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

Bakalářská práce

Systém pro prezentaci a správu elektronických oznámení

Plzeň 2022 Michaela Janková

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Michaela JANKOVÁ**
Osobní číslo:	**A20B0119P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Systém pro prezentaci a správu elektronických oznámení**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte webové technologie pro tvorbu moderních informačních systémů.
2. Seznamte se s možnostmi webových prohlížečů a technologií, včetně zpracování a zobrazování obsahu různých mediálních typů.
3. Navrhněte systém, který umožní spravovat a v různých režimech zobrazovat obsah elektronické webové nástěnky.
4. Systém ve vhodných technologiích implementujte, ověřte jeho funkčnost, zdokumentujte a připravte pro ověřovací provoz.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Přemysl Brada, MSc., Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **24. srpna 2021**
Termín odevzdání bakalářské práce: **5. května 2022**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 14. října 2021

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2022

Michaela Janková

Abstract

The aim of this bachelor thesis is to create an electronic billboard and management backend. In other words a web application which will allow to project electronic announcements and, to logged-in users, to manage these announcements - such as add new announcement, delete, choose which ones should be projected and manage parameters of projection such as length of projection or a date to which will the announcement be projected. Because the system will have users to manage these announcements it is necessary the system will also allow management of users.

Abstrakt

Cílem této bakalářské práce je vytvořit systém pro prezentaci a správu elektronických oznámení, jinými slovy webovou aplikaci, která umožní ve webovém prohlížeči promítat elektronická oznámení a zároveň umožní přihlášeným uživatelům tato oznámení spravovat - tedy přidávat, mazat, zvolit, která se mají promítat a nastavit jednotlivým oznámením parametry promítání jako je délka promítání, nebo datum do kterého se má oznámení promítat. Protože bude mít systém uživatele spravující tato oznámení, je nezbytné, aby umožňoval i správu uživatelů. Analytická část aplikace se zabývá výběrem vhodných webových technologií a způsobem ukládání a prezentace multimediálních souborů. Praktická část obsahuje návrh řešení a implementaci systému.

Obsah

1 Úvod

Cílem této práce je vytvořit webovou aplikaci, která bude prezentovat oznámení a upoutávky na nadcházející akce. Tato aplikace pak bude otevřena v prohlížeči na monitoru/televizi na chodbě katedry. Součástí aplikace je také správa těchto oznámení a správa uživatelů systému.

Druhá kapitola se zabývá požadavky na systém, mimo jiné jak se mají daná oznámení při promítání zobrazit, jaké parametry bude třeba uživatelům umožnit upravit, nebo jaké typy a formáty souborů bude systém promítat, stejně jako požadavky na správu uživatelů a jejich rolí v systémů.

Teoretická část se zabývá výběrem vhodných webových technologií pro tento projekt, především vhodného PHP frameworku, stejně jako analýzou způsobu ukládání multimediálních souborů a prezentování multimediálního obsahu ve webových prohlížečích.

Praktická část se soustředí na implementaci systému. Jsou zde popsány všechny tři vrstvy, které tvoří aplikaci a její databázový systém a jak jsou oznámení promítána. Jsou zde popsány jednotlivé kontrolery a jejich funcionalita, stejně tak jsou zde popsány pohledy aplikace a jednotlivé tabulky databáze a modely a nástroje, které s nimi pracují. Poslední kapitola této části se zabývá testováním výsledné webové aplikace.

2 Požadavky na systém

Tato kapitola popisuje požadavky na systém, funkce systému a specifikaci jednotlivých pohledů.

Hlavní funkcí systému je promítání aktivních oznámení. Aby se oznámení vůbec mohla promítat, systém napřed musí umožňovat vytvořit účet uživatele, přiřadit mu jednu z rolí v systému a následně se přihlásit.

Po přihlášení se uživateli zobrazí obsah podle role. Administrátorovi se zobrazí přehled všech oznámení v databázi se všemi dostupnými metadaty. Bude moci oznámení vybrat do aktivních, která se následně budou promítat, nebo jej smazat. Také mu bude umožněno nastavit délku promítání oznámení, která bude v sekundách. Vzhledem k tomu, že se jedná o systém pro prezentaci oznámení, je zřejmé, že většina oznámení bude informovat o nadcházejících událostech, které nemá smysl prezentovat po dni konání. Proto bude systém umožňovat nastavení datumů, v jejichž rozmezí se bude oznámení promítat. Není nutné je nastavit, ale systém bude umožňovat tuto možnost. Stejně tak lze nastavit pouze jeden z těchto datumů. Výpis oznámení bude možné filtrovat. Systém bude umožňovat zobrazit výpis pouze obrázků, videí, nebo dokumentů. Stejně tak bude umožněno si nechat zobrazit pouze aktivní položky. Dále je samozřejmě potřeba, aby bylo v systému možno oznámení přidat vložením jednotlivých souborů, případně zadáním URL (Uniform Resource Locator) webových stránek. Systém bude umožňovat, aby se daly vložit obrázky formátu JPG (JPEG), PNG, GIF a JFIF, videa formátu MP4 a AVI a soubory formátu PDF.

Systém bude umožňovat také správu uživatelů. Bude tak možné změnit uživateli heslo a hlavní funkčností bude změna role uživatele v systému.

Poslední věc je samotné promítání oznámení, která uživatelé vyberou do aktivních. Promítání bude mít samostatný pohled, kde bude vidět pouze aktuálně promítaná oznámení a nic jiného. Oznámení se budou prezentovat po jedno a budou se střídat podle doby, nastavené uživatelem pro jednotlivá oznámení (výchozí doba promítání bude pět sekund).

2.1 Role

Systém bude spravován osobami, které budou mít v systému odlišné role a proto je také třeba rozlišovat jejich práva ohledně přístupu k jednotlivým funkcionalitám aplikace. Jedná se o tři skupiny uživatelů. V rámci programu

jsou popsány jako *admin, r/w a r/o*. V rámci tohoto textu je označme administrátor, správce oznámení a správce promítání.

Administrátor je tedy uživatel, který má přístup ke všem funkcím programu. Co se týká oznámení, může přidávat nové obrázky, videa, PDF soubory i URL adresy k webovým stránkám, které by se zobrazily v IFramu při promítání. Může je také odstraňovat, procházet, filtrovat a vybírat, jaká oznámení se při promítání zobrazí nebo nastavit datum, od kterého (nebo do kterého) se budou zobrazovat konkrétní oznámení. Kromě toho si také mohou zobrazit všechny uživatele, přidat nového. Pokud by to bylo třeba, je možné změnit uživatelům heslo (nikoliv si stávající hesla zobrazit) a změnit uživatelům roli, jakou v systému mají.

Další skupinou uživatelů jsou správci oznámení, tedy osoby, které spravují pouze oznámení nikoliv uživatele. Ohledně oznámení mají veškerá práva (tedy stejně jako administrátor).

Poslední skupinou jsou správci promítání. Stejně jako správci oznámení nemají přístup ke správě uživatelů. Oproti nim ovšem nemohou ani soubory (nebo URL) přidávat, ani mazat. Mohou pouze měnit, která oznámení se budou promítat ve smyčce (a příslušné atributy - dobu promítání, přiřadit rozmezí datumů, kdy se budou promítat a podobně).

2.2 Správa oznámení

Pro správu oznámení se uživateli zobrazí tabulka se všemi oznámeními v databázi. Uživatel bude moct změnit stav oznámení z aktivního na neaktivní a opačně zaškrtnutím políčka (checkboxu). Změnu bude třeba potvrdit tlačítkem, které se bude nacházet pod tabulkou.

Pokud bude oznámení označeno jako aktivní, zobrazí se také políčko pro vložení celého kladného čísla, které reprezentuje dobu promítání oznámení ve smyčce. Maximální doba zobrazení oznámení bude nastavena na 3600 sekund (tedy jednu hodinu). Výchozí doba délky promítání oznámení je pět sekund.

U aktivních oznámení se ještě zobrazí dvě pole pro vložení datumu. Ty bude možné využít k nastavení intervalu promítání oznámení, ohraničeným těmito dvěma datumy, které nastaví uživatel.

Samozřejmě je také potřeba, aby oznámení bylo možné smazat. V tabulce v každém řádku bude proto na konci dostupné tlačítko pro smazání daného oznámení. Po stisknutí tlačítka vyskočí upozornění, jestli opravdu chcete oznámení smazat. V této chvíli bude ještě možné akci zrušit. V případě, že bude chtít uživatel opravdu oznámení smazat, stiskne tlačítko OK

a oznámení se smaže z databáze. V případě ukládání souborů na disk by se oznámení smazalo i na něm.

2.3 Vložení oznámení

Pod tabulkou s již vloženými oznámeními bude sekce *Vložit oznámení*. Zde se budou nacházet dvě tlačítka pro vložení souboru. První bude sloužit k vybrání souboru typu PNG, JPG, MP4, AVI, GIF, JFIF a PDF. Uživatel bude moci vybrat ze všech souborů, která má dostupná na svém zařízení do velikosti 2 000 MB. Pokud uživatel vybere jeden ze souborů v tomto formátu a bude ho chtít nahrát, pak klikne na druhé tlačítko, které nahraje soubor do databáze (nebo jen jeho metadata), případně i na disk.

Pokud by chtěl uživatel nahrát URL webových stránek, využil by možnosti vložení textu, která se bude nacházet pod těmito tlačítky. Vedle tohoto pole se bude nacházet i tlačítko pro vložení URL, na které bude třeba kliknout, aby se nahrálo do systému.

2.4 Správa uživatelů

V rámci aplikace bude třeba, aby administrátor mohl spravovat uživatele. Nejprve je třeba, aby administrátor mohl nějakého uživatele přidat. Pro to bude muset zadat e-mail uživatele, jeho uživatelské jméno, heslo a vybrat jednu z rolí v systému (tedy r/o pro správce promítání, r/w pro správce oznámení nebo admin pro administrátora aplikace).

Dále bude třeba, aby administrátor mohl uživateli změnit heslo, změnit jeho roli v systému nebo ho smazat.

2.5 Promítání

Hlavní funkcionalitou aplikace je promítání vybraných oznámení. Oznámení, která se mají promítat, se v tabulce pro správu oznámení označí jako aktivní zaškrtnutím políčka. Aplikace bude schopna promítat videa (formáty MP4 a AVI), obrázky (PNG, JPG, GIF a JFIF), PDF dokumenty a webové stránky. Oznámení budou zobrazena přes celou obrazovku. Dále bude aplikace umožňovat, aby se dala upravit délka promítání. V momentě, kdy se oznámení označí jako aktivní, mu bude přidělena nějaká výchozí hodnota. Dále bude aplikace umožňovat, aby se dalo určit, v jakém rozmezí datumů se má aktivní oznámení promítat, či určit jenom jeden z těchto krajních datumů.

2.6 Filtrování

Aplikace bude umožňovat filtrování oznámení, aby se v nich dalo lépe orientovat. Uživatel si tak bude moct nechat zobrazit pouze aktivní položky, obrázky, videa, nebo dokumenty.

3 Webové technologie

Před samotným vývojem aplikace, bude nutné vybrat nejlépe vyhovující technologie. Tyto technologie musí být prohlížeči dostatečně podporované, naučení jejich základů, musí být relativně snadné a měly by být dobře zdokumentované. Také by měly být otevřené, aby za jejich použití nebylo nutné platit.

Vzhledem k tomu, že aplikace má promítat ve smyčce oznámení, bude třeba v prohlíči tyto data měnit pomocí nějakého skriptovacího jazyka (nabízí se JavaScript, nebo TypeScript). Pro správu oznámení bude třeba nějaký jazyk na straně serveru, kde se vzhledem k tomu, že nejedná o robustní aplikaci, nabízí PHP.

3.1 JavaScript

JavaScript je multiplatformní skriptovací jazyk, který umožňuje vytvořit interaktivní webové stránky. Byl vyvíjen, aby běžel u uživatele v prohlížeči. Není navržený pro vývoj velkých a komplexních aplikací.

Jedná se o interpretovaný jazyk, což znamená, že zdrojový kód není zkompilovaný do binárního kódu, než se spustí a je třeba interpret, který zdrojový kód interpretuje procesoru.

JavaScriptový kód je spustitelný v jakémkoliv webovém prohlížeči. Každý z nich má nějakou verzi JavaScript Engine [19] (interpret) - nejpoužívanější je V8 JavaScript Engine od společnosti Google. Používá jej Google Chrome a další prohlížeče na něm založené, stejně jako aplikace, které obsahují Chromium. Používají jej také běhové systémy jako Node.js nebo Deno. Mezi další používané Enginy patří SpiderMonkey od Mozilly, který je použitý ve Firefoxu, JavaScriptCore, který je využívaný pro Safari a Chakra, který běží v Internet Exploreru (Edge je založen na Chromu, používá tudíž V8).

Pro vývoj serverů se pak využívají prostředí jako Node.js, Jaxer, Deno, AppJet, Wakanda a další. V dnešní době je pravděpodobně nejpoužívanější Node.js.

Node.js je asynchronní JavaScriptové běhové prostředí řízené událostmi založené na již zmíněném V8 a je navržené pro psaní vysoce škálovatelných webových aplikací. Byl vyvinut v roce 2009 Ryanem Dahlem původně pouze pro Linux a Mac OS X. Od roku 2011 existuje podpora i pro Windows [28].

Autor Node.js, Ryan Dahl, je také autorem prostředí Deno. Deno je běhové prostředí jak pro JavaScript, tak TypeScript, založené na enginu V8

a napsané v Rustu. Bylo vydané v roce 2018. Od Node.js se liší například zabudovaným správcem balíčků, NPM tedy není potřeba (NPM je správce balíčků pro JavaScript a Node.js.). Dalším rozdílem je, že disponuje kompilátorem pro TypeScript.

JavaScriptový kód je také možné spustit v příkazové řádce operačního systému, pokud je na něm nainstalovaný Node.js.

3.1.1 Historie

JavaScript patří mezi nejpoužívanější jazyky v dnešní době. Podle Business Insideru (2019) je to dokonce nejvíc používaný jazyk na GitHubu. [8] Využívá se především pro vývoj webu.

