiku. Počáteční snaha přiblížit programovací jazyk Mocha jazykům jako je Lisp nebo Scheme nakonec ustoupila v zájmu napodobení programovacího jazyka Java.
Tento nový programovací jazyk byl přejmenován na LiveScript a zapracován do webového prohlížeče Netscape Communicator [22]. Později došlo k uzavření dohody s firmou Sun o přejmenování na dnes již světové rozšířený název JavaScript.
První verze JavaScriptu definovala spoustu znaků, kterými je tento jazyk dodnes velmi dobře známý. Jedním z nich je třeba objektový model a funkce s ním spojené. Tento programovací přístup byl přítomen už v první verzi.
Po vydání JavaScriptu vstoupila do hry společnost Microsoft. Držela na technologickém trhu monopol a ten se jí povedlo držet i díky její strategii ovládání nových konkurentů. Microsoft díky svým zdrojům byl schopný vstoupit do nově se otevírajících potenciálních trhů, osvojil si otevřenou technologii, vytvořil vlastní verzi a následně do ní ale nad rámec přidal některé rozšiřující funkce. Lidé poté začali využívat rozšířenou verzi, vytvořenou Microsoftem. Tímto způsobem si firma Microsoft přivlastnila dříve otevřenou technologii a zabránila tak vzniku jakékoliv konkurence.
S programovacím jazykem JavaScript to dopadlo podobně. Firma Microsoft vytvořila vlastní verzi, kterou pojmenovala JScript. Tato implementace se však od původního jazyka JavaScript lyšila v několika funkcích a skripty nebyly zcela kompatibilní. Vydáním webového prohlížeče Internet Explorer, který byl výchozím prohlížečem operačního systému Windows si firma Microsoft zajistila uživatelskou většinu.
Následující války o nadvládu webu skončili soudními spory, které vyhrála firma Sun. Microsoft musel ustoupit a v dnešní době je JavaScript nejrozšířenějším skriptovacím jazykem pro web. První velká změna pro JavaScript po jeho veřejném publikování přišla ve formě ECMA standardu [16].

2.1.1 ECMAScript

Standardizace byla pro takto mladý jazyk velice důležitý krok. Už i kvůli děsivé hrozbě fragmentace, ať už ze strany Microsoftu nebo jiných firem. Z důvodu ochranné známky nemohl standard použít název JavaScript a tak byl definovaný jazyk pojmenován ECMAScript. V dnešní době je JavaScript pouze obchodní název programovacího jazyka ECMAScript [22].
První standard jazyka ECMAScript byl postaven na verzi jazyka JavaScript, která byla vydána s prohlížečem Netscape Navigator 4 a stále jí scházela spousta důležitých funkcí. Postrádala například regulární výrazy, JSON, vý-

jimky a důležité funkce pro práci s objekty.
Následovalo vydání standardu ECMAScript 2. Hlavním úkolem tohoto vydání bylo vyřešit nesrovnalosti mezi standardem ECMA a ISO, takže nedošlo k velkým syntaktickým změnám. Velké změny přišly až s verzí ECMAScript 3. Ta přinesla funkce jako regulární výrazy, do-while cykly, výjimky, vestavěné funkce pro práci s řetězci a operátory in a instanceof. Tato verze byla vydána v roce 1999 a rychle se rozšířila po celém webu. V následujícím vydání prohlížeče Internet Explorer 5, firma Microsoft obohatila tento standard o funkce, které umožňovaly odesílání asynchronních HTTP dotazů na server. To umožňovalo stránkám dynamicky načítat a měnit obsah bez potřeby obnovování webové stránky. Této technologii se později začalo říkat AJAX a stala se standardem dalším verzí standardu ECMAScript. Následující roky byly však pro programovací jazyk JavaScript poměrně obtížné. Vývojová komunita se rozdělila na dva tábory. První tábor zastával názor, že JavaScript potřebuje nové vlastnosti, které by z něj udělaly mnohem silnější jazyk, určený pro vývoj rozsáhlých aplikací. Druhá skupina zase zastávala názor, že toto není směr, kterým by se měl JavaScript udat. Začal vznikat standard ECMAScript 4. Jeho rozsah však rychle nabíral na velikosti. Dlouhá doba vývoje a nevole ze strany vývojářů webových prohlížečů zapříčinila to, že byl celý tento projekt smeten ze stolu. Místo něj vyšel zmenšený standard ECMAScript 3.1, který obsahoval jen ty funkce na kterých se všichni členové vývojové komise shodli.
Koncept standardu ECMAScript 4 vedl k vytvoření programovacího jazyka ActionScript. Firma Adobe ho implementovala jako součást jejich frameworku Flash. A dodnes je to primární vývojový jazyk pro Flash aplikace. Firma Adobe doufala, že vývojem programovacího jazyka ActionScript urychlí a nasměruje budoucí vývoj standardu ECMAScript 4, k tomu však nedošlo a ECMAScript se v budoucích vydáních od jazyka ActionScript spíše oddálil. S příchodem HTML5 pak Flash úplně vymizel. Po neúspěchu ECMAScript 4 se vývojáři zaměřili na ECMAScript 3.1, který byl následně přejmenován na ECMAScript 5, aby nedocházelo k nedorozuměním. ECMAScript 5 se stal jedním z nejvíce podporovaných verzí programovacího jazyka JavaScript. ECMAScript 5 byl v plné míře podporován prohlížeči Firefox 4, Chrome 19, Safari 6, Opera 12.10 a Internet Explorer 10.
Asi nejdůležitější verzí standardu je ECMAScript 6, který vyšel v roce 2015. Spousta funkcí, které byly původně naplánovány pro zrušený ECMAScript 4, byly odsunuty a předělány právě pro toto vydání. Skoro každá změna, která vyžadovala změnu jazykové syntaxe byla naplánována až pro tuto verzi. Tentokrát se však komunita shodla a už během vydání, pracovalo mnoho vývojářů webových prohlížečů na implementaci nových změn. Změn bylo však

opravdu hodně a některé prohlížeče je dodnes nepodporují [14].
Vydání standardu ECMAScript 6 také zapříčinilo obrovský nárůst používání transpilátorů. To je překladač, který překládá zdrojový kód z jednoho jazyka do druhého. Umožňuje tedy přeložit kód určený pro prohlížeče podporující ECMAScript 6 na kód, který splňuje pouze standard ECMAScript 3. Mezi tyto transpilátory patří například Babel nebo Traceur.

2.2 Plyr

Plyr je jednoduchý, modifikovatelný a velmi dostupný HTML5 video přehrávač. Podporuje jak přehrávání souborů umístěných na internetu, tak videí nahraných na serverech jako je YouTube a Vimeo.
Plyr rozšiřuje standardní HTML5 elementy médií <audio> a <video>. Pro přehrávání videí ze serverů YouTube a Vimeo se používá <iframe>. Jako HTML5 přehrávač Plyr přímo nezpracovává různé typy a formáty přehrávaných médií. Tuto problematiku přenechává konkrétnímu webovému prohlížeči. Plyr lze do stránky vložit jako klasický JavaScriptový skript, ale je možného použít i jako npm modul. Poté stačí iniciovat konstruktor a předat mu identifikátor HTML elementu, který má přehrávat videa. Tento způsob je vidět v ukázce kódu 2.1

```
const player = new Plyr('#player');
```

Kód 2.1: Inicializace přehrávače

Plyr umožňuje i iniciování hned několika přehrávačů na jedné stránce. K tomu je třeba využít funkci querySelectorAll(). Inicializace přes třídu elementu je ukázána v ukázce 2.2.

```
const players = Array.from(document.querySelectorAll('.js-player')).map(p
    => new Plyr(p));
```

Kód 2.2: Hromadná inicializace přehrávače

Během inicializace a nebo klidně v průběhu přehrávání je možné měnit vlastnosti přehrávače. Tyto vlastnosti se předávají jako klasický JavaScriptový objekt v konstruktoru. Způsob, kterým je možné nastavit nadpis je definován v ukázce č.2.3.

```
const player = new Plyr('#player', {
    title: 'Example Title',
});
```

Kód 2.3: Nastavení nadpisu videa

Nebo jako JSON pole v html atributu data-plyr-config v ukázce 2.4.

```
<video src="/path/to/video.mp4" id="player" controls data-plyr-config='{
    "title": "Example Title" }'></video>
```

Kód 2.4: Alternativní nastavení nadpisu videa

Veškeré možnosti nastavení se pokusím shrnout v následujících odstavcích. Nastavením položky enabled lze kompletně vypnout funkce knihovny Plyr. Nabývá hodnot true a false a defaultně je nastavena na true. Pokud bychom iniciovali přehrávač způsobem v ukázce 2.5, nedošlo by k jeho načtení okamžitě, ale až ve chvíli kdy bychom v nějakém skriptu tuto hodnotu nastavili zpět na true.

```
const player = new Plyr('#player', { enabled: false });
```

Kód 2.5: Skrytí přehrávače

Další užitečné nastavení je hodnota debug. Funguje podobně jako nastavení enabled. Po jeho zapnutí se do konzole v prohlížeči vypysují všechny stavy a události, kterými přehrávač prochází. Pro zapnutí debug módu je třeba vytvořit instanci přehrávače způsobem z ukázky 2.6.

```
const player = new Plyr('#player', { debug: true });
```

Kód 2.6: Režim ladění

Následující nastavení vyžaduje jako vstup pole. Jedná se o nastavení ovládacích prvků. Pokud toto pole není uvedeno tak jsou v základu zapnuty skoro všechny ovládací prvky. Výchozí hodnota je ekvivalentní nastavení z ukázky 2.7. Odstraněním určitých prvků z pole je možné konkrétní prvky schovat.

```
const player = new Plyr('#player', { controls: ['play-large', 'play', '
    progress', 'current-time', 'mute', 'volume', 'captions', 'settings',
    'pip', 'airplay', 'fullscreen'] });
```

Kód 2.7: Nastavení ovládacích prvků

Pokud je do pole vložena funkce, tak je očekáváno, že její návratová hodnota bude buď element a nebo textový řetězec v HTML, který tento element reprezentuje. Do této funkce jsou pak předány 3 hodnoty. Unikátní identifikátor, čas současného stavu přehrávání a nadpis videa. S tímto nastavením souvidí nastavení setting z ukázky 2.8.

```
const player = new Plyr('#player', { setting: ['captions', 'quality', '
    speed', 'loop'] });
```

Kód 2.8: Prvky nastavení přehrávače

Pokud jsou totiž nastaveny základní ovládací prvky, tak lze tímto nastavením specifikovat, které položky má zobrazovat menu nastavení.

Možnost autoplay je ve výchozím stavu nastavena na false. Nastavením na hodnotu true je možné vynutit spuštění videa hned po načtení stránky. Toto je však považováno za špatnou praktiku. Většině uživatelů se nelíbí, když se při otevření nového okna se bez interakce automaticky pustí přehrávání videa nebo zvuku. Z toho důvodu je tato funkce na většině moderních prohlížečů zablokována.
Další funkce s podobným názvem je autopause. Ta zajišťuje, že bude přehrávání probíhat vždy jen z jednoho přehrávače. Při spuštění nového videa se momentálně spuštěné video pozastaví. Tato funkce je však v současné verzi přehrávače Plyr implementována jen pro videa ze serveru Vimeo.

```
const player = new Plyr('#player', { seekTime: 20 });
```

Kód 2.9: Rychlost přetáčení videa

Upravením hodnoty seekTime v ukázce 2.9 je možné modifikovat čas v sekundách, o který se posune časová osa při stisknutí tlačítka pro přetočení videa.

```
const player = new Plyr('#player', { volume: 0.8 });
```

Kód 2.10: Výchozí hlasitost

Hodnota volume definuje hodnotu na kterou se nastaví hlasitost po vytvoření přehrávače. Tato hodnota se musí nacházet mezi 0 a 1. Podobným způsobem, jako v ukázce 2.10 lze nastavit hodnotu muted na true, aby došlo k automatickému ztišení.
Pokud chceme uživatelům usnadnit tažení přehrávaného videa, lze přepnout hodnotu disableContextMenu na false. Ve výchozím nastavení je totiž blokováno vyskakovací okno, které se zobrazuje při stisknutí pravého tlačítka na přehrávači.
Při nečinnosti myši se v přehrávači po 2 vteřinách automaticky schovají ovládací prvky. Při přejetí myši, zastavení přehrávání nebo zvýraznění elementu dojde zpětně k instantnímu zobrazení. Pokud bychom chtěli toto chování vypnout, lze nastavit hodnotu hideControls na false.

```
const player = new Plyr('#player', { keyboard: { focused: true, global:
    false } });
```

Kód 2.11: Ovládání pomocí klávesnice

Nastavení keyboard z ukázky 2.11 upřesňuje zda má přehrávač reagovat na stisky klávesových zkratek, buď globálně nebo jen při označení přehrávače.
Podobným nastavením lze zapnout nebo vypnout funkci zobrazování kontextové nápovědy. Při podržení kurzoru myši nad jednotlivým nastavením je

možné zobrazit název vybraného zařízení a při podržení kurzoru nad časovou osou, dojde k zobrazení momentálního času.

```
const player = new Plyr('#player', { tooltips:  { controls: false, seek:
    true } });
```

Kód 2.12: Zobrazení nápovědy

Nastavením invertTime lze definovat zda se má časová osa inkrementovat od nuly a nebo naopak dekrementovat od délky videa.

```
const player = new Plyr('#player', { fullscreen: { enabled: true,
    fallback: true, iosNative: false, container: null } });
```

Kód 2.13: Režim celé obrazovky

Nastavením v ukázce 2.13 je možné ovlivnit povolení režimu celé obrazovky. Fallback umožňuje použití alternativního způsobu zobrazení na plnou obrazovku. iosNative určuje, zda se má použít původní přehrávač na mobilních zařízeních od společnosti Apple. Tento režim však neumožňuje využití vlastních ovládacích prvků.
Po inicializaci je možné na objektu přehrávače volat různé metody nebo zjišťovat a nastavovat proměnné. Objekt lze získat jako návratovou hodnotu konstruktoru a nebo jeho instanci lze získat z událostí, které přehrávač vyvolává. Tento příklad je uveden v ukázce 2.14.

```
element.addEventListener('ready', event => {
    const player = event.detail.plyr;
});
```

Kód 2.14: Získání instance přehrávače

Ze všech proměnných, které je možné přehrávači nastavit je nejdůležitější proměnná source, která definuje zdroj přehrávaného videa. Korektně nastavený zdroj pro video ve více rozlišeních, s počátečním snímkem poster.jpg a 2 soubory titulků je deklarován v ukázce 2.15.

```
player.source = {
    type: 'video',
    title: 'Example title',
    sources: [
        {src: '/path/to/movie.mp4', type: 'video/mp4', size: 720},
        {src: '/path/to/movie.webm', type: 'video/webm', size: 1080}
    ],
    poster: '/path/to/poster.jpg',
    previewThumbnails: {
        src: '/path/to/thumbnails.vtt'
    },
    tracks: [
```

```
        {kind: 'captions', label: 'English', srclang: 'en', src: '/path/
    to/captions.en.vtt', default: true},
        {kind: 'captions', label: 'French', srclang: 'fr',src: '/path/to/
    captions.fr.vtt'}
    ],
};
```

Kód 2.15: Ukázka vícenásobných zdrojů

V průběhu používání přehrávače dochází k vysílání různých událostí, na které může uživatel reagovat svým kódem. Tyto události jsou například spuštění, zastavení, změna hlasitosti, ukončení videa, změna velikosti nebo dokončení inicializace.
Pokud uživatel kliknul myší do elementu přehrávače, je možné pro jeho ovládání používat i klávesové zkratky. Klávesa F pro zvětšení videa na celou obrazovku, M pro vypnutí zvuku, mezerník pro zastavení a spuštění a šipky pro posouvání po časové ose.
Přehrávač Plyr je podporován všemi moderními prohlížeči. Je však naprogramován pomocí ES6 a tak je třeba některým starším prohlížečům dodat takzvaný polyfill.

2.3 Wordpress

Wordpress je nejrozšířenější nástroj na spravování informačního obsahu na webu. Využívá ho až 36 procent webových stránek. Nástroje na správu informací uživateli umožňují vytváření a publikaci digtálního obsahu. Do této kategorie spadá množství dalších podobných nástrojů pro tvorbu blogů, fór, online krámů a spousty dalších podobných stránek. Konkrétně se jedná například o Joomla, Drupal, Shopify a Squarespace [26]. Každý z těchto nástrojů jednotlivě je však stále méně používaný než Wordpress. Pokud se zaměříme jen na tyto nástroje, tak 60 procent všech stránek, které nějaký nástroj využívají, využívají právě Wordpress. Na následující příčce je umístěna až Joomla s pouhými 6,4 procenty [31].
Wordpress začal v roce 2003, ve chvíli kdy Matt Mullenweg a Mike Little odvodili větev z blogovací platformy B2. Jedná se o open source projekt vydaný pod GPL licencí. Mullenweg založil pro Wordpress nadaci v roce 2010, aby zajistil dodatečnou podporu a udržitelnost. Tato nadace vlastní dokonce Wordpress ochrannou známku [25].
Úkol Wordpressu byl již od začátku zpřístupnění možností publikování informací. Jeho misí bylo zajistit, aby i jakákoliv netechnická osoba dokázala vytvořit webovou stránku, ale zároveň vytvořit platformu, která může být

používána i velkými podniky s komplexními potřebami. Wordpress obsahuje dokonce i programátorské rozhraní REST API, díky kterému je možné komunikovat pouze s back-end funkcemi a vybudovat si nad nimi vlastní aplikaci.
Wordpress je tvořen více než 500 tisíci řádkami kódu. Jedná se především o kód psaný v jazyce PHP. Tento kód se stará o obsluhování HTTP požadavků a dotazování se do MySQL databáze. Poslední dobou se však prudce zvyšuje využití programovacího jazyka JavaScript. Děje se tak převážně po stránce vzhledu a ovládání uživatelského rozhraní. Hlavním frameworkem pro tento nový směr se stal React [20].
V současné době tvoří přes polovinu kódu stále PHP, 30 procent je pak psáno v programovacím jazyce JavaScript a zbytek je rozdělen mezi CSS, HTML a XML. Během 17 let od vzniku vzrostlo množství řádek přidaných za rok z 16 tisíc na 600 tisíc [30].
Databázové schéma nástroje Wordpress je poměrně jednoduché. Vše je rozděleno do 12 klíčových tabulek. Kód je zase rozdělen do několika komponent, na základě jejich funkcionality. Každá komponenta je tvořena směsí procedurálních a objektově orientovaných programovacích technik. Nové funkce jsou většinou vytvářeny spíš objektově a některé starší procedurální části byly přesunuty do obalových tříd, aby bylo zajištěno lepší zapouzdření funkcí.
Wordpress momentálně využívá verzovací systém SVN a systém na sledování chyb s názvem Trac. Všechny důležité diskuze probíhají právě na něm. Existuje i repozitář na GitHubu, ale jedná se pouze o obraz SVN repozitáře, který je určený pouze pro čtení [10].
Aktuálně je nejnovější verze 5.3.2. Wordpress se snaží během svých aktualizací zachovat veškeré funkce a zachovat tak zpětnou kompatibilitu. Tím se liší od ostatních nástrojů pro správu webového obsahu. Například nástroj Drupal upřednostňuje, zlepšení svých funkcí na úkor kompatibility [2].
Z historických důvodů Wordpress nepoužíval sémantické verzování. První dvě cifry čísla verze, tedy určují hlavní verze a poslední cifra určuje o jaký se jedná patch. Donedávna docházelo k vydávání hlavních verzí průběžně. Přibližně každé čtyři měsíce. V roce 2016 však Wordpress oznámil, že se pokusí přejít na nový způsob a tím bylo vydávání hlavních verzí až v návaznosti na nové rozšíření funkcí.
Každé vydání má svého hlavního vývojáře a také hlavního designera. Tyto pozice se vydání od vydání mění a tím je zajištěno, že nedojde k příliš vysoké koncentraci informací v hlavách malého množství vývojářů. Vedoucí vývoje sice rozhodují o technických aspektech vydání, ale spoléhají převážně na Wordpress komunitu, která kód vyvíjí. Mimo tyto vedoucí pozice má několik klíčových členů přidělené právo zápisu do SVN repozitáře. Někdy je

