Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Interaktivní HTML5 video přehrávač s playlistem

David Věžník

Vedoucí práce: Ing. Tomáš Vondra

29. dubna 2015

Poděkování

V první řadě chci poděkovat vedoucímu své práce, panu Ing. Tomáši Vondrovi, za cenné rady při konzultacích. Dále musím poděkovat všem, kteří mě podporovali během celého studia.

Prohlášení

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

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

V Praze dne 29. dubna 2015 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Věžník, David. *Interaktivní HTML5 video přehrávač s playlistem.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Tato bakalářská práce se zabývá rozšířením fungující webové prezentace, která uživatelům poskytuje informace o státech USA, o přehrávač videí. Přehrávač je vybrán na základě analýzy existujících přehrávačů a následně doplněn o některé funkce tak, aby co nejvíce vyhovoval požadavkům zadavatele. Dále práce zajišťuje administraci videí, která představuje základní operace s videi jako je přidávání, upravování, mazání a některé další možnosti. Práce také obsluhuje požadavky přicházející ze související mobilní aplikace. Nakonec je provedeno testování přehrávače v různých prohlížečích a na různých zařízeních.

Klíčová slova HTML5 přehrávač, playlist, JWPlayer, webová aplikace

Abstract

This bachelor thesis is focused on extending a functioning web presentation which provides information for users about the states of the USA, about a video player. The player was chosen based on analysis of existing video players and subsequently complemented by given functions so that it meets the requirements of the client as much as possible. Further, the thesis provides an administration of videos which represents basic video operations such as adding, editing, deleting and other options. The thesis also serves the

requirements incoming from related mobile application. Eventually, the test of the video player is performed in various browsers and devices.

Keywords HTML5 player, playlist, JWPlayer, web application

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Ve světě internetu je možné návštěvníkům webových stránek poskytovat informace ve formě videí. Nejdříve se publikování videí na webu jevilo jako nemožná věc. Hlavním omezením byla rychlost internetového připojení. Až postupem času, co se internet vyvíjel a připojení se stávalo rychlejším, se na internetu začal objevovat multimediální obsah právě v podobě videí. V roce 2005 přišla na internet služba Youtube a později další podobné služby, které videa na internetu ještě více zpopularizovala. Videa se tak stala běžnými prvky webových stránek.

Cestou prezentací videí se rozhodla jít i česko-americká nezisková společnost CATV Herald, která je zadavatelem této práce. Vytvořila projekt nazvaný American Travel Show, který momentálně poskytuje návštěvníkům pomocí webové prezentace informace o všech státech USA. Tyto informace mají pomoci lidem z celého světa lépe poznat jednotlivé státy USA a případně se na základě zjištěných informací rozhodnout, do kterých ze států vycestují.

Cílem této bakalářské práce je rozšířit webové stránky o přehrávač videí, který bude schopen videa o státech USA přehrát na většině dnešních zařízení. Přehrávač bude,
v rámci jeho možností, co nejvíce přizpůsoben potřebám této webové aplikace. Dále je třeba umožnit administrátorovi těchto stránek provádět operace s videi. Mezi operace bude patřit přidávání nových videí (o některém státu či reklamní spoty). Tato nová videa bude administrátor přiřazovat k určitým státům a bude k nim přidávat informace, které shrnou obsah daného videa na jehož základě se může uživatel rozhodnout pro přehrání tohoto videa. V případě videí jako reklamních spotů bude administrátor moci určit, kdy se tato videa mají přehrávat. Administrátor bude dále moci videa ze systému mazat, upravovat informace o nich a provádět s nimi některé další operace. Součástí práce je také obsluha požadavků přicházející z mobilní aplikace pro tento projekt. Systém webové aplikace bude obsluhovat základní požadavky mobilní aplikace jako jsou přihlášení, odhlášení, registrace nového uživatele a některé další.

Motivací pro výběr této práce bylo to, že jsem chtěl, aby výsledek práce byl po realizaci skutečně využíván, což tento projekt nabízí. Dále jsem chtěl tvořit webovou aplikaci, abych zužitkoval znalosti nabyté při studiu oboru Web a multimédia.
V neposlední řadě také pomohu neziskové organizaci.

Práce je členěna do šesti částí (kapitol). V první části (1) se zabývám přehráváním videí v prohlížečích, kde začnu od historie vkládání videí do webových stránek, následují možné formáty takových videí a kapitola končí současnými způsoby vkládání videí do stránek a operace s nimi. V druhé části (2) jsou uvedeny všechny požadavky na tuto práci, které vzešly z konzultací se zadavatelem. Třetí část (3) obsahuje rešerši a analýzu dostupných webových přehrávačů, ze kterých vzejde jeden, který nejvíce vyhovuje potřebám této práce. Ve čtvrté části (4) je uveden návrh řešení všech požadavků. Pátá část (5) ukazuje samotnou implementaci požadavků. V poslední, šesté, části (6) jsou uvedeny výsledky testování funkčnosti přehrávače v různých prohlížečích na desktopových a mobilních zařízeních.

Kapitola 1

Přehrávání videí v prohlížečích

Videa zpočátku nebyla přehrávána přímo na webových stránkách, ale byla stažena a přehrána programem nainstalovaným v počítači. S příchodem služeb Youtube a Flash se svět videí na internetu začal významně rozšiřovat a přesouvat se přímo na stránky. To vedlo velké firmy k diskuzím o nutnosti standardu pro přehrávání videí na webových stránkách.

1.1 Historie

Video se na webových stránkách po dlouhou dobu zobrazovalo pouze pomocí HTML elementů <object>, <embed> nebo <img> (ten k tomu není primárně určen).[4][5] Tyto elementy označovaly část webu, jejíž obsah není prohlížečem zobrazitelný, ale za pomocí vhodného pluginu je možné ho uživateli poskytnout. To nutilo uživatele nainstalovat si tento plugin do počítače. Nejdříve pluginy fungovaly tak, že se pro přehrání videa spustily jako program běžící mimo internetový prohlížeč a video se začalo přehrávat v něm. Později bylo umožněno přehrání videa v prohlížeči, avšak ve vyskakovacím okně.[4] Nutnost pluginů také vedla k válce mezi výrobci těchto aplikací. V roce 1996 Microsoft vyvinul ActiveMovie, jenž umožňoval přehrávat právě videa dostupná z internetu. V roce 1997 přišel na trh RealPlayer, který umožňoval totéž a v roce 1999 přichází firma Apple se svým přehrávačem QuickTime. Jedná se však o období, kdy internetové připojení bylo realizováno přes vytáčené spojení, jehož rychlost byla maximálně 56 kb/s po jedné telefonní lince, takže videa se zatím netěšily vysoké popularitě.[6] Microsoft se snažil vylepšit své postavení aplikacemi Microsoft Windows Media a později Microsoft Silverlight. V roce 2002 přišla firma Macromedia se svým přehrávačem Flash Player 6, který s sebou nese možnost přehrávat videa přímo v prohlížeči. Jeho využití bylo však velmi malé, protože celý webový multimediální obsah se soustředil především na pluginy Windows Media, QuickTime a RealMedia. V roce 2005 přišel vylepšený Flash Player verze 8, který se stal hlavním a nejčastějším pluginem pro přehrávání videí v prohlížečích té doby. Podporoval videa ve formátu FLV. Spolu s příchodem služby YouTube (v roce 2005), došlo ke zvýšení popularity sledování videí na internetu. Omezení rychlostí připojení už nebylo tak markantní jako dříve, protože kolem roku 2000 přišlo širokopásmové připojení, které

umožnilo podstatně vyšší rychlost.[4][7] Hned v roce 2005 YouTube začal používat pro přehrávání videí Flash Player, na který postupem času přecházely i další služby. V témže roce byla společnost Macromedia koupena společností Adobe a vznikl přehrávač s názvem Adobe Flash. V roce 2007 Adobe Flash začal podporovat pokročilý kodek H.264, který započal postupný přesun médií z formátu kontejneru FLV do formátu MP4.[4]

Nutnost pluginů donutilo různé výrobce prohlížečů učinit první společný krok vůbec, a to ten, že prohlížeče budou mít nativní podporu médií s příchodem nového HTML5. Začaly tak vznikat návrhy pro HTML tag <video>. První experimentální implementace tohoto tagu přišly v roce 2007 a zasloužila se o ně společnost Opera. Dále se vedly diskuze o tom, jaký se bude používat kodek a kontejner. Možné kodeky byly následující: MPEG-4 AVC/H.264, VC-1, MPEG-2, Theora, Dirac, VP8. Existující kontejnery byly: MOV (QuickTime), MP4 (MPEG), WMV, AVI (oba Microsoft), FLV (Adobe), MKV (Matroska), Ogg (Xiph). Na konci roku 2007 bylo navrženo, že se pro video bude používat kontejner Ogg a kodek Theora. Tento návrh se však nesetkal s úspěchem. Například Apple tento návrh odmítal, a proto prohlížeč Safari začal podporovat pouze kodeky MPEG-4 AVC/H.264 v kontejneru MP4. Prohlížeče se tedy rozdělily a začaly používat stejné, ale i odlišné kodeky a kontejnery. V roce 2010 přišel Google s projektem nazvaným WebM, který si dal za úkol sjednotit formát multimediálních souborů pro všechny prohlížeče. WebM využívá VP8 kodeku pro video a Vorbis kodeku pro audio. Audio a video je zapouzdřeno v kontejneru odvozeného z Matroska kontejneru a kontejner je nazván stejně jako projekt – WebM.[4][8]

1.2 Kontejnery

Kontejner zapouzdřuje několik datových toků multimediálních dat do jednoho souboru. Do jednoho kontejneru lze vložit například video, několik zvukových stop v různých jazycích a titulky. Je zajištěna jejich synchronizace.[9] Uživatel si sám může při přehrávání videa vybrat, jakou kombinaci chce použít. Kontejner nic neříká o kompresi. Ta je celá v režii kodeků. Některé kontejnery mohou v sobě mít více druhů kodeků, jiné méně.

1.2.1 MP4

Je založen na kontejneru QuickTime od Apple. Byl navržen, aby nahradil starý kontejner AVI. Nabízí uložení videa, více zvukových stop, menu, titulků i 3D objektů. Video bývá kódováno obvykle kodekem H.264, zvuk pomocí MP3, AAC nebo AC3. Umožňuje streamování přes internet. V současnosti je pomale vytlačován kontejnerem MKV, který se více hodí pro HD rozlišení.[10]

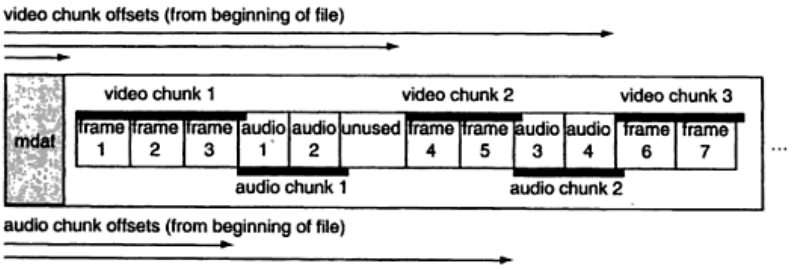
Všechna data uložená v kontejneru jsou uspořádána do atomů. Každý atom je určitého typu a délky.

Důležité atomy:

- `moov` obsahuje `header`, ve kterém jsou informace například o datu poslední změny a délce trvání videa. Dále obsahuje množinu všech stop média.

- `mdat` obsahuje video a audio snímky. Bývá rozdělen na stopy. Může být jedna stopa pro video a druhá stopa pro audio nebo může být jedna stopa, ve které je video i audio společně. Každá stopa je rozdělena na chunky a každý chunk je rozdělen na vzorky (snímky). Počet vzorků na chunk je určen v `stsc` atomu. Offset chunku je určen v `stco` atomu. Pomocí tohoto offsetu můžeme určit umístění chunku v souboru – offset udává počet bajtů od začátku souboru po začátek chunku.

  Případ videa i audia v jedné stopě:



Obrázek 1.1: MP4 struktura dat [1]

Například MPEG-4 používá obousměrně predikované snímky, tzv. B-frames. Pro dekódování B-frame dekodér potřebuje 3 snímky – snímek, který bude zobrazen před dekódovaným B-frame, dále snímek, který bude zobrazen po dekódovaném B-frame a konečně dekódovaný B-frame. V MP4 kontejneru jsou snímky uloženy v pořadí, ve kterém se budou dekódovat.[11][1]

1.2.2 Ogg

Ogg je otevřený standard vytvořen organizací Xiph.org Foundation. Je „open source-friendly“ a není zatížený žádnými patenty.[9] Umožňuje prokládané uložení videa a audia. Je vhodný pro streamování na internetu.[12]

Data uložená v kontejneru jsou rozdělena na stránky. Ve stránce jsou uchovávána samotná data (zakódované video a audio) ve formě paketů. Jednotlivé stránky mohou být proměnné velikosti, avšak maximální velikost stránky je 64 kB. Stránka obsahuje své identifikační číslo. Každý proud dat, který je těmito stránkami tvořen, začíná speciální stránkou *bos* („*begin of stream*“), která udává začátek proudu a končí speciální stránkou *eos* („*end of stream*“) udávající konec proudu. *Bos* stránka obsahuje informace o použitém kodeku a měla by obsahovat informace o zakódovaných datech. Například vzorkovací frekvenci videa. Mezi těmito speciálními stránkami jsou stránky obsahující samotná data.[13][14]

1.2.3 WebM

Otevřený formát vytvořený Googlem. Je založen na kontejneru Matroska. Taktéž vhodný pro streamování na internetu. Pro uchovávání dat a informací využívá EBML (Extensible

Binary Meta Language) struktury. EBML uchovává data ve stromové struktuře. Je to vlastně XML v binární podobě, namísto textové. WebM soubory musí obsahovat určitou množinu elementů, aby se daly považovat za WebM validní. Každý validní soubor musí obsahovat na nejvyšší úrovni (level 0) elementy `Ebml` a `Segment`. `Ebml` element říká, jaká jeho verze je použita. `Segment` element obsahuje další 4 podelementy (level 1), které by ve validním WebM neměly chybět. Prvním je element `SeakHead`. Ten obsahuje, mimo jiné, podelement `KaxCues`, který je použit při seekování – dle jeho informací se začne přehrávat určitý snímek. Druhý element se nazývá `Info`. Ten obsahuje informace o době trvání tohoto segmentu, informace o aplikaci, která toto video vytvořila, datum vytvoření a další. Třetím elementem je `Tracks`, ten obsahuje všechny stopy média. Může to být například jedna stopa pro video a jedna stopa pro audio. Stopy obsahují elementy jako `TrackUID`, což je jedinečný identifikátor stopy, `TrackType`, který udává typ stopy – může být audio, video nebo i titulky, logo a některé další. Dále stopy obsahují element `CodecID`, jenž určuje typ kodeku, kterým jsou data kódována. Pro validní WebM musí být `CodecID` pro stopu videa `V_VP8` a pro stopu audia `A_VORBIS`. Posledním čtvrtým podelementem elementu `Segment` je `Cluster`. Ten obsahuje samotné bloky dat videa (snímky + audio). Jednotlivé bloky dat obsahují informaci o čase, ve kterém mají být prezentovány.[15][16]

1.3 Video kodeky

Kodek je zkratka pro kodér-dekodér. Video kodek je software, který provede kompresi původního videa a převede ho do některého z kompresních formátů (zakóduje video).[17] Videopřehrávač pomocí dekodéru poté video dekóduje. Videa musíme kódovat, protože různá zařízení a prohlížeče podporují různé typy kodeků. Způsob komprese muže být ztrátový, nebo bezztrátový. Bezztrátový způsob komprese se pro web příliš nepoužívá, protože taková videa jsou příliš velká. Při ztrátové kompresi dochází k nenávratné ztrátě některých méně důležitých informací. Výsledná velikost videa je zlomek původní velikosti.[9]

1.3.1 H.264

Je také známý jako „MPEG-4 part 10“ nebo „MPEG-4 AVC“ nebo „MPEG-4 Advanced Video Coding“. Byl vyvinut pracovní skupinou MPEG a standardizován v roce 2003.[9] Nahrazuje starší kodeky H.263 a MPEG-4 part 2.[18] H.264 je rozdělen do profilů. Každý profil zahrnuje jiné vlastnosti a je určen pro jiná zařízení. Profily se zaměřují na zařízení, které bude mít pomalé připojení a slabý procesor (mobilní telefony) až po zařízení s vysokorychlostním připojením a silným procesorem (počítače). To, jaký použijeme při kompresi profil, má vliv na výslednou velikost souboru a taktéž na kvalitu videa. Čím vyšší profil zvolíme, tím více času je potom potřeba na dekódování videa, a tím potřebujeme rychlejší procesor.[9] Například mobilní zařízení s operačním systémem Android verze 3.0 a vyšší podporují Baseline level profil, Apple iPhone podporuje profily Baseline level a Main level.[19][20] Apple TV (3. generace) podporuje Baseline, Main i High level profil.[21] Adobe Flash v počítačích podporuje taktéž Baseline, Main i High level profily. Tento kodek se stal standardem pro Blu-Ray.[9]

Většina nepočítačových zařízení, například Blu-Ray přehrávače, smartphony, provádějí dekódování videa na samostatném čipu, protože jejich procesory nejsou dostatečně výkoné pro dekódování videa v reálném čase.[9]

Video kódované pomocí H.264 bývá nejčastěji přenášeno v kontejneru MP4 nebo MKV.[9]

1.3.2 Theora

Vyvinutý Xiph.org Foundation. Je volný a patentově nezatížený. Standardizován v roce 2004. Videa kódované pomocí Theora bývají nejčastěji přenášeny v kontejneru Ogg.[9]

1.3.3 VP8

Vytvořeno On2 Technologies, vlastněno Googlem. V roce 2010 Google získal společnost On2 a zveřejnil specifikaci kodeku a jednoduchého kodéru a dekodéru jako open source. Produkuje výstup na stejné úrovni jako H.264 High profil při zachování nízké dekódovací složitosti na úrovni H.264 Baseline profilu.[9] Videa kódované pomocí VP8 bývají nejčastěji přenášeny v kontejneru WebM.