JavaScript byl vytvořen v roce 1995 Brendanem Eichem. Během deseti dní vytvořil prototyp pod původním názvem Mocha, který firma Netscape vzala jako rovnou hotový produkt, což vysvětluje některé nedostatky jazyka v době uvedení na trh. Na trh byl předveden v září 1995. Cílem jazyka bylo doplňovat Javu při vývoji webu [12], proto se syntaxe jazyka připodobnila Javě a název se změnil na JavaScript. První průlom jazyka nastal v roce 2009, kdy Ryan Dahl vydal multiplatformní běhovou aplikaci napsanou v JavaScriptu Node.js, která umožňovala, aby JavaScript běžel na straně serveru. Další průlom nastal okolo roku 2010, když Google vydal JavaScriptový framework pro vývoj webových aplikací AngularJS.

V roce 1997 firma byl JavaScript standardizován asociací ECMA International - mezinárodní soukromá nevýdělečná organizace, která normalizuje informační a komunikační systémy s otevřeným členstvím [14].

V roce 1998 byl vydán standard ECMAScript 2, v roce 1999 pak ECMAScript 3. Došlo k rozšíření o regulární výrazy, try/catch ošetření výjimek, specifičtější definice syntaktických chyb a formátování pro číselné výstupy. V roce 2005 se k týmu ECMA přidali i Eich a Mozilla a vytvořili E4X - rozšíření ECMAScriptu o nativní podporu XML. Tato verze byla v roce 2014 označená jako zastaralá. V červnu 2011 byl vydán ECMAScript 5.1 - oproti třetí verzi přibyly například gettery a settery, podpora JSONu. Plně splňuje standard ISO/IEC 16262:2011. V červnu 2015 byl vydán ECMAScript 6 (později přejmenován na ECMAScript 2015) - změna syntaxe (například deklarace tříd), iterátory, for...of cyklus, klíčová slova let a const, typovaná pole, generátory výjimek, kolekce a další. Nejnovější verze je jedenáctá verze s názvem ECMAScript 2020 [15], ve které mezitím přibyly mimo jiné asynchronní funkce nebo exponenciální operátor.

3.1.2 Vývojová prostředí

Mezi nejlepší vývojová prostředí pro JavaScript patří jednoznačně WebStorm, Visual Studio, Komodo a Atom. Za zmínku také stojí prostředí jako Brackets, NetBeans, Eclipse, nebo také PHPStorm (WebStorm + PHP + DB) a Visual Studio Code [41] [3] [16].

3.1.3 Knihovny a frameworky

JavaScript je v dnešní době používán převážně s frameworky a knihovnami, následující patří mezi nejpoužívanější.

React [33] je otevřená JavaScriptová knihovna představená v roce 2011 Facebookem. Slouží především k vytvoření dynamického uživatelského prostředí pro webové a mobilní aplikace. Hlavní výhodou Reactu je znovu použitelnost naprogramovaných komponent, které jsou izolované, takže změny neovlivňují ostatní komponenty.

Angular [1] je multiplatformní framework (definice frameworku v sekci 3.2.1) od Googlu s podporou HTML, který pomáhá psát klientskou stranu aplikací s architekturou MVC. Obvykle se využívá pro vývoj jednostránkových webů. Jednostránkový web je webová stránka, na které jsou všechny informace umístěné na jediné HTML stránce bez dalších podstránek [30]. Navigační menu na takových stránkách neodkazuje na URL jiných stránek, ale na kotvy umístěné v jednotlivých částech webu.

Vue [39] je progresivní framework pro tvorbu uživatelského prostředí a pokročilých jednostránkových webových stránek. Umožňuje webovou aplikaci rozdělit na několik částí, které je možné vyvíjet nezávisle na sobě [23]. Byl vydán v roce 2014, je multiplatformní, otevřený a reaktivní. Používá se také pro vývoj mobilních aplikací, jejichž vývoj usnadňuje nativním vykreslováním.

Další knihovnou je jQuery. Patří mezi nejvíce populární knihovny pro vývoj klientské strany webových aplikací. Používá ji přes 70 % webových stránek např. Google, MSN, DailyMotion nebo BBC. Byla vytvořena v roce 2006 Johnem Resignem. Nabízí například funkce pro procházení a změnu DOM, JavaScriptové pluginy, manipulaci s CSS, efekty a animace, AJAX aj.

Bootstrap [5] je otevřený CSS framework pro tvorbu responzivních webových stránek. Obsahuje návrhářské CSS a JavaScriptové šablony, které slouží k úpravě komponent rozhraní jako jsou tlačítka, formuláře apod. Vznikl v roce 2011.

3.1.4 Struktura

JavaScript se implementuje do HTML kódu webové stránky pomocí značek <script> JavaScriptový kód </script> [20].

JavaScript ignoruje mezery, tabulátory a nové řádky. Programátor těchto znaků může využít k formátování kódu. Je zvykem každou instrukci ukončit středníkem. Nicméně pokud jsou instrukce na různých řádcích kód se vykoná bezchybně. V případě více instrukcí na jedné řádce jsou středníky nezbytné.

Tento jazyk je citlivý na velikost písmen (proměnné line a LINE budou dvě různé proměnné). Komentáře jsou ve stejném stylu jako v jazyce C, Javě apod. Jednořádkové komentáře jsou za //, blok komentářů pak mezi /* a */. JavaScript také rozezná značku pro HTML komentář (<!- -) a považuje ji za jednořádkový komentář. Nicméně nepozná značku pro konec HTML komentáře (- ->), proto by měla být napsána jako //- ->.

Před ECMAScriptem 6 (2015) bylo možné deklarovat proměnnou pouze klíčovým slovem var. Od ES6 je možné použít i let a const, jejichž platnost je v kódu vymezena složenými závorkami. Const deklaruje konstantní proměnnou, proto musí být hodnota přidělena hned při její deklaraci.

Existují tři druhy proměnných - globální, lokální a proměnné s působností v rozsahu bloku. Globální i lokální se deklarují klíčovým slovem var. Výchozí hodnota proměnné je undefined. Proměnná musí začínat písmenem, podtržítkem, nebo znakem dolaru. Ve zbytku názvu proměnné se mohou objevit také číslice. Název se musí lišit od klíčových slov jazyka [34].

JavaScript má čtyři základní datové typy - undefined, number, string a boolean (od ES6 také symbol). Celá čísla jsou přesná na 15 číslic. Maximální počet desetinných míst je 17. Dělení nulou nevyhodí výjimku, výsledek je plus nebo mínus nekonečno. Číslo začínající nulou může být některými verzemi JavaScriptu interpretováno jako osmičkové.

3.1.5 TypeScript vs. JavaScript

TypeScript je nadstavba ECMAScriptu 6.

TypeScript VS JavaScript

- Lze spustit v prohlížeči, pokud je přidaná podpora pro TS. Nicméně to zvýší velikost souboru a zhorší to výkon. Řešením je kód zkompilovat předem a použít na stránce přímo JavaScript **VS** Lze spustit v jakémkoliv prohlížeči
- umožňuje statického typování i dynamické **VS** umožňuje pouze dynamické typování [13]

- kompilovaný jazyk **VS** interpretovaný jazyk
- TypeScript má poměrně malou komunitu **VS** JavaScriptová komunita je naopak velmi velká, snadno najdete řešení při problémech
- navržen pro velké komplexní aplikace **VS** navržen pro krátké skripty jako doplněk k Javě
- TypeScript podporuje moduly, rozhraní, generické programování **VS** JavaScript nic z toho nepodporuje [36]
- některé chyby lze díky tomu, že se jedná o kompilovaný jazyk, odchytit již při kompilaci, což usnadňuje vývoj aplikace **VS** veškeré chyby se projeví až při běhu programu
- TypeScript je poměrně složitý na naučení pro někoho, kdo nemá žádnou předchozí zkušenost se skriptováním **VS** JavaScript je poměrně jednoduchý dynamický jazyk, který je naopak vhodný pro začátečníky [37]
- Převážně využívání pro programování klientské strany aplikace **VS** Využíván pro obě strany aplikací[31]
- přípona souborů se zdrojovými kódy: .ts, .tsx **VS** .js
- TypeScript poskytuje přehlednější kód a statické typování, což je výhodné obzvlášť v týmu více vývojářů **VS** JavaScript je oproti tomu flexibilní jazyk, který disponuje pouze dynamickým typováním, což může v některých případech urychlit a zjednodušit vývoj

3.2 PHP

PHP (PHP: Hypertextový preprocesor v minulosti Personal Home Pages) je skriptovací programovací jazyk pro vytváření dynamických a interaktivních webových stránek. Je používaný na straně serveru, což znamená, že je třeba aby byl nainstalovaný pouze na něm a nevyžaduje instalaci na straně uživatele. Poprvé byl vydán v červnu 1995 a je distribuován pod licencí PHP License (otevřený software). K červnu 2021 je nejaktuálnější verzí 8.0.6. Nové verze vycházejí poměrně často, obvykle v řádu několika týdnů (podverze například 8.x.x). Je poměrně jednoduchý na naučení, má velkou komunitu a rozsáhlou dokumentaci. Mezi další jeho výhody bezpochybně patří, že je multiplatformní. Podporuje mnoho různých databázových systémů (mimo

jiné PostgreSQL, Oracle, MS SQL Server), pro MySQL má přímo zabudovanou podporu [32].

3.2.1 PHP Framework

Framework je sbírka mnoha funkcí (ideálně ale tříd a objektů), které programátorovi šetří práci a tím urychlují vývoj aplikace. Kód je také kratší a přehlednější. Mezi další důvody, proč používat framework, patří také větší stabilita aplikace.

Laravel

Laravel je framework pro vývoj webových aplikací s MVC architekturou. Byl vyvinut Taylorem Otwellem, který chtěl vytvořit framework s funkcionalitami, které neposkytuje CodeIgniter [6]. Vydán byl v červnu 2011 a je poskytován jako otevřený projekt pod MIT licencí. Mezi základní funkcionalitu Laravelu patří autentizace, směrování, nástroje pro komunikaci s databází, práce s relacemi a ukládání dat do mezi paměti.

Laravel kromě obsáhlé dokumentace poskytuje také neplacené video tutoriály k samotnému frameworku i jazyku PHP. Ke spravování závislostí využívá Composer (nástroj spravující závislosti v PHP v rámci konkrétního projektu [10]). Součástí Laravelu je šablonovací nástroj Blade, který na rozdíl od jiných šablonovacích nástrojů umožňuje v pohledech používat PHP.

Popis Laravelu na stránkách neoficiální české dokumentace: *Je postaven na návrhových vzorech a dodržuje principy jako jsou například DRY, KISS, SOLID atd. Snahou Laravelu je dodržovat jednoduchost a čitelnost kódu. Framework poskytuje opravdu jednoduchou práci s databází a nabízí vlastní ORM Eloquent či migrace, které pomáhají práci v týmu, aby byla databáze jednoduše synchronizována bez nutnosti složitých mechanismů. Laravel také zcela podporuje RESTful architekturu a umožňuje tak snadno vytvářet cesty (routes) či API* [22].

Symfony

Symfony je množina znovupoužitelných PHP komponent a zároveň PHP framework pro webové projekty. Byl vydán v listopadu 2005 a je z velké části inspirován jinými webovými aplikačními frameworky jako Ruby on Rails, Django nebo Spring. Stejně jako Laravel je otevřený a vydáván pod licencí MIT. Symfony podporuje nejvíc databázových systémů z frameworků zmíněných v této kapitole – Drizzle, MySQL, Oracle, PostgreSQL, SAP Sybase SQL Anywhere, SQLite a SQL Sever. Součástí frameworku je i nástroj pro

ladění. Pro pohledy využívá šablonovacího nástroje Twig. Twig neumožňuje používat PHP. Výhodou Twigu je bezesporu zvýšení bezpečnosti aplikace za pomocí automatického escapování obsahu proměnných [40]. Symfony má také na rozdíl od ostatních frameworků komerční podporu od francouzské firmy SensioLabs. Ta také nabízí placené online kurzy SensioLabs University pro Symfony a PHP.

CakePHP

CakePHP je otevřený webový framework napsaný v jazyce PHP a distribuovaný pod licencí MIT. Dodržuje MVC architekturu a jeho koncept vychází z Ruby on Rails. CakePHP je navržen tak, aby časté úkony při vývoji webových aplikací byly snadné a rychlé. Pravděpodobně největší výhodou tohoto frameworku je menší konfigurace než u jeho konkurentů. Má zabudovanou validaci vstupních dat, která chrání před CSRF (Cross-site Request Forgery – jedna z metod útoku na webové aplikace) [7].

Laminas

Laminas Project je otevřený projekt, který je pokračováním Zend Frameworku jednoho z nejpopulárnějších PHP frameworků vůbec. Byl vytvořen v roce 2006 a distribuován pod BSD licencí. Laminas poskytuje nástroje pro vkládání závislostí (dependency injection), ukládání dat do mezi paměti, validaci vstupních dat formulářů, autentizaci, směrování, stránkování, práce s relacemi, diagnostické testy a další. Laminas podporuje poměrně hodně databázových systému mazi které patří i MySQL, MS SQL Server, Oracle, MariaDB, PostgreSQL [18]. Modulární architektura aplikace minimalizuje závislosti mezi komponentami, což mimo jiné přispívá ke znovu použitelnosti jednotlivých komponent [24]. V dokumentaci je poměrně složité se orientovat, alespoň v porovnání s ostatními uvedenými frameworky.

4 Způsob ukládání souborů

Oznámení v podobě souborů (pro URL pouze metadata) bude třeba někam ukládat a způsob jejich uložení má vliv na rychlost, škálovatelnost, bezpečnost i cenu (pronájem cloudového úložiště, cena disku apod.).

V každém případě bude třeba vybrat databázový systém, aby se alespoň metadata (případně data) měla kam ukládat. Je třeba, aby byl tento databázový systém podporován vybraným PHP frameworkem.

Mezi nejpoužívanější metody pro ukládání multimediálních souborů patří:

- cloudová úložistě (AWS a podobné služby) a do databáze uložit cestu k souboru a ostatní informace
- ukládat soubory jako BLOB v databázi
- ukládat soubory na disk a do databáze cestu k souboru na disku a ostatní informace

4.1 Cloudová úložiště

Cloudové úložiště je služba, která umožňuje ukládat data tak, že se přenesou přes internet nebo jinou síť do úložného systému mimo budovu (obvykle mimo pracoviště nebo domov), který je spravovaný třetí stranou. Existují stovky různých systémů cloudových úložišť, od osobních úložišť pro ukládání nebo zálohování e-mailů, obrázků, videí a dalších osobních souborů konkrétního uživatele až po podniková úložiště, která firmám umožňují využít cloudové úložiště jako komerčně podporované řešení vzdáleného zálohování, kde mohou bezpečně přenášet a ukládat datové soubory nebo je sdílet mezi jednotlivými umístěními [9].