toto právo rozšířeno i na členy, kteří pracují na specifických komponentách. Každý týden je svolána schůze na které jsou řešeny nalezené chyby. Bezpečnostní slabiny jsou však řešeny často okamžitě. Pokud je to nutné, dojde k vytvoření bezpečnostní záplaty a všechny Wordpress stránky jsou automaticky aktualizovány.
Wordpress se stal multimiliardovým podnikem a všechno nasvědčuje tomu, že bude v budoucnu jen růst. Klíčovým faktorem v tomto růstu je především komunita lidí, která se kolem tohoto nástroje vytvořila. Wordpress například organizuje tzv. Wordcampy. Což jsou conference s odbornými prezentacemi, které se dějí všude po světě. Minulý rok těchto konferencí proběhlo více než sto a zúčastnili se jich statisíce programátorů.
Kromě organizování konferencí a schůzek členové této komunity pomáhají celému tomuto ekosystému i tvorbou pluginů a témat. Plugin rozšiřuje využití nástroje Wordpress o nové funkce a témata umožňují uživatelům přizpůsobit jak Wordpress vypadá a jak se chová. Wordpress poskytuje obrovské množství tzv. 'Hooků', které umožňují navázat nový kód na již stávající jádro Wordpress systému, aniž by ho bylo třeba nějak změnit. Existuje kolem dvou tisíc hooků a každý odpovídá nějaké obecné Wordpress události. Mezi tyto události patří například publikace článku, komentování příspěvku nebo vytvoření nového uživatele. V reakci na tyto události dokáží pluginy a témata provést nějakou akci, nebo změnit způsob, kterým je obsah zobrazen uživateli [11].
Oficiální repozitář obsahuje kolem padesáti tisíc pluginů, které byly staženy až šest set milionkrát. Pluginy mohou být zdarma a nebo placené, občas i měsíčně. Kvalita pluginů se drasticky liší a většinou je to chyba právě některého pluginu, když dojde k napadení webové stránky, která Wordpress používá.
Tým kontrolující kvalitu pluginů zajišťuje, aby se do repozitáře nedostal nějaký plugin nebo téma, které by nesplňovalo zásady nástroje Wordpress [6]. Pravidla definující tyto zásady jsou poměrně striktní, ale zároveň umožňují prostor pro interpretaci. Díky tomu za historii nástroje Wordpress vznikly desítky kontroverzních případů rozhodnutí ohledně inkluze nebo exkluze konkrétních pluginů a témat. Vznik nového projektu s názvem Tide má tento proces ulehčit tím, že vývojářům umožní své pluginy před nahráním do repozitáře otestovat automatizovaným systémem.
Přestože jsou tvůrci pluginů a témat především nezávislí vývojáři, existují kolem nástroje Wordpress i společnosti, které se zabývají především poskytováním Wordpress hostingu a jiných služeb za měsíční poplatky. Wordpress je možné nainstalovat na jakýkoliv internetový hosting, některé hostingy však uživatelům poskytují rozšířenou podporu. Mezi tyto společnosti patří

například Mullenwegs Automattics, která své služby poskytuje pod doménou www.wordpress.com. Neplést si s oficiální Wordpress webovou stránkou umístěnou na www.wordpress.org [1]. Nejrozšířenějším poskytovatelem v České republice s podporou nástroje Wordpress je pak Wedos.
Wordpress ušel už obrovskou cestu od svých začátku jako nástroj pro správu blogů až po gigantický, ale stále flexibilní framework. Musí se stále vyvíjet, aby zůstal relevantní. Trh s nástroji pro správu digitálního obsahu se stále mění a noví konkurenti vznikají skoro každý den. Většina z nich se však soustředí na konkrétní uživatelské skupiny, narozdíl od nástroje Wordpress, který se snaží být univerzálním řešením pro všechny [13].
Jako reakce na tuto problematiku vznikla iniciativa Five for the Future, která požaduje po všech společnostech žijících na této platformě, aby věnovali pět procent zaměstnanců rozvoji Wordpress systému. Ať už se jedná o vývoj, psaní dokumentace, textovou podporu na fórech, testování, překlad nebo cokoliv co pomůže této platformě v posunu vpřed.
Současná verze nástroje začala používat tzv. Gutenberg. Tento posun lze považovat za největší architektonickou změnu, kterou Wordpress během svojí historie prošel. Gutenberg se snaží zjednodušit koncepty, které Wordpress už několik let využívá. Koncepty jako je menu, různé widgety a použití speciálních tagů s názvem shortcode. Rozhraní Gutenberg právě tyto jednotlivé prvky nahradí jedním společným, který má jednoduchý název tzv. “Blok”. Nově se bude tedy rozhraní skládat z různých do sebe zapadajících bloků. Podle zakladatele Mullenwega, je rozhraní Gutenberg budoucností celé platformy Wordpress, co se týče psaní, upravování a customizace. Stejně jako jakákoliv jiná velká změna i Gutenberg rozdmýchal ve Wordpress komunitě velké množství sporů. Kvůli těmto změnám bude muset velká část vývojářů přepsat své pluginy a témata.
Tyto změny jsou v souladu s cílem platformy Wordpress, stát se dominantní platformou pro tvorbu webových stránek. Někteří uživatelé mohou tyto změny však vnímat, jako posun špatným směrem. Pokud se jim tento vývoj nebude líbit, mohou v budoucnu oddělit od hlavní vývojové větve a vytvořit si vlastní verzi platformy, která bude lépe reprezentovat jejich cíle. To by měla za důsledek roztříštění současné komunity na menší díly. Tomuto riziku však čelí všechny open source projekty.
Ať už dopadne vývoj platformy Wordpress jakkoliv, určitě velkou roli v něm určitě bude mít komunita, která ho obklopuje. Zajímavé je, že i přes rozsáhlou komunitu, není platformě Wordpress věnovaná dostatečná pozornost co se týče výzkumu. Rozdíl je obzvlášť dramatický při porovnání s například operačním systémem Linux, který je zkoumán po všech možných stránkách. Studií o bezpečnosti, stability, použitelnosti platformy Wordpress ještě

zkrátka tolik nevzniklo.

2.4 Sass

Pro úpravu vzhledu na webových stránkách se používají kaskádové styly. Jenže soubory s čistými kaskádovými styly mohou snadno narůst. Stávají se poté nečitelné a hůře se udržují. Samotné kaskádové styly také postrádají určité užitečné vlastnosti, jako například hodnotové proměnné, vrstvení a dědění. Sass je takzvaný preprocesor kaskádových stylů. Umožňuje nám psát modifikovanou syntaxi, která se po přeložení přetvoří na ekvivalentní verzi psanou v čistých kaskádových stylech.
Technologie Sass je moderní způsob vyvíjení kompatibilních kaskádových stylů a je extenzivně využívána nově vznikajícími JavaScript knihovnami. Jednou z těchto knihoven je i Plyr, který pomocí ní generuje vzhled přehrávače.

3 Analýza

V této kapitole se práce bude věnovat v první části historickým řešením přehrávání a následně analýze požadavků moderního Wordpress pluginu. Jsou zde popsány důvody, které vedly k zániku Flashových přehrávačů videa. Kapitola dále obsahuje doporučení, na které je dobré myslet, při vývoji pluginu pro redakční systém Wordpress. Na konci kapitoly jsou také detailně rozepsány nejčastější komponenty, které jsou v současné době využívány pro přehrávání videa ve webových prohlížečích.

3.1 Flash

V minulosti, když vznikla potřeba přenášet video soubory po síti, se objevilo několik aplikací, které umožňovaly právě to. Nejvíce používané byly přehrávače QuickTime, Windows Media Player a Real Player. Byla to velmi problematická doba, protože uživatelé Windows postrádali QuickTime, uživatelé Mac zase Windows Media Player a Real Player nebyl velmi populární. V tu chvíli se objevila stránka YouTube, která využívala přehrávač zvaný Flash Player. Přestože YouTube nebyla první stránka, která by tento přehrávač využila, rozhodně se dá říct, že ho popularizovala.
Využití přehrávače Flash player programátorovi umožnilo například vytvoření vlastního vzhledu, dodělání nových funkcí, vkládání reklam a zamaskování zdroje videa, aby ho nebylo možné snadno stáhnout. Flash player také dokázal, jako první, zvětšit video na celou obrazovku. V počátcích internetu se videa šířila spíše tak, že je uživatelé uložili na svůj počítač a pak je rozeslali svým známým. Když tedy velké stránky jako je YouTube, Hulu, cnet.com a nbc.com, zjistili, že existuje způsob jak mohou uživatelům přehrávat reklamy a zablokovat jim stahování jejich obsahu, netrvalo jim dlouho tuto technologii implementovat.

3.1.1 Boj společnosti Apple proti přehrávači Flash Player

Flash byla technologie, která se rozšířila skoro na každý počítač. Jeden z hlavních důvodů, proč tomu tak už není je i tvrdý postoj společnosti Apple. S prvním vydáním iPhonu Steve Jobs oznámil, že zanechají podpory Flashe [21]. Byl to velice odvážný tah, jelikož v té době byl Flash součástí velkého množství stránek. Důvody proč dospěl k takovému rozhodnutí popsal

v dnes známém článku “Thoughts on Flash”. Největší problémy, které Flash měl byly jeho vysoké nároky na paměť, drasticky zvyšoval spotřebu baterie a nebyl přizpůsobený na dotykové zařízení. Flash v té době zastával obrovské množství funkcí a vývojáři ho používali jako snadné multiplatformní řešení pro skoro jakékoliv aplikace. Tím že Apple zakázal Flash, znemožnil uživatelům používat jiné aplikace, než ty které se nacházeli na AppStore. Což byl další produkt, který se snažil Apple tvrdě tlačit. Nejen, že by povolením Flashe, Apple prohrál tento boj o nově vznikající trh mobilních aplikací, ale zároveň by ztratil pravomoc kontrolovat, jaký obsah uživatelé jejich produktů konzumují. Došlo by k tomu, že uživatelé by místo stáhnutí přizpůsobené aplikace z AppStoru, nad kterými má Apple kontrolu, raději navštěvovali webové stránky a získali tak špatný uživatelský dojem z celého procesu. Steve Jobs měl pravdu a po 10 letech je situace taková, že na chytrých telefonech jsou aplikace stále zprostředkovány převážně přes obchody jako je AppStore a Google Play a dokonce i samotná firma Adobe, která Flash vytvořila, se od něj rozhodla upustit a přešla na modernější a slibnější HTML5.

3.2 Požadavky na výkonnost nástroje Wordpress

Pro návštěvníky webových stránek je výkon důležitý. V dnešní době je každá vteřina strávená načítáním webové stránky kritická. Co se týče výkonnosti bude nástroj Wordpress samozřejmě zaostávat za ručně napsaným kódem, který může programátor konkrétně optimalizovat. Programovat všechno ručně pro konkrétní způsoby užití má kromě obrovské pracnosti i jiná úskalí. Nejdůležitějším z nich je čitelnost indexovacími roboty. Pokud bychom optimalizovali stránky moc, je pravděpodobné, že dojde k zhoršení právě této čitelnosti. To by mělo za následek snížení indexu v žebříčcích internetových vyhledávačů. Nižší index by pak vedl k nižšímu počtu nových návštěvníků. Vysokou čitelnost pro roboty mají webové stránky vyrobené právě za použití nástrojů jako je Wordpress. Při porovnání té samé webové stránky vytvořené pomocí ručně napsaného kódu a vytvořené pomocí nástroje Wordpress je možné pozorovat, že časy načtení webové stránky se liší jen o jednotky procent. Wordpress sice porvání při načítání mnohem více dotazů na server, získané výhody však o dost převyšují tyto mírné výkonnostní penalizace [29].
Jednotlivé faktory, které nejvíce ovlivňují dobu nahrávání webové stránky, jsou přidané scripty, délka HTML kódu a kaskádových stylů, velikost ob-

rázků, videa a textu. Při vývoji témat a pluginů je dobré na tyto faktory dbát.

3.3 Funkční požadavky vytvářeného pluginu

Výsledný plugin by měl splňovat následující funkční požadavky. Uživatel pluginu musí být schopen vytvořit seznam videí. Tento seznam je pak možné spojit s určitým příspěvkem v redakčním systému Wordpress. Návštěvníkovi stránek se po otevření patřičného příspěvku zobrazí přehrávač, který přehraje daný seznam videí. Seznam videí se tedy chová jako pořad s několika částmi. První část může být například úvodní znělka a poslední, video s titulky. Přesto, že se prostředek pořadu mění, tak počáteční a koncové soubory jsou sdíleny přes všechny díly. Konkrétní díl pořadu může být vložen do několika příspěvků v systému Wordpress. Administrátor webu může zpětně změnit nějakou část pořadu a tato změna se projeví na všech místech kde je pořad vložen. Do pořadů mohou tímto způsobem být vkládány dočasné reklamy, nebo může dojít ke změně úvodního videa.
Přehrávané soubory budou moci být umístěna jak lokálně, tak vzdáleně na serverech YouTube. Pokud uživatel umístí často opakovanou část pořadu na servery YouTube dojde k ušetření zátěže lokálních serverů.
Vytváření seznamů videí by mělo být dostatečně jednoduché, aby k němu uživatel nepotřeboval žádné předchozí programátorské zkušenosti.
Celé řešení musí být zapouzdřeno ve Wordpress pluginu a musí jít snadno nahradit nebo vypnout skrze ovládací centrum systému Wordpress. Mělo by pro svoji činnost co nejvíce využívat vestavěné funkce a řešení definované systémem Wordpress.
Kromě těchto základních funkčních požadavků je třeba dodržet i následující požadavky definované komunitou redakčního systému Wordpress, zvlášť pokud dojde k budoucímu vystavění vytvořeného pluginu do veřejného repozitáře.
Plugin musí být kompatibilní s GNU licencí. Veškerý kód, data, obrázky nebo jakékoliv soubory umístěné ve kořenové složce pluginu musí splňovat obecnou veřejnou licenci nebo s ní být kompatibilní. Kód musí být čitelný člověkem. Nesmí tedy být nijak skrytý nebo zpřeházený. Některé převážně JavaScriptové knihovny totiž umožňují kód znepřehlednit aby tak zabránily dalšímu rozšíření nebo úpravě. Výsledný plugin nemůže fungovat ve formě zkušební verze, která by po nějaké době omezila svou funkci nebo přestala fungovat kompletně. Plugin nesmí sbírat uživatelská data bez souhlasu. Ne-

smí docházet ke stahování kódu ze služby třetí strany. To znamená, že nesmí dojít ani k využití tzv. CDN. Content delivery network je název pro online služby hostující specifický kód nebo kaskádové styly. Plugin by měl pro svou funkci využívat vestavěné WordPress knihovny a nepřidávat zbytečně alternativní knihovny, které realizují stejné funkce. Jedná se především o využití jQuery namísto ostatních alternativ. Poslední důležité pravidlo pro Wordpress plugin je zachování autorských práv. Plugin, který tato pravidla nesplňuje nesmí být umístěn do veřejného repozitáře pluginů [23].
Jako poslední požadavky bych chtěl zmínit obecné standardy, které budou při vývoji dodrženy. Kód by neměl využívat funkcí označených jako deprecated. Veškeré třídy, proměnné a i záznamy v databázi budou označeny prefixem, aby nedocházelo ke konfliktům. Při využívání databáze nebude použitý přímý přístup, ale místo něj bude veškerý přístup realizován přes databázové aplikační rozhraní nástroje Wordpress [4]. Plugin nebude vytvářet nové tabulky, ale bude využívat ty stávající definované systémem Wordpress. Všechny dotazy budou ošetřeny proti SQL injection. Plugin bude využívat výchozí verzi jQuery, u které nikdy nedojde k odregistraci. Všechny skripty a kaskádové styly budou vloženy skrze registrační rozhraní systému Wordpress. Plugin nebude modifikovat nebo odstraňovat data v reakci na svoji deaktivaci. Kód bude přehledně odřádkovaný pomocí tabulátorů. Soubory budou využívat obecné značky <?php a ne jejich zkrácené verze. Soubory budou uloženy v kódování UTF-8. Všechny bloky budou označené pomocí složených závorek [7].

3.4 Komponenty pro přehrávání videa

Historicky bylo vkládání videa do obsahu webové stránky komplikovaná záležitost. V roce 2011 existoval jediný webový standard s názvem HTML4, který umožňoval vkládat videa do webových stránek skrze elementy <iframe> nebo <object>. Naštěstí se toto všechno změnilo s nástupem HTML5 v roce 2015. Tento standard představil element <video>, který umožňuje vkládání videí přímo, bez žádného externího zprostředkovacího softwaru. Obrovskou výhodou tohoto přístupu je kompatibilita, protože tento nový způsob je interpretován stejným způsobem jak na stolních počítačích, tak na telefonech či tabletech.
Nejjednodušším a nejrozšířenějším způsobem přehrávání videí na webu je tedy HTML5 přehrávač. Výhody tohoto přístupu je kompatibilita s širokým množstvím internetových prohlížečů. Mezi nevýhody pak patří potřeba uklá-

dání videí na FTP server. Mezi nejpopulárnější HTML5 přehrávače patří:

- Plyr
- Video.js
- Afterglow
- MediaElement.js
- jPlayer
- JW Player

Většina těchto přehrávačů je skoro totožná. Nejzajímavějším přehrávačem je Plyr. Tento přehrávač je open-source a vytvořila se kolem něj aktivní komunita, která tlačí vývoj stále dopředu. Přehrávač JW Player, který je na poslední pozici, byla skvělá volba pro publikování videí v redakčním systému Wordpress. Tento přehrávač však přesel na nový obchodní model, který umožňuje využití zdarma pouze do omezené měsíční datové kapacity.
Dalším velice populárním řešením je vložení přehrávače ze stránky YouTube. Tento server totiž zprostředkovává přehrávač, který konvertuje uložené video do obrovského množství formátů a pro přehrávání implementuje jak Flash tak moderní technologie HTML5. Hlavní výhody jsou jednoduchost implementace, absence hostování souborů s videem. Nevýhodami je pak závislost na externím hostingu, možné nežádoucí reklamy a také riziko toho, že uživatelé opustí vaši stránku a budou videa sledovat raději přímo z YouTube.
Posledním způsobem je vložení přehrávače Flash Player. Tato možnost je na ústupu od doby, kdy společnost Apple odmítla implementovat podporu na nových iOS zařízeních. Mezi nevýhody patří potřeba instalace externího pluginu pro internetové prohlížeče a absence kompatibility na většině mobilních zařízeních.

4 Návrh

V této kapitole se práce věnuje jednotlivým částem procesu, které musí být uživatel schopný provést. Uživatel našeho pluginu bude moci vytvořit nové video. Toto video pak následně spáruje s příspěvkem v redakčním systému Wordpress. Při otevření daného příspěvku dojde k přihrání spárovaného videa. Obsahem této kapitoly je návrh těchto jednotlivých kroků.