1.4 Podpora videoformátů

Z důvodu neexistujícího standardu pro formát videa na webu musíme vědět, jaké formáty jsou podporovány různými prohlížeči a mobilními zařízeními, abychom byli schopni video prezentovat uživatelům.
V následujících tabulkách je znázorněna podpora video formátů v hlavních desktopových a mobilních prohlížečích. Do výběru jsou zařazeny nejčastější a nejběžnější kombinace kodeků a kontejnerů. Jsou to: kodek H.264 v kontejneru MP4, kodek Theora v kontejneru Ogg a kontejner WebM, který používá kodek VP8.

1.4.1 Podpora v prohlížečích

Do porovnání zahrneme nejoblíbenější prohlížeče včetně jejich starších verzí, které jsou však stále používány.

Prohlížeč	**MP4 H.264**	**Ogg Theora**	**WebM**
Internet Explorer	Ano	Ne	Ne
Google Chrome	Ano	Ano	Ano
Mozilla Firefox	Ne	Ano	Ano
Safari	Ano	Ne	Ne
Opera	Ne	Ano	Ano

Tabulka 1.1: Podpora videoformátů v prohlížečích v roce 2010

Prohlížeč	MP4 H.264	Ogg Theora	WebM
Internet Explorer	Ano	Ne	Ne
Google Chrome	Ano	Ano	Ano
Mozilla Firefox	Ano	Ano	Ano
Safari	Ano	Ne	Ne
Opera	Ano	Ano	Ano

Tabulka 1.2: Podpora videoformátů v prohlížečích v roce 2014

Z výše uvedených tabulek vyplývá, že pokud chceme zaručit přehrání videa ve všech prohlížečích, včetně jejich starších verzí (do roku 2010), tak musíme vložit video v minimálně dvou formátech, a to ve formátu MP4 H.264 a WebM.

1.4.2 Podpora v mobilních prohlížečích

Pro srovnání podpory v mobilních prohlížečích zahrnujeme nejoblíbenější mobilní prohlížeče. Jedná se o jejich nejnovější verze. Ve verzích starších šlo ve většině případů přehrát video pouze ve formátu MP4 H.264.

Prohlížeč	MP4 H.264/AAC	Ogg Theora	WebM
IE Mobile	Ano	Ne	Ne
Chrome for Android	Ano	Ne	Ano
iOS Safari	Ano	Ne	Ano
Opera Mini	Ne	Ne	Ne
Android Browser	Ano	Ne	Ano

Tabulka 1.3: Podpora videoformátů v mobilních prohlížečích v roce 2014

Pro mobilní zařízení je tedy vhodné mít video ve formátu MP4 H.264. Když prohlížeč video nepodporuje, tak ho většinou nabídne ke stažení a následně je možnost ho zkusit přehrát přehrávačem nainstalovaným v zařízení.

1.4.3 Fallback na Adobe Flash

Zajímavý je tah společnosti Adobe, která ohlásila podporu pro formát videí WebM. To znamená, že když na některém webu bude video pouze ve WebM formátu a prohlížeč tento formát nebude podporovat (například Safari, Internet Explorer), tak zprostředkovatel webu může nabídnout fallback na přehrávač Adobe Flash, aniž by měl video ve formátu pro Flash (formát FLV).[4] Zatím ale není přesné datum, kdy bude Flash plně WebM podporovat.

1.5 HTML <video> element

HTML 5 nám přináší párový element <video>, který dává možnost, jak zobrazovat video uživatelům. Uvnitř tohoto elementu můžeme uvádět další elementy, které jsou jeho potomky. Pokud uvedeme prostý text nebo element jiný, který není specifikován jako potomek, tak se na něj díváme jako na „fallback content“. Tento „fallback content“ bude vykreslen na stránku, pokud budeme používat prohlížeč, který element <video> nepodporuje nebo nepodporuje žádný z formátů, ve kterých je video vloženo.[4]

1.5.1 Atributy elementu

Element <video> má několik atributů, které určují jeho vzhled, chování nebo zdroj videa.

- `src` určuje cestu ke zdroji videa.
- `autoplay` je typu boolean a určuje, zda má přehrávač začít video přehrávat hned, jakmile je to možné.
- `loop` taktéž typu boolean. Když je nastaven, tak se video bude přehrávat stále dokola. Bude v nekonečné smyčce.[22]
- `poster` určuje cestu k souboru obrázku, který chceme zobrazit v přehrávači, když ještě není spuštěno přehrávání. Je to úvodní obrázek videa. Pokud tento atribut není nastaven, tak se vezme náhodný obrázek (frame) z videa.[22]
- `width` udává šířku videopřehrávače v pixelech.
- `height` udává výšku videopřehrávače v pixelech.
- `controls` je typu boolean. Pokud je nastaven, tak jsou v přehrávači zobrazeny standardní ovládací prvky. Standardní ovládací prvky jsou: *Play, Pauza, Seekování, Hlasitost, Zobrazení na celou obrazovku, Popisky a titulky* (pokud jsou dostupné), *Audio stopy* (pokud je jich k dispozici více).[23]
- `preload` určuje prohlížeči, kolik dat videa má stáhnout při jeho zobrazení na stránce. Může nabývat tří hodnot:
  - `none` – prohlížeč nezačne stahovat žádná data videa. Pokud je nastaven atribut `poster`, tak ani ten se nezobrazí. Je vhodné tuto hodnotu použít, pokud chceme ušetřit co nejvíce stahovaných dat.[4]
  - `metadata` – prohlížeč stáhne jen nejdůležitější data a informace o videu. Mezi tato data patří například obrázek v atributu `poster` a doba trvání videa.[4]
  - `auto` – prohlížeč začne stahovat všechna data související s videem.[22]

Příklad použití:

```
<video src="myVideo.mp4" poster="myVideoPoster.png" width="300"
       height="176" controls autoplay>
   <p>Váš prohlížeč nepodporuje video ve formátu MP4.</p>
</video>
```

Uvedený příklad ukazuje použití elementu a jeho atributů v praxi.

1.5.2 Podelement <source>

Atribut `src` v elementu <video> dovoluje určit pouze jeden zdroj videa. Jelikož různé prohlížeče podporují různé formáty, je vhodné určit pro video více zdrojů. To umožňuje HTML element <source>, který je definován jako potomek elementu <video>. Elementů <source> může být libovolné množství.[22] Prohlížeč pak prochází jednotlivé <source> od shora dolů. První, který reprezentuje video v podporovaném formátu, je vybrán k přehrávání.[4] Tento element má následující atributy:

- `src` – určuje cestu ke zdroji videa.
- `type` – slouží ke sdělení informací prohlížeči o tom, jaký je použitý kodek a kontejner pro video (tzv. *MIME type*). Prohlížeč může na základě této informace rozhodnout, zda přehraje video z tohoto <source>, či nikoliv, aniž by musel stahovat metadata. Prohlížeč ohodnotí, zda může video přehrát, jednou hodnotou z množiny `No`, `Maybe`, `Probably`.[4]
  V praxi to může vypadat například takto:

  ```
  <video poster="myVideo.png" controls>
    <source src="myVideo.webm" type="video/webm">
  </video>
  ```

  Případně můžeme v atributu `type` specifikovat i typy kodeků použitých pro zakódování videa. Prohlížeč potom bude v odhadování podpory přesnější:

  ```
  <video poster="myVideo.png" controls>
    <source src="myVideo.webm" type='video/webm;
     codecs="vp8, vorbis"'>
  </video>
  ```
- `media` – umožňuje definovat, pro jaká zařízení je video určeno. Tento atribut však není podporován žádným prohlížečem.[24]

1.5.3 Podelement <track>

Tento element přidává k videu textové stopy. Stopy jsou textový dokument například ve formátu WebVTT, který je podporován všemi prohlížeči. Struktura takového dokumentu může být následující:

```
radek1
00:00:10.000 -> 00:00:12.500
Tento text se objeví na obrazovce v desáté sekundě videa.

radek2
00:00:13.200 -> 00:00:16.900
A tento text se objeví v čase 13.2 s.
```

Každá položka v souboru se nazývá cue. Každý cue má svůj čas začátku, konce a text. Může mít i své ID. V našem příkladě uživáme ID `radek1` a `radek2`. Jednotlivé cue jsou odděleny prázdným řádkem.[25]

Element je specifikován následujícími atributy:

- `src` – cesta ke zdrojovému souboru.
- `kind` – udává typ textové stopy.[22] Je to hodnota z následující množiny:
  - `subtitles` – textovou stopu budou tvořit titulky. Titulky jsou kompletní přepis dialogů ve videu. Jsou vhodné zejména při nekvalitním audiu videa nebo pro překlad do jiného jazyka.[22][26]
  - `captions` – textovou stopu budou tvořit popisky. Popisky textově reprezentují nejen dialogy videa ale i veškeré zvukové efekty (např. start motoru). Jsou vhodné k použití při vypnutém audiu videa nebo pro sluchově postižené osoby.[22][26]
  - `descriptions` – textová reprezentace obsahu videa. Umožňuje sdělit obsah videa těm, kteří ho nevidí.[22][27]
  - `chapters` – textová reprezentace kapitol videa. Umožňuje přeskočení ve videu na určitou kapitolu.[22][27]
  - `metadata` – atribut je využíván skripty. Jeho nastavení není viditelné. Umožňuje programátorům spustit skript, jakmile přehrávání videa dosáhne času určitého cue. Text tohoto cue je předán skriptu.[28]

  Pokud je v elementu <track> atribut `kind` vynechán, tak se bere jeho výchozí hodnota `subtitles`.[22]
- `srclang` – v tomto atributu je uveden světový jazyk, ve kterém je stopa napsána. Je povinný pokud atribut `kind` je nastaven na `subtitles`.[29]
- `label` – udává název stopy. Musí být unikátní pro všechny elementy <track> se stejným atributem `kind` v daném elementu <video>.[22]
- `default` – atribut typu boolean. V případě více elementů <track> se stejným `kind` prohlížeč na základě preferencí uživatele rozhoduje, který vybrat. Pokud se na tomto základě nemůže prohlížeč rozhodnout, vybere se ten element <track>, který má nastaven atribut `default`.[30]

Příklad použití elementu <track> v elementu <video>:

```
<video width="320" height="240" controls>
  <source src="myVideo.mp4" type="video/mp4">
  <track src="subtitlesEn.vtt" kind="subtitles"
   srclang="en" label="English" default>
  <track src="subtitlesCz.vtt" kind="subtitles"
   srclang="cs" label="České">
</video>
```

1.6 Javascript a DOM

Javascript je programovací jazyk, který je hojně používaný v prohlížečích. Umožňuje provádět skripty na straně klienta (v prohlížeči). Zajišťuje interakci mezi uživatelem a prohlížečem, asynchronní komunikaci a dokáže měnit obsah zobrazovaný uživateli. Umožňuje přístup k prvkům DOM a jejich editaci.
DOM (Document Object Model) je hierarchická struktura ve formě stromu, obsahující všechny elementy webové stránky jako objekty. U těchto objektů se uchovávají jejich atributy. Ke každému objektu této struktury lze přistoupit a využívat metody a atributy, které jsou pro něj definované. Různé HTML elementy mají různé metody a atributy, ovšem existuje množina metod a atributů společná všem HTML elementům.

1.7 <video> jako DOM objekt

Pro získání kontroly nad videopřehrávačem musíme získat objekt, který je reprezentován elementem <video>. Využijeme Javascriptu:

```
<video controls>
  <source src="myVideo.mp4" type="video/mp4">
</video>
<script type="text/javascript">
  objectsVideo = document.getElementsByTagName("video");
  objectVideo = objectsVideo[0];
</script>
```

`document` odpovídá kořenu stromu v DOM. Metoda `getElementByTagName` vrací všechny elementy, jejichž název je předán v parametru funkce, jako objekty. Objekty jsou navráceny ve formě `NodeList` a lze k nim přistupovat pomocí indexů.[31] V příkladu výše, proměnná `objectVideo` reprezentuje objekt videopřehrávače. Nyní lze využívat metody a atributy tohoto objektu. Mezi jeho atributy patří, mimo jiné, všechny ty, které jsme přehrávači nastavili již při psaní elementu <video>. Například pro zjištění, zda přehrávač zobrazuje ovládací prvky využijeme boolean atributu `control`: `objectVideo.control`

Video DOM objekt poskytuje atributy, které se dají nastavit, i takové, které se nastavit nedají a slouží jen pro čtení. Ty především informují o tom, v jakém stavu se video nachází.[32]

1.7.1 Obecné atributy

currentSrc – určuje, z jakého ze zdrojů je video přehráváno. Musí se vyčkat, až budou stažena metadata jednotlivých zdrojů (pro jednotlivé elementy <source>).
V tu chvíli bude vyhodnoceno, ze kterého se bude video přehrávat a nastaví se atribut `currentSrc`. Pokud se zeptáme dříve, než se načtou metadata nebo nebude vyhovovat žádný ze zdrojů, atribut bude prázdný.[4] V opačném případě vrátí řetězec obsahující URL zdroje např. `http://www.w3schools.com/jsref/movie.mp4`.[33]

duration – vrací délku videa v sekundách. Tato hodnota je určena po stažení metadat. Předtím má duration hodnotu `NaN` (`Not-a-Number`). Pokud se jedná o živý přenos, má hodnotu `Infinity`. Může se stát, že prohlížeč nedokáže vyhodnotit délku trvání po stažení metadat přesně a v průběhu přehrávání videa ji změní. Při této změně se vyvolá událost `durationchange`.[4]

volume – určuje hlasitost přehrávaného videa. Hodnoty jsou v intervalu <0.0, 1.0>, kde 1.0 znamená nejvyšší hlasitost a 0.0 znamená vypnutý zvuk. Výchozí hodnota je 1.0. Hodnotu můžeme měnit. Pokud zadáme číslo mimo interval <0.0, 1.0>, tak se vyvolá výjimka `INDEX_SIZE_ERR`. Kdykoli se manipuluje s hodnotou atributu volume, tak je vyvolána událost `volumechanged`.[4]

muted – udává, zda je zvuk videa vypnutý, či ne. Pokud je jeho hodnota `true`, zvuk je vypnutý, pokud je `false`, tak se hlasitost řídí podle atributu `volume`. Je nadřazený atributu `volume`, takže i když hlasitost videa bude maximální, můžeme zvuk vypnout nastavením `muted` na hodnotu `true`. Při změně hodnoty atributu se vyvolá událost `volumechanged`, stejně jako u `volume`.[4]

videoWidth, videoHeight – atributy pouze ke čtení. Udávají šířku a výšku videa v pixelech. Pokud rozměry videa nejsou známy, tak vrací nulu. Rozdíl oproti atributům `width` a `height`, které se dají nastavit přímo elementu <video> je takový, že `videoWidth` a `videoHeight` uvádí skutečné rozměry videa tak, jak přijde do dekódovací pipeline, narozdíl od rozměrů videopřehrávače. Změna hodnot atributů `width`, `height` nemá žádný vliv na hodnoty `videoWidth` a `videoHeight`.[4]

1.7.2 Atributy spojené s přehráváním

currentTime – sděluje aktuální pozici přehrávaného videa v sekundách. Pokud se chceme v přehrávání posunout, můžeme hodnotu atributu změnit. Pokud nastavovaná hodnota bude ve videu dosažitelná, přehrávání se na ni přesune a vyvolá se událost `timeupdate`. V případě, že dosažitelná nebude nebo video nebude umožňovat seekování (posouvání se v čase videa), bude vyvolána výjimka. Pokud se přesouváme na pozici, která je dosažitelná, ale ještě pro ni nemáme připravená data, tak se přehrávání zastaví, vyvolá se událost `waiting` a čeká se, až budou data dostupná. [4]

seeking – atribut, který je nastaven na hodnotu `true`, pouze pokud probíhá seekování. Jinak má hodnotu `false`. [4]

paused – indikuje, zda je video pozastavené – hodnota `true`, či nikoliv – hodnota `false`. Změna hodnoty atributu vyvolá událost `timeupdate`.[4]

ended – udává, zda přehrávání videa skončilo. V takovém případě je nastaven na hodnotu `true` a vyvolá události `timeupdate` a `ended`. Pokud má videopřehrávač nastaven

atribut `loop` na hodnotu `true`, tak `ended` bude vždy `false` a bude probíhat nekonečné přehrávání.[4]