Jednou z výhod ukládání dat na cloudová úložiště je, že se není třeba starat o zálohování dat na cloudovém úložišti, je třeba zálohovat pouze databázi s metadaty. Zároveň jsou data přístupná kdekoliv na světe, kde je připojení k internetu. Další obrovskou výhodou je velmi nízká pravděpodobnost ztráty dat. V dnešní době není u cloudových úložišť třeba ani velká obava o bezpečnost. Mnoho cloudových úložišť je zabezpečených 256-bit AES šifrováním, nebo dokonce Zero knowledge šifrováním - jen vy znáte kód k vašim datům, takže i kdyby se podařilo útočníkovi k vašim datům dostat, nerozšifroval by je [17]. Další výhodou je jednoduchá a vysoká škálovatelnost. Největší nevýhodou cloudových úložišť je bezpochyby cena.

4.1.1 Amazon Web Services

Jednou z nejvyužívanějších cloudových služeb je Amazon Web Services. Amazon Web Services je dceřiná společnost Amazonu, která poskytuje cloudové úložiště a výpočetní výkon. Služba je placená formou předplatného [2]. Služba využívá serverových farem po celém světě. Mezi asi nejpoužívanější produkt této společnosti pro ukládání dat je Amazon S3. Služby jsou také navržené tak, že je téměř nulová pravděpodobnost ztráty dat [38].

4.2 BLOB

Multimediální soubory se dají ukládat do databáze jako BLOB (binary large object), což jsou blíže nespecifikovaná binární data v databázi [4]. Jedna z výhod tohoto způsobu uchovávání dat je snadná záloha, protože jsou všechna data a metadata na jednom místě oproti ukládání na disk. Mezi nevýhody pak patří to, že se tím může podstatně zvětšit velikost databáze (například pokud budete v databázi takto ukládat videa) a následně možný snížený výkon databáze.

4.3 Disk

Soubory jsou v tomto případě uloženy na disku na serveru a informace o nich (metadata) včetně cesty k souboru na disku jsou uloženy v databázi. Přístup k souboru je možný procházením adresářů, servery jsou poměrně jednoduše škálovatelné pro nízké objemy dat a lze využít Storage Area Network (SAN), což je dedikovaná datová síť, která slouží k připojení externích zařízení k serverům (disková pole a jiná zálohovací zařízení) [35]. V případě jednoduché aplikace s několika uživateli se toto jeví jako nejjednodušší řešení, avšak v případě velké veřejné aplikace třeba s videi, by se asi jen těžko dalo vyhovět požadavkům na kapacitu disku. Tedy toto je vhodné pro aplikace, kterým bude stačit kapacita několika TB. Jednou z nevýhod je možná nekonzistentnost. Může dojít k vymazání souboru na disku, ale zůstat záznam v databázi a naopak. Tato nevýhoda platí i pro cloudová úložiště. Stejně tak stačí, aby složku na disku někdo přesunul jinam a najednou se není k datům jak dostat. Záloha souborů je také složitější. Musí se zálohovat data z disku i data z databáze.

5 Možnosti prohlížečů

Webových prohlížečů je nepřeberné množství, avšak většina uživatelů používá několik nejběžnějších prohlížečů - Google Chrome, Mozillu Firefox, Apple Safari (jak už z názvu vyplývá výchozí prohlížeč pro Apple zařízení), Microsoft Edge (výchozí prohlížeč pro Windows) a Operu. Někteří ještě používají Internet Explorer, ale jeho podpora již v červnu 2022 skončí a není ani zdaleka tak používaný jako dřív. Jeho oblíbenost klesla a na novějších zařízeních ho nahradil jeho nástupce Microsoft Edge. Je potřeba při vývoji webových aplikací brát na vědomí to, že jednotlivé prohlížeče mohou zobrazovat pohledy různě a mohou podporovat různé verze jednotlivých technologií a podle toho tomu vývoj také uzpůsobit a testovat alespoň na několika těchto prohlížečích, ideálně na všech. Co se týká multimédií, podpora jejich zobrazení a následně jejich zobrazení, je plně v rukou programátora daného prohlížeče. V dnešní době jsou však nejčastější formáty multimédií podporovány ve všech jmenovaných prohlížečích.

5.1 Technologie pro zobrazování multimédií

Nejčastějším způsobem zobrazování multimédií ve webových prohlížečích je pomocí jazyka HTML s využitím značek <video>, <img>, <iframe> nebo <object>. Dají se tak zobrazit obrázky a videa různých formátů, animace, PDF dokumenty i jiné webové stránky (například YouTube video). Nicméně HTML pouze dokáže zobrazit soubor daný atributem *src*. Pokud je třeba jakkoli s médii interagovat (zastavit, spustit video nebo například při prohlížení fotek obrázky měnit), HTML nebude stačit. Nabízí se tedy JavaScript, pomocí kterého lze změnit atribut *src*, disponuje funkcemi pro ovládání videa, stejně tak dokáže změnit styl zobrazování HTML elementu i po načtení webové stránky.

Dříve se také velmi často využívalo Javovských apletů, pluginů pro zobrazování multimédií. Asi nejznámějším pluginem je Adobe Flash Player, který byl před deseti lety naprosto nepostradatelný. V současné době jeho podpora již skončila (ke konci roku 2020).

5.2 Možnosti nastavení prohlížeče

Webový prohlížeč může ovlivňovat zobrazení také svým uživatelským nastavením. Například, pokud uživatel nastaví zobrazení na jinou hodnotu než 100 %, obraz se adekvátně zvětší, či zmenší, a přebije tak nastavení naprogramované ve webové aplikaci - typicky kaskádové styly.

Webové prohlížeče také disponují tzv. kioskovým módem. Kioskový mód nabízí většina webových prohlížečů a znamená to, že aplikace je zobrazená přes celou obrazovku a schovají se všechny ostatní věci, které obvykle uživatel vidí – menu, nástroje, záložky a podobně. Cílem kioskového módu je obvykle zabránění uživateli v jiných akcích, než je zobrazení webové aplikace a těch, která webová aplikace nabízí [21]. Přesně to bude potřeba u zobrazování oznámení této aplikace, kdy chceme uživateli pouze ukázat nadcházející akce, důležité informace a podobná oznámení, ale nechceme, aby s aplikací, potažmo webovým prohlížečem, jakkoliv interagoval.

6 Implementace

Na základě analytické části jsem zvolila vhodné technologie a řešení pro danou problematiku. Kapitola pojednává o popisu a implementaci tohoto řešení.

6.1 Použité řešení a technologie

Server bude tvořit třívrstvá aplikace s MVC architekturou, která bude poměrně jednoduchá, proto pro její implementaci stačí PHP. Z PHP frameworků jsem vybrala Laravel kvůli tomu, že je jeden z jednodušších, pokud někdo s vývojem webových aplikací teprve začíná, je rychlejší na naučení, jeho syntax je jednoduchá a přehledná a má rozsáhlou dokumentaci. Disponuje všemi potřebnými funkčnostmi pro účely této aplikace - autentizace, šablony, přístup k databázi.

Při výběru mezi TypeScriptem a JavaScriptem jsem se nakonec rozhodla, že pro účely této aplikace bude lepší použít JavaScript, hlavně kvůli tomu, že jsem před vývojem této aplikace neměla ani s jedním jazykem žádné zkušenosti. Před programováním v TypeScriptu je dobré znát alespoň JavaScriptové základy (tím že se stejně překládá zpět do JavaScriptu), učení jazyka je složitější a delší a má režii navíc, která je v případě této jednoduché aplikace zbytečná.

Pro ukládání dat jsem vybrala možnost ukládat data na server a metadata do databáze. Objem dat pro tuto aplikaci by neměl být tak velký, aby mu nedostačoval pevný disk (maximálně v řádu jednotek TB, pravděpodobně méně). MySQL databázi jsem zvolila proto, že s ní mám nejvíc zkušeností.

Pro zajištění přenositelnosti aplikace jsem použila Docker.

K vývoji aplikace byly použity tyto technologie:

- PHP verze 8.1.2
- MySQL verze 8.0
- HTML verze 5
- CSS verze 3
- Laravel verze 8.83.7

- Composer 2.2.5
- JavaScript verze 5 a vyšší
- Docker verze 20.10.13

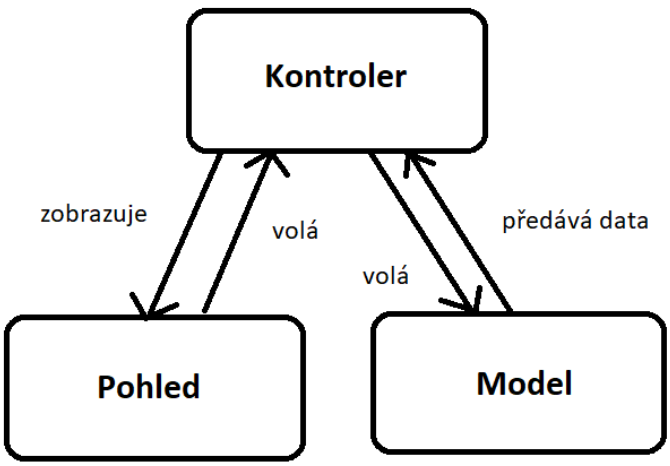
6.2 MVC model

MVC (Model, View, Controller) je často aplikovaný model při vývoji webových aplikací (znázorněn na obrázku 6.2). MVC architektura odděluje aplikační logiku od zobrazovaní výstupu. Aplikace je pružnější a snadno modifikovatelná – jednoduše se vymění jedna vrstva aplikace, nebo se změní databázový systém (například z MySQL na SQLite). Tato architektura také zajišťuje přehlednější kód a roste znovu použitelnost jednotlivých částí aplikace pro vývoj dalších aplikací.

Model zajišťuje komunikaci s databází, konverzi a kontrolu dat. Pohled (View) zajišťuje formátování výstupu.

Kontroler (Controller) nebo také ovladač či řadič zpracovává vstup a předává ho modelu. Řídí aplikaci. V rámci webových aplikací to znamená, že existuje hlavní soubor aplikace, který do projektu zahrne všechny ostatní potřebné soubory. Stejně tak podle dostupné konfigurace vytvoří spojení s databází, Podle URL pak určí typ obsahu, který se pomocí PHP wrapperu uloží do proměnné a vše se odešle do šablony k zobrazení.

V šabloně (a pouze v šabloně) by se měla nacházet celá logika zobrazení a HTML. Model zajišťuje veškerou logiku – výpočty, databázové dotazy, validace a podobně. Kontroler tyto dvě vrstvy propojuje spolu s uživatelem.



Obrázek 6.2 MVC model

6.3 Struktura aplikace

Úplná struktura aplikace je popsána v dokumentaci laravelu [26]. Nejdůležitější soubory (z pohledu vývoje a údržby aplikace) jsou v následujících adresářích.

6.3.1 app

V podadresáři *Models* se nacházejí soubory s modely pro práci s databází. V podadresáři *Http\Controllers* se nacházejí všechny kontrolery aplikace. V adresáři *Http\Controllers\Auth* se nachází kontrolery pro autentizaci, které poskytuje přímo Laravel.

6.3.2 resources\css

Obsahuje soubory s kaskádovými styly použitými napříč aplikací.

6.3.3 resources\views

Obsahuje veškeré pohledy použité v aplikaci. V podadresáři *auth* se nachází pohledy pro autentizaci poskytované Laravelem. V podadresáři *layouts* jsou šablony pro aplikaci. Podadresář *vendor\pagination* obsahuje styly pro stránkování.

6.3.4 routes

V souboru `web.php` se nastavuje směrování aplikace.

6.3.5 config

V této složce se nachází konfigurační soubory aplikace. Zvlášť důležité při vývoji a konfiguraci aplikace byly soubory *app.php* a *database.php*.

6.4 Model

Jak již bylo zmíněno, Laravel podporuje čtyři databázové systémy – SQL Server, PostgeSQL, SQLite a MySQL. Laravel poskytuje pro práci s databázemi hned několik nástrojů – třídu DB, Schema Builder, Eloquent ORM a Query Builder.

6.4.1 Třída DB

Nachází se v adresáři *vendor\laravel\framework\src\Illuminate\Support\-Facades*. Pro používání třídy DB musí být správně nastavené údaje přístupu k databázi v souboru *config\database.php*.

6.4.2 Schema Builder

Schema je třída, která umožňuje pracovat s tabulkami dané databáze. Například může vytvářet nové tabulky, přidávat sloupce do již existující tabulky, mazat a měnit sloupce (například jméno sloupce).

6.4.3 Elouquent ORM

ORM (Objektově relační mapování) je programovací technika v softwarovém inženýrství, která zajišťuje konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem [29].

Elouqent ORM je knihovna, která je součástí Laravelu pro práci s databází (lze ji použít i samostatně). Využívá architektonického návrhového vzoru Active Record. Každá tabulka má odpovídající „model“ (třídu), který knihovna použije při interakci s danou tabulkou. Aby knihovna fungovala, je stejně jako u DB třeba správně nakonfigurovat připojení k databázi v souboru *config\database.php* [27].

Elouquent model lze vytvořit příkazem `php artisan make:model název_modelu`. V takovém případě se vytvoří třída *Název_modelu* a pokud se explicitně nespecifikuje jiný název tabulky, bude použito množné číslo názvu (v angličtině) – například model *User*, tabulka *users*.

6.4.4 Query Builder

Query Builder je rozhraní, které poskytuje pohodlnější a snazší práci s databází (zadávání příkazů) než samotná třída DB. Chrání před SQL útoky (využívá PDO).

6.4.5 Databáze

Databáze aplikace obsahuje pět tabulek. Jednu pro uživatele `users`, druhou pro oznámení `files`, třetí pro pořadí promítání `projection_order`, čtvrtou pro režimy `modes` a poslední je vazební tabulka mezi `files` a `modes` - `files_modes`. K těmto tabulkám se přistupuje pomocí tříd *File*, *User* a

Mode (pro `modes` a `files_modes`) a `ProjectionOrder`, které se nacházejí v adresáři *app/Models*. Modely využívají Eloquent ORM a třídu DB.