4.1 Vytvoření videa

Nejprve je nutné navrhnout datovou strukturu, do které bude možné vytvořené video uložit a ze které ho bude možné nahrát. Současné požadavky na plugin vyžadují, aby bylo uloženo pole, ve kterém každá položka představuje jedno přehrávané video. V budoucnu je ale možné, že bude třeba datový objekt rozšířit a začít ukládat více nastavení. Například úpravy vzhledu, obrázek videa před spuštěním nebo různá metadata. Datová struktura, která video bude reprezentovat tedy musí být dostatečně netriviální, aby toto rozšíření umožňovala. Bylo by vhodné, aby jednotlivá videa byla uložena jako záznamy v databázové tabulce. Wordpress v současnosti využívá tabulek hned několik.

4.1.1 Wordpress databáze

Databáze systému Wordpress je vytvořena při každé instalaci. Všechna data, která systém Wordpress využívá, jsou v ní uložena. Jedná se o příspěvky, stránky, komentáře a nebo nastavení. Standartě existuje jedenáct hlavních tabulek [3].

1. wp_posts
2. wp_postmeta
3. wp_options
4. wp_users
5. wp_usermeta
6. wp_term_taxonomy
7. wp_terms

8. wp_term_relationships
9. wp_links
10. wp_comments
11. wp_commentmeta

Většina z těchto tabulek obsahuje navíc relační vztahy. To znamená, že řádek v jedné tabulce může být spojen s jedním nebo více řádky v jiné tabulce. Všechny tyto relace jsou zobrazeny na obrázku A.1. Pro uložení informací potřebných pro přehrávání videí je třeba vybrat jednu nebo více těchto tabulek. Některé z nich jsou vhodnější než ostatní a tak je třeba analyzovat k čemu obvykle slouží a vybrat ty správné.

wp_posts

Tabulka ve které jsou uloženy jednotlivé příspěvky a stránky. Dalo by se říct, že se jedná o vůbec nejdůležitější tabulku databáze systému Wordpress. Obsahem je text, revize, položky menu a datové přílohy.

wp_postmeta

Je to rozšíření tabulky wp_posts. Jsou v ní uloženy extra informace patřící položkám z této tabulky. Některé pluginy do ní ukládají svá data.

wp_options

Tabulky nastavení se od předchozích tabulek mírně liší. Namísto ukládání obsahu stránek, obsahuje jednotlivé položky a hodnoty konfigurace. Je v ní uložen například název stránek, popis a časová zóna. Narozdíl od ostatních tabulek, tato tabulka tedy nevlastní žádnou relační vazbu s ostatními tabulkami.

wp_users

Tabulka wp_users obsahuje seznam všech registrovaných uživatelů. Obsahuje základní informace jako je přihlačovací jméno, hash hesla, email, identifikátor, přezdívku, datum registrace a tak dále. Převážně hash uživatelského hesla je velice užitečný. Systém Wordpress neukládá hesla jako běžný text, protože by to bylo bezpečnostní riziko. Pokud má ale uživatel přístup do databáze, může si vygenerovat nový hash, přepsat ten současný a následně se přihlásit jako jakýkoliv uživatel. Navrácením původního hash řetězce je

možné vrátit původní heslo, takže uživatel nic nepozná. Tato funkce je obzvlášť užitečná pokud je třeba plugin nasadit na systém u kterého postrádáte administrátorské přihlašovací údaje, ale zároveň máte přístup k databázi.

wp_usermeta

V této tabulce jsou uloženy informace o uživatelích, které jsou méně důležité. Je zde uložené například příjmení.

wp_terms

Tato tabulka ukládá všechny možné kategorie, ať už se jedná o příspěvky, stránky nebo jmenovky. Tato tabulka obsahuje relační vazby na následující dvě tabulky.

wp_term_taxonomy

Tabulka wp_term_taxonomy obsahuje popisy jednotlivých kategorií z tabulky wp_terms.

wp_term_relationships

Jak už název napovídá, tato tabulka udržuje relační vazby. Pokud je některý příspěvek například spojený s určitou kategorií a je označen několika jmenovkami, tyto vztahy budou uloženy právě v této tabulce.

wp_links

Donedávna systém Wordpress obsahoval funkci s názvem blogrolls. Jednalo se o menu umístěné v postraní liště. Toto menu obsahovalo odkazy na externí stránky. Uživatelé systému Wordpress však začali zneužívat této funkce a tak došlo k jejímu zrušení v novějších verzích. Tato tabulka udržovala informace ohledně této zrušené funkce. Přesto, že nové verze jí již neobsahují, tak je tato tabulka zachována z důvodů zpětné kompatibility.

wp_comments

Všechny typy komentářů, ať už schválené nebo zamítnuté, které uživatelé zanechávají na příspěvcích a stránkách jsou uloženy právě v této tabulce. Další specifická data jako je jméno autora, jeho emailová adresa a zda se jedná o odpověď nebo obyčejný komentář, jsou zde také uložena. Je dobré myslet na to, že pokud uživatel systému Wordpress používá pro ukládání

komentářů vlastní plugin, tak se pravděpodobně komentáře ukládají jiným způsobem.

wp_commentmeta

Tabulka wp_commentmeta obsahuje extra informace týkající se komentářů. Je zde uložena například relace k jakému příspěvku nebo stránce je komentář vztažený.

4.1.2 Výběr tabulky

Jak už jsem se dříve zmínil, je třeba vybrat některou z tabulek pro uložení datového modelu odpovídajícímu vytvořenému videu. Samozřejmě, že nejsnadnějším řešením je vytvoření nové tabulky, ale v rámci zachování architektury systému Wordpress je rozumnější použít jednu z již existujících tabulek. Pokud by každý příspěvek měl jen jedno video, bylo by rozumné využít pro implementaci tabulku wp_postmeta. Ke každému příspěvku nebo stránce by tedy bylo připojeno několik záznamů z této tabulky. Každá hodnota by však v této tabulce musela být uložena zvlášť a mohla by vlastnit realční vztah pouze s jedním příspěvkem. Proto je nejlepším řešením výběr nejobecnější tabulky wp_post. Architektonicky tedy budou jednotlivé příspěvky vlastnit vazbu na další příspěvky. Tyto vlastněné příspěvky však budou již reprezentovat jednotlivá videa. To nám dokonce umožňuje videím přidat další extra informace pomocí vazeb na tabulku s metadaty [9].

4.2 Spárování s příspěvkem

V tuto chvíli je navržený způsob na ukládání informací o videu. Jednotlivé příspěvky a stránky jsou tedy uloženy v tabulce wp_post a mají přidělené identifikační číslo. Ve stejné tabulce jsou uloženy i vytvořené videa a ty mají také přidělené identifikátory. Je třeba tyto 2 identifikátory spojit. Prvním možným řešením by bylo vytvoření relační vazby přes tabulku wp_postmeta. Tato vazba by ale musela být udržována a při každé změně modifikována. Architektura systému Wordpress realizuje vkládání videí přes editor příspěvků. Nový systém editování příspěvku s názvem Gutenberg dokonce umožňuje přetáhnout prvek reprezentující video přímo na místo, kde se má video zobrazit. Rozhodl jsem se tedy navrhnout spárování videa nebo videí takovým způsobem, že uživatel vytvářející příspěvek vloží objekt s identifikačním číslem videa přímo do textového obsahu příspěvku.

4.3 Zobrazení videa

V tuto chvíli je spárován konkrétní příspěvek s jedním nebo několika videi. Při otevření stránky s příspěvkem dojde k vykreslení přehrávače. Primární volbou pro tento přehrávač je javascriptový přehrávač s názvem Plyr. Je nejrozšířenější, neustále udržovaný a funkčně jednoduchý přehrávač, který je open source. Je třeba předat data ze serveru běžícím v jazyce PHP na front-end vrstvu na které běží přehrávač. To lze realizovat třemi způsoby.
První způsob využívá pro získání dat Ajax. Jedná se o technologii, která umožňuje vytvářet asynchronní dotazy na server. Vytvořením přímého volání PHP funkce, lze získat návratové hodnoty z back-end serveru. Výhodou je, že toto volání může probíhat asynchronně. Nevýhodou je složitost implementace a realizace. Každé spuštění tohoto volání totiž vyvolá vlastní HTTP dotaz.
Druhou možností je vložení dat přímo do stromové struktury webové stránky. Do objektového modelu webové stránky lze přidat na serveru element, který obsahuje potřebná data. Tento element zůstane skrytý, ale lze ho zpřístupnit na straně klienta pomocí skriptu. Nevýhodou tohoto řešení je to, že může dojít k vygenerování nevalidního HTML kódu. Poslední možností je obalení vložených dat HTML elementem <script>. Obsah se odešle na front-end a při zobrazení se vnitřní skript automaticky provede. Je možné tedy ukládat data rovnou do kontextu jazyka JavaScript. Tento způsob se zdá nejlepší, jelikož jsou data po načtení webové stránky konstantní.

5 Implementace

Tato kapitola se bude věnovat samotné implementaci. Obsahuje detailní vysvětlení jednotlivých částí vytvořeného pluginu. V kapitole Obalový projekt bude čtenář seznámen se způsobem, kterým je přehrávač zakomponován do redakčního systému Wordpress. Gutenberg je nový editor na který se systém Wordpress pokusí časem přejít. Plugin obsahuje rozpracovaný blok, kterým bude možné v budoucnu přidávat videa skrze tento nový editor. Většina webů však tento editor ještě nevyužívá a tak je se kapitola dále věnuje takovému řešení, které je funkčně kompatibilní se staršími verzemi redakčního systému Wordpress. Následující podkapitoly také odpovídají předchozím oddílům v kapitole Návrh. V podkapitole Video editor je vytvořeno řešení, které umožňuje vytvářet videa. V podkapitole Zástupné kódy je pak vyřešeno spárování s příspěvkem a nakonec v kapitole Přehrávač je vytvořeno řešení pro zobrazení videa.

5.1 Obalový projekt

Výsledný plugin, pro přehrávání videí v nástroji Wordpress je koncipován jako dva projekty. Jeden obalový, který řeší korektní napojení, nastavení a komunikaci s nástrojem Wordpress a pak jeden vnitřní, který se stará o přehrávání videa. Tato kapitola se věnuje obalové části jejíž kód je interpretován na straně serveru.

5.1.1 Gutenberg

Jak už bylo zmíněno výše v této práci, nástroj Wordpress nově přešel na nový systém tvorby webového obsahu Gutenberg [5]. Tento nový editor funguje na bázi technologie React a přístup k implementaci nových funkcí se velmi liší od klasického přístupu. Počáteční myšlenka byla tedy vyrobit plugin jako nový prvek tohoto editoru. Problém ale nastává ve chvíli, kdy by měl být plugin zpětně kompatibilní se stránkami, které stále běží na starší verzi nástroje Wordpress, která systém Gutenberg nepodporuje. Rozhodl jsem se nejprve napsat detekční kód 5.1, který je spuštěn před registrací pluginu. Podle výsledku této funkce je pak rozhodnuto, jestli má dojít k vytvoření a registraci blocku a nebo registraci nastavení starajících se o fallback, který zprostředkuje stejné funkce, ale bez použití systému Gutenberg.

```
function is_gutenberg_active()
```

```
{
    $gutenberg    = false;
    $block_editor = false;
    if (has_filter('replace_editor', 'gutenberg_init')) {
        // Gutenberg is installed and activated.
        $gutenberg = true;
    }
    if (version_compare($GLOBALS['wp_version'], '5.0-beta', '>')) {
        // Block editor.
        $block_editor = true;
    }
    if (!$gutenberg && !$block_editor) {
        return false;
    }
    include_once ABSPATH . 'wp-admin/includes/plugin.php';
    if (!is_plugin_active('classic-editor/classic-editor.php')) {
        return true;
    }
    $use_block_editor = (get_option('classic-editor-replace') === 'no-
    replace');

    return $use_block_editor;
}
```

Kód 5.1: Funkce pro zjištění existence Gutenberg editoru

Díky této funkci jsme schopni detekovat, zda má dojít k registraci Gutenberg blocku. Nejprve je tedy třeba registrovat nový typ blocku. Pro registraci je nutné použít vstupní funkci nástroje Wordpress register_block_type(). Tato funkce má dva vstupní argumenty. Prvním je název blocku a druhým je pole obsahující konfiguraci.
Jako název nového blocku jsem zvolil řetězec 'wplyr-better-video/wplyr-video-block'. První část tvoří název pluginu, která je identická s názvem složky, ve které je umístěný. Pole s konfigurací pak obsahuje názvy již registrovaných scriptů, které se starají o realizaci blocku. V první řadě jde o script, který definuje jak se má block vykreslit při zobrazení na webové stránce. Ten je v poli uložen pod klíčem 'script'. Druhý script definuje, jak se bude block vykreslovat při editaci příspěvků v editoru. Ten je v poli uložen pod klíčem 'editor_script'.
Nové blocky jsou ve Wordpressu realizovány syntaxí, která je vystavěná na frameworku React. Základním prvkem blocku je wp.element. Jedná se o tenkou abstrakční vrstvou nad javascriptovým objektem React.Component. Gutenberg tento základní prvek používá pro vytvoření blocku a statického HTML výstupu. Tento statický HTML výstup pak slouží k uložení a zachování stavu blocku. Gutenberg editor tedy používá komponenty frameworku

React, které následně serializuje a ukládá. Jednotlivým návštěvníkům je pak servírován statický HTML kód místo React komponent, které by se na straně klienta teprve interpretovaly. Skript vytvoření Gutenberg bloku je iniciován pod registračním řetězcem s názvem 'wp_wplyr_video_gutenberg_block'. Obecně všechny registrační řetězce budou mít prefix wp_ pro označení, že patří do nástroje Wordpress, následuje další prefix wplyr_, který definuje konkrétní plugin a pak tento řetězec obsahuje sufix _gutenberg_block, který znamená, že se jedná o právě o skript obsahující implementaci blocku.
Z důvodů uvedených v následující kapitole však vývoj bloku pro nový editor nebyl kompletně dokončen. Ve zdrojových souborech je tedy možné najít základní návrh výsledného bloku, který je možné přidávat do nového editoru. Není však dokončené napojení na konkrétní soubory s videi a tak je tento blok v publikované verzi zatím vypnutý.

5.1.2 Video editor

Jelikož začal nástroj Wordpress využívat nový editor Gutenberg teprve nedávno a to v roce 2018, využívala ho jen velmi malá část webových stránek. Po konzultaci s administrátorem stránek www.catvusa.com, vyšlo najevo, že pro sdílení obsahu používají Wordpress 4. Jde o mnohem starší verzi, která editor Gutenberg ještě ani neobsahuje. Primárním cílem projektu se tedy stalo to, aby výsledný plugin fungoval na webech, které ještě editor Gutenberg nepodporují [19]. Došlo tedy nejprve k vytvoření zpětně kompatibilní verze, které umožní korektní fungování pluginu i na starších verzích systému Wordpress.
Bylo nutné vytvořit komponentu, která simuluje chování bloku a umožňuje vytvoření a publikování videa. Jelikož je možné při přehrávání vytvořit i video, složené z více oddělených souborů, je nutné uživateli zobrazit editor, který takovou konfiguraci umožňuje. Pokud by se jednalo o Gutenberg block, objevily by se tyto možnosti na obrazovce během přesunutí nového video blocku do editoru nového příspěvku. Informace o jednotlivých videích by pak byly serializovány a uloženy do obsahu daného příspěvku. Pro použití pluginu v nástroji Wordpress, který ještě nepodporuje Gutenberg, bylo třeba tedy vymyslet jiný postup.
Video by mělo být možné umístit na jakoukoliv stránku v nástroji Wordpress a tak slučovat informace o videu s konkrétním příspěvkem nepřipadá v úvahu. Wordpress využívá pro ukládání dat jednoduché objekty, které odpovídají databázovým tabulkám. Jedním z nejjednodušších a nejobecnějších datových objektů je "příspěvek". Pro uložení dat týkající se konkrétního videa byl tedy vytvořen nový typ příspěvku 5.2. Tento nový příspěvek si v sobě

udržuje stav daného videa. Po úspěšném vytvoření je možné tento datový příspěvek vložit do jakéhokoliv jiného obsahového příspěvku.

```
function wp_wplyr_register_content_type()
{
    //Labels for post type
    $labels = array(
        'name'               => 'WPlyr Video Manager',
        'singular_name'      => 'Video',
        'menu_name'          => 'Videos',
        'name_admin_bar'     => 'Video',
        'add_new'            => 'Add New',
        'add_new_item'       => 'Add New Video',
        'new_item'           => 'New Video',
        'edit_item'          => 'Edit Video',
        'view_item'          => 'View Video',
        'all_items'          => 'All Videos',
        'search_items'       => 'Search Videos',
        'parent_item_colon'  => 'Parent Video:',
        'not_found'          => 'No Videos found.',
        'not_found_in_trash' => 'No Videos found in Trash.',
    );
    //arguments for post type
    $args = array(
        'labels'            => $labels,
        'public'            => false,
        'publicly_queryable' => false,
        'show_ui'           => true,
        'show_in_nav'       => true,
        'query_var'         => true,
        'hierarchical'      => false,
        'supports'          => array('title'),
        'has_archive'       => true,
        'menu_position'     => 20,
        'show_in_admin_bar' => true,
        'menu_icon'         => 'dashicons-video-alt',
        'rewrite'            => array('slug' => 'videos', 'with_front' =>
     'true')
    );
    //register post type
    register_post_type('wp_wplyr_videos', $args);
}
```

Kód 5.2: Definice vlastního WordPress příspěvku

Pro zaregistrování nového typu příspěvku je nutné naplnit dvě pole předdefinovanými argumenty. V prvním poli jsou převážně názvy použité pro uživatelské rozhraní nástroje Wordpress a druhé pole obsahuje konkrétní přepínače nastavení. Důležité pro nás je převážně nastavení přepínače public na

false. Což zabrání zobrazení datového příspěvku jako obsahového příspěvku na webových stránkách. A pak také přepínače show_in_nav, který, jak už název napovídá definuje, zda dojde k zobrazení datových příspěvků s videi v administrační navigační liště.
V tuto chvíli nám Wordpress sám zařídí rozhraní, které umožňuje vytvoření nového příspěvku, prohledávání a editaci již vytvořených příspěvků. Jelikož nově přidané datové příspěvky ještě nemají žádný obsah, je třeba přidat nějaká editační pole, která umožní ukládat data.
Úsek kódu 5.3 zajišťuje zobrazení editoru seznamu videí pro přehrání. V první části dojde k vložení kaskádových stylů a javascriptových souborů. Toto embedování je provedeno skrze funkce wp_enqueue_*, které v nástroji Wordpress zajišťují, že se načtou až ve chvíli kdy jsou potřeba a hlavně dojde k načtení pouze jednou.

```
function wp_wplyr_option_box_html($post)
{
    wp_enqueue_script('wplyr-file-tree-script', plugin_dir_url(__FILE__)
    . '/video_editor/phpFileTree/php_file_tree.js');
    wp_enqueue_style('wplyr-file-tree-stylesheet', plugin_dir_url(
    __FILE__) . '/video_editor/phpFileTree/styles/default/default.css');
    wp_enqueue_script('wplyr-editor-script', plugin_dir_url(__FILE__) . '
    /video_editor/editor_script.js');
    wp_enqueue_style('wplyr-editor-stylesheet', plugin_dir_url(__FILE__)
    . '/video_editor/editor_style.css');
    ?>
```