1.7.3 Stavové atributy

networkState – reprezentuje momentální stav elementu <video> z pohledu síťové aktivity. Může se nacházet v následujících stavech:

- `NETWORK_EMPTY` – nebyl identifikován žádný `currentSrc` atribut. K tomu může dojít, pokud nebyl zatím inicializován element <video> nebo element <src> respektive atribut `src`.[22]
- `NETWORK_IDLE` – nastává, pokud se čeká na uživatelovu interakci. Běžně, pokud není nastaven atribut `autoplay` a čeká se, až uživatel spustí přehrávání. Dále může nastat v případě potíží s připojením, při poškozeném zdrojovém souboru nebo v případě, že je již dostatek dat připravených pro prezentování a čeká se, až bude potřeba připravit další data. Stav se také projeví, pokud je již stažen celý zdrojový soubor. Při přechodu do tohoto stavu je vyvolána událost `suspend`.[4]
- `NETWORK_LOADING` – nastane, pokud se prohlížeč pokouší stáhnout část videa. Pokud stahuje první část, tak se vyvolá událost `loadstart`. Když později znovu nastane tento stav, tak se volá událost `progress`. Pokud server náhle přestane poskytovat data, tak je vyvolána událost `stalled`.[4][22]
- `NETWORK_NO_SOURCE` – značí, že se nepodařilo načíst zdroj nebo nebyl zdroj nalezen – žádný atribut `source` či element <source>.[22]

readyState – reprezentuje momentální stav elementu <video> ve vztahu k pozici v přehrávaném videu. Může se nacházet v následujících stavech:

- `HAVE_NOTHING` – nejsou dostupné žádné informace o zdroji videa. Když je atribut `networkState` ve stavu `NETWORK_EMPTY`, tak je typicky atribut `readyState` nastaven na tuto hodnotu.
- `HAVE_METADATA` – stav, kdy přehrávač dostal informace o zdroji, který bude přehrávat. Mezi informace patří například výška a šířka videa, délka trvání videa (případně jeho odhad). Také je sestavena dekódovací pipeline. Při dosažení tohoto stavu je vyvolána událost `loadedmetadata`.[4]
- `HAVE_CURRENT_DATA` – v tomto stavu má přehrávač připravena dekódovaná data k přehrávání pro aktuální pozici ve videu, ale nemá jich dostatečně mnoho, aby mohl začít s plynulým přehráváním. Tento stav může být přeskočen na stavy `HAVE_FUTURE_DATA` či `HAVE_ENOUGH_DATA`.[4]
- `HAVE_FUTURE_DATA` – data pro přehrávání jsou připravena jak pro současnou, tak pro nadcházející pozici ve videu (je připraven snímek pro současnou pozici i snímek pro následující pozici). Jestliže je přehrávač v tomto stavu poprvé, tak je vyvolána událost `canplay`. Přehrávač začne přehrávat video. Vyvolá se událost `playing`.[4]
- `HAVE_ENOUGH_DATA` – jsou připravena data pro současnou i více nadcházejících pozic pro přehrávání. Je umožněno plynulé přehrávání a je vyvolána událost `canplaythrough`.[22]

error – představuje poslední chybu elementu <video>. Chyba je reprezentována objektem `MediaError`, který vypadá následnovně:

```
interface MediaError {
const unsigned short MEDIA_ERR_ABORTED = 1;
const unsigned short MEDIA_ERR_NETWORK = 2;
const unsigned short MEDIA_ERR_DECODE = 3;
const unsigned short MEDIA_ERR_SRC_NOT_SUPPORTED = 4; }
```

Pokud žádná chyba není, tak má atribut hodnotu `null`, v opačném případě získáme kód chyby pomocí `error.code`.
Význam jednotlivých chyb:

- `MEDIA_ERR_ABORTED` – nastane, pokud komunikace, za účelem získání dat pro přehrávání, mezi prohlížečem a serverem byla ukončena prohlížečem. Například uživatel přejde na jinou webovou stránku. Je vyvolána událost `abort`.[4]
- `MEDIA_ERR_NETWORK` – nastane, pokud z nějakého důvodu přestane server komunikovat s prohlížečem, a tedy prohlížeč nemůže získávat další části dat ze zdroje videa. Je vyvolána událost `error`.[4]
- `MEDIA_ERR_DECODE` – pokud některou část dat zdroje videa není možné dekódovat, je nastavena tato chyba. Nastává v případě poškozených dat zdroje videa nebo data obsahují něco, co daný prohlížeč nepodporuje. Chyba vyvolá událost `error`.[4]
- `MEDIA_ERR_SRC_NOT_SUPPORTED` – nastane, pokud prohlížeč nepodporuje formát videa nebo na URL zdroje videa nebyla nalezena žádná odpovídající data. Je vyvolána událost `error`.[4]

buffered – uchovává části dat zdroje videa (tzv. „*ranges*“), které prohlížeč doposud nabufferoval. Tyto části jsou uchovávány v jednom objektu `TimeRanges`, který vypadá následovně:

```
interface TimeRanges {
     readonly attribute unsigned long length;
     float start(in unsigned long index);
     float end(in unsigned long index); }
```

Atribut `length` udává počet částí (*ranges*) v tomto objektu. `start(i)` vrací začátek části `i` v sekundách a `end(i)` její konec.
Sousední části se většinou spojují do jedné větší části. Pouze v případě velkých videí se vyskytuje částí více.[4]

seekable – reprezentuje oblast ve videu, na kterou je možné se přesunout. Tyto oblasti jsou taktéž reprezentovány pomocí `TimeRanges`.[4]

1.7.4 Metody

DOM objekt HTML elementu <video> poskytuje metody, pomocí kterých můžeme ovlivňovat výše zmíněné atributy, a tím i provádět operace s videopřehrávačem.

load() – přenačte <video> element. Veškerá dosavadní komunikace a všechny operace, které se nyní s objektem prováděly se zruší a začne se od začátku.[4] Kroky, které se provedou po zavolání této metody a stavy atributů:

- Inicializace – `networkState` je nastaven na `NETWORK_EMPTY`, `readyState` je nastaven na `HAVE_NOTHING`, `paused` je nastaven na `true`, `seeking` je nastaven na `false`, `ended` je nastaven na `false`, `currentTime` je nastaven na `0`, `error` je nastaven na `null`, `buffered, played, seekable` jsou prázdné.
- Výběr zdroje – `currentSrc` je nastaven podle atributu `src`, nebo elementu <source>, `networkState` je nastaven na `NETWORK_LOADING`, `loadstart` událost je vyvolána.
- Stahování metadat.
- Metadata jsou stažena – `duration` je určen, `videoWidth, videoHeight` jsou určeny, `seekable` je určen, `readyState` je nastaven na `HAVE_METADATA`, `loaded metadata` událost je vyvolána.
- Je připravený dostatek dat pro přehrávání – `readyState` je nastaven minimálně na `HAVE_FUTURE_DATA`, `loadeddata` událost je vyvolána, `canplay` událost je vyvolána, `buffered` je aktualizováno.
- Spustí se přehrávání – `paused` je nastaven na `false`, `play, playing` události jsou vyvolány.
- Video se přehrává. [4]

play() – nastaví atribut `paused` na `false` a provede kroky nutné pro přehrávání videa tak, jak je popsáno výše.[4]

pause() – nastaví atribut `paused` na `true`, vyvolá událost `pause` a pozastaví přehrávání videa. Stahování buď pokračuje, nebo je zastaveno. To závisí na množství dat, která jsou připravena k přehrávání.[4]

canPlayType() – zkontroluje, zda prohlížeč dokáže video přehrát.[34]Vrátí jednu z následujících hodnot:

- `probably` – prohlížeč s vysokou pravděpodobností dokáže přehrát tento formát videa.[34]
- `maybe` – prohlížeč si není jistý podporou formátu videa.[34]
- `""` – prázdný řetězec. Prohlížeč nepodporuje tento formát.[34]

Kapitola 2

Požadavky na projekt

V této kapitole budou stanoveny všechny funkční a nefunkční požadavky na tento projekt. V následujícím textu se budou objevovat 2 typy videí. Prvním typem budou videa, která sdělují uživatelům webu informace o státech USA. Těm budu dále říkat jen videa. Druhým typem budou reklamní videa, a ty budu nazývat spoty.
Některé z požadavků budou rozebrány detailněji.

2.1 Funkční požadavky

2.1.1 Obecné funkční požadavky

- F1.1. Přehrávání videí.
- F1.2. Správa uživatelských účtů.
- F1.3. Nabízení videí pro přehrání dle preferencí uživatelů.

2.1.2 Funkční požadavky kladené na přehrávač

- F2.1. Možnost vytvoření playlistu.
- F2.2. Přehrávání videí z Youtube.
- F2.3. Možnost úpravy vzhledu přehrávače.
- F2.4. Nastavení hlasitosti videa při startu přehrávání.
- F2.5. Ukládat informace o zhlédnutých videích.

2.1.3 Funkční požadavky kladené na administraci

- F3.1. Správa videí.
- F3.2. Správa spotů.

2.1.4 Funkční požadavky kladené na API

- F4.1. Registrace uživatele.
- F4.2. Přihlášení uživatele.
- F4.3. Odhlášení uživatele.
- F4.4. Získání informací o uživateli.
- F4.5. Získání seznamu videí.

2.2 Nefunkční požadavky

- N1. Videa přehrávat technologií HTML5 video, případně Flash.
- N2. Technologie PHP, Javascript, jQuery, CSS, HTML5, MySQL.
- N3. Podpora v novějších prohlížečích Chrome, Firefox, IE, Safari, Opera.
- N4. Podpora na moderních mobilních zařízeních.

2.3 Požadavky detailněji

- F1.1. Přehrávání videí – web bude umožňovat uživatelům přehrávat videa. Budou dva druhy přehrávání. Prvním druhem je přehrání jednoho videa o některém ze států a druhým spuštění tzv. *Programu*, kde se přehrávají všechna videa ze systému za sebou. Pokud bude chtít návštěvník přehrát pouze jedno video, tak se postupně přehrají: spot určený k přehrávání před tímto videem, ono požadované video, spot určený k přehrávání za tímto videem. *Program* bude složený z úvodního spotu a následně se bude střídat vždy video o některém ze států se spotem.
- F1.2. Správa uživatelských účtů – uživatelé se mohou do systému registrovat. Registrovaný uživatel se může přihlásit a měnit informace pro svůj účet.
- F1.3. Nabízení videí pro přehrání dle preferencí uživatelů – přihlášený uživatel si pro svůj účet může nastavit preferované státy. O těchto státech mu potom bude systém nabízet videa pro přehrání.
- F2.5. Ukládat informace o zhlédnutých videích – o každém návštěvníkovi stránek se systém bude snažit uchovávat informace, která videa mu byla přehrána. Na základě této informace potom budou generována videa pro *Program* tak, že dříve zhlédnutá videa budou zařazena na konec *Programu*.
- F3.1. Správa videí – administrátor bude moci přidávat nová videa, mazat videa, upravovat informace související s videi. Při úpravě videa bude moci videu nastavit hlasitost, která mu bude nastavena přehrávačem při přehrávání.
- F3.2. Správa spotů – administrátor bude moci spoty přidávat, mazat a upravovat jejich informace. Pro přidání nového spotu administrátor vybere zda bude spot uploadovat z počítače, nebo zadá URL videa z Youtube. Budou 3 druhy spotů. Prvním je úvodní spot pro playlist – ten bude v systému právě jeden. Druhým jsou spoty, které se budou přehrávat v *Programu*. Posledním druhem spotů jsou spoty, které administrátor přiřadí k určitému státu a zvolí, zda se tento spot bude přehrávat za, či před videi z tohoto státu.

Kapitola 3

Analýza dostupných přehrávačů

V této kapitole se budu zabývat již hotovými přehrávači. Budu zkoumat jejich vlastnosti a možnosti a porovnávat je s potřeby našeho přehrávače. Nejideálnější přehrávač vyberu pro moji práci a rozeberu ho detailně.

3.1 Obecně

HTML5 přehrávač představuje knihovna napsaná v jazyce Javascript, poskytující metody pro manipulaci s HTML elementem <video> a určující jeho vzhled pomocí CSS. Těchto přehrávačů je velká řada. Liší se v nabízených možnostech a případně cenách. Některé javascriptové knihovny využívají API elementu <video> a jiné na tomto API staví ještě API vlastní.

3.2 Výběr přehrávačů pro porovnání

V následující tabulce jsou uvedeny nabízené možnosti různých aktuálních HTML5 přehrávačů. Výběr jsem provedl na základě hrubé rešerše, která byla založena na rozšíření přehrávače na webu, doporučení uživatelů a spokojenosti uživatelů. Vybrané přehrávače splňují alespoň většinu požadavků. U některých přehrávačů jsou nabízeny i jeho placené varianty. Ty nejsou ve výběru zahrnuty.

Přehrávač	**MP4 H.264**	**Playlist**	**Podpora mobilních zařízení**	**Fallback na Flash**	**Youtube**	**Úprava vzhledu**	**Více zdrojů v playlistu**
FlowPlayer	ano	ano	ano	ano	ano	ano	ne
jMediaelement	ano	ano	ano	ano	ano	ano	ne
JWPlayer	ano	ano	ano	ano	ano	ano	ano
mediaelement.js	ano	ano	ano	ano	ano	ano	ne
Video.js	ano	ano	ano	ano	ano	ano	ne
SublimeVideo	ano	ano	ano	ano	ano	ano	ano
LeanBack	ano	ne	ano	ano	ano	ano	ne

Tabulka 3.1: Přehled podporavaných vlastností přehrávačů[35]

3.3 Výběr nejvhodnějšího

Z tabulky vyplývá, že naše potřeby nejvíce pokrývají přehrávače SublimeVideo a JWPlayer.

3.4 SublimeVideo

SublimeVideo přehrávač byl vyvinut v roce 2010 firmou Jilion. Byl prvním tzv. *cloud-based* přehrávačem, což znamená, že není třeba nahrávat skripty pro přehrávač na server, kde bude fungovat, ale vše potřebné se získává z cloudu.[36]
SublimeVideo dokáže přehrát videa v následujících formátech: MP4/H.264, WebM/VP8, Ogg/Theora.

3.4.1 API

SublimeVideo má své vlastní API postavené na API pro DOM objekt tagu `<video>` a využívá knihovnu jQuery. Pro získání kontroly nad přehrávačem a pro využití tohoto API musí mít element `<video>` nastavený atribut `class="sublime"`. Poté lze reprezentovat přehrávač jako objekt a využívat jeho metody a reagovat na události. To se v případě SublimeVideo provede následujícím způsobem:

```
sublime().ready(function(){
     var player = sublime.player("myVideoPlayer"); });
```

Proměnná `player` představuje kýžený objekt.

API poskytuje například následující metody:

- `play(), pause(), stop()` – pro spuštění, pozastavení a ukončení přehrávání.
- `duration()` – pro zjištění délky trvání videa.
- `videoHeight(), videoWidth()` – pro zjištění výšky a šířky přehrávače.

API umožňuje zpracovávát například tyto události:

- `start, pause, end` – události vyvolané na začátku, při pozastavení a na konci přehrávaní videa.
- `seek` – událost je vyvolána, jestliže probíhá seekování.
- `metadata` – událost je vyvolána, jakmile jsou dostupná metadata k videu.
- `timeUpdate` – událost je vyvolána pokaždé, kdy se změní čas přehrávání videa.[37]

Všechny metody a události jsou analogické k API, které je poskytováno DOM objektu elementu <video>.

3.4.2 Podpora na mobilních zařízeních

SublimeVideo slibuje přehrávání videa na všech mobilních platformách od nejnižších verzí.[38]. Při testování se však objevil problém. Pro operační systém Android verze 4.4 a vyšší přehrávač v nativním prohlížeči pro Android nefunguje. Vývojáři neplánují tento problém v blízké budoucnosti řešit.[39]

3.4.3 Vytvoření playlistu

Playlist podporuje kombinaci zdrojů videa. Můžeme tak jako zdroj videí využívat Youtube v kombinaci s videi z lokálního serveru. Playlist se vytvoří pomocí HTML elementů a ty jsou následně zpracovávány javascriptem.

Příklad vytvoření playlistu:

```
<div id="playlist1" class="sv_playlist">
  <div class="video_wrap">
    <video id="video1" width="640" height="360">
      <source src="video1.mp4" />
    </video>
  </div>
  <div class="video_wrap">
    <video id="video2" width="640" height="360"
     data-youtube-id="UNg9gQsck1c">
    </video>
  </div>
</div>
```

Uvedený kód představuje playlist, ve kterém budou dvě videa. Jedno z lokálního serveru a druhé z Youtube. Pro přehrání videa z Youtube musíme nastavit atribut `data-youtube-id` na hodnotu odpovídající ID videa. Toto ID se udává na Youtube u videa v URL jako hodnota parametru `v`.
O přehrávání se dále stará javascript.