Tabulka users

Tabulka uživatelů musí kromě identifikátoru uživatele uchovávat také heslo. Kvůli bezpečnosti musí být toto heslo zahašované (Laravel využívá pro hašování Bcrypt). Dále je nutné, aby obsahovala e-mail, který slouží k přihlášení do systému. Jelikož je v rámci aplikace stěžejní, aby všichni uživatelé neměli stejná práva, je třeba uživateli přiřadit ještě jednu ze tří rolí (popsané v sekci 2.1), která určí pravomoce uživatele.
`id` - identifikátor uživatele
`username` - uživatelské jméno
`password` - heslo uživatele
`role` - role uživatele v systému, může nabývat hodnot r/o, r/w nebo admin
`email` - e-mail uživatele nutný pro přihlášení
`remeber_token` - pokud není NULL, systém uživatele automaticky neodhlásí (čeká na ruční odhlášení uživatele)
`created_at` - časová značka
`updated_at` - časová značka
`email_verified_at` - časová značka

Tabulka files

Tabulka pro oznámení uchovává metadata o souborech a parametry promítání pro daná oznámení (je oznámení aktivní, délka promítání apod.)
`fileID` – identifikátor souboru
`name` – název souboru
`size` – velikost souboru
`type` – typ souboru (přípona/ případě URL url)
`path` – cesta k souboru na disku, v případě URL celé URL
`video_length` - délka videa
`active` – nabývá hodnot Y/N – Y, pokud je oznámení aktivní (zobrazuje se ve smyčce), N, pokud není aktivní
`active_length` – na jak dlouho se oznámení zobrazí ve smyčce - jedná se o celé kladné číslo vyjadřující dobu v sekundách
`date` – datum přidání oznámení
`date_from` – v případě aktivních oznámení může být nastaveno od jakého datumu, se má oznámení ve smyčce zobrazit (např. oznámení bude aktivní, datum od bude nastaven na 1.2. 2021 – oznámení se bude promítat, pokud je 1. 2. 2021 nebo po tomto datumu)

`date_to` – stejně jako u *date_from* v případě aktivních oznámení určuje mez promítání. V tomto případě se jedná o datum, do kterého se oznámení bude promítat
`created_at` - časová značka
`updated_at` - časová značka

Tabulka modes

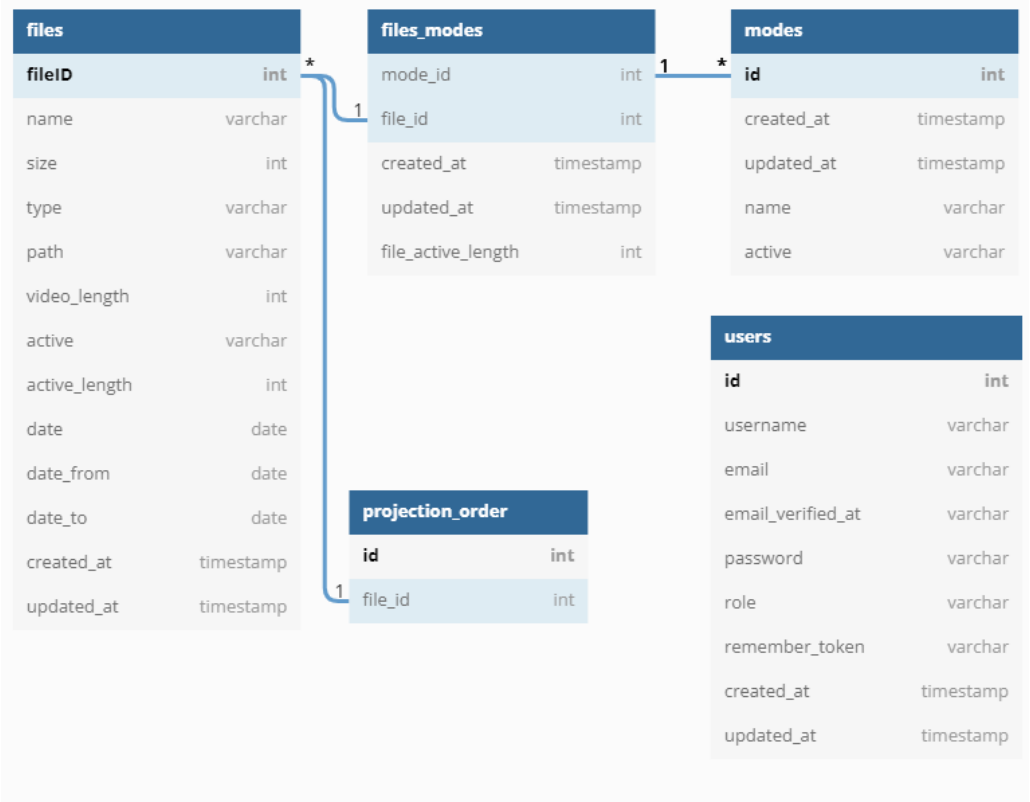
Tato tabulka uchovává informace k jednotlivým režimům promítání. Kromě identifikátoru obsahuje název a jestli je režim aktivní (má se promítat).
`id` - identifikátor režimu
`name` - název režimu
`active` - nabývá hodnot Y/N – Y, pokud je režim aktivní, N, pokud není aktivní
`created_at` - časová značka
`updated_at` - časová značka

Tabulka files_modes

Jedná se o vazební tabulku mezi tabulkami `files` a `modes`. Slouží k přiřazování oznámení jednotlivým režimům a k zaznamenávání délky jejich promítání, která se může lišit od výchozího promítání aktivních oznámení i mezi jednotlivými režimy.
`mode_id` - cizí klíč, odkazuje na identifikátor v tabulce *modes*
`file_id` - cizí klíč, odkazuje na identifikátor v tabulce *files*
`file_active_length` - délka promítání oznámení v rámci toho režimu
`created_at` - časová značka
`updated_at` - časová značka

Tabulka projection_order

`id` - identifikátor režimu
`file_id` - cizí klíč, odkazuje na identifikátor v tabulce *files*



Obrázek 6.4.5 Schéma databáze

6.5 View

Pohledy (view) oddělují zobrazovací logiku, která by se měla nacházet jen zde, od aplikační logiky. V Laravelu se pohledy ukládají do složky *resources/views*. Jména pohledů jsou ve formátu *název.blade.php*. Přípona *.blade.php* informuje Laravel, že soubor obsahuje Blade šablonu. Taková šablona může obsahovat HTML, PHP a Blade příkazy, které zjednodušují a zpřehledňují psaní kódu. Tyto příkazy se pak kompilují do čistého PHP. Pokud pak při směrování vrací metoda daný pohled, stačí napsat název souboru před příponou. Tedy například pro *users.blade.php -> users*. Ukázka:

```
Route::get('/', function ()
   return view('nazev_pohledu');
);
```

Pohledu lze předat také proměnné. Toto pole proměnných se předá jako druhý argument.

```
return view('nazev_pohledu', [
   'jmeno' => 'Jakub',
```

```
   'prijmeni' => 'Novak'
]);
```

Pro vypsání hodnoty proměnné, pak stačí v pohledu napsat `{{ $jmeno }}`, což je Blade příkaz pro výpis (samozřejmě lze také použít příkaz *<?php echo $jmeno; ?>)*. Alternativně lze pro pohled použít i zápis:

```
return view('nazev_pohledu')
   ->with('jmeno' => 'Jakub')
   ->with('prijmeni' => 'Novak');
```

Pohledy se také můžou nacházet v podsložkách adresáře *resources/view*. Pak je potřeba zadat před jméno pohledu název složky(složek), ve které se nachází oddělený tečkou.

```
return view('podslozka.nazev_pohledu', [
   'jmeno' => 'Jakub',
   'prijmeni' => 'Novak'
]);
```

6.5.1 Složka files

Složka `files` obsahuje tři pohledy - index, insert a files a podložku `modes`.

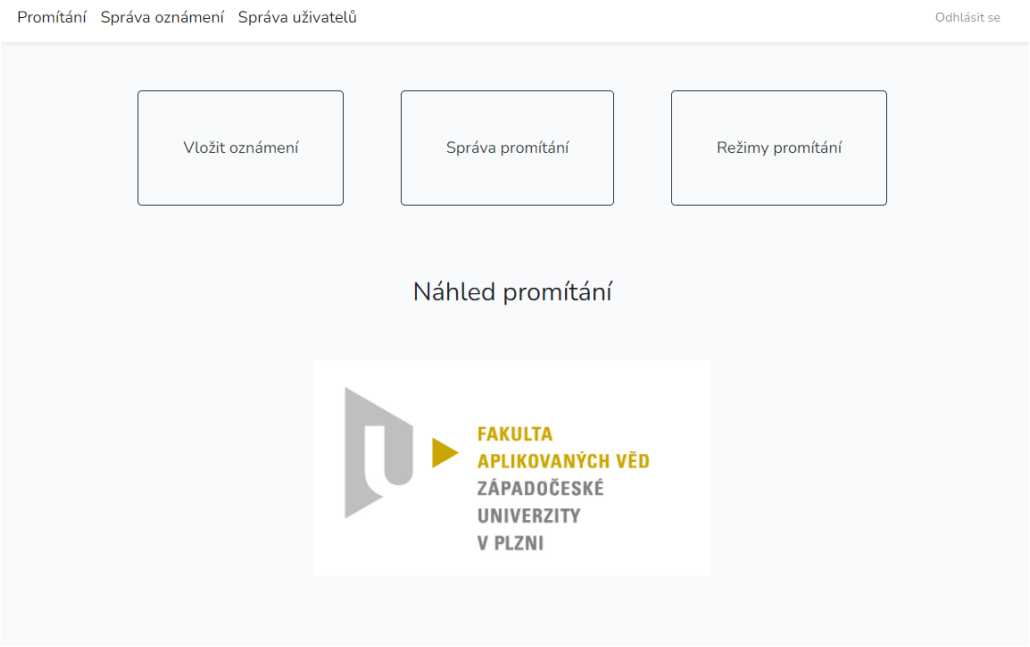
index.blade.php

Pohled *files.index* slouží jako rozcestník mezi ostatními pohledy ve složce `files`.

Pro nepřihlášeného uživatele se pouze vypíše, že je třeba se přihlásit a odkaz na stránku s přihlášením.

Přihlášenému uživateli se zobrazí tři tlačítka, která uživatele následně přesměrují (pohled je vidět na obrázku 6.5.1. a)). První slouží k přesměrování na *files.insert*, druhé na *files.files* a poslední na *files.modes.index*.

Pod tlačítky se nachází náhled promítání. Pomocí HTML značek iframe, img a video se promítají vybraná oznámení (buď jeden z režimů, nebo aktivní oznámení) s prodlevou jedna vteřina.



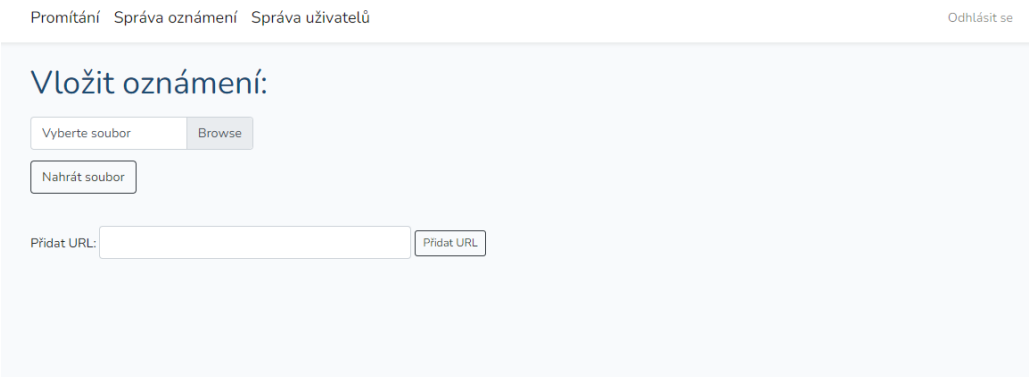
Obrázek 6.5.1 a) Správa oznámení - rozcestník

insert.blade.php

Pohled *files.insert* slouží k vložení nového oznámení.

Pro nepřihlášeného uživatele se pouze vypíše, že je třeba se přihlásit a odkaz na stránku s přihlášením.

Pro přihlášeného uživatele s rolí *r/o* se vypíše, že nemá dostatečná oprávnění. Pro ostatní se zobrazí sekce *Vložit oznámení*, kde je možné nahrát soubor nebo lze přidat URL. Pohled je vidět na obrázku 6.5.1 b).



Obrázek 6.5.1 b) Správa oznámení - vložení oznámení

files.blade.php

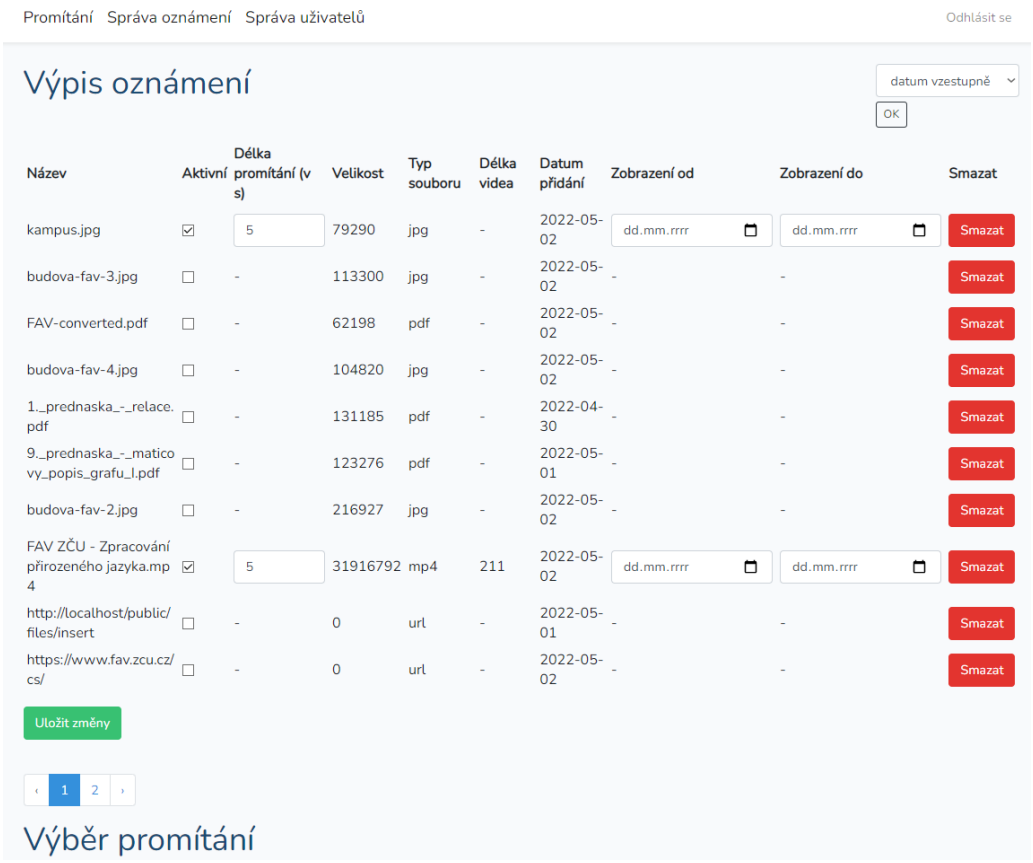
Tento pohled slouží pro správu oznámení a využívá kaskádový styl ze souboru *public/css/files.css*. Pro nepřihlášeného uživatele se pouze vypíše, že je

třeba se přihlásit a odkaz na stránku s přihlášením.