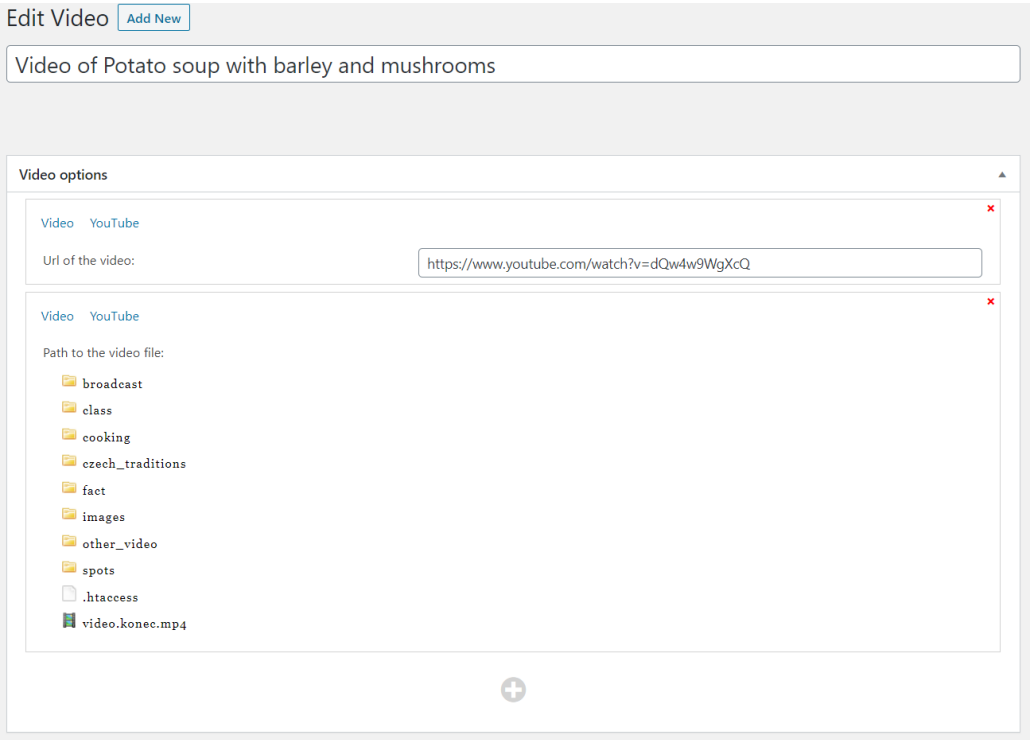
Kód 5.3: Editor seznamu 1/3 - Deklarace skriptů

Editor se skládá z bloků, které se dají řetězit za sebe. Definice bloku je vidět v ukázce 5.4. V každém bloku se nachází pole, která umožňují volbu zdroje videa. Protože lze tento blok opakovat, rozhodl jsem se uložit ho do DOM. Při vkládání nového videa se tento schovaný blok zkopíruje a odkryje. Tímto způsobem je možné k bloku přistupovat jak na serveru z programovacího jazyka PHP, tak následně ze strany uživatele z programovacího jazyka JavaScript.

```
    <div class="wplyr_editor_border" id="wplyr_hidden_path_editor" hidden
    >
        <a class="wplyr_editor_remove_button" onclick="
    remove_editor_field(this.parentNode)"></a>
        <div class="wplyr-bar">
            <a class="wplyr-bar-item wplyr-button" onclick="openTab(this
    ,'Video')">Video</a>
            <a class="wplyr-bar-item wplyr-button" onclick="openTab(this
    ,'YouTube')">YouTube</a>
```

```
    </div>
    <input name="wp_wplyr_video_type[]" value="video" class="
wp_wplyr_video_type_value" type="text" readonly hidden>
    <input name="wp_wplyr_video_source[]" type="text" class="
wp_wplyr_video_source_value" readonly hidden>
    <div class="wplyr-tab wplyr-video-tab">
        <div>
            <label>Path to the video file:</label>
            <b class="wp_wplyr_video_source_value"></b>
        </div>
        <?php
        echo php_file_tree(wplyr_video_path, "change_picked_file('[
link]',this);");
        ?>
    </div>
    <div class="wplyr-tab wplyr-youtube-tab" style="display: none;">
        <div>
            <label>Url of the video:</label>
            <input type="text" class="wp_wplyr_youtube_input
wp_wplyr_video_source_value" onchange="updateInputs()" value="">
        </div>
    </div>
</div>
```

Kód 5.4: Editor seznamu 2/3 - Deklarace panelu



Obrázek 5.1: Editor seznamu videí s alternativní možností vložení YouTube url

Vykreslený blok umožňuje volbu zdroje videa. Na obrázku 5.1 je vidět vzhled bloku při zvolení zdroje umístěného na stránkách YouTube. Teď je třeba ještě obnovit současný stav seznamu videí, aby odpovídal naposledy uloženému modelu. Tuto funkci realizuje poslední část kódu 5.5. Z metody get_post_meta získám seznam zdrojů videí a informaci o tom jestli se jedná o video ze serveru YouTube nebo lokálně uložený soubor.
Na poslední řádce je pak vykresleno plusové tlačítko, přes které je možné přidat nový blok do řetězce.

```
<div id="wplyr_editor_container">
    <?php
    $sources = get_post_meta($post->ID, '_wp_wplyr_video_source',
true);
    $types = get_post_meta($post->ID, '_wp_wplyr_video_type', true);
    if(!empty($sources)){
        for ($i = 1; $i < sizeof($sources); $i++) {
            wp_wplyr_print_editor($sources[$i], $types[$i]);
        }
    }
?></div>
```

```
    <a class="wplyr_editor_add_button" onclick="add_editor_field()"></a>
```

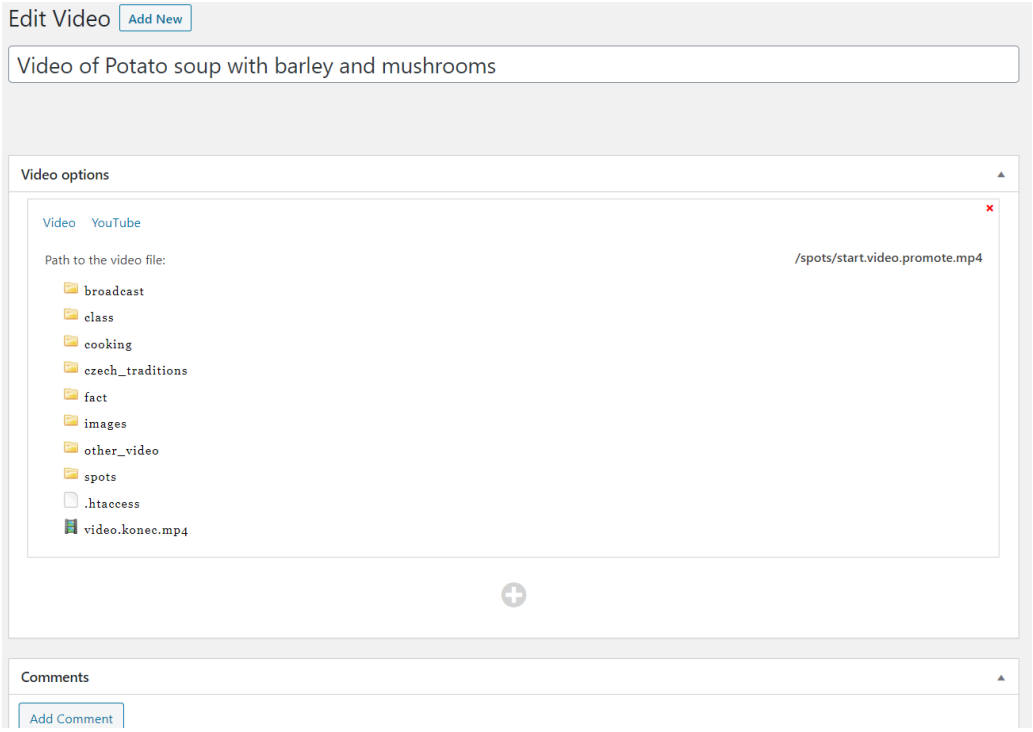
Kód 5.5: Editor seznamu 3/3 - Načtení uložených panelů

Následuje část kódu která zajišťuje vykreslení editoru z obrázku 5.2. Editor je realizovaný jako okno se dvěma záložkami. Na poslední okno je možné přes plusové tlačítko připojit další identické okno. Do každého okna je možné nastavit konkrétní video a tak je uživatel schopen vytvořit libovolně dlouhý seznam videí.
Během editování je za pomoci javascriptu dynamicky generovaný formulář, jehož obsah je po uložení příspěvku odeslán metodou POST na server. Data jsou poté uložena do patřičné databázové tabulky.
Pokud uživatel otevře příspěvek s videem, který již dříve editoval, je třeba vygenerovat dynamický formulář do takové podoby, která odpovídá uloženým datům. K tomu slouží poslední část kódu uvedeného výše. Je v něm v cyklu volaná funkce wp_wplyr_print_editor($sources$[i], $types$[i]) , která zajišťuje vložení minulých oken editoru.
Uživatel tedy může vytvořit nové video a nastavit mu požadované vlastnosti. Toto nastavení je následně uloženo do databáze. Dalším krokem je umožnění vložení vytvořeného videa do příspěvku s obsahem, který je zobrazen návštěvníkovi webových stránek. Tuto akci realizujeme pomocí funkce nástroje Wordpress, které se říká shortcode.

5.1.3 Zástupné kódy

Zástupný kódy neboli shortcodes byly do nástroje Wordpress uvedeny s verzí 2.5 a od té doby se zapsal jako jedna z nejužitečnějších funkcí. Průměrný uživatel je díky nim schopen publikovat dynamický obsah, aniž by k tomu potřeboval znalost jakéhokoliv programovacího jazyka [17].
Při vložení shortcode do těla příspěvku nebo stránky, vytvořené v nástroji Wordpress, je jeho text nahrazen nějakým jiným obsahem. Jinými slovy, při použití shortcode říkáme nástroji Wordpress, že má vyhledat všechny hranaté závorky a místo nich na stránce zobrazit konkrétní dynamický obsah, vygenerovaný specifikovanou funkcí [8].

```
function wp_wplyr_video_shortcode($id)
{
    extract(shortcode_atts(array('id' => 'id'), $id));
    //shortcode HTML element
    $html = '<div class="wplyr_container">
            <video controls crossorigin playsinline
            class="wplyr_player wplyr_video_' . $id . '">
            </video></div>
    <script>';
```



Obrázek 5.2: Editor seznamu videí

```
    //passing post data from backend to frontend
    for ($i = 1; $i < sizeof(unserialize(get_post_meta($id)["
    _wp_wplyr_video_source"][0])); $i++) {
        $path = unserialize(get_post_meta($id)["_wp_wplyr_video_source"
    ][0])[$i];
        if (empty($path)) {
            $source = '';
        } else {
            $source = wplyr_video_url. $path;
        }
        $type = unserialize(get_post_meta($id)["_wp_wplyr_video_type"
    ][0])[$i];
        $html .= 'wp_wplyr_add_source(' . $id . ',"' . $source . '","' .
    $type . '");';
    }
    $html .= '</script>';
    return $html;
}

add_shortcode("wplyr", "wp_wplyr_video_shortcode");
```

Kód 5.6: Nastavení zástupného kódu

Funkce deklarovaná v 5.6 prohledá publikovaný příspěvek a pokud v něm na-

jde shortcode ve formátu [wplyr id=123], tak ho nahradí html přehrávačem. Zároveň přidá do javascript kontextu cesty ke zdrojům videa s identifikačním číslem 123.
Způsob, kterým je shortcode implementován v nástroji Wordpress by umožňoval i rozsáhlejší využití vstupních parametrů použitého shortcode. Vstupem by místo identifikačního čísla mohl být kompletní seznam videí a ostatní nastavení spojené s konkrétním videem. Tento přístup by ale určitě působil nezkušeným uživatelům potíže. Ukládání pouze identifikačního čísla nám navíc umožňuje v budoucnu měnit obsah, který se daným shortcode vykresluje. Například, při změně úvodního videa v seznamu videí není následně nutné editovat všechny příspěvky, ve kterých je daný shortcode použit.
V tuto chvíli je možné vkládat vytvořená video do jakékoliv stránky vytvořené v nástroji Wordpress. Uživatel si ale musí pamatovat identifikační číslo videa, které chce do svého příspěvku vložit. Pro lepší uživatelský komfort byla vytvořena komponenta seznamu videí, ve které může uživatel procházet již vytvořené videa a následně přes stisk tlačítka vložit shortcode s korektním identifikačním číslem do textového pole příspěvku [24].
Při registraci komponenty na poslední řádce je také nutné určit, na jakých stránkách se bude zobrazovat. Použitím typu ‘post’ je zaručeno, že se komponenta seznamu videí zobrazí ve všech příspěvcích [28].
Následující kód registruje zmíněnou komponentu. Za zmínku stojí úsek s obsluhou stisku tlačítka psaným v programovacím jazyce JavaScript. Tato obsluha zajistí, že dojde k vložení textu pouze do textového pole určeného pro obsah příspěvku. Nejprve se opět registrují skripty a kaskádové styly pro korektní funkci komponenty. Pro zobrazení tabulky videí je použit skript s názvem DataTables, který obohacuje základní HTML tabulku novými funkcemi.
Po načtení skriptů v první části je třeba načíst data, které má komponenta zobrazovat. Tabulka bude zobrazovat seznam vytvořených videí, které je možné do příspěvku vložit. Je tedy třeba získat všechny příspěvky typu wp_wplyr_videos. Toto načtení je realizováno pomocí WordPress funkce get_posts() definované v 5.7.

```
function wp_wplyr_post_picker_html($post)
{
    //script declaration
    ...
    //fetching all relevant posts with data
    $args = array(
        'post_type' => 'wp_wplyr_videos',
        'posts_per_page' => -1,
        'numberposts' => -1
```

```
    );
    $posts = get_posts($args);
    ?>
```

Kód 5.7: Vyhledávač videí 1/4 - Typ příspěvku videa

V proměnné post je teď uložen seznam videí. Následující kód vykreslí záhlaví tabulky a do této tabulky pak přidá všechny záznamy z proměnné post. To je ralizováno iterací v 5.8.

```
    //HTML table header elements
    ...
    //HTML table content echo
    <?php
    foreach ($posts as $key => $post) {
        $posts[$key]->acf = get_post_meta($post->ID);
    ?>
    <tr>
        <td> <?php echo $post->post_date ?> </td>
        <td> <?php echo $post->post_title ?> </td>
        <td> <b>[wplyr id=<?php echo $post->ID ?>]</b> </td>
        <td> <button class="button" type="button"
            onclick="insert_shortcode_into_editor(
                <?php echo $post->ID ?>
            )">
            Insert shortcode</button> </td>
    </tr>
```

Kód 5.8: Vyhledávač videí 2/4 - Tělo tabulky

Každý záznam má také na konci řádky tlačítko, které po stisknutí vytvoří zástupnou zkratku v textovém poli WordPress editoru. Obsluha stisku tlačítka je vidět v 5.9.

```
<script>
    function insert_shortcode_into_editor(video_id) {
        if (tinyMCE && tinyMCE.activeEditor) {
            tinymce.activeEditor.execCommand('mceInsertContent', false,
            "[wplyr id=" + video_id + "]"
);}}</script>
```

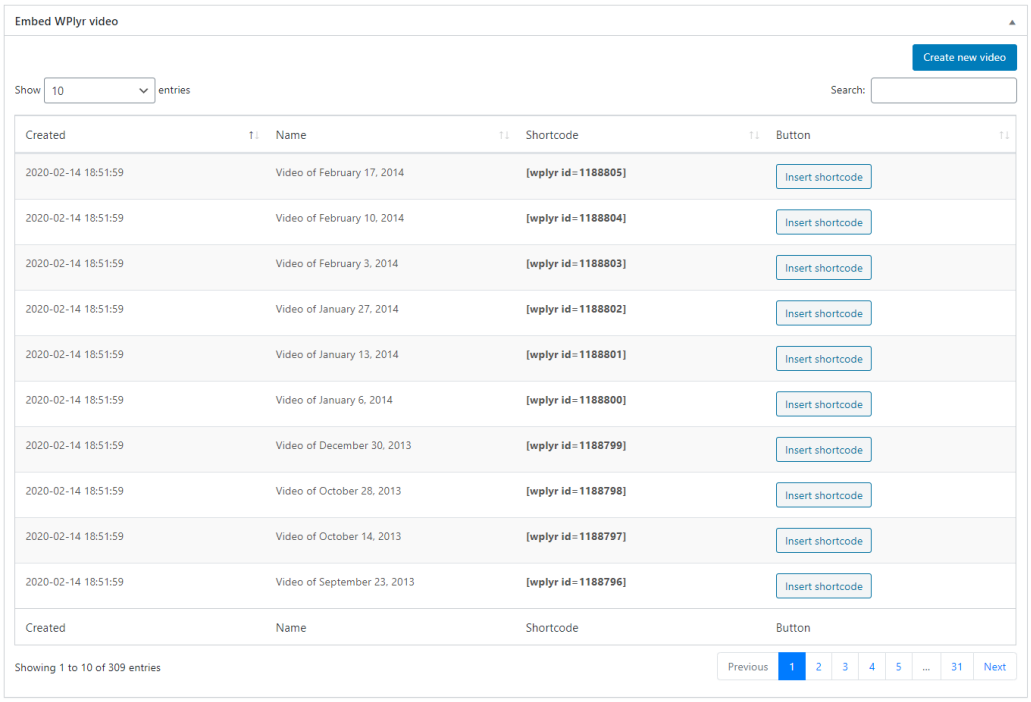
Kód 5.9: Vyhledávač videí 3/4 - Vložení zástupného kódu

Po vytvoření tabulky je třeba iniciovat plugin DataTable za pomoci kódu 5.10, který u HTML tabulky vytvoří stránkování a možnost jednoduchého prohledávání. Výsledný vzhled vytvořené komponenty je vidět na obrázku 5.3.

```
$(document).ready(function() {$('#dataTable').DataTable();});
}
```

Kód 5.10: Vyhledávač videí 4/4 - Inicializace tabulky

Uživatel tedy může vytvářet nové videa a umisťovat je na své příspěvky skrze funkce poskytované nástrojem Wordpress. Následující kapitola se bude věnovat realizaci samotného přehrávání.



Obrázek 5.3: Vyhledávač videí

5.2 Přehrávač

Jak už bylo zmíněno na začátku předchozí kapitoly, tento projekt se skládá ze dvou částí. První část zajišťuje komunikaci s nástrojem Wordpress a druhá část se věnuje samotnému přehrávání videa.

Vnitřní projekt se nachází ve složce player. Tato složka obsahuje několik podsložek. Důležitou složkou je složka dist, která pak obsahuje výsledný přeložený script. Tento skript je namapovaný do obalového projektu a aplikuje se na html prvek videa. Pro samotné nasazení není třeba vnitřnímu projektu rozumět. Pokud by však uživatel chtěl provést nějaké logické nebo vizuální změny, je nutné následně celý projekt znovu sestavit. Během tohoto procesu dojde k vygenerování nového souboru ve složce dist. Tento postup je běžný v jakémkoliv moderním projektu, který využívá programovací jazyk JavaScript.

Vnitřní projekt využívá pro kontrolu závislostí balíčkovací manažer npm a

pro sestavení JavaScriptovou knihovnu Gulp. V kořenové složce se tedy nachází konfigurační soubor package.json, který obsahuje seznam veškerých závislostí a pak také soubor gulpfile.js, ve kterém je definován způsob sestavení.
Veškeré zdrojové soubory se pak nachází ve složce src. Tato složka obsahuje dvě další podsložky. První s názvem sass a druhou s názvem js.