3.4.4 Výsledek

Ač po prvotní analýze byl přehrávač SublimeVideo vyhovující pro naši aplikaci, tak se po hlubším testování jeví jako nepoužitelný. Nemožnost přehrávat videa na nových verzích operačního systému Android je závažný problém.

Více vyhovující bude JWPlayer, který budu analyzovat podrobněji.

3.5 JWPlayer

JWPlayer byl vyvinut americkou firmou LongTail Ad Solutions, Inc. Přehrávač je vlajkovou lodí této firmy. Jedná se o open source, který je poskytován zdarma pro nekomerční účely. Pro komerční účely si lze zakoupit některou z licencí.
JWPlayer je nabízen v několika variantách:

- *FREE* – základní konfigurace, která je zdarma. V přehrávači bude stále vidět logo JWPlayer.
- *PRO* – konfigurace, která je stojí 149 dolarů ročně. Mimo základní funkce nabízí navíc skiny, možnost vlastního loga, analýzy.
- *PREMIUM* – tato verze stojí 299 dolarů za rok. Nabízí navíc ještě podporu Apple HLS streamingu a pokročilé analýzy, jako například Google Analytics.[40]

JWPlayer podporuje následující formáty videí: MP4/H.264, WebM/VP8, FLV/H.263.

3.5.1 API

JWPlayer má na API pro DOM objekt <video> postavené své API a uživatel využívá výhradně toto API.
Některé zajímavé metody API budou uvedeny níže, ostatní metody, které jsou při tvoření projektu využívány se nacházejí v příloze B. Popis kompletního API je přístupný z: `http://support.jwplayer.com/customer/portal/articles/1413089-javascript-api-reference`

3.5.1.1 Inicializace

Inicializace přehrávače probíhá pomocí HTML a javascriptu následovně:

```
<div id="myPlayer"></div>
<script type="text/javascript">
    jwplayer("myPlayer").setup({
      file: "myVideo.mp4",
      height: 360,
      width: 640
   });
</script>
```

Metoda `setup` je základní a pro vytvoření přehrávače musí být zavolána. Obsahuje veškerou konfiguraci přehrávače. Konfigurace se zapisuje ve formátu JSON. Metoda `setup` musí mít minimálně jednu položku, a to `file`.
Na základě této konfigurace jsou vykonávány operace nad objektem `<video>`, případně je použita Flash technologie.
Možné atributy pro metodu `setup` včetně popisu jejich implementace jsou dostupné v příloze B.1.

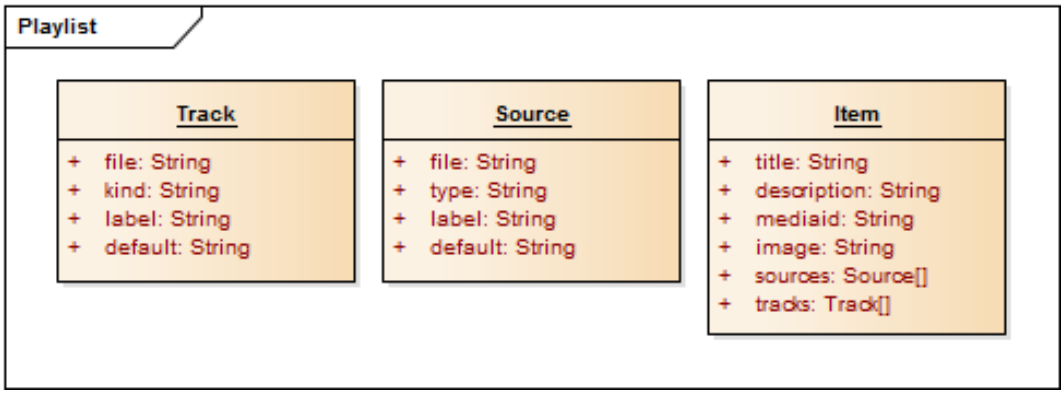
3.5.1.2 Playlist

Playlist umožňuje načíst více videí, která budou přehrávána automaticky za sebou. Zapisuje se jako položka konfigurace v metodě `setup`. Má následující tvar:

```
jwplayer("myPlayer").setup({
    playlist: [{
       image: "preview1.jpg",
       sources: [{
          file: "video1.mp4" }],
       title: "My Video"
    }]
}];
```

Atributy a metody pro `playlist` jsou popsané v příloze B.2.

Playlist pro přehrávač je implementován jako pole, které obsahuje jednu nebo více položek pro přehrávání. Položky tohoto pole představují objekty typu `Item`, jejichž struktura je zobrazena na obrázku 3.1. Atribut `sources` je pole obsahující zdroje videí ve formě objektu `Source` a atribut `tracks` je pole obsahující stopy videí ve formě objektu `Track`. Struktura objektů `Source` a `Track` je zobrazena taktéž na obrázku 3.1.



Obrázek 3.1: Reprezentace playlistu

Přehrávač pracuje s playlistem tak, že se pro každý `item` začne procházet jeho pole `sources` a testuje se, zda se video, jehož zdroj udává `sources[].file`, dá přehrát jako HTML5 video, nebo pomocí Flash, případně, jestli se jedná o video z Youtube. Ty zdroje, které nelze přehrát se odstraní. Pro jedno video se vždy vybere jen jeden zdroj, a to buď na základě volby `sources[].default`, nebo se vezme první přehratelný zdroj.
Obdobně se projde pole `tracks`, kde se vybírá stopa.
Po těchto operacích jsou položky připraveny k přehrávání.

3.5.2 Fallback na Flash

Ne všechny prohlížeče a přístroje dokáží přehrát video jako HTML5 video. JWPlayer proto nabízí fallback na Flash, který video přehraje i ve starších prohlížečích a přístrojích pomocí pluginu např. Adobe Flash Player.
Vyhodnocení, zda prohlížeč je schopen přehrávat jako HTML5 video probíhá následovně:

- Určí se typ videa. Jedna z hodnot:
  - video/ogg,
  - video/mp4,
  - video/webm.
- Zavolá se metoda DOM objektu elementu <video> `canPlayType(type)`. Jako parametr se předá určený typ videa.
- Vyhodnotí se návratová hodnota:
  - `probably` – bude se přehrávat pomocí HTML5,
  - `maybe` – bude se přehrávat pomocí HTML5,
  - `""` – bude se přehrávat pomocí Flash.

3.5.3 Videa z Youtube

JWPlayer nabízí podporu přehrávání videí z Youtube. Videa mohou být přehrávána samostatně i v playlistu.
Pro přehrání videa stačí v inicializačním atributu `file` předat odkaz na video na Youtube. V závislosti na prohlížeči a jeho podpoře HTML5 videí se zvolí, zda se bude využívat Youtube API pro Flash, nebo Youtube API pro iframe (HTML5). API, které pro videa poskytuje JWPlayer, se sesynchronizuje s vybraným Youtube API. Pro volání metod tedy není rozdíl, zda se video přehrává z Youtube, nebo z lokálního serveru. Videa z Youtube obsahují reklamy, které nelze potlačit.

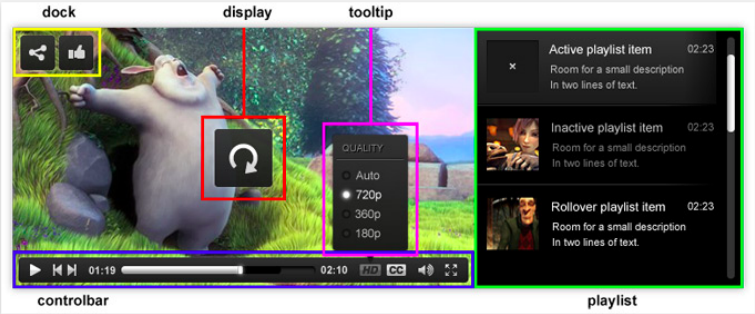
3.5.4 Úprava vzhledu přehrávače

JWPlayer umožňuje upravit vzhled přehrávače pomocí skinů. Nabízí 8 variant pro výběr a umožňuje vytvářet skiny vlastní.
Skiny jsou definovány XML souborem. Struktura XML souboru je následující:

```
<skin target="6.0" name="mySkin" author="David">
    <components>
      <component name="controlbar">
        <settings>
          <setting name="..." value="..." />
          ...
        </settings>
        <elements>
          <element name="..." src="..." />
        </elements>
```

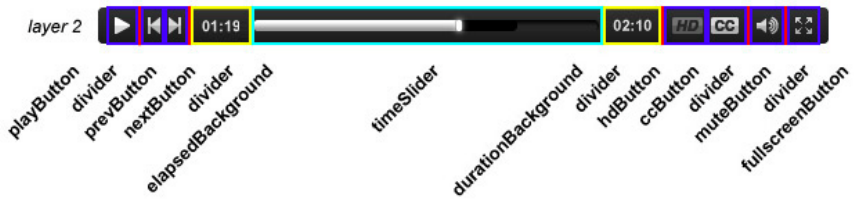
```
        </component>
        ...
    </components>
</skin>
```

XML soubor obsahuje komponenty (elementy `component`). Komponenty jsou části, ze kterých se přehrávač skládá. V jednom souboru je jich celkem 5. Jejich názvy jsou `controlbar`, `display`, `dock`, `playlist` a `tooltip`. Odpovídají částem přehrávače, znázorněným obrázkem:



Obrázek 3.2: Struktura skinu přehrávače[2]

Každý element `component` obsahuje dva podelementy – `settings` a `elements`. `settings` se skládá z elementů `setting`, které jsou tvořeny atributy `name` a `value`. Nastavují se zde neobrázkové vlastnosti komponenty, jako například velikost písma. Vlastnosti, které zde nejsou nastavené, budou mít standardní hodnoty.
`elements` se skládá z elementů `element`. Ty obsahují atribut `name`, udávající název prvku přehrávače, a `src`, určující cestu k obrázku pro tento prvek. Nastavuje se zde tedy vzhled prvků pomocí obrázku. Prvky, které zde vynecháme, nebudou v přehrávači zobrazeny. Názvy prvků zobrazuje následující obrázek:



Obrázek 3.3: Názvy prvků přehrávače[2]

Kapitola 4

Návrh

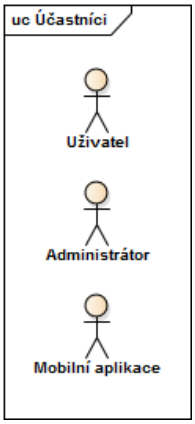
V této kapitole bude nejprve pomocí modelování případů užití (*Use Cases*) zobrazen současný stav aplikace a jak tento stav bude rozšířen. Ze současného stavu budou vybrány pouze ty funkce systému, které souvisí s tématem této práce. Dále budou uvedeny informace o architektuře aplikace a používaných technologiích. Následovat bude databázová vrstva a její rozšíření. Nakonec přijde návrh implementace nových částí aplikace.

4.1 Současný stav a jeho rozšíření

Při modelování případů užití budou oranžovou barvou znázorněny ty funkce systému, které jsou již hotové a nebude se v nich nic měnit. Zelenou barvou budou znázorněny ty funkce, které budou v aplikaci nové nebo v aplikaci již jsou, ale budou v nich vykonávány změny.

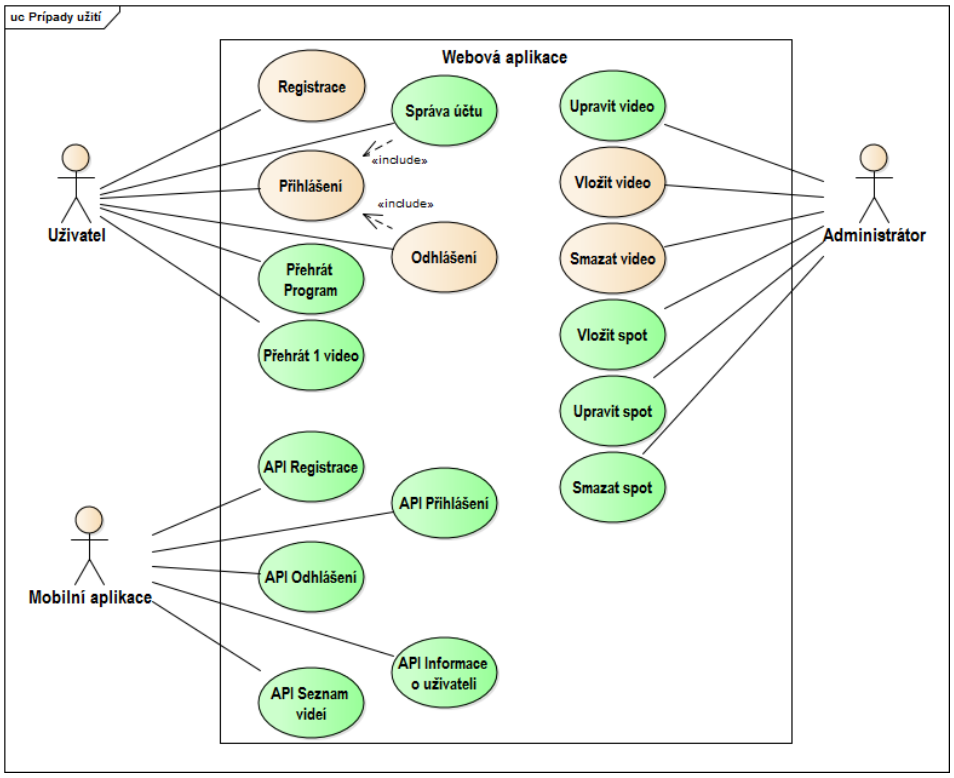
4.1.1 Seznam účastníků

- **Uživatel** – uživatel je jakýkoliv návštěvník stránek. Uživatel se může do systému registrovat. Pokud je již zaregistrován, tak se může přihlásit a upravovat nastavení svého účtu.
- **Administrátor** – stará se o obsah webu. Přidává, maže a mění informace o videích a spotech.
- **Mobilní aplikace** – zasílá do systému požadavky a očekává odpovědi.



Obrázek 4.1: Seznam účastníků

4.1.2 Diagram případů užití



Obrázek 4.2: Případy užití

4.2 Používané technologie

Stávající aplikace je realizována pomocí technologií uvedených níže. Tyto technologie budou použity i při rozšiřování aplikace.

PHP – *Hypertext Preprocessor* je programovací jazyk, který se stal nejrozšířenějším skriptovacím jazykem pro web. Je navržený především pro programování dynamických webových aplikací. PHP je interpretován na serveru a generuje HTML (či jiný) výstup. Oblíbený se stal díky své jednoduchosti a velkému množství funkcí. PHP je nezávislý na platformě, to umožňuje přenášet PHP skripty mezi různými operačními systémy takřka bez úprav.
PHP podporuje mnoho knihoven, například knihovnu pro přístup k databázi.[41][42]

HTML – *HyperText Markup Language* je značkovací jazyk. Je jedním z hlavních jazyků pro vytváření webových stránek. HTML dokument má předepsanou strukturu a množinu značek (tagů, elementů).[43] Značky se dělí na: *strukturální*, které udávají formu dokumentu (např. odstavec `<p>`), *sémantické*, které popisují povahu prvku (např. nad-

pis `<title>`) a *stylistické*, které určují vzhled prvku (např. kurzíva `<i>`). V této webové aplikaci se používá nejnovější verze jazyka – HTML5.

CSS – kaskádové styly (*Cascading Style Sheets*) jsou kolekcí metod pro grafickou úpravu webových stránek. Cílem je oddělit strukturu dokumentu s obsahem od vzhledu. Definice stylu se váže na HTML elementy a určuje jejich vzhled.[43]

Javascript – javascript byl popsán v kapitole 1.6.

jQuery – malá, rychlá javascriptová knihovna. Klade důraz na interakci mezi HTML a Javascriptem. Umožňuje měnit atributy a CSS hodnoty HTML elementů. Také dokáže zpracovávat události. Zvládá také různé animace, Ajax a další. [44]

MySQL – multiplatformní databázový systém. Využívá se pro webové služby a je nainstalovaný na serveru. Je ideální pro malé i velké aplikace. Ke komunikaci využívá dialekt jazyka SQL.[45]

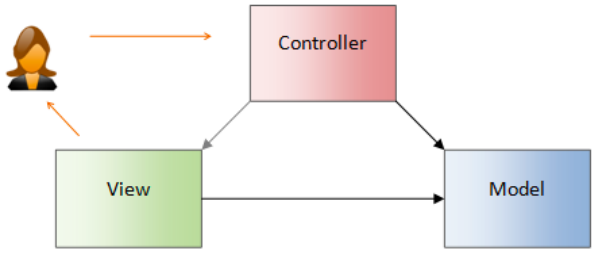
4.3 Architektura

Stávající aplikace je postavená na architektuře MVC (Model-View-Controller). Tato architektura rozděluje datový model (M), uživatelské rozhraní (V) a řídící logiku (C) do tří nezávislých komponent. Změna jedné části má tak minimální vliv na zbylé 2 části.[46] Komponenty jsou následující:

- *Model* – reprezentuje data a business logiku aplikace.
- *View* – zobrazuje data z modelu a další prvky uživatelského rozhraní.
- *Controller* – stará se o funkčnost a provázanost aplikace. Reaguje na události, aktualizuje Model a View.