Přihlášenému uživateli se může zobrazit různý obsah podle role v systému. Všem, bez ohledu na roli, se zobrazí tabulka s oznámeními a jejich metadaty, která jsou uložena v databázi v tabulce *files*. Tabulka obsahuje sloupečky název, aktivní, délka promítání, velikost, typ souboru, datum přidání, zobrazení od a zobrazení do (sloupečky představují datum, do kterého nebo od kterého se má oznámení promítat). Správcům oznámení a administrátorům se ještě zobrazí sloupeček smazat, který obsahuje tlačítka pro smazání jednotlivých oznámení. Pohled pro administrátory a správce oznámení lze vidět na obrázku 6.5.1 c).

Vpravo od tabulky se nachází filtr oznámení. Oznámení lze filtrovat podle data přidání (buď sestupně nebo vzestupně). Dále lze zobrazit výpis pouze aktivních položek, které se promítají ve smyčce, nebo pouze videa (soubory s příponou .avi a .mp4), obrázky (přípona .png, .jpg, .jpeg, .gif a .jfif) nebo PDF soubory. Výchozí volba je zobrazení všech oznámení podle data přidání vzestupně.

Pod tabulkou se nachází sekce *Výběr promítání*, kde jsou vypsány všechny režimy aplikace a výchozí hodnota promítání *Aktivní oznámení*, vedle kterých se nacházejí přepínače (radio button). Uživatel tak může přepínat mezi promítáním jednotlivých režimů a aktivních oznámení.



Obrázek 6.5.1 c) Správa oznámení - správa promítání

6.5.2 Složka modes

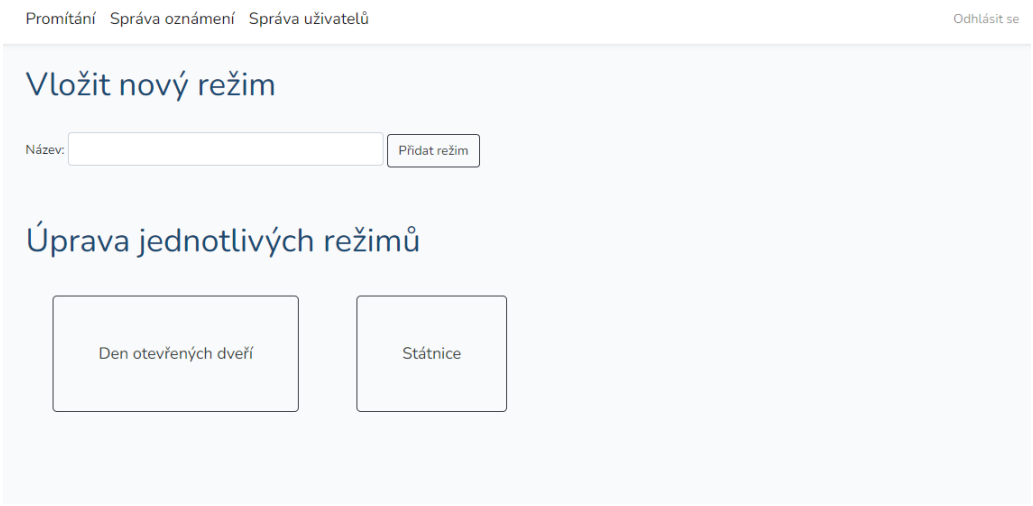
Složka `modes` je podsložkou `files` a obsahuje pohledy pro správu režimů promítání.

Pro nepřihlášené uživatele se u všech pohledů této složky pouze vypíše, že je třeba se přihlásit pro přístup k administraci a odkaz na stránku s přihlášením.

index.blade.php

Pro přihlášeného uživatele pohled *files.modes.index* obsahuje sekci pro vložení nového režimu a stejně jako v případě *files.index* slouží také jako rozcestník mezi jednotlivými režimy. Pohled můžete vidět na obrázku 6.5.2 a).

Pod sekcí pro vložení nového režimu se nachází jednotlivá tlačítka, jejichž id je ve formátu `mode_id`, kde `id` je identifikátor režimu v databázové tabulce *modes*. Po stisknutí tlačítka je uživatel přesměrován url adresu *název_serveru/public/files/modes/id*, kde se uživateli zobrazí pohled `files.modes.modes`.



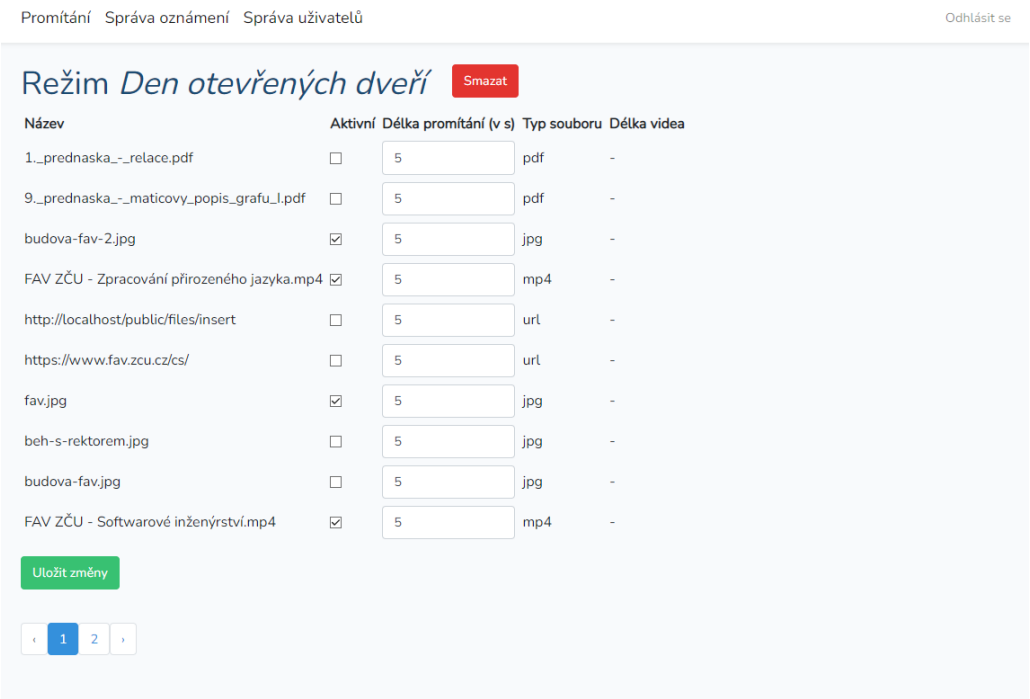
Obrázek 6.5.2 a) Správa oznámení - správa režimů

modes.blade.php

Pohled slouží pro správu jednotlivých režimů. Identifikátor režimu je předán pomocí url adresy.

Přihlášenému uživateli se zobrazí stránka s nadpisem Režim *název režimu*, vedle kterého se nachází červené tlačítko pro smazání. Po stisknutí tlačítka vyskočí upozornění, jestli uživatel opravdu chce režim smazat. Po potvrzení aplikace přesměruje uživatele zpět na pohled *files.modes.index*, který je vidět na obrázku 6.5.2 b).

Pod nadpisem se nachází tabulka s výpisem všech oznámení nahraných v aplikaci. Tabulka obsahuje sloupečky název, aktivní, délka promítání, typ souboru a délka videa. Délku promítání lze změnit pouze u aktivních souborů.



Obrázek 6.5.2 b) Správa oznámení - správa jednotlivých režimů

6.5.3 show.blade.php

Pohled show řeší promítání aktivních oznámení nastavených v pohledu *files*. Uživateli se zobrazí promítané oznámení na velmi tmavě šedém pozadí, nebo bílý text *Momentálně se nepromítají žádná oznámení* (pohled s promítaným oznámením lze vidět na obrázku 6.7). Pohled obsahuje značky <iframe>, <img> a <video>. Pohledu jsou předány PHP proměnné `$file_paths, $active_lengths a $file_paths`. Hodnoty těchto proměnných se pomocí cyklů foreach postupně překopírují do JavaScriptových proměnných, aby se s nimi dalo dále pracovat (názvy proměnných filePaths, activeLengths a filePaths). Poté se zkontroluje, jestli jsou v proměnné `filePaths` cesty k souborům, které se mají promítat. Pokud by byla proměnná filePaths prázdná (ať už z technických důvodů, nebo proto, že žádný soubor nebyl uživateli systému vybrán), vypíše se pouze hlášení *Momentálně se nepromítají žádná oznámení.*

Pokud jsou dostupné cesty k oznámením, zavolá se asynchronní metoda `showContent()` (kód na obrázku 6.5.3 a)), která v sobě má zanořené dva cykly. Nekonečný cyklus while, který zajišťuje, aby se promítání nezastavilo a v něm je vnořený cyklus for, který iteruje přes všechna aktivní oznámení, jejichž cesty byly předány pohledu v proměnné `$file_paths`. Ve for cyklu se postupně zavolají dvě metody spolu s klíčovým slovem `await`, které zajišťuje,

že se v cyklu bude pokračovat až po vykonání kódu zavolané metody.

```
async function showContent() {
    while (true) {
        for (i = 0; i < "{{sizeof($file_paths)}}"; i++) {

            await changeContent(filePaths[i], fileTypes[i]);
            await sleep(activeLengths[i] * 1000); // seconds to ms

        }
    }
}
```

Obrázek 6.5.3 a) Metoda showContent()

Nejdříve se zavolá metoda `changeContent()` a jako parametry jsou jí předány cesta k oznámení a typ oznámení. V této metodě se podle typu oznámení rozhodne, jak se bude promítat. Pro obrázky se použije HTML element <img>, jehož výška je nastavena na celou obrazovku a jeho zobrazení se nastaví na *block*. Do atributu *src* se nastaví cesta k obrázku. U elementů <iframe> a <video> se naopak zobrazení nastaví na *none*, aby se schovaly a byl vidět pouze obrázek. Také se zastaví video, aby neběžel zvuk v pozadí v případě, že by předtím bylo puštěné. U ostatních typů oznámení se postupuje obdobně, jen se u videí použije značka <video> a následně se spustí a pro PDF dokumenty a URL se použije <iframe>. Popsaná část kódu je vidět na obrázku 6.5.3 b).

```
async function changeContent(filePath, fileType) {
    document.getElementById("diviframe").style.display = "block";

    var frame = document.getElementById("frame");
    var img = document.getElementById("img");
    var video = document.getElementById("video");


    if (!fileType.localeCompare("pdf") || !fileType.localeCompare("url")) {
        img.style.display = "none";
        video.style.display = "none";
        video.pause();
        frame.style.display = "block";
        frame.src = filePath;
    } else if (!fileType.localeCompare("mp4") || !fileType.localeCompare("avi")) {
        img.style.display = "none";
        frame.style.display = "none";
        video.style.display = "block";
        video.src = filePath;
        video.play();
    } else {
        video.style.display = "none";
        video.pause();
        frame.style.display = "none";
        img.style.display = "block";
        img.src = filePath;
    }
}
```

Obrázek 6.5.3 b) Metoda changeContent()

Poté se zavolá metoda `sleep()`, které je jako argument předána délka promítání oznámení vynásobená tisícem („spánek“ je v milisekundách). Počká se, až se tato metoda dokončí (jinými slovy, než uběhne doba, po kterou se mělo oznámení promítat) a cyklus pokračuje pro další oznámení.

6.5.4 usersadmin.blade.php

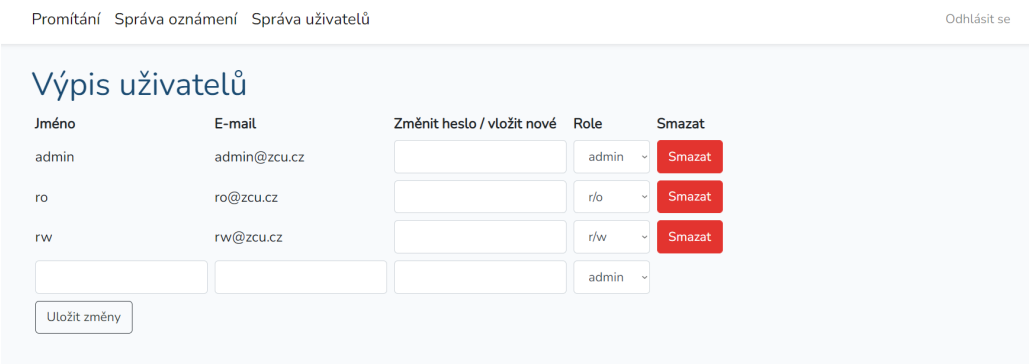
Pohled slouží k zobrazení správy uživatelů. Jsou mu předány proměnné `$users, $insert_user_ok, $insert_user_empty_input` a `$insert_user_with _existing_email`. Pro nepřihlášeného uživatele se pouze vypíše *Pro přístup k administraci je třeba se přihlásit* a vedle toho odkaz na přihlášení. Pro přihlášeného uživatele s rolí r/w nebo r/o se vypíše *Pro přístup k administraci uživatelů nemáte potřebná oprávnění.*

Pokud chce pohled zobrazit administrátor, zobrazí se mu tabulka se všemi uživateli, která obsahuje sloupečky se jménem uživatele, e-mailem, polem pro vložení nového hesla (za účelem změny hesla pro stávajícího uživatele nebo nového hesla pro nového uživatele), vybranou rolí a tlačítkem pro smazaní uživatele. Tento pohled lze vidět na obrázku 6.5.4.