5.3 Souborová struktura

Složka sass tedy obsahuje soubory definující vizuální vzhled přehrávače videí. Tyto soubory jsou poté použity při překladu pro vygenerování *.css souborů ve složce dist.
Z názvů jednotlivých složek lze snadno odhadnout jakou část vizuálního vzhledu ovlivní. Například pro úpravu barev může vývojář otevřít složku settings a upravit v ní soubor colors.scss. Pokud jde pouze o nějakou menší úpravu vzhledu, která má za úkol pouze překrýt nějaké dosavadní chování, může vývojář využít soubor custom.scss. Ten se při překladu vkládá jako poslední a tak jeho nastavení přepíší ty dosavadní.
Druhá složka má název js a obsahuje jediný soubor s názvem player.js. Tento soubor obsahuje veškeré funkce, které realizují přehrávání videí. Pro samotné přehrávání videí je použit populární open-source přehrávač s názvem Plyr. Na první řádce 5.11 se nachází jeho import. Je zde také importována knihovna alertify, která zapouzdřuje jednoduché vyskakovací okna. Je použita pro informování uživatele o selhání inicializace video přehrávače [27].

```
import Plyr from 'plyr';
import alertify from 'alertifyjs';
```

Kód 5.11: Přehrávač 1/6 - Vložení knihoven

Následuje inicializace přehrávače. Proces stažení, načtení a provedení skriptu obsahujícího implementaci přehrávače může trvat poměrně dlouhou dobu. Alespoň ve srovnání s ostatními skripty, které zprostředkovávají nebo zpříjemňují například uživatelské rozhraní a jsou tedy pro uživatelský komfort důležitější. Na zařízeních s nízkým výpočetním výkonem, jako jsou například mobilní telefony, se může tato doba ještě prodloužit. Z tohoto důvodu je samotná inicializace obalena kusem kódu 5.12.

```
(() => {
    document.addEventListener('DOMContentLoaded', () => {
        try {
...
```

```
        } catch (error) {
alertify.error(
"<b>Video playback might not be working. Please use more modern browser
    .</b></br>" + error
);
            throw error;
        }
    });

})();
```

Kód 5.12: Přehrávač 2/6 - Obal inicializace

Ten zajistí, že se inicializace provede, až po načtení a vykreslení celého obsahu stránky. Zároveň je zobrazena chybová hláška, pokud by došlo během inicializace k výjimce. Jelikož byla převážná většina výjimek způsobena zastaralým webovým prohlížečem, je hláška zakončena slovy "Please use more modern browser."
Následuje několik funkcí, které slouží k vylepšení uživatelského komfortu a opravy, které ještě nebyly zaneseny do nové verze přehrávače. Tyto funkce byly převážně převzaty ze stránek projektu přehrávače na stránkách Github [27].
To čím se má implementace liší od té, kterou doporučují autoři přehrávače je možnost zobrazení více přehrávačů na jedné stránce. V příkladu uvedeném autory je vytvořen HTML element s identifikátorem "wplyr_player". Protože musí být na webové stránce každý identifikátor unikátní, musel jsem pro svoje rozšíření použít stejně pojmenovanou třídu. Kód 5.13 tedy definuje identifikátor třídy, definující div elementy, které jsou po inicializaci modifikovány na samotné okno přehrávače videí. Tento selektor je následně použit pro vybrání všech elementů na stránce a přiřazení nové instance přehrávače, každému z nich.

```
const selector = '.wplyr_player';
const players = Array.from(document.querySelectorAll(selector)).map(p =>
    new Plyr(p, {debug: true, controls: ['play-large', 'play', 'progress'
    , 'current-time', 'mute', 'volume','airplay', 'fullscreen']}));
```

Kód 5.13: Přehrávač 3/6 - Inicializace

Vstupní parametry konstruktoru jsou HTML element pro modifikaci a pole nastavení. Toto pole může být velmi rozsáhlé. Přehrávač Plyr totiž umožňuje, skrze tato nastavení,velké množství úprav.
Po inicializaci jednotlivých přehrávačů je potřeba každému z nich přiřadit seznam videí, které má přehrávat. Tento seznam je definován identifikačním číslem, které je mu přiřazeno při uložení do Wordpress tabulky v databázi. Při vykreslování stránky nahrazuje nástroj Wordpress veškeré zástupné kódy

novým HTML obsahem. V našem případě vykreslí prázdné elementy typu div, které mají přiřazenou třídu wplyr_player, díky které jsme přehrávače iniciovali. Zároveň jim je ale přiřazena i třída wplyr_video_*, kde je místo hvězdičky dosazen právě identifikátor seznamu videí. Právě iniciovaným přehrávačům je tedy nutné tento identifikátor předat. K tomu slouží kód 5.14.

```
for (let i = 0; i < players.length; i++) {
                const player = players[i];
                player.sourceIndex = -1;
                for (let j = 0; j < player.elements.original.classList.
    length; j++) {
                    var classString = player.elements.original.classList[
    j];
                    if (/wplyr_video_\d+/.test(classString)) {
                        var string = classString.replace("wplyr_video_",
    "");
                        player.videoId = parseInt(string, 10);
                    }
                }
                nextSource(player)();
                player.on('ended', nextSource(player));
                Player.controls;
}
```

Kód 5.14: Přehrávač 4/6 - Nastavení videí

Index přehrávaného videa je nastaven na hodnotu -1. Pak dojde ke kontrole všech tříd a pokud třída splňuje regulární výraz, je z hodnoty vybrán řetězec zakončující její název. Tento řetězec je převeden na číslo v desítkové soustavě a uložen do objektu přehrávače pod názvem videoId. Poté dojde k spuštění funkce pro přehrání následujícího videa. Tato funkce zvětší index přehrávaného videa z hodnoty -1 na hodnotu 0 a dojde k přehrání prvního videa v seznamu. Předposlední řádka zaregistruje funkci pro přehrání následujícího videa jako událost, která se spustí poté co přehrávač vyšle signál, že došlo k ukončení přehrávání. Přehrávání jednotlivých videí v seznamu, je tedy realizováno ve funkci nextSource z 5.15.

```
function nextSource(player) {
    return function() {
        if (player.currentTime > 0 || player.sourceIndex == -1) {
            if (typeof window.videoSourceMap[player.videoId] !== '
    undefined' &&
            window.videoSourceMap[player.videoId].length > player.
    sourceIndex) {
                do {player.sourceIndex++;}
                while (typeof window.videoSourceMap[player.videoId][
    player.sourceIndex] !== 'undefined' && isEmpty(window.videoSourceMap[
```

```
player.videoId][player.sourceIndex].source) && window.videoSourceMap[
player.videoId].length > player.sourceIndex)
                if (window.videoSourceMap[player.videoId].length <=
player.sourceIndex) {
                    player.sourceIndex = 0;
                }

                        ... //Nastaveni konkretniho zdroje videa

                if (player.sourceIndex != 0) {
                    var playPromise = player.play();
                    if (playPromise !== undefined) {
                        playPromise.then(function() {}).catch(
function(error) {
                            alertify.error("<b>Video playback might
not be working. "+
                                           "Please use more modern
browser.</b></br>" + error);
                            });
                        }
                    }
                }
            }
        }
```

Kód 5.15: Přehrávač 5/6 - Posouvání zdrojů videa

Samotné přepnutí zdroje videa je jen vybrání korektního prvku z objektu videoSourceMap a nastavení těchto hodnot přehrávači v 5.16

```
                        var source = window.videoSourceMap[player.
videoId][player.sourceIndex].source;
                        var type = window.videoSourceMap[player.
videoId][player.sourceIndex].type
                        var sources = [];
                        if (type === 'youtube') {sources = [{src:
source, provider: 'youtube'}];}
                        else if (type === 'video') {sources = [{src:
source, type: 'video/mp4'}];}
                        player.source = {type: 'video', sources:
sources}
```

Kód 5.16: Přehrávač 6/6 - Nastavení konkrétního zdroje videa

Tato funkce vrací další funkci, aby mohlo dojít k její registraci na událost ukončení přehrávání. Jelikož na různých prohlížečích docházelo k falešnému spouštění událostí o ukončení videa, obsahuje funkce nextSource velké množství podmínek, které zamezují tomu, aby tyto události ovlivnily samotné

přehrávání videa.
První podmínka zajišťuje, že dojde k přepnutí videa jen pokud již došlo k částečnému přehrání, ale zároveň se nejedná o první video ve frontě. Tato podmínka se zde nachází, protože prohlížeč Edge občas nestihl načíst první video včas a došlo k přepnutí na následující video ve frontě, aniž by došlo ke spuštění přehrávání.
Po této kontrole dojde k zvýšení indexu přehrávaného videa. Pokud index přesahuje velikost fronty, je nastaven zpět na 0. Dvojice hodnot Index videa a identifikátor konkrétní fronty videí definuje objekt v poli videoSourceMap. Tento objekt obsahuje cestu k videu a typ zdroje. Typy zdroje jsou dva. První typ je definován řetězcem video. Cesta je pak plnohodnotný odkaz na soubor s videem. Druhý typ je definován řetězcem youtube. V tomto případě je cesta odkaz na stránku videa na serveru YouTube.
Funkce je zakončena podmínkou, která zajistí spuštění právě načteného videa. K procesu spuštění může dojít se zpožděním a nebo k němu nemusí dojít vůbec. I když by k této chybě dojít nemělo, pokud se tak stane a k automatickému přehrání nedojde, je uživateli opět zobrazena generická chybová hláška.

5.4 Balíčkovací systém

Výsledný přehrávač byl umístěn do balíčkovacího systému NPM. Jedná se o výchozí balíčkovací systém nástroje Node.js. Proces publikování veřejného uživatelského balíčku je jednoduchý. Nejprve se bylo třeba registrovat. Skrze příkazovou řádku se poté lze přihlásit pomocí příkazu

```
npm login
```

Poté je třeba vytvořit strukturu potřebnou pro vytvoření balíčku. To lze realizovat příkazem

```
npm init
```

Po vyplnění potřebných informací vznikne ve složce soubor package.json. Po umístění zdrojových souborů do složky s tímto souborem je možné výsledný balíček publikovat pomocí příkazu

```
npm publish --access public
```

Balíček je uložen do veřejného repositáře, ze kterého si ho mohou ostatní uživatelé instalovat příkazem

```
npm install plyr_showplayer
```

Pro použití je třeba zachovat DOM strukturu elementů a tříd přehrávače, ale z toho důvodu je v balíčku přidán i index.html a v popisu je odkaz na stránky GitHub s funkční demonstrací.

6 Testování

Předchozí kapitoly se věnovali analýze, návrhu a následné implementaci pluginu pro přehrávání videí. Tato kapitola rozděluje širokou oblast testování do několika konkrétních kategorií, které mají největší význam pro webové aplikace. V každé této kategorii se práce věnuje krátkému popisu a pak následně způsobu, kterým byly tyto principy aplikovány na otestování vytvořeného pluginu.
Přesto, že je testování často podceňované je to velmi důležitá část vývoje softwaru. Šance vzniku chyby s každou řádkou narůstá. Stejně tak narůstá i cena opravy čím pozdější je moment, ve kterém je chyba během vývojového procesu objevena. Pokud se jedná o agilní vývoj, tak je proces vývoje a testování spojen ještě těsněji [15].

6.1 Jednotkové testy

Jednotkové testy slouží k ověření fungování a korektní implementace systému. Testovaná jednotka je určitá část programu. Může to být funkce, procedura nebo například proměnná [18].
Přestože bývají jednotkové testy obvykle automatizované, tento projekt byl otestován převážně manuálně. Zvolil jsem tento způsob převážně kvůli různorodosti programovacích jazyků. Při vývoji serverové části kódu, která obsluhovala vytváření videí v administrační zóně systému Wordpress, byly vytvořeny mockové vstupy pro komponenty editoru a vyhledávače videí. Tímto postupem často rovnou probíhal i vývoj pluginu. Nejprve byl vytvořen záznam v databázi. Následný kód ho transformoval na korektní formulář. Ve chvíli kdy byl výstup korektní pro daný vstup, byly vytvořeny a načteny i ostatní vstupy a zkontrolováno zobrazení nových dat.
Nejvíce byl testován samotný přehrávač, který je naprogramován v programovacím jazyce JavaScript. Videa se načítají z globální proměnné videoMap a tak pro mockování jiného vstupu stačilo tento objekt iniciovat na jinou hodnotu. Tyto rychlé úpravy je možné provádět rovnou z konzole webového prohlížeče a tak bylo odladění chyb odhalených pomocí jednotkových testů velmi jednoduché. Absence frameworku určeného pro automatizované jednotkové testování, byla nahrazena extenzivním testováním funkčnosti.

6.2 Testování funkčnosti

Jedná se o nejzákladnější formu testů. Jeho cílem je ověřit zda je aplikace funkčně korektní. Během funkčního testování ověřuje databázová spojení, všechny odkazy, cookies a funkčnost tlačítek. Mělo by být prováděno již v průběhu vývoje. V podstatě spočívá v provedení série kroků, buď automatizovaným skriptem a nebo lidským testerem. Po získání testovacích výsledků dojde k porovnání s očekávanými hodnotami. Je provedeno několikrát s rozdílnými vstupy a pokud je přesnost výsledků pro všechny případy stejná, lze aplikaci prohlásit za funkčně korektní.
Hlavní testovací scénář bylo otevření stránky s vloženým videem. Spuštění videa. Změna hlasitosti. Posunutí časové osy na konec prvního videa. Kontrola ukončení přehrávání prvního videa a spuštění přehrávání videa druhého. Tento rychlý test často odhalil základní chyby a umožnil tak určit zda plugin funguje správně. Mezi nejčastější chyby patřilo, nezobrazení přehrávače, zobrazení více přehrávačů, selhání přehrání videa a selhání přepnutí videa. Nezobrazení přehrávače bylo nejčastěji způsobeno nekorektním vložením zástupného kódu, vypnutím pluginu, nebo chybou v umístění. K zobrazení více přehrávačů docházelo především při nasazování na server CzechAmericanTV. Ten totiž pro přehrávání používal již vestavěný přehrávač a ten bylo nutný při nasazení vypínat. Při testování se ale osvědčilo mít zapnuté oba přehrávače, kvůli porovnání jejich obsahu. Při nasazování tedy stačilo zapomenout na odstranění původního přehrávače a tak k této chybě docházelo poměrně často při rychlém přepínání způsobu přehrávání. Selhání přehrání videa je nejčastěji způsobeno špatnou cestou k videu. V tu chvíli je dobré zkontrolovat vývojářské nástroje v prohlížeči. V výstupu je často zobrazena chybová hláška a nebo je vidět chybějící video v seznamu stažených souborů. Pokud dojde k nekorektnímu přepnutí videa je způsobeno poškozeným datovým modelem a nebo nekorektním spuštění událostí na úrovni skriptu. Každý prohlížeč tyto události implementuje jiným způsobem a tak se občas stalo, že dojde například k přeskočení videa, které trvalo dlouho načíst. Na tuto chybu trvalo dlouho přijít, protože se projevovala jen v naprostém minimu případů a chovala se nedeterministicky. Výsledný plugin však již obsahuje všechny opravy, takže k těmto chybám nedochází [12].
Testovací scénář definovaný v předchozím odstavci byl proveden po každé větší změně. Před nasazením s ním byla ověřena podpora na následujících prohlížečích.

- Google Chrome 81
- Opera 65

- Internet Explorer 11
- Microsoft Edge 44
- Mozzila Firefox 68
- Safari 13

Při prvotních testech bylo odhaleno, že prohlížeč Microsoft Edge implementuje jiné spouštění událostí. Při načtení videa okamžitě odeslal informaci o jeho ukončení. Efekt této chyby byl takový, že celý přehrávač začal v nekonečné smyčce načítat následující video, až dokud se některé nenačetlo. K přehrání tedy došlo, ale přehrávač několikrát problikl. Tuto chybu jsem opravil tak, že k přepnutí videa nedojde pokud nebyla alespoň část již přehrána.
Při přehrávání videa v přehrávači Opera je zobrazena ikona PIP. Po jejím stisknutí je přehrávané video zobrazeno v módu picture in picture. Tomuto chování nelze zabránit, protože je implementováno na úrovni prohlížeče.
Kromě těchto dvou odchylek se přehrávač choval ve všech testovaných prohlížečích stejně.

6.3 Testování použitelnosti

Testování použitelnosti by mělo být prováděno s externími testery, kteří simulují průřez očekávanými cílovými uživateli.
Výsledný plugin byl prezentován skupině testovacích uživatelů. Bohužel žádný z těchto uživatelů neměl předchozí znalosti systému Wordpress a tak jim bylo třeba základní použití pluginu třeba vysvětlit. Technicky zdatnější uživatelé byli schopni po vysvětlení systému vytvořit a publikovat video. Ti ostatní však měli s tímto procesem velké problémy. Z těchto testů lze usoudit, že vytváření videí není intuitivní. Uživatel pro správné pochopení funkce pluginu potřebuje krátké zaškolení a nebo alespoň uživatelskou příručku. Komplexita uživatelského rozhraní je ale způsobena především systémem Wordpress. A jelikož je plugin určen pro uživatele, kteří systém Wordpress využívají na každodenní bázi, tak by v reálných situacích k těmto problémům docházet nemělo.
Další závěr, který z tohoto testování vyplynul byl, že by pluginu značně prospělo implementování Gutenberg bloku. Ten by totiž proces vkládání nového videa značně zjednodušil. Uživatelé měli totiž největší problém s vkládáním vytvořených videí do jednotlivých článků. Pokud by šla veškerá nastavení realizovat na úrovni bloku, bylo by rozhraní značně zjednodušeno.

Při finálním nasazení byla aplikace představena i administrátorovi stránek CzechAmericanTV. Na základně jeho připomínek byla část uživatelského rozhraní ještě dodatečně zjednodušena. Došlo k odstranění možnosti uložit rozpracované videa jako návrh. Tato možnost totiž postrádala smysl. Video uložené jako návrh nelze nikam vložit a díky tomu, že není třeba publikované videa nikam vkládat, je lepší rozpracované video při ukládání rovnou publikovat.

6.4 Testování rozhraní

Tento typ testování je důležitý obzvlášť u webových aplikací. Testuje komunikační vrstvu mezi serverem a webovou aplikací. Mělo by také ověřit, zda jsou správně zpracovány různé výpadky.
Ve výsledném pluginu se o veškerou komunikaci se serverem nebo databází stará rozhraní systému Wordpress. Nebylo tedy třeba testovat komunikační vrstvu. To co bylo třeba otestovat bylo chování výpadků. Jelikož žádný ze skriptů není umístěn na serveru třetí strany, může dojít jen k výpadku celé webové stránky. Při testování rozhraní bylo otestováno především chování výpadku jednotlivých videí. Jsou uložena v databázi jen jako cesty a tak nelze zajistit, že se videa na dané cestě budou nacházet věčně. Pokud dojde k poškození odkazu na přehrávané videa, dojde při přehrávání k jeho přeskočení. Uživatel je navíc o této skutečnosti informován pomocí neblokující chybové hlášky.

6.5 Testování výkonu

Během testování výkonu by mělo dojít k otestování aplikace při větším množství současně připojených uživatelů a nebo chování aplikace při nedostatečném množství systémových zdrojů.
Otestování výkonu bylo provedeno za pomoci vývojářských nástrojů integrovaných do prohlížeče Google Chrome. Umožňují omezit rychlost stahování nebo výkonnost zařízení. I při šestinásobném zpomalení výkonu a nastavení rychlosti připojení na pomalé 3G se přehrávač načetl do 3 vteřin. Celé webové stránky v systému Wordpress se při takovém nastavení načítají okolo 30 vteřin a tak je zdržení způsobené načítáním přehrávače zanedbatelné.
Výkonnostní omezení způsobené připojením více uživatelů v tomto případě nezávisí na rychlosti pluginu nebo přehrávače, ale na výkonu serveru, který

tyto uživatele obsluhuje. Jelikož dochází k obsluze přehrávání na straně uživatele je možné plugin plně škálovat zvyšování výkonu webového hostingu.