Tok událostí:

1. Uživatel vykoná akci v uživatelském prostředí.
2. Akce je zachycena Controllerem.
3. Controller reaguje – změní hodnoty v Model nebo přímo ovlivní View.
4. View zobrazí změny uživateli.



Obrázek 4.3: MVC architektura [3]

4.3.1 Registr

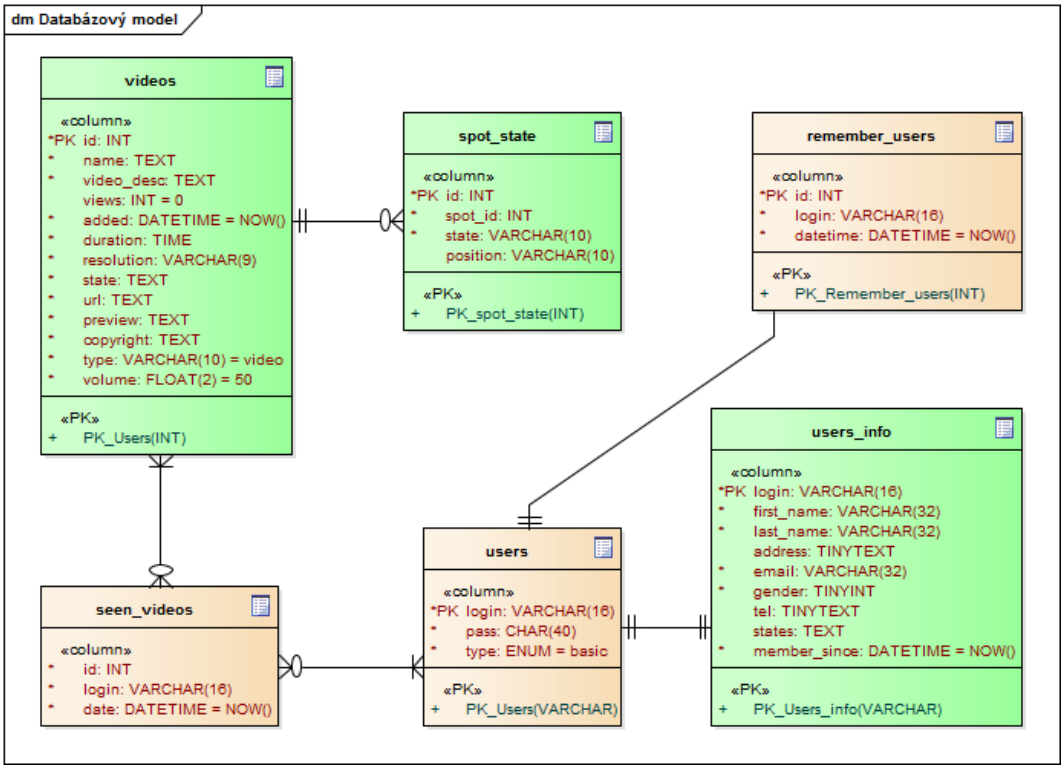
V aplikaci je využito návrhového vzoru *Singleton* pro implementaci centrálního registru. Tento registr je přístupný ze všech vrstev architektury a umožňuje tak předávat data mezi vrstvami. Registr nepatří do žádné z vrstev architektury MVC, stojí samostatně. Registr je asociativní pole. Ve vrstvách Controller a View k němu přistupujeme pomocí `$this->registry`. Data do něj se mohou vložit například následovně: `$this->registry["start_spot"] = "startSpot1.mp4"`.

4.4 Rozšíření aplikace

Rozšíření aplikace o spoty, nové požadavky na přehrávání a administraci si vyžadují zásah do stávající aplikace. Rozšiřování bude aplikaci doplňovat, základní stavební kameny aplikace tedy zůstanou nezměněny.
V zobrazovaných modelech budou nově přidané nebo pozměněné prvky zobrazovány zelenou barvou. Původní prvky budou zobrazovány oranžovou barvou.

4.4.1 Databáze

Následující databázový model zobrazuje, v jakých tabulkách jsou data uchovávána.



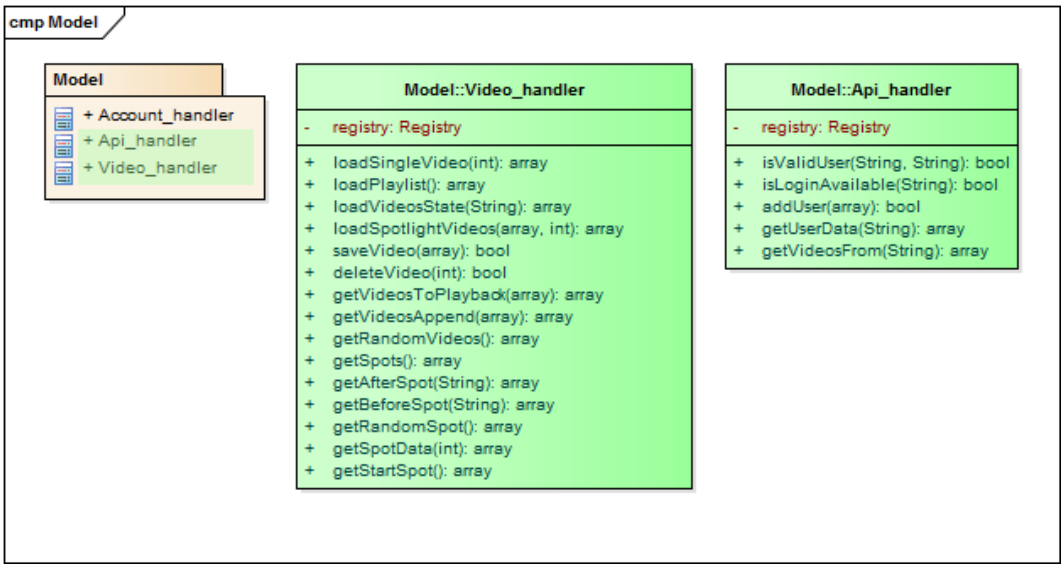
Obrázek 4.4: Databázový model

Dále budou popsány vyobrazené tabulky. U tabulek budou popsány jen ty sloupce, u kterých je třeba něco zdůraznit či vysvětlit.

- *videos* – představující všechna videa v systému, včetně reklamních spotů.
  Význam některých sloupců tabulky:
  - `state` – stát, ke kterému je video přiřazeno.
  - `url` – cesta ke zdroji videa na lokálním serveru, nebo URL na Youtube.
  - `type` – určuje typ videa. Může být video o některém státu, úvodní spot nebo spot.
  - `volume` – hodnota hlasitosti videa. Výchozí hodnota bude 50, aby šla hlasitost zvýšit i snížit.
- *spot_state* – jelikož bude umožněno nastavit pro videa určitých států reklamní spoty, které se budou přehrávat na začátku a na konci videí, vzniká tato tabulka.
  Význam některých sloupců tabulky:
  - `spot_id` – identifikátor videa z tabulky *videos*, které představuje tento spot.
  - `state` – stát, ke kterému je spot přiřazen.
  - `position` – jedna z hodnot `before`, `after` určující, zda tento spot bude přehrán před, nebo za videi státu `state`.
- *seen_videos* – uchovává videa zhlédnutá přihlášeným uživatelem.
- *users* – uchovává přihlašovací údaje a roli uživatele. Význam některých sloupců tabulky:
  - `type` – role uživatele. Běžný uživatel má roli `basic`, administrátor má roli `admin`.
- *users_info* – uchovává informace o uživatelích.
  Význam některých sloupců tabulky:
  - `login` – přihlašovací jméno uživatele.
  - `states` – státy preferované uživatelem. Pokud uživatel preferuje více států, tak tvoří řetězec, kde jsou jednotlivé státy odděleny čárkou.
  - `member_since` – datum uživatelovy registrace.
- *remember_users* – uchovává uživatele, kteří chtějí, aby si web pamatoval jejich přihlašovací údaje.

4.4.2 Vrstva Model

V modelové vrstvě se nacházejí tři třídy. Třídy komunikují přímo s databází – v jejich metodách se provádí databázové dotazy.



Obrázek 4.5: Modelová vrstva – třídy

Api_handler

Třída využívaná při zpracování požadavků z mobilní aplikace. Jediným atributem je `registry`. Ten je využíván pro přenos různých dat napříč celou aplikací. Více o Registru v kapitole 4.3.1. Popis jednotlivých metod je v následující tabulce.

Metoda	Popis
isValidUser	Jako parametry přijímá uživatelské jméno a heslo. Následně zkontroluje, zda uživatel s takovými údaji v systému existuje. V případě, že existuje, tak vrátí true, jinak false.
isLoginAvailable	Parametrem je uživatelské jméno. Metoda zjistí, zda se již takové uživatelské jméno v systému nachází. Pokud se v systému nachází, tak vrací true, jinak false.
addUser	Přidá uživatele do systému. Jako parametr metodě byla předána data ve formě asociativního pole s informacemi o uživateli. V případě úspěšného přidání vrací true, jinak false.
getUserData	Vrátí informace o uživateli. Uživatele identifikuje pomocí parametru, kterým je uživatelské jméno. Výsledek je ve formě asociativního pole.
getVideosFrom	Vrátí všechna videa, která byla přidána do systému od data předaného jako parametr. Datum se předává metodě jako timestamp. Výsledek je ve formě asociativního pole.

Tabulka 4.1: Popis metod třídy Video_handler

Video_handler

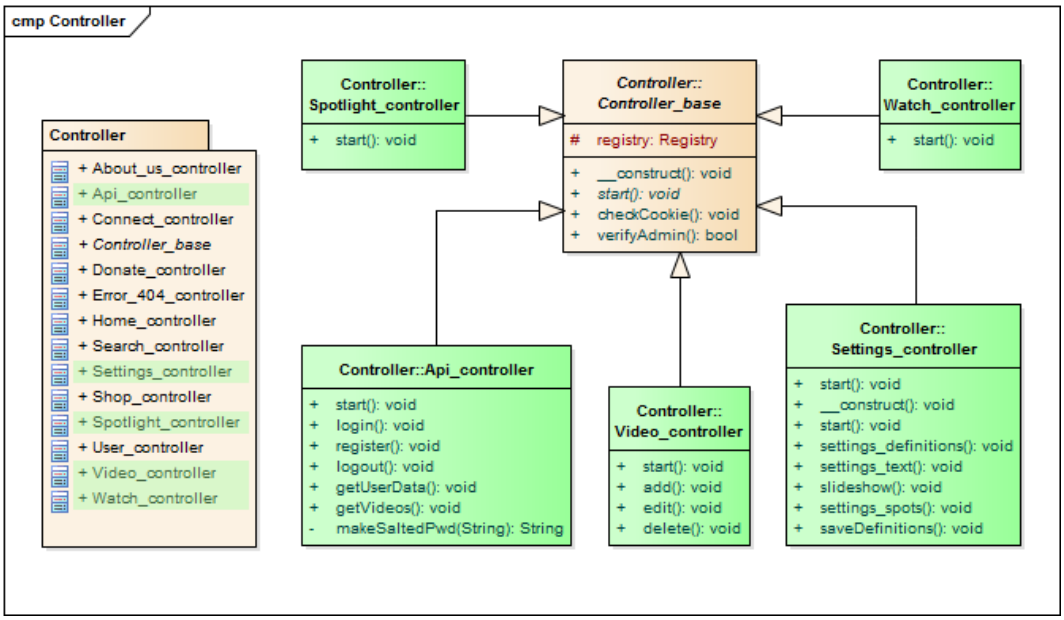
Třída, která získává, aktualizuje a maže data spojená s videi. Stejně jako třída *Api_handler* má jeden atribut `registry`. Popis metod třídy je v následující tabulce.

Metoda	**Popis**
loadSingleVideo	Získá z databáze video podle ID předaného jako parametr.
loadPlaylist	Získá videa pro sestavení playlistu.
loadVideosState	Získá videa patřící ke státu předaného jako parametr.
loadSpotlightVideos	Získá videa patřící ke státům preferovaným uživatelem. Preferované státy jsou předány jako parametr ve formě pole. Požadavek na maximální počet takových videí je určen druhým parametrem.
saveVideo	Přidá nové video do systému. Údaje o novém videu jsou předány jako parametr. V případě úspěšného přidání vrací true, jinak false.
deleteVideo	Smaže video ze systému. Jako parametr se předá ID videa, které má být smazáno.
getVideosToPlayback	Získá z databáze taková videa, která nejsou v poli předaném jako parametr. Videa vrací ve formě pole. Metoda se využívá při generování playlistu, kdy chceme videa, která uživatel zatím neviděl.
getVideosAppend	Získá z databáze taková videa, která jsou v poli předaném jako parametr. Videa vrací ve formě pole. Metoda se využívá při generování playlistu, kdy chceme videa, která uživatel už viděl.
getRandomVideos	Provede výběr všech videí z databáze v náhodném pořadí. Videa vrací ve formě pole. Využívá se, pokud uživatel ještě nezhlédl žádné video.
getSpots	Vybere a vrátí ve formě pole všechny spoty z databáze.
getAfterSpot	Získá z databáze spoty pro videa, které mají být přehrány po těchto videích, patřící ke státu předanému jako parametr. Spoty vrací ve formě pole.
getBeforeSpot	Získá z databáze spoty pro videa, které mají být přehrány před těmito videi, patřící ke státu předanému jako parametr. Spoty vrací ve formě pole.
getRandomSpot	Vybere náhodný spot z databáze a vrátí ho ve formě pole.
getSpotData	Získá z databáze informace o spotu. Identifikace spotu probíhá na základě ID předaného jako parametr.
getStartSpot	Získá z databáze úvodní spot a vrátí ho ve formě pole.

Tabulka 4.2: Popis metod třídy Api_handler

4.4.3 Vrstva Controller

O každou stránku webové prezentace se stará některá třída z této vrstvy. Třída se jmenuje stejně jako stránka, o kterou se stará. Základním stavebním kamenem této vrstvy je třída s názvem *Controller_base*. Od této třídy dědí třídy ostatní. V metodách jednotlivých tříd se využívá služeb objetků *Video_handler* a *Api_handler*.
Na následujícím obrázku je zobrazen seznam všech tříd spadající do této vrstvy a také modely tříd, které vznikaly nově nebo byly za účely této práce změněny.



Obrázek 4.6: Vrstva Controller – třídy

Controller_base – má atribut `registry` (viz. 4.3.1) a některé metody. Jelikož je rodičovskou třídou všem ostatním třídám v této vrstvě, tak tento atribut a metody mohou využívat i tyto třídy. Metoda *`start()`* je abstraktní a je nutné, aby byla implementována v třídách potomků. Metoda `checkCookie()` zkontroluje uložené cookies o tomto uživateli a metoda `verifyAdmin` ověří, zda je tento uživatel administrátor.

Api_controller – třída zpracovává požadavky přicházející z mobilní aplikace. Některé požadavky posílají POST nebo GET parametry. Metody požadavky zpracují a vygenerují výstup ve formátu JSON (struktura odpovědí bude uvedena v následující kapitole Implementace). Popis metod:

Metoda	Popis
start()	Připraví HTML stránku pro vypisování výstupu Api_controlleru.
login()	Přihlásí uživatele do systému. Očekává dva POST parametry – uživatelské jméno a heslo.
register()	Registruje do systému nového uživatele. Údaje o uživateli budou obsaženy v POST parametrech.
logout()	Odhlásí uživatele.
getUserData()	Získá informace o právě přihlášeném uživateli.
getVideos()	Získá videa, která byla přidána do systému od určitého data. Datum bude zadáno jako timestamp v GET parametru.
makeSaltedPwd()	Převede heslo zadané v parametru do jeho zaheshované podoby. Funkce se využije pro ověření správnosti přihlašovacích údajů.

Tabulka 4.3: Popis metod třídy Api_controller

Settings_controller – tato třída je jedna z těch, které obsluhují administraci webu. Pro účely této práce bude do této třídy přidána metoda `settings_spots()`. Ta bude zpracovávat všechny požadavky na manipulaci se spoty (přidávání, editace, mazání).

Spotlight_controller – třída obsluhující na webu stránku Spotlight, na které jsou uživateli nabízena videa dle jeho preferencí. Obsluha bude prováděna v metodě `start()`.

Video_controller – třída obsluhující veškeré operace s videi v administraci. Pro tuto práci bude upravena její metoda `edit()`. Rozšíření této metody bude spočívat v přidání údaje o hlasitosti videa.

Watch_controller – třída obsluhující stránku Watch, na které probíhá veškeré přehrávání videí na webu. Obsluha bude prováděna v metodě `start()`.