V případě, že uživatel klikne na tlačítko smazat, vyskočí administrátorovi upozornění, jestli opravdu chce daného uživatele smazat. V případě

změny hesla, také vyskočí upozornění se jménem uživatele a je třeba, aby ho administrátor potvrdil. Pohled umožňuje uživateli roli změnit na jednu z hodnot admin (administrátor), r/w (správce oznámení) nebo r/o (správce promítání).

Pohled kontroluje, jestli uživatel v poli e-mail zadal hodnotu obsahující zavináč. Pohled také zobrazuje v horní části obrazovky upozornění aplikace. Jestliže administrátor úspěšně vložil nového uživatele, zobrazí se zelené upozornění s textem *Uživatel byl úspěšně přidán!*. Pokud by došlo k nějaké chybě při vkládání uživatele do databáze zobrazilo by se červené oznámení *Uživatele se bohužel nepodařilo přidat. Zkuste to, prosím, znovu.* Pokud by administrátor vkládal nového uživatele, vyplnil jméno, ale zapomněl by na některou z dalších položek, zobrazilo by se opět červené upozornění, tentokrát s nápisem *Pole nesmí být prázdná!*. Pokud by vyplnil všechna pole, ale e-mail by již patřil některému z existujících uživatelů, zobrazilo by se rovněž červené upozornění s textem *Tento e-mail již existuje.*



Obrázek 6.5.4 Správa uživatelů

6.6 Controller

Nejjednodušší směrování v Laravelu obsahuje URI a uzávěr, který obsahuje metodu definující chování a směrování aplikace. Pokud bychom ovšem chtěli použít komplikovanější metody, kód by stal velmi nepřehledným. Komplikovala by se i možnost jeho znovu použitelnosti (určitě už by to nebylo tak jednoduché jako použít již vytvořenou třídu) a nedodržovali bychom MVC architekturu. Proto tyto metody soustředíme do tematicky podobných tříd, které se budou lépe spravovat, budou přehlednější pro ostatní programátory, kteří by s kódem přišli do styku a velmi nám to usnadňuje použití jednotlivých třídy i pro nadcházející projekty. Defaultně jsou kontrolery umístěny ve složce *app\Http\Controllers*. Všechny kontrolery v Laravelu rozšiřují třídu

Laravelu `App\HTTP\Controllers\Controller`. Směrování aplikace se nachází v souboru *resources/web.php*.

6.6.1 Autentizace

Laravel také poskytuje kontrolery pro autentizaci, které se nacházejí ve složce *app\Http\Controllers\Auth*. Obsahuje pět různých kontrolerů jmenovitě `ForgotPaswordController.php`, `LoginController.php`, `RegisterController.php`, `ResetPaswordController.php` a `VerificationController.php`. Autentizace uživatelů je obvykle velmi klíčová a ani v případě této aplikace tomu není jinak.

Nepřihlášený uživatel smí vidět pouze smyčku promítání. Přihlášený uživatel si pak podle role smí zobrazit seznam oznámení, přidávat je či odebírat ze smyčky, v případě rozsáhlejších práv je i mazat a přidávat či dokonce spravovat uživatele.

Příkazem **php artisan make:auth** se vytvoří pohledy odpovídající jednotlivým kontrolerům. Všechny tyto pohledy ve výchozím nastavení využívají vybranou šablonu Bootstrapu, ale dá se změnit a vytvořit jiný vzhled podle potřeby. Také se nastaví všechno potřebné směrování a vytvoří se HomeController.php, který zobrazí domovskou stránku po přihlášení, jejíž odpovídající pohled se nachází v souboru *resources\views\home.blade.php* [25].V případě potřeby lze uživatele po přihlášení přesměrovat jinam pomocí globální proměnné `$redirectTo` ve třídě `LoginController`, nastavením jiné URI.

Díky těmto kontrolerům je tedy možné, aby se uživatel přihlásil pomocí e-mailu, odhlásil se a pokud uživatel zapomene heslo, může vyplnit formulář, nové mu bude zasláno na e-mail a může si ho obnovit. Také lze nastavit, aby se místo e-mailu uživatel přihlašoval uživatelským jménem. Toho lze docílit vložením metody **public function username(){return 'username';}** do souboru *LoginController.php*. K práci s databází využívá Laravel ve výchozím nastavení Eloquent ORM. Pro zjištění, jestli je uživatel přihlášený slouží metoda **check()** fasády `Auth`, která v případě že je uživatel přihlášen vrátí true, jinak false. V případě, že uživatel není přihlášen, bude přesměrován na stránku pro přihlášení. Toto chování se dá změnit v souboru *app\Http\Middleware\Authenticate.php* v metodě `redirectTo()`.

6.6.2 ShowController.php

Třída ShowController obsahuje tři metody. Metodu `index()`, která se zavolá při požadavku GET a metodu `getProjectingFiles()`, kterou volá metoda

`index()` a `FilesController::index()` a metodu `getProjectingFilesArray()`. Třída slouží k získání dat, která se budou promítat ve smyčce a jejich předání odpovídajícímu pohledu (*show*).

Metoda `getProjectingFiles()` nejprve získá všechny režimy z tabulky *modes* pomocí modelu *Mode*. Zkontroluje, jestli má některý z nich nastavenou hodnotu ve sloupci *active* na `Y`. Pokud ano, do proměnné $active_files se načtou oznámení, která jsou přiřazená k tomuto modelu, jinak se zavolá metoda `getActiveFiles()` modelu `File`.

Vytvoří se proměnné `$file_paths, $active_lengths a $file_types`. Následuje for cyklus, kdy se prochází všechna aktivní oznámení. Nejprve se zkontroluje, jestli jsou oba datumy, od kdy a do kdy se má oznámení promítat, zadány. V případě, že jsou, se zkontroluje, jestli je dnešní den v tomto rozmezí a pokud by nebyl, oznámení by se přeskočilo. Jestliže nejsou zadány oba datumy, zkontroluje se každý zvlášť. Postup je analogický, jestliže jsou nastaveny a dnešní datum je mimo toto rozmezí, oznámení se přeskočí. Oznámením, která jsou v rozmezí těchto datumů (oznámení se má promítat), nebo žádné datum není nastavené, tudíž se má oznámení promítat vždy, se přiřadí cesta k oznámení do proměnné `$file_paths` na základě typu oznámení. Pro URL se nastaví jako cesta k oznámení název oznámení, kde je celá URL uložená. Pro ostatní soubory se nastaví cesta ke složce *files* na disku, kde jsou soubory uložené a za ní se připojí název souboru. Pro PDF dokumenty se na konec řetězce připojí ještě nastavení pro upravení jejich zobrazení v prohlížeči. Následně se uloží do proměnné `$active_lengths` délka zobrazení souboru a do `$file_types` typ souboru. Metoda vrátí pole s výše zmíněnými proměnnými.

Metoda `index()` zavolá `getProjectingFiles()` a pole, které vrátí, uloží do tří proměnných o stejném názvu jako v zavolané metodě, tedy `$file_paths, $active_lengths a $file_types`. Vrací pohled *show*, kterému tyto proměnné předá.

Metoda `getProjectingFilesArray()` je podobná metodě `getProjectingFiles()`, ale místo pole polí vrací pole všech oznámení, která se mají promítat.

6.6.3 FilesControler.php

Třída FilesController obsluhuje všechny požadavky GET a POST pro pohledy ve složce *files*.

index()

Metoda se zavolá při zadání URL *https://název-serveru/files*. Zavolá metodu *ShowControlleru* `getProjectingFiles()`. Do jednotlivých proměnných vloží prvky pole, které následně předá pohledu *files.index*.

insert()

Metoda `insert()` zpracovává GET požadavek *https://název-serveru/files/insert*. Metoda pouze vrátí pohled *files.insert* s proměnnými pro tento pohled.

files()

Metoda se zavolá při zadání URL *https://název-serveru/files/manage*. Nejprve načte z relace (session) hodnotu klíče *filter* a zavolá metodu této třídy `getFilteredFiles()`, která vrátí filtrovaná oznámení stránkovaná podle nastaveného počtu položek. Následně zjistí, zda se promítá některý z režimů nebo aktivní oznámení a tuto informaci uloží do proměnné $projection_mode. Ještě do proměnné *$projecting_files* uloží oznámení, která se mají promítat (pro úpravu pořadí) a vrátí pohled *files.files*.

post_insert_file()

Metoda zpracovává POST požadavek URL adresy *https://název-serveru/-files/insert*.

Pokud se uživatel snaží nahrát soubor na server, zkontroluje se, jestli nějaký vybral. Pokud ano, načte se jeho jméno, uloží se na disk a zjistí se jeho přípona (typ souboru). Poté metoda ověřuje, zda se jedná o video (přípona .mp4 nebo .avi) a zjistí délku videa v sekundách, nebo nastaví hodnotu na NULL. Tato metadata se pokusí vložit do databáze zavoláním metody třídy `File::insertFile()` s parametry jméno souboru, jeho velikost na disku, typ souboru, cesta k souboru na disku a délka videa. Jestliže by došlo k chybě, smazal by se soubor z disku a vypsalo by se uživateli hlášení, že se soubor nepodařilo načíst. V případě, že by uživatel nevybral žádný soubor k nahrání a kliknul na tlačítko *Nahrát soubor*, které by odpovídalo této akci, by se vypsalo upozornění na tuto skutečnost s textem *Nebyl vybrán žádný soubor!*. Kód je vidět na obrázku 6.6.3 a).

```
if ($request->input()['action'] == "fileupload") {
    $insert_file_ok = true;

    if ($request->hasFile('fileupload')) {
        $name = $request->file('fileupload')->getClientOriginalName();

        $path = $request->file('fileupload')->storeAs(
            'files', $name
        );

        $extension = pathinfo($path, PATHINFO_EXTENSION);

        if ((!strcmp($extension, 'mp4')) || (!strcmp($extension, 'avi'))) {
            $getId3 = new getId3;
            $video_length = $getId3->analyze(config('app.path_to_project') .
                "/storage/app/" . $path);
            if (array_key_exists('playtime_seconds', $video_length)) {
                $video_length = (int)$video_length['playtime_seconds'];
            } else {
                $video_length_error = true;
            }
            print_r($video_length);
        } else {
            $video_length = NULL;
        }


        $ret = File::insertFile($name, Storage::size($path), $extension, $path, $video_length);

        if (!$ret) {
            Storage::delete($path);
            $insert_file_ok = false;
        }

    } else {
        $insert_file_file_exists = false;
        $insert_file_ok = NULL;
    }
}
```

Obrázek 6.6.3 a) Kód kontroleru pro načtení obrázku

V případě, že chce uživatel přidat URL, pouze se zkontroluje, jestli není pole prázdné (v opačném případě se pouze uživateli vypíše upozornění) a zavolá metodu `File::insertFile()` s parametry URL adresa, velikostí souboru nula, protože se nejedná o soubor, který by na disku nějaké místo zabíral, typ souboru 'url' a jako cesta k souboru na disku se opět nastaví dané URL.

Metoda vrátí pohled *files.insert*, kterému předá pouze proměnné obsahující případné chybové stavy.

post_files_manage()

Metoda zpracovává POST požadavek URL adresy *https://název-serveru/-files/manage*. Nejprve získá ze relace(session), jaký je nastavený filtr, do proměnné `$post` načte požadavek a zjišťuje, k jaké došlo akci.

V případě, že chce uživatel uložit provedené změny ohledně oznámení, se nejdříve opět zavolá metoda `getFilteredFiles()`, aby se našla mno-

žina oznámení, jejichž nastavení uživatel upravoval. Potom se najde stránka tabulky, na které se uživatel nachází a její poslední položku. Těchto informací využije pro for cyklus, který běží od první položky zobrazené stránky do poslední položky stránky. Postupně se kontroluje, jestli je hodnota 'aktivní' nastavená na stejnou hodnotu ve formuláři jako v databázi. Nejprve se vytvoří proměnná `$value`, která se nastaví na prázdný řetězec. Pokud je nastavena hodnota *activeFileID* v požadavku, znamená to, že je ve formuláři zaškrtnutý atribut aktivní. Ověří se, jestli je tato hodnota 'Y' nastavena i v databázi. V případě, že by nebyla, by se hodnota v proměnné *$value* nastavila na 'Y'. Pokud není zadán požadavek ve formuláři na atribut aktivní u dané položky, zkontroluje se, jestli je hodnota v databázi nastavena na neaktivní, tedy 'N' a jestliže není, nastaví se hodnota *$value* na 'N'. Poté se zkontroluje hodnota proměnné *$value* a pokud neobsahuje prázdný řetězec, nastaví se proměnná *$length* buď na výchozí hodnotu délky promítání, pokud je *$value* 'Y', nebo na NULL v případě, že by *$value* byla 'N'. Nakonec se zavolá metoda třídy `File::updateActive()`, která oznámením s odpovídajícím identifikátorem nastaví sloupečky *active* a *active_length* na hodnoty `$value` a *$length*.

Po atributu aktivní se kontroluje atribut délka promítání. V případě, že tato hodnota není stejná s hodnotou v databázi, zavolá se metoda `updateActiveLength()` opět modelu `File`, která nastaví délku promítání na novou hodnotu. Jako poslední se zkontrolují data od kdy a do kdy se mají oznámení zobrazovat. V případě, že se hodnota nastavená uživatelem liší od té v databázi v sloupečku *date_from*, respektive *date_to*, zavolá se příslušná metoda modelu a hodnota se aktualizuje. Pro datum zobrazení od se zavolá metoda modelu `updateDateFrom()`, pro datum zobrazení do se zavolá metoda `updateDateTo()`.

Další akcí, kterou může uživatel provést, je změna filtru. Filtr je uložený jako globální proměnná v této třídě. Dojde tedy k porovnání hodnot a pokud nejsou stejné, přiřadí se do proměnné nová hodnota.