6.6 Testování kompatibility

V současné době se webové aplikace zobrazují na velkém množství zařízení s různými obrazovkami. Je dobré zajistit, aby se aplikace zobrazovala na všech těchto zařízeních stejně. Během testování kompatibility dochází k ověření chování aplikace na různých zařízeních, ale i rozdílných operačních systémech.
Jako možná nejdůležitější část testování, bylo testování kompatibility. Osobně jsem byl schopný otestovat přehrávání videa na poměrně velkém množství zařízení a operačních systémech. Testování bylo provedeno na následujících zařízeních:

- Desktop
  - Windows 10
  - Linux
  - MacOS
- Mobilní zařízení
  - Android 4
  - Android 7
  - iOS 12

Největší problém bylo testování desktopové verze prohlížeče Safari. Ta je sice podporována na systému Windows, ale jen do verze 3. Současná verze prohlížeče Safari, která je používána na drtivé většině zařízení s operačním systémem MacOS, s touto starou verzí již nemá nic společného. Webový prohlížeč Safari je navíc úzce spjatý s operačním systémem na kterém běží. Současná nejnovější verze MacOS Sierra obsahuje prohlížeč verze 12.1.2 a pokud by uživatel chtěl otestovat jinou verzi prohlížeče, musel by přeinstalovat celý operační systém.
V testování kompatibility, také velice pomohla webová stránka `www.browsershots.org`. Jedná se o službu, která po vložení URL postupně spustí až 40 webových prohlížečů na různých zařízeních. Po pár vteřinách pořídí snímek obrazovky a ten pak zobrazí uživateli. Lze tak rychle zjistit, s jistou mírou nepřesnosti,

na kterých prohlížečích přehrávač selhává. Některá zařízení jsou totiž pomalá a tak dojde k pořízení snímku předtím, než dojde ke kompletnímu načtení všech zdrojů.

6.7 Vzdálené ukládání chybových hlášení

V prvotních fázích projektu bylo vynaloženo úsilí pro vytvoření takzvané Sentry. Jedná se v podstatě o hlídače, který ve svém kontextu spouští webovou aplikaci. Pokud dojde při běhu k nějaké výjimce nebo neočekávané chybě, tak dojde k odeslání chybového hlášení na vzdálený server. Později se tyto nahromaděné chyby dají na serveru zkontrolovat a opravit. Je to užitečný nástroj při vývoji aplikací v programovacím jazyce JavaScript. Nejprve byl pro kontrolu chyb použit balíček Sentry.io a následně podobná alternativa s názvem Raven. Obě služby jsou však placené. Nakonec bylo testování provedeno lokálně a tak jejich využití postrádalo smysl.

7 Nasazení

Nasazení vytvořeného pluginu na server CzechAmericanTV je snadné. Stačilo zkopírovat nový plugin do již existující složky s pluginy a poté plugin aktivovat skrze administrační centrum. Tyto stránky však již několik let využívají jiný způsob přehrávání videí a tak obsahují již poměrně velký katalog stránek s videi. Odkaz na video je uložen v metadatech konkrétní stránky. Ve Wordpress tématu, které stránky využívají je pak naprogramována logika, která nejprve přehraje soubor videa s úvodní znělkou, následuje video z metadat a po něm se přehraje koncové videa s titulky a poděkováním.
Tímto způsobem jsou vytvářeny na webových stránkách různé pořady. V současné době jsou na stránkách čtyři typy pořadů. Každý pořad má vlastní typ příspěvku. Klíčový slova, pod kterými jsou pořady uloženy jsou class, cooking, fact a broadcast. Pořady typy broadcast jsou jediné, které mají úvodní video. Všechny pořady mají pak jedno obsahové video a jedno závěrečné.
Přestože by bylo migraci možné realizovat pouze pomocí databázového skriptu v jazyce SQL. Pravděpodobně bude třeba data ještě nějak transformovat a tak jsem se rozhodl migrační skript udělat v jazyce Java. Mám v tomto jazyce největší zkušenosti a tak šlo čistě o osobní preferenci.

7.1 Web hosting

Pro vývoj byla použita lokálně uložená Wordpress databáze. Před nasazením však bylo nutné plugin řádně otestovat a nejlépe rovnou na stránkách CzechAmericanTV, které budou plugin definitivně používat. Majitel webu mi poskytl zálohu, ze které jsem byl schopný celé webové stránky lokálně obnovit.
Během testovací části se ukázalo, že aby plugin mohl pohodlně otestovat i někdo jiný, bude třeba nasadit web CzechAmericanTV na veřejně přístupný hosting. To se nakonec ukázalo problematické. Na internetu se nachází obrovské množství webových hostingů, které tvrdí, že zvládnou hostovat redakční systém Wordpress bez žádných poplatků. Jenže realita je takové, že každá z těchto stránek se snaží pouze získat uživatele a po době přibližně jednoho týdne hosting zablokují. Občas je to z důvodu směšně nízkého maximálního povoleného počtu návštěvníků za týden, jindy vyčerpaný počet uzlů na Linuxovém FTP, nebo spotřebování určitých kreditů. Stránky CzechA-

mericanTV navíc i bez většiny obrázků a videí zabírají několik GB. Navíc hostovat tyto soubory z originálního webu bylo kvůli nastavení systému Wordpress obtížné. Nejkvalitnější hostovací služba, která dokáže hostovat systém Wordpress bez jakýchkoliv poplatků byla stránka www.000webhost.com. Kvůli pomalé rychlosti a obrovským komplikacím s hostováním videí na externím FTP serveru byl nakonec zakoupen hosting na českém serveru Wedos. Tento nový server byl zaregistrován pod doménou http://www.testcatv.ga/ a byl během testování extenzivně používán jak mnou, tak ochotnými členy týmu CzechAmericanTV, kteří pomohli s otestováním.

7.2 Migrace dat

Při spuštění skriptu je třeba vytvořit databázové spojení. Za zmínku stojí informace, že při migraci došlo několikrát k rozbití speciálních znaků. Specifikováním způsobu kódování v 7.1 bylo možné tento problém vyřešit.

```
public static void main(String[] args) {
      try (
         Connection conn = DriverManager.getConnection(
               "jdbc:mysql://localhost:3306/migration?characterEncoding=
   utf8",
               "root", "");
         Statement stmt = conn.createStatement();
      ) {
```

Kód 7.1: Migrace 1/12 - Připojení do databáze

Po vytvoření databázového spojení je vytvořen dotaz pro získání záznamů všech příspěvků z tabulky wp_posts 7.2, které odpovídají záznamům jednotlivých pořadů.

```
         String strSelect = "select * from wp_posts where post_type like
   'broadcast' OR post_type like 'cooking' OR post_type like 'class'  OR
    post_type like 'fact'";
         ResultSet rset = stmt.executeQuery(strSelect);
```

Kód 7.2: Migrace 2/12 - Výběr příspěvků s videi

Po provedení SQL dotazu je třeba projít seznamem výsledků 7.3.

```
         int rowCount = 0;
         while(rset.next()) {
```

Kód 7.3: Migrace 3/12 - Iterace přes příspěvky

Pro každý pořad zjistím jeho identifikátor, název a typ. Původní cesta k videu je uložena v tabulce wp_postmeta. Záznam je spárován s pořadem přes jeho identifikátor a je definován hodnotou meta_key = 'video'. Každý příspěvek by měl mít přiřazen jen jeden záznam s informaci o videu v tabulce wp_postmeta. Pokud to tak není, tak se vybere ten poslední.

```
        Statement stmt2 = conn.createStatement();
        int    postId   = rset.getInt("ID");
        String    postName   = rset.getString("post_title");
        String    postType   = rset.getString("post_type");
           String query1 = "SELECT * FROM wp_postmeta WHERE post_id = "+
  postId+" AND meta_key LIKE 'video'";
           ResultSet query1ResultSet = stmt2.executeQuery(query1);
           int rowCount2 = 0;
           long newVideoPostId = -1;
           while(query1ResultSet.next()) {
```

Kód 7.4: Migrace 4/12 - Načtení hodnot

V záznamu z tabulky wp_metavalue je tedy pod klíčem meta_value uložen název videa v ukázce 7.5.

```
              String videoPath = query1ResultSet.getString("meta_value")
  ;
              if(videoPath.isEmpty()) {
               continue;
              }
```

Kód 7.5: Migrace 5/12 - Načtení názvu původního videa

Z názvu odstraním příponu video souboru za pomoci kódu 7.6.

```
              String trimmedVideoPath = videoPath;
              while( trimmedVideoPath.lastIndexOf('.') != -1 &&
  trimmedVideoPath.lastIndexOf('.') == trimmedVideoPath.length() - 4 )
  {
               trimmedVideoPath = trimmedVideoPath.substring(0,
  trimmedVideoPath.lastIndexOf('.'));
              }
```

Kód 7.6: Migrace 6/12 - Úprava názvu

Teď je třeba vytvořit nový objekt přehrávaného videa. Ten je realizovaný jako záznam v tabulce wp_posts. Jako název nového videa je nastavena kombinace názvu souboru s videem a název příspěvku s pořadem. To je provedeno částí kódu 7.7.

```
              String query2 = "INSERT INTO wp_posts "
                + "(post_author, post_date, post_date_gmt, "
                + "post_content, post_title,post_excerpt, "
                + "post_status, comment_status, "
```

```
                + "post_name, "
                + "to_ping, pinged, "
                + "post_modified, post_modified_gmt,
post_content_filtered,"
                + "post_parent, menu_order, post_type, comment_count )
VALUES "
                + "(999, NOW(), NOW(),'',?,'',"
                + " 'publish','closed',"
                + " ?,"
                + "'','', "
                + "NOW(), NOW(), '',0, 0, 'wp_wplyr_videos',0)";
            PreparedStatement insertVideoStatement = conn.
prepareStatement(query2, Statement.RETURN_GENERATED_KEYS);
            insertVideoStatement.setString(1, trimmedVideoPath+" video
 of "+postName);
            insertVideoStatement.setString(2, new String(
trimmedVideoPath+" video of "+postName).toLowerCase().replace(' ','-'
).replaceAll("[^a-zA-Z0-9]", "").trim());
            int query2ResultSet = insertVideoStatement.executeUpdate()
;
```

Kód 7.7: Migrace 7/12 - Vytvoření nového videa

Po vygenerování videa je třeba přidat zdroje k souborům pro přehrání. V ukázce 7.8 nejprve získáme identifikátor nově vytvořeného videa.

```
            try (ResultSet generatedKeys = insertVideoStatement.
getGeneratedKeys()) {
                if (generatedKeys.next()) {
                 newVideoPostId = generatedKeys.getLong(1);
```

Kód 7.8: Migrace 8/12 - Zjištění identifikátoru

Následně je nutné vytvořit metadata. To provádí ukázka 7.9. Pro pořady class, fact a cooking, je vytvoření metadat realizováno pomocí metody createOtherPostMeta().

```
                 if(postType.equals("class")) {
                  createOtherPostMeta(newVideoPostId,
                  postType+"/"+trimmedVideoPath+".mp4",
                  conn,  "spots/za.czech.class.mp4");
                 }else if(postType.equals("fact")) {
                  String dynamicRegion = getDynamicRegion(postId,conn)
;
                  if(dynamicRegion == null) {
                   createOtherPostMeta(newVideoPostId,
                       postType+"/"+trimmedVideoPath+".mp4",
                       conn,  "spots/video.konec.mp4");
                  }else {
                   createOtherPostMeta(newVideoPostId,
```

```
                        postType+"/"+trimmedVideoPath+".mp4",
                        conn,  "spots/za-"+dynamicRegion+".mp4");
                    }
                    }else if(postType.equals("cooking")) {
                        createOtherPostMeta(newVideoPostId,
                            postType+"/"+trimmedVideoPath+".mp4",
                            conn,  "spots/video.konec.mp4");

                    }
```

Kód 7.9: Migrace 9/12 - Vytvoření pořadu

Zatímco pro pořady typu broadcast je třeba použít metodu createVideoPostMeta(). Obsah této metody je vidět v kódu 7.15.

```
                    else {
                     createVideoPostMeta(newVideoPostId, postType+"/"+
trimmedVideoPath+".mp4", conn, "spots/start.video.promote.mp4", "
spots/video.konec.mp4");
                    }
                   }
```

Kód 7.10: Migrace 10/12 - Vytvoření alternativního pořadu

Následuje úsek 7.11 propagace výjimky a inkrementace počítadla příspěvků s videem.

```
                   else {
                       throw new SQLException("Creating user failed, no
ID obtained.");
                   }
               }
               ++rowCount2;
           }
```

Kód 7.11: Migrace 11/12 - Ošetření výjimky

Jako poslední věc, po vytvoření nového objektu s videem, je přidat toto video do původního příspěvku přes shortcode. To je realizováno v metodě appendVideoToPost() v části 7.12.

```
           String postContent = rset.getString("post_content");
           appendVideoToPost(postContent,postId,newVideoPostId,conn);
           ++rowCount;
       }
    } catch(SQLException ex) {
       ex.printStackTrace();
    }
}
```

Kód 7.12: Migrace 12/12 - Přidání zástupného kódu do příspěvku

Při migraci bylo zjištěno, že některé pořady typu facts mají speciální ukončovací video, podle kraje ke kterému se pořad vztahuje. Metoda 7.13 zjišťuje patřičný kraj a její návratová hodnota je využita při vytváření záznamů v tabulce wp_postmeta.

```
private static String getDynamicRegion(int postId, Connection conn) {
 Statement stmt;
 try {
  stmt = conn.createStatement();
    String query1 = "SELECT term_taxonomy_id FROM wp_term_relationships
    WHERE object_id = "+postId;
    ResultSet query1ResultSet = stmt.executeQuery(query1);
    while(query1ResultSet.next()) {   // Move the cursor to the next row,
     return false if no more row
     Statement stmt2 = conn.createStatement();
     int    termId   = query1ResultSet.getInt("term_taxonomy_id");
        String query2 = "SELECT slug FROM wp_terms WHERE term_id = "+
    termId;
        ResultSet query2ResultSet = stmt2.executeQuery(query2);
        query2ResultSet.next();
        return query2ResultSet.getString("slug");

    }
 } catch (SQLException e) {
  e.printStackTrace();
 }
 return null;
}
```

Kód 7.13: Migrační funkce 1/4 - Zjištění regionu

Metoda appendVideToPost z 7.14, realizuje spojení videa s příspěvkem. Nejdříve načte původní příspěvek, do jeho obsahu uloží shortcode s hodnotou vkládaného videa a následně přepíše obsah původního příspěvku. Při migraci dat bylo zjištěno, že 3 příspěvky obsahovaly prázdné znaky u00A0. Tyto znaky se po uložení z nějakého důvodu poškodily. Protože tyto znaky neměly v příspěvcích žádný význam tak byly nahrazeny obyčejnou mezerou.

```
private static void appendVideoToPost(String oldPostContent, int postId,
    long newVideoPostId, Connection conn) {
  String query = "update wp_posts set post_content = ? where ID = ?";
     try {
  PreparedStatement preparedStmt = conn.prepareStatement(query);
  preparedStmt.setString(1, "[wplyr id="+newVideoPostId+"] </br> " +
    oldPostContent.replace('\u00A0',' '));
  preparedStmt.setInt(2, postId);
  preparedStmt.executeUpdate();
```

```
 } catch (SQLException e) {
  e.printStackTrace();
 }
}
```

Kód 7.14: Migrační funkce 2/4 - Připojení zástupného kódu

Metoda createVideoPostMeta slouží k vytvoření odkazů na soubory s videem pro pořady typu broadcast.

```
public static void createVideoPostMeta(long id, String videoPath,
    Connection con, String introPath, String outroPath)
   {
    try {
    String query = "INSERT INTO wp_postmeta "
            + "(post_id,meta_key,meta_value) VALUES "
            + "("+id+",'_wp_wplyr_video_source','"
            + "'a:4:{i:0;s:0:\"\";"
             + "i:1;s:"+(introPath.length()+1)+":\"/"+introPath+"\";"
             + "i:2;s:"+(videoPath.length()+1)+":\"/"+videoPath+"\";"
             + "i:3;s:"+(outroPath.length()+1)+":\"/"+outroPath+"\";}')";
          PreparedStatement insertVideoMetaStatement = con.
    prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
   int query2ResultSet = insertVideoMetaStatement.executeUpdate();
   query = "INSERT INTO wp_postmeta "
          + "(post_id,meta_key,meta_value) VALUES "
          + "("+id+",'_wp_wplyr_video_type','a:4:{i:0;s:5:\"video\";i:1;s
    :5:\"video\";i:2;s:5:\"video\";i:3;s:5:\"video\";}')";
   insertVideoMetaStatement = con.prepareStatement(query, Statement.
    RETURN_GENERATED_KEYS);
   query2ResultSet = insertVideoMetaStatement.executeUpdate();
  } catch (SQLException e) {
   e.printStackTrace();
  }

   }
```

Kód 7.15: Migrační funkce 3/4 - Vytvoření pořadu

Metoda createOtherPostMeta je skoro identická předchozí metodě. Na rozdíl od ní vytvoří pořad bez úvodního videa. Je použita pro pořady class, fact a cooking.

```
public static void createOtherPostMeta(long id, String videoPath,
    Connection con, String outroPath)
{
    try {
    String query = "INSERT INTO wp_postmeta "
         + "(post_id,meta_key,meta_value) VALUES "
         + "("+id+",'_wp_wplyr_video_source','"
```

```
        + "'a:3:{i:0;s:0:\"\";"
         + "i:1;s:"+(videoPath.length()+1)+":\"/"+videoPath+"\";"
         + "i:2;s:"+(outroPath.length()+1)+":\"/"+outroPath+"\";}')";
      PreparedStatement insertVideoMetaStatement = con.prepareStatement(
   query, Statement.RETURN_GENERATED_KEYS);
      System.out.println(videoPath);
   int query2ResultSet = insertVideoMetaStatement.executeUpdate();
   query = "INSERT INTO wp_postmeta "
         + "(post_id,meta_key,meta_value) VALUES "
         + "("+id+",'_wp_wplyr_video_type','a:3:{i:0;s:5:\"video\";i:1;s
    :5:\"video\";i:2;s:5:\"video\";}')";
   insertVideoMetaStatement = con.prepareStatement(query, Statement.
    RETURN_GENERATED_KEYS);
   query2ResultSet = insertVideoMetaStatement.executeUpdate();
  } catch (SQLException e) {
   e.printStackTrace();
  }
}
```

Kód 7.16: Migrační funkce 4/4 - Vytvoření alternativního pořadu

Při migraci na servery webového hostingu Wedos bylo zjištěno, že jí nelze realizovat přímým spojením do databáze. Pro migraci tedy bylo třeba stáhnout zálohu databáze lokálně, spustit skript, databázi exportovat a pak teprve nahrát na servery služby Wedos.