4.4.4 Vrstva View

Ve vrstvě View je implementováno kompletní uživatelské rozhraní. Jsou zde skripty, které definují vzhled stránek. Typicky je to tak, že každý controller má v této vrstvě odpovídající view. Skripty zobrazují obsah pomocí HTML. Obsah může být v některých případech statický a v některých je generován pomocí PHP a zobrazuje data předaná controllerem. Z controlleru do view se data předávají pomocí registru. Skripty z této vrstvy data nijak nezpracovávají, pouze je prezentují.
Za zmínku stojí skript watch.php, který reprezentuje stránku pro přehrávání Watch. Tento skript zobrazuje, mimo jiné, přehrávač JWPlayer. Nachází se zde také veškerá obsluha přehrávače, která probíhá v jazyce javascript a také se zde nastavuje a aktualizuje cookie, která uchovává informaci o tom, jaká videa uživatel zhlédnul. Obsluha nemohla být přesunuta do odděleného javascriptového souboru, protože se ve scriptu využívá proměnných z registru.

Kapitola 5

Implementace

V této kapitole bude popsána implementace požadavků na aplikaci. Ukázky zdrojových kódů jsou redukovány tak, aby pouze nastínily řešení. Nejsou v nich nezajímavé operace, jako například inicializace polí.

5.1 Přehrávání videí

Popis implementace 2 druhů požadavků na přehrávání videí.

1. Přehrání jednoho videa.
   URL požadavku na přehrání jednoho videa může vypadat například takto:
   `http://www.americantravelshow.com/watch/64/?watch=no`
   Číslo 64 v URL určuje ID videa, které má být přehráno. Parametr `watch=no` je indikátor přehrávání pouze jednoho videa. To, jestli se bude přehrávat pouze 1 video, nebo celý *Program* je důležité, protože pro každý druh přehrávání se zobrazují na stránce některé odlišné ovládací prvky.
   Požadavek je zpracováván *Watch_controllerem* následovně:
   a) Vyvolá se metoda `start()` a zjistí se, zda se bude přehrávat pouze 1 video.
   b) V metodě se vytvoří objekt:
   `$videoHandler = new Video_handler();`
   c) Získá se záznam požadovaného videa z databáze:
   `$video = $videoHandler->loadSingleVideo(64);`. Tato metoda vrátí výsledek jako jednorozměrné asociativní pole. Klíče do pole jsou pojmenovány podle sloupců tabulky *videos*.
      - Metoda `loadSingleVideo(64)` provede databázový dotaz:
        `"SELECT * FROM videos WHERE id = '64'"`
   d) Stát, ke kterému video patří, se uloží do proměnné:
   `$videoState = $video["state"];`
   e) Získá se spot, který bude přehrán před tímto videem a spot, který bude přehrán za tímto videem:
   `$spotBefore = $videoHandler->getBeforeSpot($videoState);`
   `$spotAfter = $videoHandler->getAfterSpot($videoState);`

- Metoda `getBeforeSpot($videoState)` provede rekurzivní databázový dotaz:
```
"SELECT * FROM videos WHERE id =
(SELECT spot_id FROM spot_state
WHERE state = '" . $videoState . "'
AND position = 'before' ORDER BY rand() LIMIT 1)"
```
- Metoda `getAfterSpot($videoState)` provede stejný dotaz, akorát místo `position = 'before'` bude `position = 'after'`
- Pokud pro daný stát nebude v systému žádný spot, metoda `getBeforeSpot` nebo `getAfterSpot` nevrátí žádný záznam, tak se vybere náhodný spot, patřící k jinému státu. Pokud ani takový spot v systému nebude, tak video bude přehráno samostatně.

f) Nyní se vytvoří pole, které bude předáno do *Watch*. Ve *Watch* potom bude využito pro definování playlistu přehrávače.
Vytvoření a naplnění pole:
```
...
$data[$index]["mediaid"] = $spotBefore["id"];
$data[$index]["title"] = $spotBefore["name"];
$data[$index]["sources"][0]["file"] =
$spotBefore["url"];
$data[$index]["image"] = $spotBefore["preview"];
$data[$index]["description"] = $spotBefore["video_desc"];
$data[$index]["volume"] = $spotBefore["volume"];
...
```
Dále se bude plnit pole `$data` hodnotami z `$video` a na konec ze `$spotAfter`.

g) pole `$data` tedy obsahuje 3 videa, uložená v pořadí, ve kterém se mají přehrávat. Toto pole se uloží do registru, kde ho poté využije view *Watch*:
```
$this->registry["playlist"] = $data;
```

Následuje popis implementace view *Watch*:

a) HTML část.
V hlavičce se do stránky vkládá řídící skript JWPlayeru a kaskádové styly:
```
<head>
  <script src="http://jwpsrv.com/library/
   kX6TDkf6EeKq6iIACp8kUw.js"></script>
  <link href="styles.css" rel="stylesheet" type="text/css">
</head>
```

V těle stránky bude vyhrazen prostor pro přehrávač:
```
<body>
  <div id="videoPlayer">Loading the player...</div>
</body>
```

b) Javascriptová část.
Veškeré operace s přehrávačem, se provádí pomocí javascriptu.
Nejprve bude popsána implementace inicializační metody `setup`. Popis bude rozdělen do několika částí:

- Vzhled.
  Pro přehrávač bude použit vlastní skin:
  `skin: "/skins/atsSkin.xml"`
  Do přehrávače bude umístěno logo American Travel Show:
  `logo: { file: "/images/logo.png" }`
- Playlist.
  Controller *Watch_controller* připravil data pro playlist jako pole, jehož struktura odpovídá struktuře playlistu. Playlist je zapsán ve formátu JSON, takže bude použita PHP funkce `json_encode`. Tato funkce vytvoří z pole řetězec odpovídající formátu JSON:
  ```
  playlist: <?php echo json_encode(
            $this->registry["playlist"]);?>
  ```
- Ostatní.
  Video se začne přehrávat ihned po načtení stránky.
  `autostart: true`

Metoda `setup` tedy bude vypadat následovně:

```
jwplayer("videoPlayer").setup({
   skin: "/skins/atsSkin.xml"
   logo: { file: "/images/logo.png" },
   playlist: <?php echo json_encode(
               $this->registry["playlist"]);?>,
   autostart: "true" });
```

2. Přehrání playlistu.
   Pro přehrání playlistu bude URL vypadat takto:
   `http://www.americantravelshow.com/watch`
   Zpracování *Watch_controllerem* je následující:
   a) Vyvolá se metoda `start()`.
   b) V metodě se vytvoří objekty:
   `$videoHandler = new Video_handler();`
   c) Z databáze se získá úvodní spot
   `$startSpot = $videoHandler->getStartSpot();`
   - Metoda `getStartSpot()` provede následující dotaz:
     `"SELECT * FROM videos WHERE type = 'start_spot'"`
     Výsledkem bude 1 záznam, protože úvodní video bude v systému právě jedno.

   d) Získají se spoty, které se budou přehrávat vždy, po dokončení přehrávání videa o některém ze států:
   `$spots = $videoHandler->getSpots();`

- Metoda `getSpots()` provede následující dotaz:
  `"SELECT * FROM videos WHERE type = 'spot'"`

e) Spoty jsou uloženy v dvourozměrném asociativním poli `$spots`, které má strukturu `$spots[`*`index`*`][`*`db_sloupec`*`] = `*`value`*`.`
Budou uloženy do pole požadovaného formátu:

```
...
foreach($spots as $spotData) {
  $spotsArray[$spotIndex]["id"] = $spotData["id"];
  $spotsArray[$spotIndex]["name"] = $spotData["name"];
  $spotsArray[$spotIndex]["source"] =
  $spotData["url"];
  $spotsArray[$spotIndex]["preview"] =
  $spotData["preview"];
  $spotsArray[$spotIndex]["volume"] =
  $spotData["volume"]; }
...
```

f) Získají se videa o jednotlivých státech, která se budou přehrávat v playlistu. K tomu slouží dvě metody: `getVideosToPlayback` a `getVideosToAppend`. Obě jako parametr očekávají pole s videi, která uživatel již zhlédl. Uživatelem zhlédnutá videa se ukládají do cookie jako řetězec ve tvaru *12x87x6x100x2...*, kde čísla jsou ID zhlédnutých videí a znak "x" je použit jako oddělovač jednotlivých identifikátorů videí. Pro získání jednotlivých ID a jejich uložení do pole se použije funkce `explode`, která rozdělí řetězec podle znaku a výsledek uloží do pole. Zhlédnutá videa se pomocí funkce `array_merge` uloží do pole za ještě nezhlédnutá videa.

i.
```
$played = explode("x", $_COOKIE["played_videos"]);
$unSeen = $videoHandler->getVideosToPlayback($played);
$seen = $this->videoRepository->getSeenVideos($played);
$videos = array_merge($unSeen, $seen);
```

- Metoda `getVideosToPlayback` provede databázový dotaz, který vybere všechna videa kromě těch, která byla zhlédnuta. Videa budou vybrána v náhodném pořadí, aby se generovaly různé playlisty.
  `"SELECT * FROM videos WHERE type = 'video' AND id NOT IN ("`
  `. implode(",", $played) . ") ORDER BY rand()"`
- Metoda `getSeenVideos` se liší od metody `getVideosToPlayback` pouze v databázovém dotazu. Zde:
  `"SELECT * FROM videos WHERE type = 'video' AND id IN ("`
  `. implode(",", $played) . ") ORDER BY rand()"`

g) Vytvoří se pole `$data`, do kterého budou vkládány spoty a videa v pořadí, v jakém budou přehrávány. Na první pozici v poli se vloží úvodní spot:

```
$index = 0;
$data[$index]["mediaid"] = $startSpot["id"];
$data[$index]["title"] = $startSpot["name"];
$data[$index]["sources"][0]["file"] =
$startSpot["url"];
$data[$index]["image"] = $startSpot["preview"];
$data[$index]["description"] = $startSpot["video_desc"];
$data[$index]["volume"] = $startSpot["volume"];
$index++;
```

Na další pozici se vloží video o některém státu z pole `$videos`, které bude vždy následováno reklamním spotem:

```
$spotsIndex = 0;
foreach($videos as $videoData) {
   $data[$index]["mediaid"] = $videoData["id"];
   $data[$index]["title"] = $videoData["name"];
   $data[$index]["sources"][0]["file"] =
   $videoData["url"];
   $data[$index]["image"] = $videoData["preview"];
   $data[$index]["description"] =
   $videoData["video_desc"];
   $data[$index]["volume"] = $videoData["volume"];
   $index++;

// $spotsCount je počet spotů navrácených z databáze
   if($spotsCount > 0) {
      $spotsArrayIndex = $spotsIndex % $spotsCount;
      $data[$index]["mediaid"] =
      $spotsArray[$spotsArrayIndex]["id"];
      $data[$index]["title"] =
      $spotsArray[$spotsArrayIndex]["name"];
      $data[$index]["sources"][0]["file"] =
      $spotsArray[$spotsArrayIndex]["url"];
      $data[$index]["image"] =
      $spotsArray[$spotsArrayIndex]["preview"];
      $data[$index]["description"] = "";
      $data[$index]["volume"] =
      $spotsArray[$spotsArrayIndex]["volume"];
      $spotsIndex++;} }
```

h) Pole s videi se uloží do registru a bude přístupné z view *Watch*

```
$this->registry["playlist"] = $data;
```

Implementace view *Watch* je stejná jako při přehrávání jednoho videa (1). Liší se pouze javascriptová část, kde byly doplněny reakce na některé události. Metody umožňující události odchytávat jsou popsány v příloze B.
Rozšíření javascriptové části:

- Nastavení hlasitosti videa:
```
jwplayer("videoPlayer").onPlay(function() {
   var player = jwplayer("videoPlayer");
   var videoObject = player.getPlaylistItem();
   var volume = videoObject["volume"];
   player.setVolume(volume); )}
```
- Videa, která nelze přehrát, budou přeskočena:
```
jwplayer("videoPlayer").onError(function() {
   var player = jwplayer("videoPlayer");
   player.playlistItem(player.getPlaylistIndex() + 1); )}
```
- Po přehrání videa se toto video uloží do cookie:
```
jwplayer("videoPlayer").onComplete(function() {
   var player = jwplayer("videoPlayer");
   var videoObject = player.getPlaylistItem();
   var videoId = videoObject["mediaid"];
   updateCookie(videoId); )}
```

Nastavení hlasitosti a přeskočení videa bylo přidáno i k obsluze přehrávání jednoho videa.

5.2 Interaktivní výběr

Interaktivní výběr na stránce *Spotlight* bude řídit *Spotlight_controller* v metodě `start`. Nejprve se pomocí objektu *Account_handler* zjistí uživatelovy preferované státy. Z těchto států se vytvoří pole. Dále se vytvoří objekt `Video_handler` a zavolá se jeho metoda `loadSpotlightVideos`. Ta očekává 2 parametry. 1. parametr je pole s preferovanými státy a 2. parametr je limit, kolik videí má být vybráno. Metoda `loadSpotlightVideos` provede následující dotaz: `"SELECT * FROM videos WHERE state IN (" . implode(",", $favouriteStates) . ") LIMIT . $limit ."`
Pokud bude takových videí méně, než je potřeba, tak se doplní o náhodná videa.

5.3 Administrace

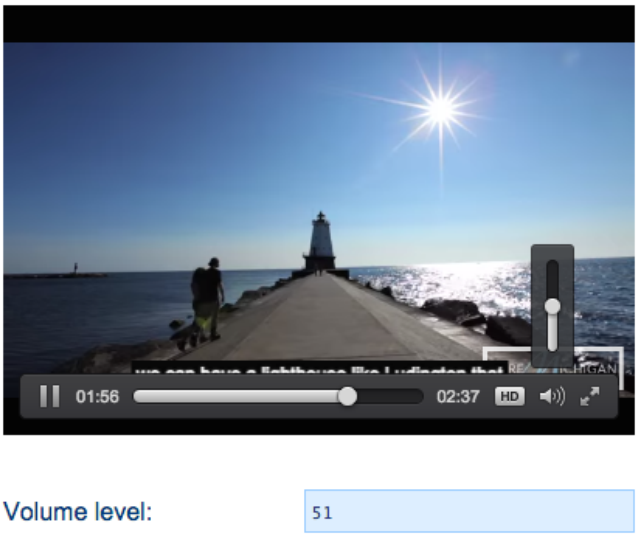
5.3.1 Administrace videí

Do formuláře poskytujícího úpravu videa, bude přidán prvek, který ponese hodnotu hlasitosti tohoto videa. Formulář je vytvořen ve view *Edit_video*.
Formulářový prvek:
```
<input type="text" id="volume" name="volume"
value="<?php echo $this->registry["volume"];?>">
```

Ve stránce bude standardním způsobem vložen JWPlayer, ve kterém bude možnost si upravované video spustit. Na přehrávač bude napojen javascript, který indikuje změnu hlasitosti videa a mění hodnotu formulářového prvku pro hlasitost.
Při odeslání formuláře jsou hodnoty zpracovány v *Video_controlleru* v metodě `edit` a video je aktualizováno v databázi.
Javascript:

```
jwplayer("videoPlayer").onVolume(function(event) {
   var volume = event.volume;
   var volumeInput = document.getElementById("volume");
   volumeInput.value = volume; } )
```

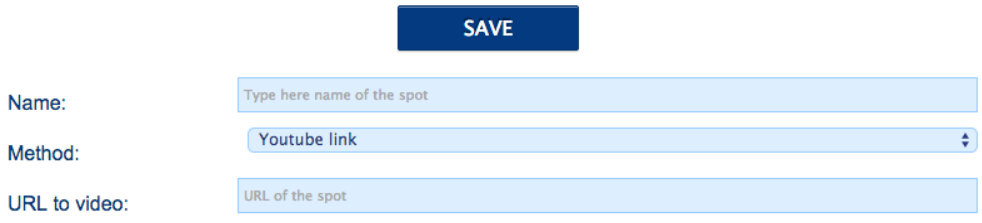
Náhled:



Obrázek 5.1: Nastavení hlasitosti videa

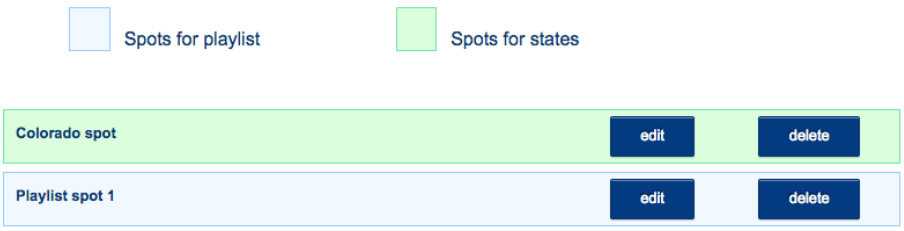
5.3.2 Adminisrace spotů

Ve view *Add_spot* bude formulář pro přidání nového spotu.
Náhled:



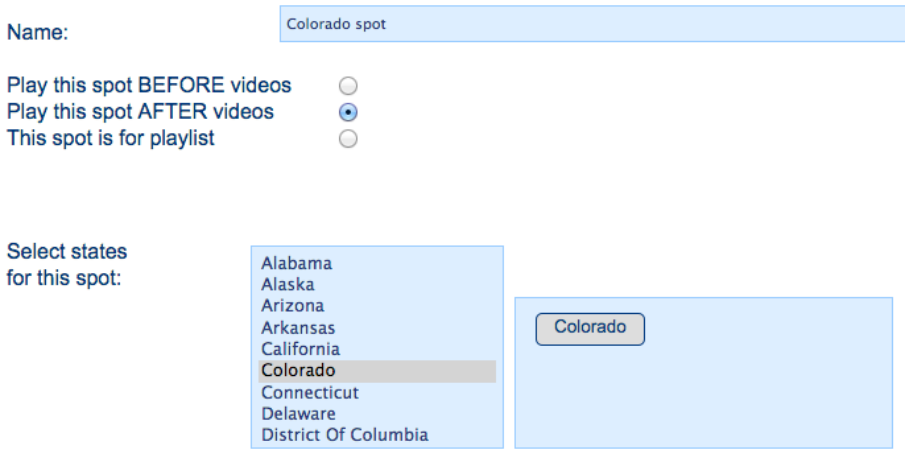
Obrázek 5.2: Formulář pro přidání spotu

Formulář je zpracován *Settings_controllerem* metodou `settings_spot`. Výsledkem je spot uložený v databázi. Spot v tomto stavu je určen pro playlist. Pokud má být spot přiřazen k přehrávání videí určitého státu, tak se musí dále editovat.
Ve view *Add_spot* je mimo formulář ještě seznam všech spotů v systému. Spoty v tomto seznamu je možné editovat či smazat.
Náhled:



Obrázek 5.3: Seznam spotů v systému

Editací se administrátor ocitne na stránce odpovídající view *Edit_spot*. Na této stránce je formulář, umožňující změnit typ spotu ze spotu pro playlist, na přehrávání spotu před, nebo za videem pro určitý stát, nebo státy.
Státy, pro které je tento spot určen, administrátor vybírá z nabídky.
Náhled:



Obrázek 5.4: Formulář pro editaci spotu

Formulář je zpracován metodou `settings_spot` controlleru *Settings_controller*. Aktualizuje položka v databázové tabulce *videos*. Pokud má být editovaný spot přehrán před, nebo za videi určitých států, tak se tato informace uloží do tabulky *spot_state*.