Poslední věc, kterou může uživatel s oznámeními udělat, je smazat je. V takovém případě se načtou všechna oznámení z databáze a postupně pomocí cyklu foreach se prochází jednotlivá oznámení. Jestliže se jejich identifikátor shoduje s hodnotou v `$post['delete']`, pak se odstraní z databáze, a pokud se nejedná o URL, tak i z disku. Pokud by se oznámení nepodařilo smazat, zobrazí se nad tabulkou upozornění *Oznámení se nepodařilo smazat.*

Pokud by chtěl uživatel změnit pořadí promítání, dojde ke smazání všech záznamů z dabáze (z tabulky *projection_order* a následně se všechna oznámení vloží zpátky ve správném pořadí.

Poslední akcí je výběr promítání. Uživatel může volit mezi promítáním

jednotlivých režimů a aktivních oznámení. Pokud uživatel vybere režim, nastaví se v databázi jeho hodnota aktivní na 'Y'.

Metoda vrací pohled *files.files*, kterému předá mimo jiné filtrovaná oznámení a filtr.

modes()

Metoda `insert()` zpracovává GET požadavek *https://název-serveru/files/-modes*. Pouze vrátí pohled *files.modes.index*, kterému předá všechny existující režimy a proměnné pro chybové stavy nastavené na NULL.

post_modes()

Metoda zpracovává POST požadavek URL adresy *https://název-serveru/-files/modes*. Pokud uživatel přidává nový režim, zkontroluje se, jestli pole se jménem není prázdné a následně se ověří, jestli režim s tímto jménem již neexistuje. V případě chyby se nastaví příslušná proměnná na chybovou hodnotu, jinak se nový režim vloží do databáze. Metoda vrací pohled *files.modes.index*.

modes_manage()

Metoda `insert()` zpracovává GET požadavek *https://název-serveru/files/-modes/id*. Nejprve tedy zkontroluje, že režim s identifikátorem předaným jako parametr v URL existuje. Pro existující režim načte aktivní oznámení režimu, všechna oznámení a sloučí je do jednoho pole tím, že změní hodnoty *active* a *active_length* u pole se všemi oznámeními a přiřadí jim hodnoty aktivních oznámení režimu. Toto pole spolu s ostatními proměnnými předá pohledu *files.modes.modes*.

post_modes_manage()

Metoda zpracovává požadavek POST URL adresy *https://název-serveru/-files/modes/id*. Zkontroluje, jestli existuje režim s tímto id a následně kontroluje, k jakým došlo akcím.

Pokud chce uživatel přidat nebo odebrat aktivní oznámení modelu, projdou se pomocí for cyklu všechna oznámení aplikace. Pokud jej uživatel ve formuláři označil jako aktivní, zkontroluje se, jestli je oznámení přiřazeno tomuto modelu i v databázi a pokud není, načte se i hodnota pro délku promítání a vloží se záznam do databáze. V případě, že se záznam nachází v databázi, se pouze porovnají hodnoty délky promítání a v případě nerovnosti se aktualizuje hodnota v databázi. Jestliže byla položka ve formuláři

označena za neaktivní, zjistí, jestli nebyla přiřazena modelu v databázi v minulosti (uživatel oznámení "zneaktivnil") a pokud model záznam najde, smaže ho z databáze.

Užitel může režim také smazat a v takovém případě se po úspěšném pokusu o smazání aplikace přesměruje na URL *https://název-serveru/files/-modes*. Metoda vrací pohled *files.modes.modes*.

6.6.4 UsersAdminController.php

Tento kontroler slouží pro správu uživatelů podle požadavků v kapitole 2.5. Obsahuje pouze dvě metody – `index()` a `post(Request $request)`.

Metoda `index()` je zavolána při požadavku GET URL adresy *https://-název-serveru/usersadmin* a pouze vrací pohled se všemi potřebnými proměnnými (pole všech uživatelů a NULL nebo false hodnoty pro upozornění).

Metoda `post()` je zavolána při požadavku POST. Vytvoří se instance třídy *User* a proměnné, které slouží k tomu, aby se zobrazila upozornění, se nastaví na výchozí hodnotu, stejně jako v metodě `index()` (upozornění zůstanou skryta).

Dále se ověří, k jakým došlo změnám. Prochází se postupně pomocí cyklu foreach() všichni uživatelé. Zkontroluje se, jestli role uživatele odeslaná ve formuláři, se shoduje s hodnotou v databázi. Pokud se liší, zavolá se metoda modelu *User* `updateRole()`, která ji změní na novou hodnotu, nastavenou administrátorem.

Dále se zkontroluje, jestli administrátor nezadal nové heslo pro některého z uživatelů. Pokud ano, zavolá se metoda modelu *User* `changePassword()`, která heslo změní.

Následně se zkontroluje, jestli administrátor nezadal údaje pro nového uživatele. Pokud zadal všechny údaje, zkontroluje se, jestli je e-mail jedinečný v rámci aplikace a pokud je, přidá se nový uživatel zavoláním metody *User* insertUser(). V případě, že uživatel vyplní jméno, ale ne e-mail nebo heslo, nastaví se hodnota proměnné `$insert_user_empty_input` na hodnotu true, která se předá pohledu, což zajistí, že se zobrazí uživateli upozornění v horní části obrazovky.

Jako poslední se zkontroluje, jestli administrátor nechtěl některého uživatele smazat. V takovém případě by se zavolala metoda *User* `deleteUsers()`.

Následně se vrátí pohled a jsou mu předány proměnné, které zajišťují zobrazení, nebo naopak skrytí upozornění a pole se všemi uživateli.

6.7 Docker

Docker je otevřený software pro běh aplikací v izolovaném prostředí (kontejnerech). Používá se obvykle místo virtualizace, která má podstatně víc režie. Díky tomu, že se jedná o prostředí izolované od prostředí hostujícího počítače, je možné aplikaci využívající Docker, spustit kdekoliv, kde je Docker nainstalovaný [12].

Tato aplikace se skládá z pěti kontejnerů. Kontejner *app-data* slouží pouze k předání dat aplikace ostatním kontejnerům. Kvůli tomu se po spuštění všech kontejnerů vypne.

Pro server, na kterém následně běží tato aplikace slouží dva kontejnery - *nginx* a *php*. Oba přebírají data z kontejneru *app data* a závisí na kontejneru databáze. *Nginx* běží na portu 80, *php* na portu 9000. V kontejneru *nginx* se nastavují údaje pro přihlášení k databázi.

Databáze běží v kontejneru *mysql* na portu 3310 u hostujícího počítače, 3306 v rámci Dockeru. Načítá se do ní skript ze složky *cesta-ke-korenovemu-adresari-serveru/database*. Nastavují se zde přístupové údaje k databázi. Poslední kontejner je *phpmyadmin*, který běží na portu 8081 (host) a mapuje se na port 80. Kontejnery běží na síti *sail*.

6.8 Promítání

Pomocí třídy `File` se ve třídě ShowController načtou všechna aktivní oznámení (buď výchozí aktivní oznámení nebo aktivní oznámení vybraného režimu) z databáze do proměnné `$active_files`. Poté se všechny tyto položky projdou a kontroluje se, jestli je nastavené datum od kdy a do kdy se má oznámení promítat. Jestliže ano, zkontroluje se, jestli do toho rozsahu patří i dnešní den. Jestli ne, toto oznámení se přeskočí. Pro oznámení, která projdou tímto sítem, se zjistí, jakého je typu a uloží se do proměnné `$file_types`. Podle typu oznámení se uloží cesta k souboru na disku/celé URL do proměnné `$file_paths` a nakonec se uloží délka zobrazení do proměnné `$active_lengths`.

V souboru `show.blade.php` se tyto proměnné překopírují do JavaScriptových proměnných se stejným názvem, jen s jmennou konvencí pro JavaScript (filePaths, activeLengths a fileTypes). Protože mi nešlo použít funkci *json_encode()*, která se pro tento účel obvykle použije, každá proměnná má svůj php foreach cyklus, ve kterém se postupně data překopírují. Potom, pokud je počet aktivních souborů větší než 0, se zavolá asynchronní funkce `showContent()`. V této funkci se nachází nekonečný while cyklus, ve kterém je zanořený for cyklus od nuly do počtu aktivních souborů. Ve for cyklu se

nejprve zavolá funkce `changeContent()`, ve které se podle typu oznámení rozhodne, jestli se bude promítat pomocí HTML značky *img* (pro obrázky), *video* (pro videa) nebo *iframe* (pro dokumenty a webové stránky). Poté se pomocí asynchronní funkce `sleep()` aplikace zastaví na dobu, po kterou se má oznámení promítat. Ukázka promítání je zobrazena na obrázku 6.7.

Obrázek 6.7 Promítání

7 Testování

Výsledná aplikace byla testována pomocí scénářů v prohlížečích Google Chrome verze 101.0.4951.41, Opera verze 85.0.4341.75, Mozilla Firefox verze 100.0 a Microsoft Edge verze 100.0.1185.39. Nasazení systému (pomocí Dockeru) bylo testováno na třech různých systémech. Jmenovitě Windows 10 Pro verze 21H1, Windows 11 Home verze 10.0.22000 a Ubuntu verze 20.04.4. V rámci Windows jsem zkoušela pouštět aplikaci z úložiště *C:* a z úložiště *\\wsl$*. Došlo k velkému rozdílu rychlostí. Zatímco spuštění z *\\wsl$* mělo normální odezvu (načítání webových stránek během maximálně jedné, dvou vteřin, obvykle kratší), při spuštění z disku *C:* byla odezva mnohonásobně delší (až 20 sekund). Doporučuji tedy aplikaci spouštět z úložiště *\\wsl$*. Avšak kromě rychlosti odezvy aplikace funguje podle očekávání v obou případech. Většina metod modelu aplikace byla testován automatickými testy, které jsou uloženy ve složce *tests/Feature*.

7.1 Přihlášení

Popis scénáře

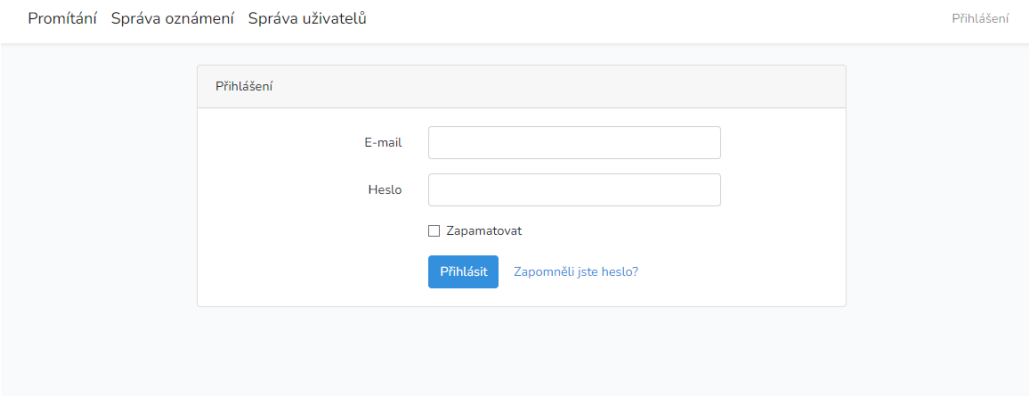
Pro přihlášení do prohlížeče zadejte adresu https://nazev-serveru/login. Pro nepřihlášeného uživatele se zobrazí obrázek 7.1. Pro úspěšné přihlášení do pole „E-mail“ zadejte e-mailovou adresu a do pole „Heslo“ heslo k tomuto účtu a klikněte na tlačítko přihlásit. Po přihlášení se Vám zobrazí stránka se správou souborů, která je vidět na obrázku 7.5.

Zadejte neplatnou e-mailovou adresu a stiskněte tlačítko *Přihlásit*. Políčko s neplatným e-mailem se ohraničí červeně a vypíše se pod ním červeným písmem *Zadané údaje nesouhlasí*.

Zadejte libovolný e-mail a špatné heslo a stiskněte tlačítko *Přihlásit*. Opět se políčko s neplatným e-mailem se ohraničí červeně a vypíše se pod ním červeným písmem *Zadané údaje nesouhlasí*.

Výsledek scénáře

Ve všech prohlížečích došlo k očekávanému průběhu i výsledku přesně podle scénáře.



Obrázek 7.1 Přihlášení

7.2 Vložit soubor

Popis scénáře

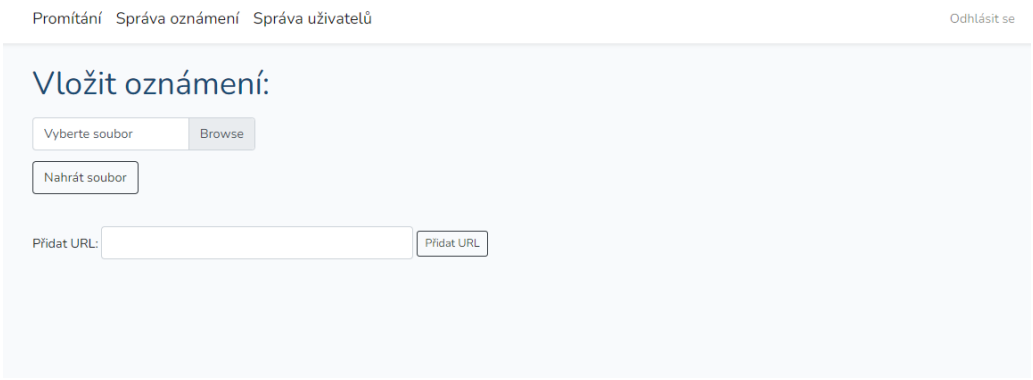
Do prohlížeče zadejte webovou adresu https://nazev-serveru/files/insert. Pro přihlášeného uživatele se načte pohled 7.2 a). Klikněte na tlačítko

Pokud nejste přihlášen, klikněte na odkaz *zde* a pokračujte podle scénáře pro přihlášení. Klikněte na tlačítko *Vyberte soubor* nebo *Browse*. Otevře se dialogové okno 7.2 b), ve kterém vyberte ze svého zařízení libovolný soubor do velikosti 2 000 MB, s příponou .avi, .mp4, .jpg, .jpeg, .jfif, .gif, nebo .pdf a klikněte na tlačítko *Otevřít*. Dialogové okno zmizí. Následně klikněte na tlačítko *Nahrát soubor*. Nad nápisem *Vložit oznámení* vyskočí zelené oznámení *Soubor byl úspěšně vložen!*. Přejděte na webovou adresu https://nazev-serveru/files/manage. Pod tabulkou s oznámeními se nachází stránkování. Klikněte na poslední stránku a ověřte, že byl soubor opravdu vložen. Jestliže by se soubor nepodařilo nahrát na disk nebo do databáze, vyskočilo by červené oznámení s nápisem *Soubor se nepodařilo načíst. Zkuste to, prosím, znovu.*