8 Závěr

Tato práce v kapitole s názvem ***Použité technologie*** shrnuje jak obecné informace o technologiích potřebných pro vytváření pluginu pro přehrávání videa na webu v systému Wordpress tak jejich historický vývoj. Také je zde detailně prozkoumán v současné době nejpopulárnější webový přehrávač s názvem Plyr. V kapitole ***Analýza*** se čtenář může dozvědět, co vedlo k zániku Flashových přehrávačů videa. Také jsou zde rozebrány požadavky, které je dobré ale občas i nutné dodržet při vývoji pluginů pro redakční systém Wordpress. V kapitole ***Návrh*** je popsán způsob, kterým je možné realizovat snadné přehrávání videa na webu včetně tvorby playlistů. Po analýze potřebných funkcí se práce v kapitole ***Implementace*** věnuje navržení Wordpress pluginu. Výsledné řešení umožňuje přehrávání videí jak z FTP serveru, tak ze serveru YouTube. Pro přehrávání je použita modifikovaná existující komponenta HTML5 přehrávače Plyr z kapitoly ***Použité technologie***. V kapitole ***Balíčkovací systém*** je pak popsán způsob, kterým byl plugin publikován do systému NPM a jakým způsobem je možné ho stáhnout. Vlastní plugin pro použití v redakčním systému Wordpress je pak popsán v kapitole ***Obalový projekt***. V kapitole ***Testování*** se práce věnuje všem odvětvím testování, ve kterých byla ověřena funkčnost výsledného pluginu. Vytvořené řešení bylo detailně otestováno a následně nasazeno na produkční servery CzechAmericanTV. Po instalaci pluginu do redakčního systému Wordpress je možné vytvářet a publikovat videa. K přehrávání jsou použity moderní technologie jako je HTML5 a JavaScript. Toto řešení je jednoduché, univerzální a s vyhlídkou budoucího růstu. Ukládání videa jako samotného Wordpress objektu datového modelu umožňuje definici extenzivních nastavení. Využitím těchto technologií se také podařilo dosáhnout kompatibility s velkým množstvím prohlížečů a zařízení.

Výsledný plugin byl úspěšně nasazen na server www.catvusa.com, kde se s jeho pomocí vysílají jak nové, tak původní pořady. Přestože plugin splňuje základní funkce, během návrhu řešení byl rozsah projektu postupně navyšován. Nad rámec zadání měla být implementována například podpora nového editoru systému Wordpress. Odladění a nasazení pluginu zabralo neočekávaně velké množství času. Z tohoto důvodu zatím nebyla dokončena podpora nového Gutenberg editoru. Vývoj pluginu bude pokračovat i po ukončení této práce s cílem implementace plné podpory pro nový editor Gutenberg a zveřejněním pluginu v oficiálním repositáři systému Wordpress.

Literatura

[1] *WordPress.com and WordPress.org* [online]. WordPress, 2018. Dostupné z: https://en.support.wordpress.com/com-vs-org.

[2] *Make WordPress Core* [online]. WordPress, 2018. Dostupné z: https://make.wordpress.org/core/handbook/about/organization.

[3] *Database Description* [online]. WordPress, 2019. Dostupné z: https://codex.wordpress.org/Database_Description.

[4] *Detailed Plugin Guidelines* [online]. WordPress, 2019. Dostupné z: https://developer.wordpress.org/plugins/wordpress-org/detailed-plugin-guidelines/.

[5] *Say Hello to the New Editor* [online]. WordPress, 2019. Dostupné z: https://wordpress.org/gutenberg/.

[6] *Make WordPress Plugins* [online]. WordPress, 2018. Dostupné z: https://make.wordpress.org/plugins/handbook.

[7] *WordPress Requirements for Envato Market* [online]. Envato Elements, 2019. Dostupné z: https://help.author.envato.com/hc/en-us/articles/360000510603-WordPress-Plugin-Requirements.

[8] *Shortcodes documentation* [online]. WordPress, 2019. Dostupné z: https://wordpress.com/support/shortcodes/.

[9] Banu, S. *Detailed Plugin Guidelines* [online]. BlogVault, 2019. Dostupné z: https://blogvault.net/wordpress-database-schema/.

[10] Brazell, A. *WordPress Bible*. John Wiley & Sons, 2010. ISBN 9780470621035.

[11] Brown, A. *WordPress Hooks Database* [online]. 2017. Dostupné z: https://adambrown.info/p/wp_hooks.

[12] Doğan, S. – Betin-Can, A. – Garousi, V. Web application testing: A systematic literature review. *Journal of Systems and Software*. 2014, 91, s. 174 – 201. ISSN 0164-1212.

[13] Field, D. *6 Major Tech Companies Have Doubled Their Design Hiring Goals in the Last Half Decade* [online]. TechCrunch, May 2017. Dostupné z: https://techcrunch.com/2017/05/31/here-are-some-reasons-behind-techs-design-shortage.

[14] Flanagan, D. *JavaScript : the definitive guide*. O'Reilly, 2006. ISBN 9780596101992.

[15] Garousi, V. et al. A systematic mapping study of web application testing. *Information and Software Technology*. 2013, 55, 8, s. 1374 – 1396. ISSN 0950-5849.

[16] Goodman, D. *JavaScript bible*. Wiley Pub, 2007. ISBN 9780470146231.

[17] Hughes, J. *What Are Shortcodes in WordPress? Explained for Beginners* [online]. Themeisle, 2018. Dostupné z: `https://themeisle.com/blog/what-are-shortcodes-in-wordpress/`.

[18] Hunt, A. *Programator pragmatik: Jak se stat lepsim programatorem a vytvaret kvalitni software*. Computer Press, 2007. ISBN 978-80-251-1660-9.

[19] Jackson, B. *Diving Into the New Gutenberg WordPress Editor (Pros and Cons)* [online]. Kinsta Inc., 2019. Dostupné z: `https://kinsta.com/blog/gutenberg-wordpress-editor/`.

[20] JACOBY, J. *PROFESSIONAL WORDPRESS PLUGIN DEVELOPMENT*. WILEY-BLACKWELL, 2020. ISBN 978-1119666943.

[21] Jobs, S. P. Thoughts on Flash. April 2010. Dostupné z: `https://www.apple.com/hotnews/thoughts-on-flash/`.

[22] Keith, J. *DOM scripting : web design with JavaScript and the Document Object Model*. Friendsof, 2005. ISBN 978-1-59059-533-6.

[23] Koskinen, T. – Ihantola, P. – Karavirta, V. Quality of WordPress Plug-Ins: An Overview of Security and User Ratings. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, s. 834–837, 2012.

[24] Kouratoras, K. *WordPress Shortcodes: A Complete Guide* [online]. Kinsta Inc., 2012. Dostupné z: `https://www.smashingmagazine.com/2012/05/wordpress-shortcodes-complete-guide/`.

[25] Krogsgard, B. *Interview with Matt Mullenweg on the Word Press Ecosystem—Draft Podcast* [online]. Post Status, Nov 2017. Dostupné z: `https://poststatus.com/interview-matt-mullenweg-wordpress-ecosystem-draft-podcast`.

[26] Patel, S. K. – Rathod, V. R. – Parikh, S. Joomla, Drupal and WordPress - a statistical comparison of open source CMS. In *3rd International Conference on Trendz in Information Sciences Computing (TISC2011)*, s. 182–187, 2011.

[27] Potts, S. *Plyr GitHub page* [online]. 2020. Dostupné z: https://github.com/sampotts/plyr.

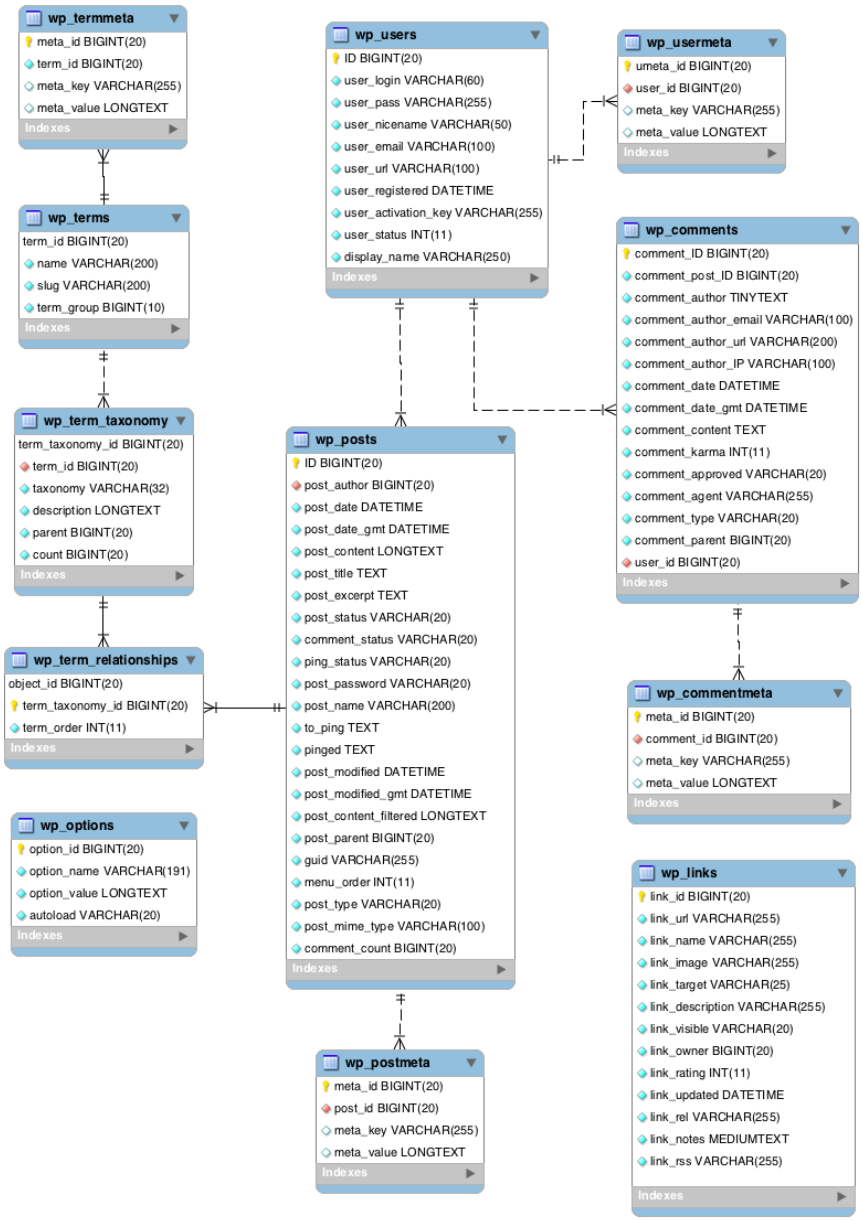
[28] Prelovac, V. *WordPress plugin development : beginner's guide : build powerful, interactive plugins for your blog and to share online.* Packt Pub, 2009. ISBN 978-1847193599.

[29] Tomiša, M. – Milković, M. – Čačić, M. Performance Evaluation of Dynamic and Static WordPress-based Websites. In *2019 23rd International Computer Science and Engineering Conference (ICSEC)*, s. 321–324, 2019.

[30] van Genuchten, M. – Hatton, L. Compound Annual Growth Rate for Software. *IEEE Software*. 2012, 29, 4, s. 19–21.

[31] *Usage statistics of content management systems* [online]. W3Techs, 2020. [cit. 2020/05/10]. Dostupné z: https://w3techs.com/technologies/overview/content_management.

A Schéma

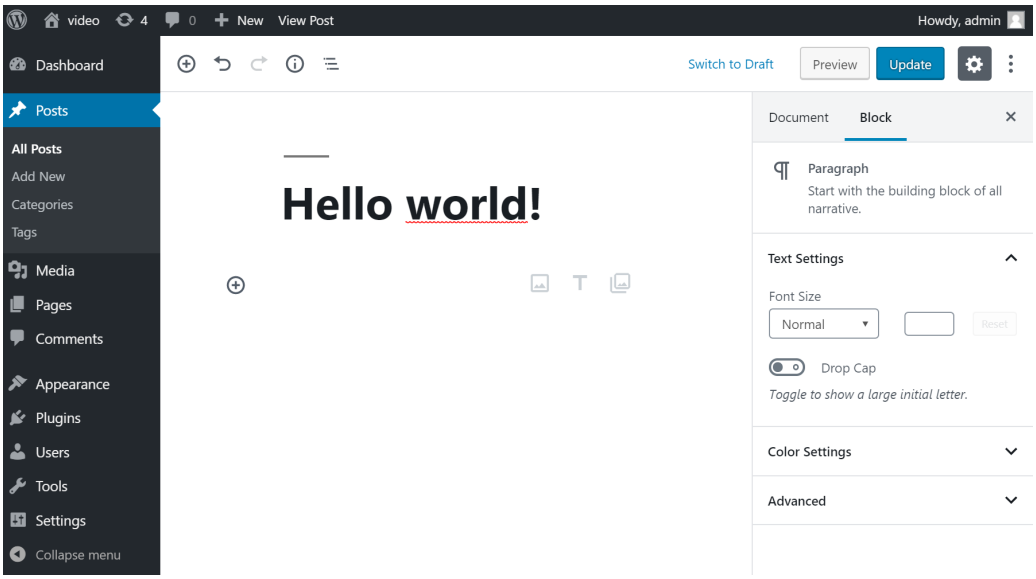


Obrázek A.1: Databázové schéma systému Wordpress

B Uživatelská příručka

Pro použití pluginu je třeba nejprve nainstalovat redakční systém Wordpress. Po přihlášení do administrační zóny nás uvítá obrazovka na obrázku B.1. Čerstvá instalace nejnovější verze systému Wordpress však již využívá interaktivní editor Gutenberg. Vytvořený plugin sice obsahuje blok, který v editoru lze použít, ale tento způsob vkládání videa ještě není kompletně odladěn. Uživatel tedy musí otevřít příspěvek ve starém editoru.

Nejsnadnější způsob jak otevřít příspěvek ve starém editoru je instalace



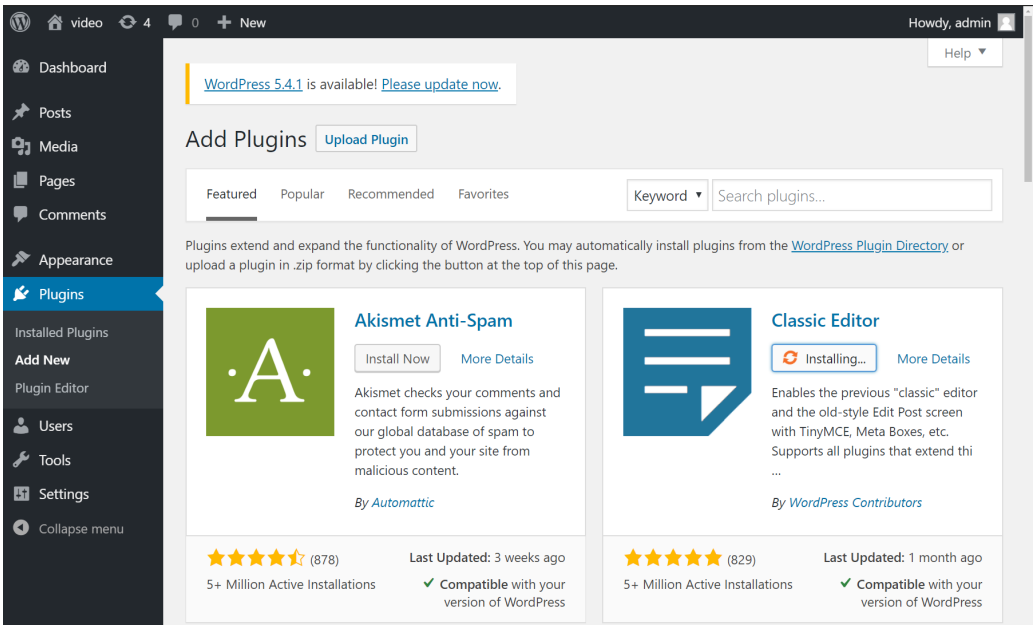
Obrázek B.1: Wordpress Editor Gutenberg

pluginu s názvem Classic Editor. Po navigaci do online katalogu pluginů je třeba tento plugin vyhledat a nainstalovat. Jedná se o krok z obrázku B.2.

Po úspěšné instalaci pluginu starého editoru, může rovnou uživatel nainstalovat i plugin pro přehrávání videí. Ten se zatím bohužel nenachází ve veřejném katalogu a tak je třeba složku s pluginem manuálně nakopírovat do cesty /wp-content/plugins. Když tak uživatel úspěšně učiní, je možné v administrační zóně systému Wordpress, pod položkou Plugins vidět 2 neaktivní pluginy.

Oba tyto pluginy z obrázku B.3 je třeba označit a aktivovat. Při následném otevření rozpracovaného příspěvku se zobrazí starý editor z obrázku B.4.

Teď je už redakční systém Wordpress připraven na vytvoření nového videa. Po úspěšném zapnutí pluginu pro přehrávání videí se v levé navigační liště



Obrázek B.2: Povolení starého editoru

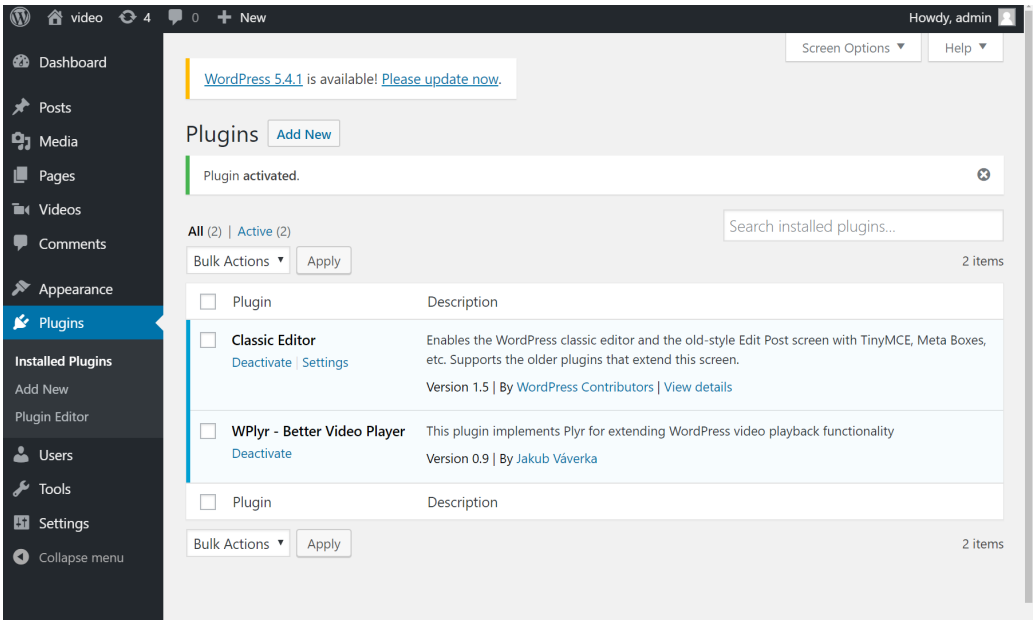
objeví položka Videos. Po stisknutí se zobrazí stránka všech v pluginu vytvořených videí. Náhled je zachycen na obrázku B.5.

Po vytvoření nového videa se zobrazí stránka na obrázku B.6. Video je třeba pojmenovat a pak lze přes tlačítko ve tvaru šedého plus přidávat za sebe videa do fronty.