5.4 Vzhled přehrávače

Vzhled přehrávače se dá přizpůsobit pomocí skinů (viz. 3.5.4).
Pro tuto aplikaci nebude vytvořen kompletně vlastní skin, ale bude upraven skin existující. Úpravy budou následující:

- Změna velikosti písma v komponentě `controlbar`:
```
<component name="controlbar">
  <settings>
    <setting name="fontsize" value="12" />
    ...
  </settings>
...
</component>
```
- Změna barevnosti přehrávače – barvy prvků přehrávače jsou definovány pomocí obrázků. Některé z obrázků tedy budou upraveny, aby barvy ladily s designem webu.
- Pro mobilní zařízení budou ovládací prvky zvětšeny, aby na ně šlo lépe ťuknout prstem. Zda se jedná o mobilní zařízení bude zjištěno pomocí javascriptu.

5.5 API

Požadavky přicházející z mobilní aplikace jsou zpracovávány *Api_controllerem* v jehož metodách se využívá služeb objektu *Api_handler*.

- Registrace.
  Adresa pro registraci je: *http://www.americantravelshow.com/api/register*. Adresu je třeba volat s nastavenými POST parametry `login, pwd, first_name, last_name a email`. Při registraci je vyvolána metoda `register`. Ta zvaliduje POST parametry a zjistí dostupnost uživatelského jména. Následně buď zaregistruje nového uživatele, nebo ne. V případě úspěchu je struktura odpovědi, ve formátu JSON, pro mobilní aplikaci následující:
  `{"status" : "success", "login" : "`*`login`*`", "pwd" : "`*`heslo`*`", "first_name" : "`*`jmeno`*`", "last_name" : "`*`prijmeni`*`", "email" : "`*`email`*`"}`.
- Přihlášení.
  Adresa pro přihlášení je: *http://www.americantravelshow.com/api/login*. Adresu je třeba volat s POST parametry `login` a `pwd`. Metoda `login` ověří platnost přihlašovacích údajů a v případě úspěchu uživatele přihlásí. Odpověď potom vypadá:
  `{"status" : "success", "session_id" : "`*`ID`*`", "login" : "`*`login`*`", "first_name" : "`*`jmeno`*`", "last_name" : "`*`prijmeni`*`", "email" : "`*`email`*`"}`.
- Odhlášení.
  Adresa pro odhlášení je: *http://www.americantravelshow.com/api/logout*. Odhlášení je zpracované metodou `logout` a neočekává žádné parametry. Odhlásí uživatele na základě session id, které je odesláno v hlavičce požadavku. V případě úspěchu zasílá: `{"status" : "success"}`.

- Informace o uživateli.
  Adresa požadavku: *http://www.americantravelshow.com/api/getuserdata*. Požadavek je zpracováván metodou `getUserData` a neočekává žádné parametry. Vrátí informace o uživateli, identifikovaného pomocí session id v hlavičce požadavku. V případě úspěšné identifikace je odpověď: `{"status" : "success", "login" : "login", "first_name" : "jmeno", "last_name" : "prijmeni", "email" : "email"}`.
- Seznam videí.
  Adresa požadavku: *http://www.americantravelshow.com/api/getvideos*. Požadavek je zpracováván metodou `getVideos`. K požadavku je třeba připojit GET parametr `from`. Hodnota tohoto parametru musí být ve tvaru Unix timestamp. Tento parametr určuje, jak stará videa se budou vybírat. Vybraná videa budou navrácena ve dvourozměrném JSON poli. Pro každé video budou uvedeny všechny údaje z databázové tabulky *videos*.

Pokud kterýkoliv z požadavků selže nebo neprojde některou z validací, tak výsledkem je: `{"status" : "fail"}`

Kapitola 6

Testování

Přehrávač bude testován napříč různými prohlížeči a jejich verzemi. Testování bude probíhat jak na desktopových, tak mobilních zařízeních. Výběr prohlížečů pro testování bude na základě aktuální statistiky podílu prohlížečů na trhu.
Dále bude otestováno API a odlišný vzhled ovládacích prvků přehrávače na mobilních a desktopových zařízeních.

6.1 Desktopová zařízení

Statistika podílu desktopových prohlížečů na trhu:

Prohlížeč	**Podíl**
Chrome	61,9 %
Firefox	23,4 %
Internet Explorer	7,8 %
Safari	3,8 %
Opera	1,1 %

Tabulka 6.1: Podíl prohlížečů na trhu v lednu 2015[47]

Ostatní prohlížeče představují podíl na trhu menší než 1 % a nebudou do testování zahrnuty.
Předpokládá se, že některé prohlížeče nebudou podporovat přehrávání HTML5 videa a projeví se fallback na Flash. Testování bude prováděno s nejnovější verzí Adobe Flash Playeru.
Následovat bude statistika používání jednotlivých verzí prohlížečů. V těchto verzích budou prohlížeče testovány. Položka „ostatní“ představuje starší verze prohlížeče a také novější, které jsou však ještě ve vývoji.
Jelikož jsou stránky napsány v HTML5, tak je možné, že nebudou v některých starších prohlížečích správně fungovat.

6.1.1 Zastoupení verzí prohlížečů na trhu

Verze	Podíl
40	21,0 %
39	67,4 %
38	4,8 %
37	2,1 %
Ostatní	4,7 %

Tabulka 6.2: Chrome [48]

Verze	Podíl
35	37,3 %
34	38,2 %
33	3,4 %
Ostatní	23,1 %

Tabulka 6.3: Firefox [49]

Verze	Podíl
11	47,4 %
10	14,1 %
9	18 %
8	15,4 %
7	2,6 %
Ostatní	2,5 %

Tabulka 6.4: IE [50]

Verze	Podíl
8	57,9 %
7	26,3 %
6	7,9 %
5	7,9 %

Tabulka 6.5: Safari [51]

Verze	Podíl
27	9,1 %
26	63,6 %
12	9,1 %
Ostatní	18,2 %

Tabulka 6.6: Opera [52]

6.1.2 Výsledek testování

Při testování byly vyzkoušeny oba dva druhy přehrávání – jedno video, celý *Program*. Byly vyzkoušeny všechny ovládací prvky přehrávače a stránky s přehrávačem. Dále se testovalo, jakou technologií je video přehráváno, zda HTML5, nebo Flash.
V Chrome a Firefoxu proběhlo všechno naprosto bez problémů. Video se ve všech verzích přehrávalo pomocí technologie HTML5. Pokud byl přehrávač překonfigurován, aby používal Flash, tak se video v této technologii přehrávalo také bez problémů.
Prohlížeč Safari fungoval bez problémů do verze 5. Verze 5 nepřehrávala video pomocí HTML5, ale pouze pomocí Flash, i když by Safari mělo zvládat HTML5 video již od verze 4 [53].
Opera ve verzích 27 a 26 také přehrávala bez problémů pomocí HTML5. Verze 12 však přehrávala pouze pomocí Flash, i když by měla podporovat HTML5 video od verze 10.5 [53].
Přehrávání v Internet Exploreru 11 a 10 funguje bez problémů. Ve verzi 9 probíhalo přehrávání pouze pomocí Flash, i když by už v této verzi měla být podpora HTML5 videa [53]. Ve verzích 8 a 7 přehrávání sice funguje, ale web se špatně zobrazuje. Tyto verze Internet Exploreru nepodporují vše z HTML5 a mají problém také s CSS animacemi.

6.2 Mobilní zařízení

Statistika podílu mobilních operačních systémů na trhu:

Operační systém	Podíl
Android	57,6 %
iOS	26,2 %
Windows Phone	9,2 %
Ostatní	7 %

Tabulka 6.7: Podíl mobilních OS na trhu v lednu 2015[54]

Uživatelé OS Android využívají nejvíce prohlížečů Chrome a nativního prohlížeče pro Android. Uživatelé iOS využívají nejvíce služeb prohlížeče Safari a taktéž Chrome.
U Windows Phone jsou nejrozšířenějšími prohlížeči Internet Explorer Mobile a UC Browser.
Na výše uvedených platformách a prohlížečích bude prováděno testování přehrávače. Pro přehrávání na mobilních zařízeních uvádí JWPlayer několik rozdílů oproti přehrávání v prohlížeči [55]:

- Mobilní zařízení ignoruje volbu `autostart`, takže se přehrávání nespustí automaticky.
- Není možné spustit více videí najednou.
- Není zde možnost *ztlumit*.
- Na iPhonech jsou videa přehrávána pouze v celoobrazovkovém režimu.

6.2.1 Výsledek testování

Na webu jsou momentálně všechna videa přehrávána z Youtube. Při testování na mobilních zařízeních se objevil problém, spojený právě s přehráváním videí z Youtube v playlistu. Na všech platformách (Android, iOS, Windows Phone) a ve všech výše zmíněných oblíbených prohlížečích se nepřehrají všechna videa, která tvoří playlist. Pokud chceme přehrát pouze jedno video, tak se utvoří playlist, kde je spot, video a zase spot. Z této trojice byl přehrán pouze první spot a video. Poslední spot už nebyl přehrán. Pokud chceme přehrát *Program*, tak se vytvoří playlist s úvodním spotem a pak se vždy střídá video se spotem. Z tohoto playlistu byly přehrány vždy jen první 3 nebo 4 položky, ne více.
Pokud všechna videa a spoty v playlistu jsou načítána z lokálního serveru, nikoliv z Youtube, tak vše funguje bez problémů. Při kombinaci zdrojů, lokální server i Youtube, spadne prohlížeč po druhém videu z Youtube.
Další postřeh je, že všechna videa z Youtube v playlistu vyžadují manuální spuštění přehrávání, zatímco videa z lokálního serveru se kromě prvního spouští automaticky.

6.3 Testování API

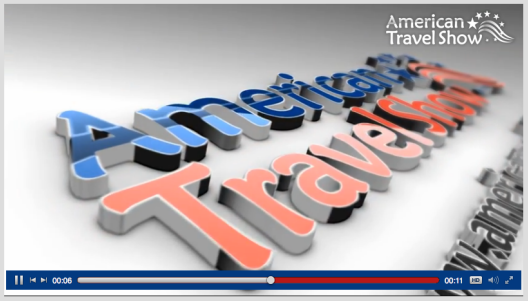
Při vývoji bylo API nejprve testováno pomocí doplňku do prohlížeče Chrome s názvem Postman. Byly zasílány jednotlivé požadavky s parametry a byly kontrolovány odpovědi. Následně služeb API začal využívat kolega, pracující na mobilní aplikaci pro Android.

Všechny jeho připomínky a nové požadavky byly operativně řešeny, takže momentální API je přizpůsobené jeho aplikaci. Ve spolupráci s ním se bude pravděpodobně API dále rozšiřovat.

6.4 Testování vzhledu přehrávače

Ovládací prvky přehrávače by měly vypadat odlišně na desktopových a mobilních zařízeních. Na mobilních by měly být prvky zvětšené. Také by zde mělo být prvků méně. Výsledek testu:

Obrázek 6.1: Mobilní zařízení



Obrázek 6.2: Desktop

6.5 Zhodnocení testování

Na desktopových zařízeních v moderních prohlížečích funguje vše tak, jak má. Na mobilních zařízeních způsobuje problémy přehrávání videí z Youtube v playlistu. Problém by se dal vyřešit tak, že by se videa přesunula z Youtube na vlastní server. Postupem času by se měla videa skutečně začít přesouvat, ale bude to nějakou dobu trvat. Potom už bude přehrávač plně sloužit i na mobilních zařízeních. Problém s přehráváním na mobilních zařízeních vyřeší i nová aplikace pro Android, která přehrávání videí bude nabízet taktéž.

Závěr

Cílem této bakalářské práce bylo rozšířit existující webové stránky o přehrávač videí. Tento cíl se mi podařilo splnit. Na základě rešerše existujících webových přehrávačů videí jsem jich vybral 7 k podrobnější analýze. V analýze jsem se zaměřil na to, jak jednotlivé přehrávače dokáží uspokojit požadavky na přehrávač od zadavatele. Z analýzy nejlépe vyšel přehrávač JWPlayer, a tak jsem ho vybral a zabudoval do webových stránek. Při samotné realizaci této práce se objevily některé další události, které byly třeba při přehrávání ošetřit a nebyly uvedeny v požadavcích. Například přeskočení přehrávání vadného či nepodporovaného videa.

Dalším cílem bylo také umožnit přehrávání videí na moderních zařízeních. Tento cíl byl splněn zčásti. Na desktopových zařízeních funguje vše tak, jak má. Na mobilních zařízeních se objevuje problém v případě, kdy chceme přehrát playlist, v němž jsou videa z Youtube. Při řešení tohoto problému jsem kontaktoval podporu JWPlayeru, ta to označila jako problém s videi z Youtube na mobilních zařízeních, jehož oprava není v plánu. Řešením tohoto problému je tedy nepřehrávat videa z Youtube nebo použití mobilní aplikace pro tento projekt.

Práce také měla zajistit administrační rozhraní pro správu videí. Tento požadavek byl splněn a administrátor může přidávat videa (video o některém ze států, reklamní spot) a nastavovat pro ně určité parametry jako například ke kterému státu video patří, informace o videu či hlasitost se kterou má být video přehráváno. V případě reklamních spotů ještě může administrátor určit, kdy se bude daný spot přehrávat.

Za poslední cíl si tato práce kladla obsluhu požadavků mobilní aplikace. Cíl se mi podařilo splnit. Webová aplikace dokáže zpracovat požadavky na registraci, přihlášení a odhlášení uživatele. Dále dokáže poskytnout informace o uživateli a seznam videí. Při realizaci tohoto cíle jsem spolupracoval s kolegou, který současně pracuje na mobilní aplikaci, takže funkčnost tohoto rozhraní byla přizpůsobována jeho aplikaci.

Všem implementovaným prvkům, popsaným výše, předcházel návrh. Při návrhu jsem využíval různé diagramy, které jsem tvořil pomocí softwaru Enterprise Architect, se kterým jsem se při studiu seznámil. Dále jsem při návrhu dbal na to, abych při rozšiřování webové aplikace zachoval její architekturu.

Po implementaci jsem webovou aplikaci testoval. V testování jsem se zaměřil na podporu přehrávání videí v různých prohlížečích a na přehrávání na mobilních zařízeních.

Literatura

[1] Fernando Manuel Bernardo Pereira, T. E.: *The MPEG-4 book*. Prentice Hall, 2002, ISBN 978-0130616210.

[2] Building JW Player Skins [online]. [cit. 2015-27-01]. Dostupné z: `http://support.jwplayer.com/customer/portal/articles/1412123-building-jw-player-skins`

[3] Bernard, B.: Prezentační vzory z rodiny MVC [online]. 2009 [cit. 2015-02-02]. Dostupné z: `http://www.zdrojak.cz/clanky/prezentacni-vzory-zrodiny-mvc/`

[4] Pfeiffer, S.: *The Definitive Guide to HTML5 Video*. Apress, 2010, ISBN 978-1-4302-3090-8.