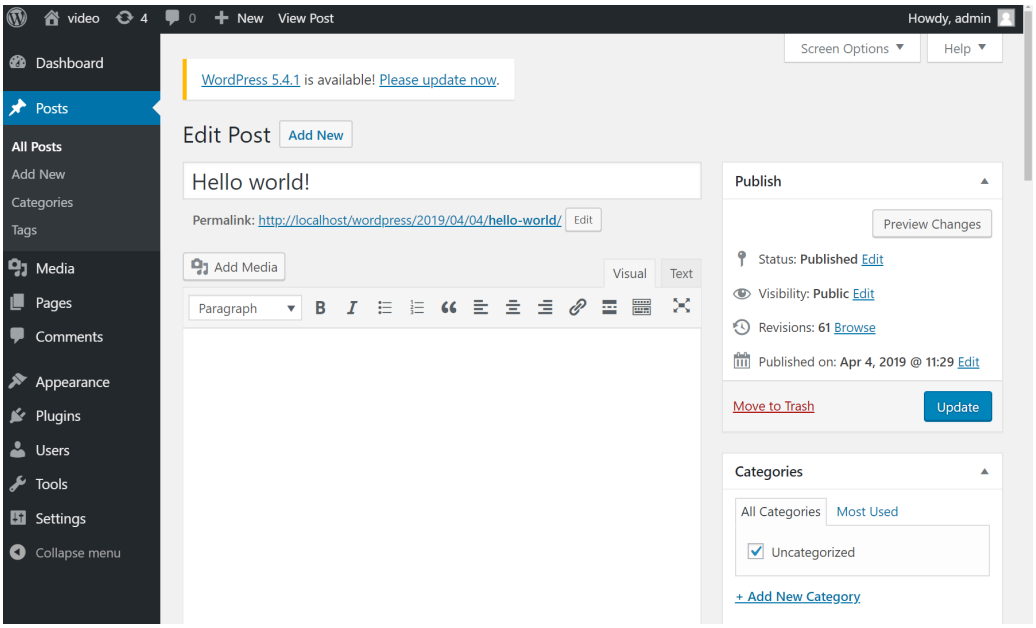
Každý nově přidaný panel má dvě záložky pomocí kterých může být přepnutý buď do stavu klasického videa a nebo do stavu videa ze serveru YouTube.

V prvním případě se zobrazí adresářová struktura složky na kterou je plugin nastavený a v případě druhé se zobrazí textové pole do kterého je možné vložit odkaz na YouTube. Po vytvoření playlistu je nutné příspěvek publikovat. Tímto způsobem se vytvořené video zapíše do databáze.

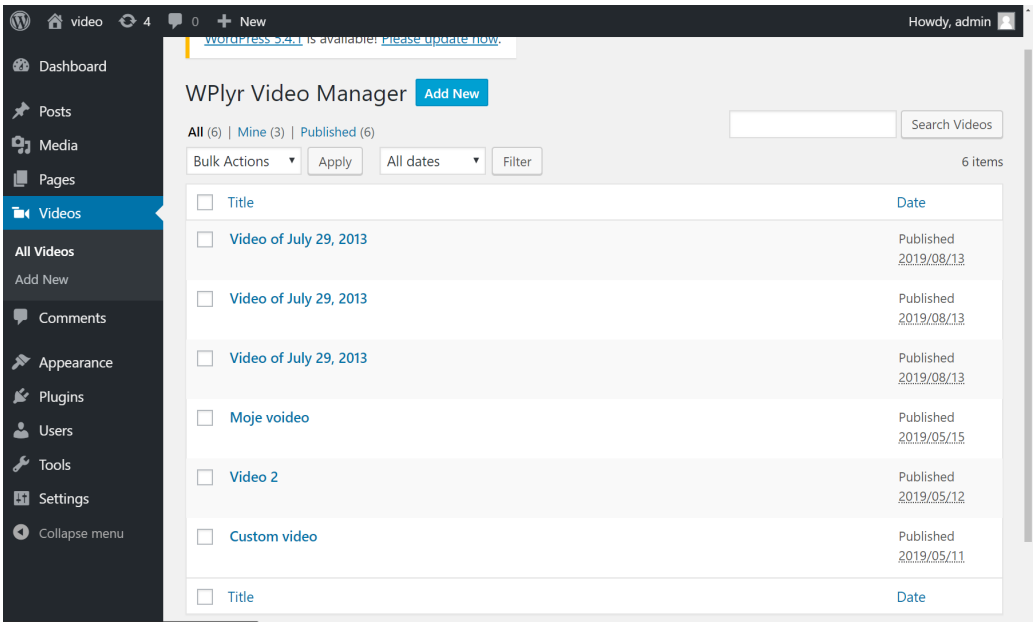
V následujícím kroku je ještě třeba video vložit do příspěvku. To je nutné realizovat využitím zástupného kódu ve tvaru [wplyr id=*] do jakéhokoliv textového pole publikovaného příspěvku. Pokud si uživatel pamatuje identifikační číslo právě vytvořeného videa je možné zástupný kód napsat rovnou do editoru, jako je zobrazeno na obrázku B.9. Jinak je možné pro vložení zástupného kódu použít komponentu, která se zobrazí na spodní straně každého editoru nových příspěvků. Tato komponenta je vidět na obrázku B.8. Skrze ní je možné vyhledat jakékoliv vytvořené video a pak následně přes tlačítko Insert shortcode je možné vložit zástupný kód přímo do textového pole.



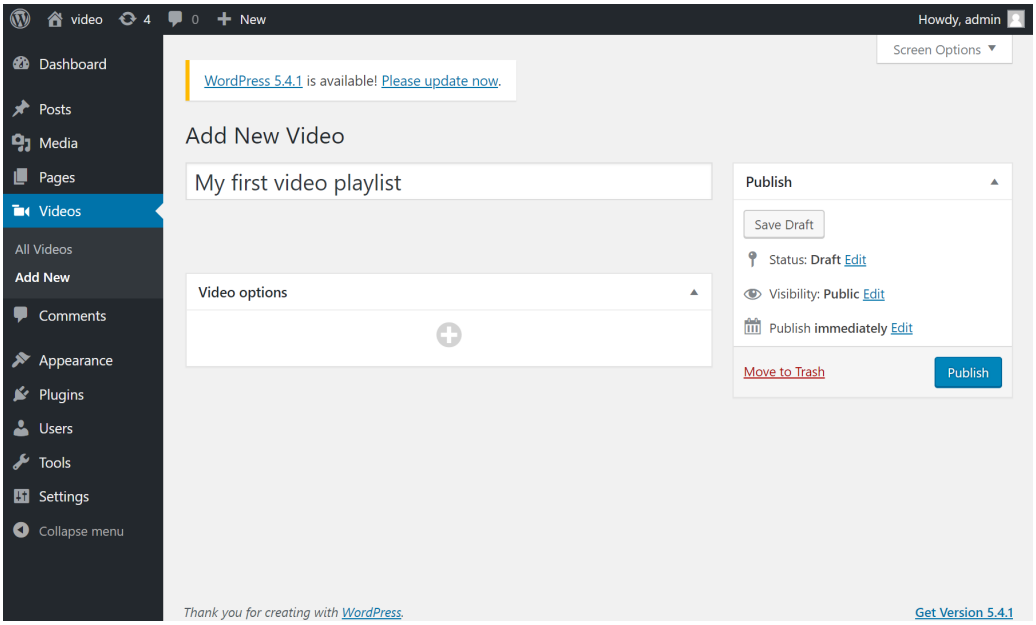
Obrázek B.3: Zapnutí neaktivních pluginů



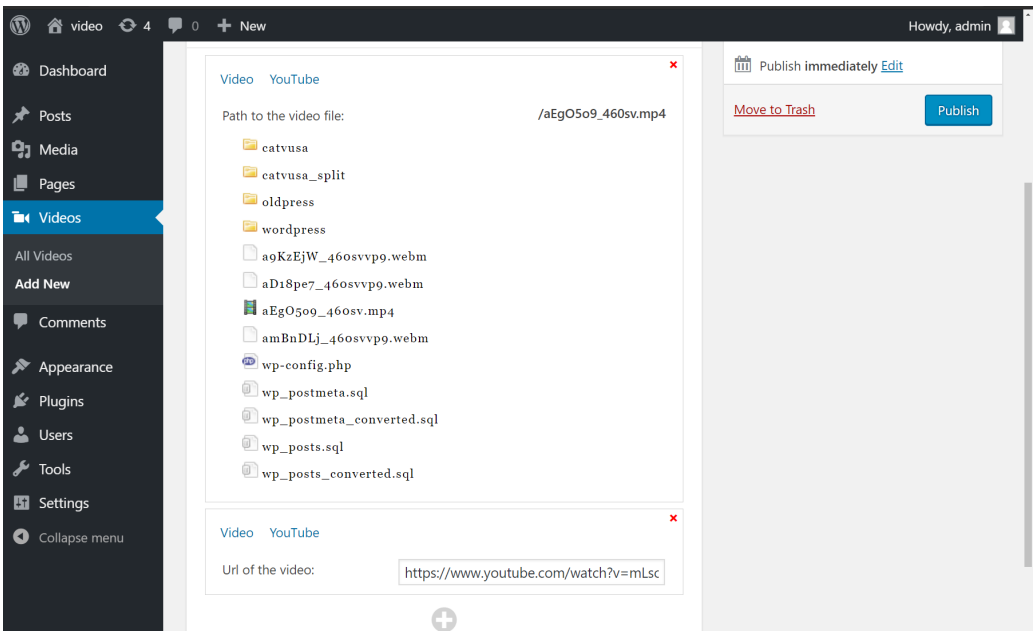
Obrázek B.4: Vzhled starého editoru



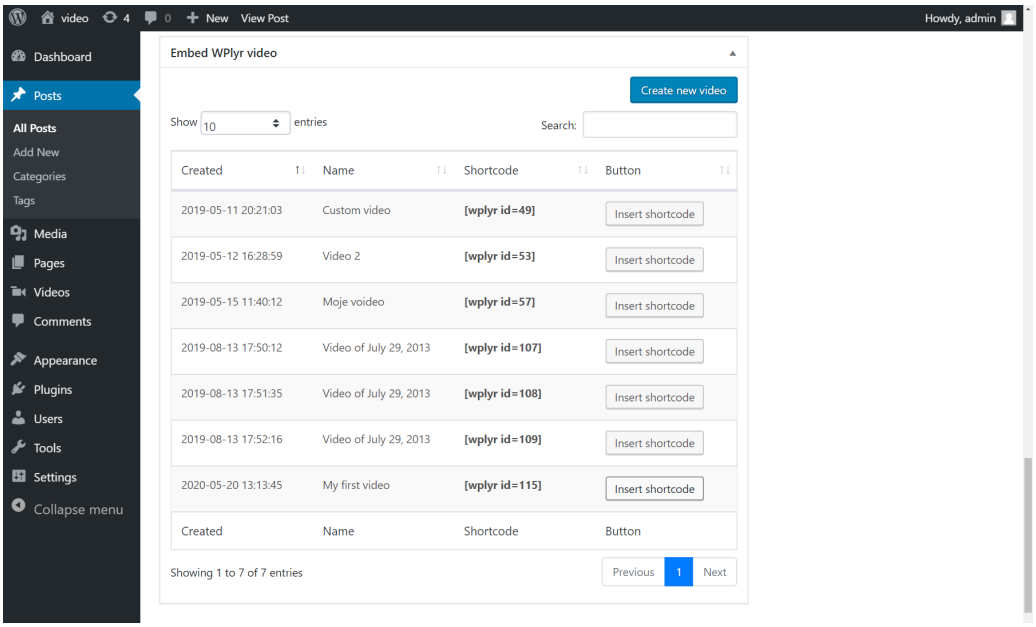
Obrázek B.5: Seznam videí



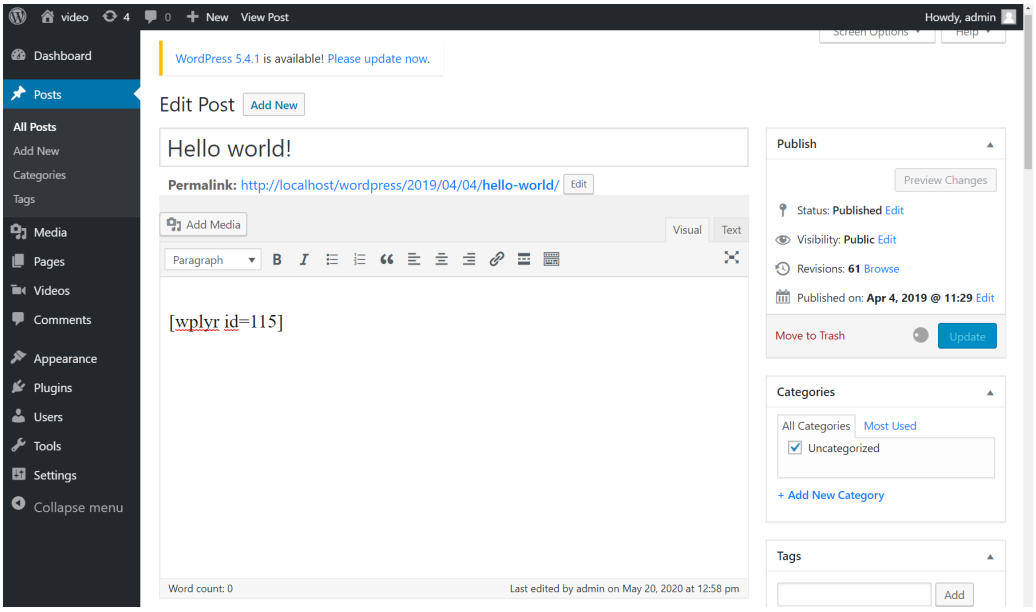
Obrázek B.6: Stránka nového videa



Obrázek B.7: Naplněné video



Obrázek B.8: Vyhledávač videí



Obrázek B.9: Vložení zástupného kódu

C Programátorská příručka

Na obrázku C.1 je vidět přibližná souborová struktura vytvořeného pluginu. Většina podstatných implementačních částí této práce byla již popsána v kapitole Implementace a nebo je vysvětlena v kódu. Tato příručka se bude tedy soustředit hlavně na strukturu pluginu a případné budoucí rozšíření.
Nejdůležitějším souborem je kořenový soubor wplyr-better-video.php. Ten v sobě obsahuje základní napojení do systému Wordpress a také je v něm nutné definovat cesty do složky s přehrávanými videi.

```
if(!defined('wplyr_video_path')){
    define('wplyr_video_path',$_SERVER['DOCUMENT_ROOT'] . "/videos");
    define('wplyr_video_url',get_site_url() . "/");
}
```

Kód C.1: Cesta do složky s videi

Toto nastavení je pro korektní funkci pluginu klíčové. Aby mohla být videa v budoucnu přesunuta ukládají se do databáze jen konce cesty k videu. Zde je tedy nutné nastavit korektní prefix cesty, aby byla cesta korektní jak ze strany prohlížeče, tak ze strany souborového systému. Zároveň je proměnná wplyr_video_path použita k vykreslení stromové struktury pro vybrání videa v editoru videí. Pokud uživatel nechá nastavenou cestu do složky, která obsahuje spoustu souborů dojde k zpomalení načítání tohoto editoru. Tato cesta v souborovém systému by tedy měla odkazovat do složky ve které se nachází jen video soubory.
Kořenový soubor wplyr-better-video.php podle potřeby využívá ostatní soubory. Jedním z nich je wplyr-gutenberg-block.js. Ten obsahuje signaturu potřebnou pro vytvoření bloku nového Wordpress editoru. Tato funkčnost není zcela podporována a funguje jen z části. Tento soubor tedy není pro současné fungování pluginu zcela zapotřebí.
Soubor wplyr-options-menu.php obsahuje registrační funkce pro vytvoření nových položek ve Wordpress nastavení. Tento soubor se používal za začátku a je s ním možné nastavit cestu do složky s videi rovnou skrze administrační panel redakčního systému Wordpress. Konfigurace se ale nakonec přesunula do souboru wplyr-better-video.php a tak už toto nastavení není potřeba. Tento soubor je v pluginu zachovám z důvodu možného rozšíření možností konfigurace.
Následující soubory wplyr-video-editor.php se stará o veškeré funkce spjaté s editorem videí. Jsou zde uloženy funkce pro generování komponenty pro

vytváření playlistů a i tabulkové komponenty pro procházení již vytvořených videí. Editor pro vytváření playlistů ukládá svá data do Wordpress databáze a tak se zde nachází i funkce pro uložení a načtení potřebných dat skrze databázové rozhraní. S tímto souborem je úzce spjatá složka video_editor, která obsahuje skripty, kaskádové styly a zdrojové kódy pro vykreslení stromu souborového systému.
Následující soubor wplyr-video-element.js se stejně jako wplyr-gutenberg-block.js stará o serializaci a deserializaci bloků nového editoru Gutenberg. Pro současný běh pluginu není tento soubor používaný, ale pokud bude v budoucnu chtít administrátor webu CzechAmericanTV přejít do nového editoru, je třeba blok upravit, aby umožňoval zadávání identifikátoru videa. Momentálně je možné vkládat jen prázdný blok.
Soubor wplyr-video-list.php obsahuje alternativní zobrazení tabulky pro výběr ze seznamu vytvořených videí. Při testování ale bylo zjištěno, že tato tabulka není vyhovující pro zobrazení velkého množství videí a tak byla implementována tabulka datatables.
Ostatní složky obsahují komponenty, které plugin využívá. Nachází se zde Bootstrap což je vzhledový systém, který umožňuje využití rozšířených šablon kaskádových stylů. Dále se zde nachází přehrávač Plyr, který je využit pro realizování samotného přehrávání. Poslední taková složka je složka datatables, která obsahuje zdrojové kódy použité pro vykreslení dynamické vyhledávací tabulky.

C.1 Vnitřní projekt

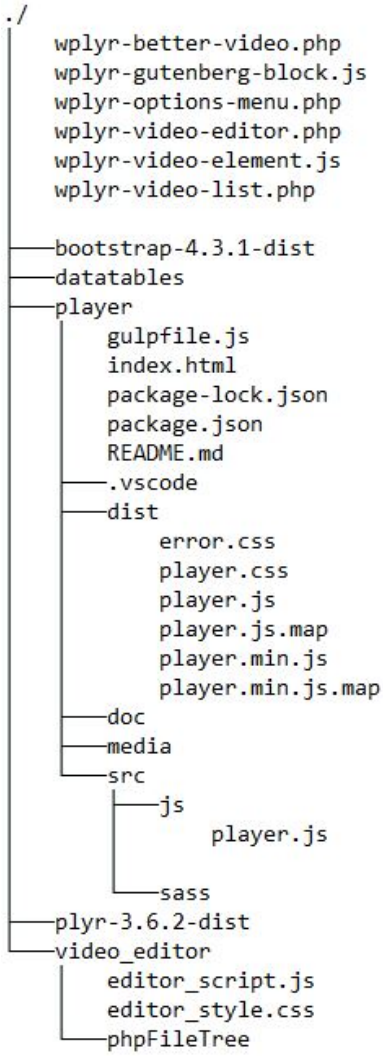
Nejdůležitější složkou pluginu je složka player, která obsahuje vnitřní projekt s implementací přehrávače videí. Tato složka je samotný projekt uložený v repositáři NPM. Je možné ho najít pod názvem plyr_showplayer a následně nainstalovat pomocí terminálového příkazu. V podsložce media se nacházejí soubory s videem použité pro demonstraci funkcí modulu na stránkách NPM. Zdrojové soubory se nachází ve složce src. Přesto, že je projekt psaný převážně v jazyce JavaScript je zapotřebí ho sestavit. Z toho důvodu se ve složce nachází soubor gulpfile.js a package.json. Pro sestavení je třeba nejprve stáhnout veškeré závislosti příkazem:

```
npm install
```

a následně sestavit projekt příkazem:

```
gulp build
```

Po úspěšném provedení těchto příkazů se vygeneruje složka dist ve které se budou nacházet výstupní soubory, které jsou minimalizované a doplněné o potřebné polyfilly pro zvýšení kompatibility.
Vnitřní projekt také obsahuje soubor index.html s jednoduchou demonstrací funkcí přehrávače. Pro korektní fungování je třeba zachovat pojmenování tříd elementů, které přehrávač obalují.



Obrázek C.1: Souborová struktura