[5] Janovský, D.: Video na stránkách [online]. 2008 [cit. 2014-16-11]. Dostupné z: `http://www.jakpsatweb.cz/video.html`

[6] Mike: A Short History of Online Video, Part 1 – Before Youtube [online]. 2012 [cit. 2014-16-11]. Dostupné z: `http://blog.treepodia.com/2012/02/a-short-history-of-online-video-part-1-before-youtube`

[7] Broadband history [online]. [cit. 2014-16-11]. Dostupné z: `http://www.uswitch.com/broadband/guides/broadband_history`

[8] About WebM [online]. [cit. 2014-16-11]. Dostupné z: `http://www.webmproject.org/about`

[9] Pilgrim, M.: Video on the Web [online]. 2009 [cit. 2014-17-11]. Dostupné z: `http://kniha.html5.cz/video.html`

[10] Kontejnery pro video [online]. [cit. 2014-18-11]. Dostupné z: `http://avnavody.cz/?sekce=vrch_kontejnery`

[11] Ng, T.: MP4 File Format Part 1 [online]. 2010 [cit. 2014-18-11]. Dostupné z: `http://thompsonng.blogspot.cz/2010/11/mp4-file-format.html`

[12] The Ogg container format [online]. [cit. 2014-18-11]. Dostupné z: http://xiph.org/ogg

[13] Ogg logical bitstream framing [online]. [cit. 2014-18-11]. Dostupné z: http://www.xiph.org/vorbis/doc/framing.html

[14] Pfeiffer, S.: The Ogg Encapsulation Format Version 0 [online]. 2003 [cit. 2014-18-11]. Dostupné z: http://www.ietf.org/rfc/rfc3533.txt

[15] WebM Container Guidelines [online]. [cit. 2014-18-11]. Dostupné z: http://www.webmproject.org/docs/container

[16] permadi: Introduction To WebM File Structure [online]. 2010 [cit. 2014-18-11]. Dostupné z: http://permadi.com/blog/2010/06/webm-file-structure

[17] Siegchrist, G.: Codec [online]. [cit. 2014-17-11]. Dostupné z: http://desktopvideo.about.com/od/glossary/g/codec.htm

[18] Wootton, C.: *A Practical Guide to Video and Audio Compression*. Elsevier, 2005, ISBN 0-240-80630-1.

[19] HTTP Live Streaming Overview [online]. [cit. 2014-17-11]. Dostupné z: https://developer.apple.com/library/ios/documentation/networkinginternet/conceptual/streamingmediaguide/FrequentlyAskedQuestions/FrequentlyAskedQuestions.html

[20] Supported Media Formats [online]. [cit. 2014-17-11]. Dostupné z: http://developer.android.com/guide/appendix/media-formats.html

[21] Apple TV (3rd generation) - Technical Specifications [online]. [cit. 2014-17-11]. Dostupné z: http://support.apple.com/kb/sp648

[22] Devlin, I.: *HTML5 Multimedia: Develop and Design*. Peachpit Press, 2012, ISBN 978-0-321-79393-5.

[23] HTML Audio/Video DOM controls Property [online]. [cit. 2015-28-01]. Dostupné z: http://www.w3schools.com/tags/av_prop_controls.asp

[24] HTML <source> media Attribute [online]. [cit. 2014-21-11]. Dostupné z: http://www.w3schools.com/tags/att_source_media.asp

[25] Dutton, S.: Getting started with the HTML5 track element [online]. 2012 [cit. 2014-22-11]. Dostupné z: http://screenfont.ca/learn

[26] Clark, J.: Understanding captions and subtitles [online]. [cit. 2014-22-11]. Dostupné z: http://screenfont.ca/learn

[27] HTML <track> kind Attribute [online]. [cit. 2014-22-11]. Dostupné z: http://www.w3schools.com/tags/att_track_kind.asp

[28] Schklowsky, P.: What's new in HTML5: The Track Element [online]. 2012 [cit. 2014-22-11]. Dostupné z: http://www.jwplayer.com/blog/whats-new-in-html5-the-track-element

[29] HTML <track> Tag [online]. [cit. 2014-22-11]. Dostupné z: http://www.w3schools.com/tags/tag_track.asp

[30] HTML <track> default Attribute [online]. [cit. 2014-22-11]. Dostupné z: http://www.w3schools.com/tags/att_track_default.asp

[31] XML DOM getElementsByTagName() Method [online]. [cit. 2014-20-12]. Dostupné z: http://www.w3schools.com/dom/met_document_getelementsbytagname.asp

[32] HTML DOM Video Object [online]. [cit. 2014-20-12]. Dostupné z: http://www.w3schools.com/jsref/dom_obj_video.asp

[33] Video currentSrc Property [online]. [cit. 2014-20-12]. Dostupné z: http://www.w3schools.com/jsref/prop_video_currentsrc.asp

[34] Video canPlayType() Method [online]. [cit. 2014-21-12]. Dostupné z: http://www.w3schools.com/jsref/met_video_canplaytype.asp

[35] HTML5 Video Player Comparison [online]. 2003 [cit. 2014-18-12]. Dostupné z: http://praegnanz.de/html5video/

[36] About Us [online]. [cit. 2015-20-01]. Dostupné z: http://www.sublimevideo.net/about

[37] Using the JavaScript API [online]. [cit. 2015-21-01]. Dostupné z: http://docs.sublimevideo.net/javascript-api/usage

[38] Supported browsers and devices [online]. [cit. 2015-21-01]. Dostupné z: http://docs.sublimevideo.net/supported-platforms

[39] Videos won't play on some Android devices [online]. [cit. 2015-21-01]. Dostupné z: https://getsatisfaction.com/sublimevideo/topics/videos-wont-play-on-some-android-devices-leefacws8t5nl

[40] Pricing [online]. [cit. 2015-22-01]. Dostupné z: http://www.jwplayer.com/pricing/

[41] Historie a budoucnost [online]. [cit. 2015-10-01]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=171

[42] Andi Gutmans, S. S. B., Derick Rethans: *Mistrovství v PHP 5*. COMPUTER PRESS, 2005, ISBN 80-251-0799-X.

[43] Duckett, J.: *HTML and CSS: Design and Build Websites*. John Wiliey and Sons, 2011, ISBN 978-1118008188.

[44] What is jQuery? [online]. [cit. 2015-14-01]. Dostupné z: http://jquery.com/

[45] PHP MySQL Database [online]. [cit. 2015-14-01]. Dostupné z: http://www.w3schools.com/php/php_mysql_intro.asp

[46] Bernard, B.: Úvod do architektury MVC [online]. 2009 [cit. 2015-02-02]. Dostupné z: http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[47] Browser Statistics [online]. 2015 [cit. 2015-03-03]. Dostupné z: http://www.w3schools.com/browsers/browsers_stats.asp

[48] Google Chrome Statistics [online]. [cit. 2015-03-03]. Dostupné z: http://www.w3schools.com/browsers/browsers_chrome.asp

[49] Firefox Statistics [online]. [cit. 2015-03-03]. Dostupné z: http://www.w3schools.com/browsers/browsers_firefox.asp

[50] Internet Explorer Statistics [online]. [cit. 2015-03-03]. Dostupné z: http://www.w3schools.com/browsers/browsers_explorer.asp

[51] Safari Statistics [online]. [cit. 2015-04-03]. Dostupné z: http://www.w3schools.com/browsers/browsers_safari.asp

[52] Opera Statistics [online]. [cit. 2015-04-03]. Dostupné z: http://www.w3schools.com/browsers/browsers_opera.asp

[53] HTML5 Video [online]. [cit. 2015-14-03]. Dostupné z: http://www.w3schools.com/html/html5_video.asp

[54] Mobile Devices Statistics [online]. [cit. 2015-04-03]. Dostupné z: http://www.w3schools.com/browsers/browsers_mobile.asp

[55] Browser and Device Reference [online]. [cit. 2015-15-03]. Dostupné z: http://support.jwplayer.com/customer/portal/articles/1403653-browser-device-reference

Příloha **A**

Seznam použitých zkratek

HTML HyperText Markup Language
EBML Extensible Binary Meta Language
XML Extensible markup language
DOM Document Object Model
PHP Hypertext Preprocessor
CSS Cascading Style Sheets

Příloha **B**

Popis API přehrávače JWPlayer

B.1 Atributy pro setup

- `aspectratio`
  Udává poměr velikosti stran, šířky ku výšce, přehrávače . Hodnota může být například "16:9". Využívá se při responsivním designu přehrávače, kde se šířka udává v procentech.

  Implementace: Hodnota šířky, v procentech, se mění podle velikosti okna. Výška se dopočítává dle následujícího vzorce: $H = \frac{R_h}{R_w} * W$, kde R_h je výška z poměru, R_w je šířka z poměru a W je aktuální šířka přehrávače v procentech. Tato hodnota se nastaví elementu <video> jako hodnota atributu `height`.
- `autostart`
  Video se začne přehrávat ihned po načtení stránky. Možné hodnoty jsou `"true"` nebo `"false"`.
  Požadavek na automatické přehrávání je ignorován mobilními zařízeními, z důvodu šetření datových tarifů.

  Implementace: Řídící skript nejprve zkontroluje, zda zařízení, na kterém se má přehrávat není mobilní a následně spustí přehrávání.
- `controls`
  V závislosti na hodnotě, `"true"/"false"`, zobrazí, či skryje ovládací prvky přehrávače. Pokud je nastaveno na `"false"`, tak je zabráněno veškeré interakci s uživatelem. Nezobrazí ovládací prvky, ani neumožní ovládat přehrávač pomocí klávesnice.

  Implementace: Pomocí javascriptu se nastaví elementu <video> atribut `control`.
- `file`
  Cesta ke zdroji videa, které má být přehráno. Může být cesta na lokálním serveru nebo URL na Youtube, případně i URL livestreamu.

  Implementace: Do HTML elementu `<video>` se pomocí javascriptu nastaví atri-

but `src`. V případě, že je zadáno URL videa z Youtube, tak se začne využívat API, které nabízí Youtube. Parametry, které jsou nastaveny v konfiguraci, se přenáší do nastavení videa z Youtube, pokud to jeho API umožňuje.

- `height`
  Výška přehrávače v pixelech.

  Implementace: Pomocí javascriptu se nastaví výška elementu `<video>`.
- `image`
  Cesta k obrázku, který bude zobrazen před samotným začátkem přehrávání videa. Tato volba je ignorována, pokud je používán playlist.

  Implementace: Pomocí javascriptu se nastaví atribut `poster` elementu `<video>`. Pokud přehráváme video, a není nastaven atribut `image`, tak se vybere náhodný snímek z videa.
- `logo`
  Obrázek, představující logo, který bude zobrazen v přehrávači.

  Implementace: Pomocí CSS se obrázek přesune na vhodnou pozici v přehrávači.
- `mute`
  Hodnota `"true"`, nebo `"false"`. Určuje, zda má mít přehrávání vypnutý zvuk, či nikoliv. Nefunguje na mobilních zařízeních.

  Implementace: Elementu `<video>` se nastaví atribut `muted` na jednu z hodnot.
- `primary`
  Přehrávač dokáže video přehrát jako HTML5 video, nebo jako Flash. Tato volba umožňuje zvolit výchozí technologii přehrávání. Hodnota `"html5"` nastaví jako defaultní HTML5 video a hodnota `"flash"` zase Flash technologii.

  Implementace: Na základě hodnoty se buď spouští skripty pro sestavení HTML5 přehrávání, nebo flash přehrávání.
- `repeat`
  Když je nastavený na hodnotu `"true"`, tak se video přehrává stále dokola.

  Implementace: Pomocí javascriptu se nastaví atribut `loop` elementu `<video>` na `"true"`, nebo `"false"`.
- `skin`
  Určuje vzhled přehrávače. Na výběr je z pěti skinů nebo můžeme vytvořit skin vlastní. Ten je definován XML souborem. Hodnotou je cesta k XML souboru se skinem.

  Implementace: Na základě XML souboru jsou měněny obrázky, ze kterých se skládá přehrávač.

- `width`
  Šířka přehrávače v pixelech. Případně v procentech pro responsivitu (v tomto případě je nutné, aby byl nastaven `aspectratio`).

  Implementace: Pomocí javascriptu se nastaví šířka elementu `<video>`.

B.2 Atributy a metody playlistu

B.2.1 Atributy

- `title`
  Název videa.
- `description`
  Popis videa.
- `mediaid`
  Jednoznačný identifikátor pro video v playlistu.
- `image`
  Obrázek, který bude zobrazen přehrávačem před samotným přehráváním tohoto videa v playlistu.
- `sources[].file`
  Pole obsahující cesty ke zdrojům videa. Může být jak cesta k videu na lokálním serveru, tak URL videa na Youtube. Položek `file` může být v poli `sources` více a mohou představovat stejné video v různých formátech nebo kvalitách.
  Pokud je video ve více formátech a není nastaven `sources[].default`, tak se začne přehrávat první vyhovující. Pokud je ve více kvalitách, tak se v přehrávači zobrazí tlačítko pro výběr kvality videa.
- `sources[].label`
  Pojmenování zdroje videa. Pokud je video dostupné ve více kvalitách, tak kvalita je pojmenována takto.
- `sources[].default`
  Pří více zdrojích, položkách `sources[].file`, bude tento zdroj brán jako výchozí. Pokud přehrávač nebude schopen video z tohoto zdroje přehrát, tak se zvolí první vyhovující z `sources[].file`.
- `sources[].type`
  Určuje typ přehrávaného videa, např. `video/mp4`. Položka je využívána, pokud není určena přípona zdrojového souboru.
  Nepoužívá se u videí z Youtube.
- `tracks[].file`
  Pole obsahující cesty ke zdrojům stop.
- `tracks[].kind`
  Určuje druh stopy. Stopou můžou být popisky, kapitoly nebo náhledy.

- `tracks[].label`
  Pojmenování stopy. Pokud je dostupno více zdrojů popisků, tak jsou pojmenovány dle `label`.
- `tracks[].default`
  Pouze pro popisky. Pokud je nastavena hodnota `"true"`, tak se popisky objevují hned při spuštění videa. Pokud je hodnota `"false"`, tak se musí popisky spustit manuálně.

B.2.2 Metody

- `getPlaylist()`
  Vrátí playlist jako pole objektů `Item`.
- `getPlaylistIndex()`
  Vrátí index aktuálně přehrávaného videa.
- `getPlaylistItem(index)`
  Vrátí objekt `Item` ležící na zadaném indexu v poli playlistu. Pokud se na této pozici žádný nenachází, tak je vrácen aktuálně přehrávaný `Item`
- `load(playlist)`
  Načte nový playlist. Jako parametr se předává JSON, se stejnou strukturou, jaká je při inicializaci playlistu.
- `playListItem(index)`
  Přehrávač začne přehrávat video z playlistu na daném indexu.
- `onPlaylist(callback)`
  Umožňuje zpracovávání nově nahraného playlistu.
- `onPlaylistItem(callback)`
  Umožňuje reagovat na přechod mezi videi v playlistu – když jedno video skončí a druhé začíná.
- `onPlaylistComplete(callback)`
  Umožňuje reagovat na dokončení přehrávání všech videí playlistu.

B.3 Další použité metody

- `onReady(callback)`
  Metoda umožňující reagovat na stav, kdy je přehrávač připraven začít přehrávat.
- `onSetupError(callback)`
  Metoda umožňující reagovat na stav, kdy přehrávač není schopen začít přehrávat.
- `onPlay(callback)`
  Metoda umožňující reagovat na stav, kdy přehrávač začne přehrávat.
- `onError(callback)`
  Metoda umožňující reagovat na stav, kdy přehrávač není schopen přehrát video. Důvodem může být například nedostupný zdroj videa nebo video není v podporovaném formátu.

- `onComplete(callback)`
  Metoda umožňující reagovat na stav, kdy přehrávač dokončí přehrávání videa.
- `getState()`
  Vrací stav, ve kterém se přehrávač nachází. Jedna hodnota z množiny:
  - `IDLE`
    Přehrávání ještě nezačlo nebo bylo zastaveno.
  - `BUFFERING`
    Přehrávání začalo, ale čeká se na stažení dostatku dat videa.
  - `PLAYING`
    Probíhá přehrávání.
  - `PAUSED`
    Přehrávání je pozastaveno.
- `play(state)`
  Pokud `state` je `"true"`, tak se video začne přehrávat. V případě, že je `state` `"false"`, tak se nastaví pauza. Zavolání metody bez parametru přepíná mezi pauzou a přehráváním.
- `stop()`
  Přehrávání se zastaví, přestanou se stahovat další části videa a přehrávač přejde do stavu `IDLE`.
- `setVolume(volume)`
  Nastaví přehrávanému videu hlasitost `volume`. Hodnota je z intervalu 0 až 100 (%).

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Veznik_David_2015.pdf` ..... text práce ve formátu PDF