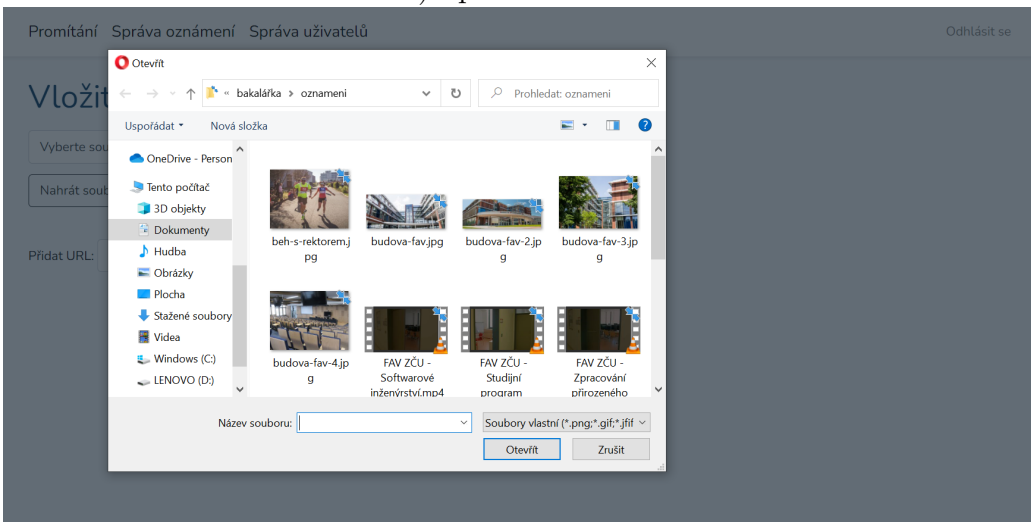
Nechte pole pro vložení souboru prázdné a klikněte na tlačítko *Nahrát soubor*. Vyskočí červené oznámení s textem *Nebyl vybrán žádný soubor.*

Výsledek scénáře

Ve všech prohlížečích došlo k očekávanému průběhu i výsledku. V prohlížeči Opera došlo k malé odchylce, kdy před tím, než vyskočilo obvyklé dialogové okno, se ukázaly poslední tři stažené položky spolu s tlačítkem „Zobrazit všechny soubory“, které dialogové okno otevře (lze samozřejmě i vybrat jeden ze tří zobrazených). Aplikace umožňuje vložení souboru podle požadavku v sekci 2.3.



Obrázek 7.2 a) Správa oznámení - rozcestí



Obrázek 7.2 b) Dialogové okno "Vybrat soubor"

7.3 Vložit URL

Popis scénáře

Do prohlížeče zadejte webovou adresu https://nazev-serveru/files/insert. Pro přihlášeného uživatele se načte pohled 7.2 a).

Pokud nejste přihlášen, klikněte na odkaz *zde* a pokračujte podle scénáře pro přihlášení. Do políčka *Přidat URL* vložte URL, které chcete přidat jako nové oznámení a klikněte na tlačítko *Přidat URL*. Následně vyskočí zelené oznámení s textem *URL bylo úspěšně vloženo*. Přejděte na webovou adresu https://nazev-serveru/files/manage. Pod tabulkou s oznámeními se nachází stránkování. Klikněte na poslední stránku a ověřte, že oznámení bylo opravdu vloženo. V případě, že by se oznámení nepovedlo přidat by vyskočilo červené oznámení s nápisem *URL se nepodařilo načíst, zkuste to, prosím, znovu.*

Nechte pole pro vložení URL prázdné a klikněte na tlačítko *Přidat URL*.

Vyskočí červené oznámení s textem *Pole nemůže být prázdné.*

Výsledek scénáře

Ve všech prohlížečích došlo k očekávanému průběhu i výsledku přesně podle scénáře. Aplikace umožňuje vložení URL podle požadavku v sekci 2.3.

7.4 Vložit uživatele

Popis scénáře

Přihlaste se jako administrátor (podle scénáře 7.1). V horní liště klikněte na nápis *Správa uživatelů*, nebo zadejte adresu https://nazev-serveru/usersadmin. Zobrazí se Vám pohled 7.4. Pod tabulkou s výpisem uživatelů se nachází prázdná pole pro zadání hodnot nového uživatele. Vyplňte tedy jméno, e-mail, heslo a zvolte roli nového uživatele a klikněte na tlačítko pod tabulkou *Uložit změny.* V horní části obrazovky se zobrazí zelené upozornění s textem *Uživatel byl úspěšně přidán* a v tabulce přibyde nová řádka s údaji nově přidaného uživatele.

Můžou nastat také tři neúspěšné scénáře, při kterých se zobrazí v horní části obrazovky červené upozornění. Pokud by se nepodařilo přidat uživatele do databáze, vypsalo by se upozornění *Uživatele se bohužel nepodařilo přidat. Zkuste to, prosím, znovu.*

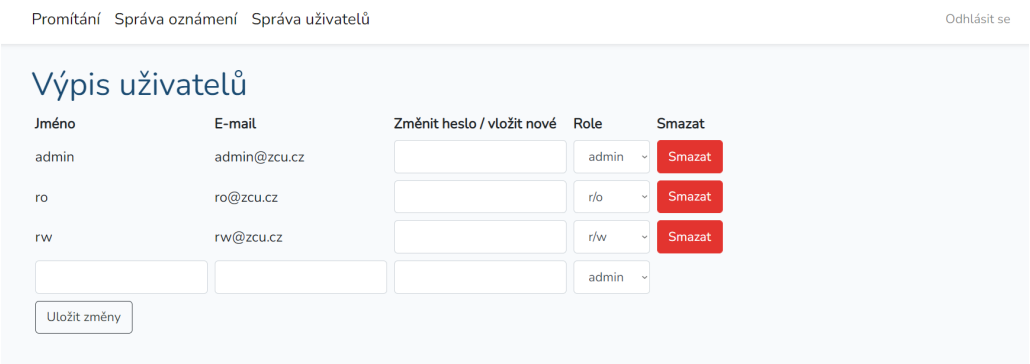
Pro další neúspěšné vložení uživatele, nechte pole heslo prázdné. Po stisknutí tlačítka *Uložit změny*, se zobrazí upozornění *Pole nesmí být prázdná!.*

Třetí neúspěšný scénář: Zadejte do pole e-mail, e-mailovou adresu přiřazenou již existujícímu uživateli. Po stisknutí tlačítka pro uložení se zobrazí upozornění s textem *Tento e-mail již existuje!.*

Do pole e-mail zadejte neplatnou e-mailovou adresu (bez zavináče). Okolo pole pro vložení se zobrazí upozornění, že to není platná e-mailová adresa.

Výsledek scénáře

Ve všech prohlížečích došlo k očekávanému průběhu i výsledku. Aplikace umožňuje vložení nového uživatele do systému.



Obrázek 7.4 Správa uživatelů

7.5 Aktivní oznámení

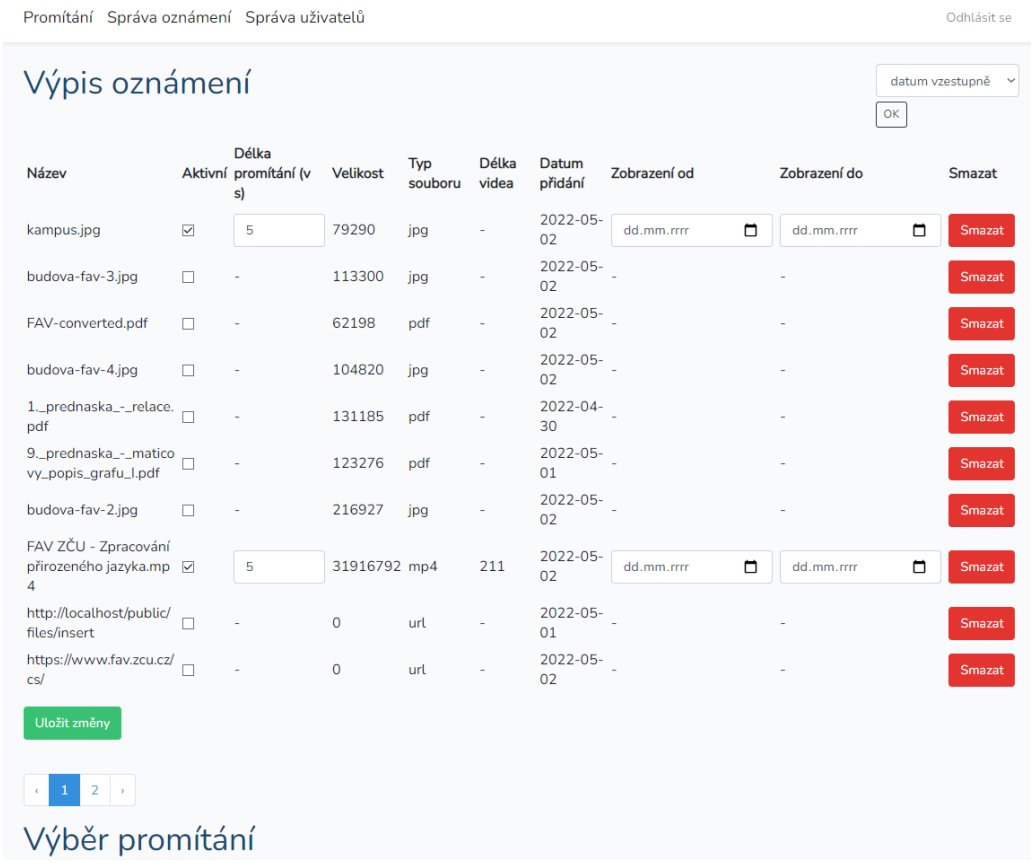
Popis scénáře

Pokud nejste přihlášeni, přihlaste se podle scénáře 7.1 a přejděte na adresu https://název-serveru/files/manage. Zobrazí se Vám pohled 7.5. Zaškrtněte první políčko v sloupci *Aktivní* a potvrďte volbu stisknutím tlačítka *Uložit změny*. Pole zůstane zaškrtnuté. V horní liště klikněte na nápis *Promítání*. Zkontrolujte, že je dané oznámení součástí promítaní.

Vraťte se na adresu https://název-souboru/files/manage, klikněte znovu na první pole v sloupci *Aktivní*. Zaškrtnutí zmizí. Potvrďte volbu kliknutím na tlačítko *Uložit změny* a klikněte na nápis *Promítání*. Zkontrolujte, že se dané oznámení přestalo promítat.

Výsledek scénáře

Ve všech prohlížečích došlo k očekávanému průběhu i výsledku. Oznámení lze zařadit mezi množinu aktivních oznámení, která se mají promítat. Stejně tak lze oznámení odebrat z této množiny a přestat promítat.



Obrázek 7.5 Správa promítání

7.6 Upravit parametry promítání

Popis scénáře

Přejděte na adresu https://název-serveru/files/manage. Pokud nejste přihlášeni, přihlaste se. V sloupci *Délka promítání* změňte hodnotu na jednu sekundu. V horní liště klikněte na nápis *Promítání*. Zkontrolujte, že se dané oznámení promítá jednu sekundu.

Vraťte se zpátky na adresu https://název-serveru/files/manage a nastavte datum ve sloupci *Zobrazení do* na včerejší datum. Přejděte na promítání a ověřte, že se oznámení opět přestalo promítat.

Vraťte se zpátky na adresu https://název-serveru/files/manage a smažte datum ve sloupci *Zobrazení do* a ověřte, že se oznámení znovu začalo promítat. Vraťte se zpátky na adresu https://název-serveru/files/manage a nastavte datum ve sloupci *Zobrazení od* na zítřejší datum. Přejděte na promítání a ověřte, že se oznámení přestalo promítat.

Výsledek scénáře

Ve všech prohlížečích došlo k očekávanému průběhu i výsledku. Doba promítání i datumy promítání uživatelé mohou upravovat podle potřeby.

7.7 Smazat oznámení

Popis scénáře

Přejděte na na adresu https://název-serveru/files/manage. Ve sloupečku *Smazat* klikněte na červené tlačítko s nápisem *Smazat*. Následně vyskočí oznámení *Opravdu chcete oznámení smazat?*. Klikněte na tlačítko *Zrušit* a ověřte, že nedošlo k žádné akci (oznámení se stále zobrazuje v tabulce). Znovu klikněte na tlačítko s nápisem *Smazat*, tentokrát však následně klikněte na tlačítko *OK*. Ověřte, že oznámení zmizelo z tabulky oznámení.

Výsledek scénáře

Ve všech prohlížečích došlo k očekávanému průběhu i výsledku. Aplikace umožňuje oznámení smazat.

7.8 Výsledek testování

Aplikace reagovala podle očekávání kromě drobných odchylek. Za nejvýznamnější odchylku považuji problém s promítáním souborů a url pomocí iframe v Opeře na Windows 10. Nejedná se o trvalý problém, nicméně během testovaní nastal (a opět pominul), proto doporučuji pro promítání (na chodbě) upřednostnit ostatní zmíněné prohlížeče. Video se zvukem se nespustí bez interakce uživatele se stránkou, proto jsem zvolila jako výchozí hodnotu *muted*. Video se tedy vždy pustí, ale bez zvuku. V případě URL je důležité si zkontrolovat, jestli je možné na danou adresu přistoupit (například můžu přistoupit a tedy zobrazit některou ze stránek této aplikace, ale nelze zobrazit stránku *https://www.fav.zcu.cz/cs/*.

8 Závěr

Byla vytvořena webová aplikace za účelem promítání oznámení na chodbách Katedry informatiky a výpočetní techniky, která umožňuje spravovat a promítat oznámení a spravovat uživatele této aplikace.

Analytická část této práce se zabývá webovými technologiemi, způsobem uložení a prezentace multimediálních souborů. Praktická část se zabývá návrhem řešení a jeho implementací a následně testováním.

Analýza byla využita k výslednému řešení, které sestává z třívrstvé webové aplikace (model MVC), jejíž backend je napsán v PHP za použití frameworku Laravel a na frontendu je využit Javascript, HTML, CSS a Bootstrap a jako databázový systém jsem zvolila MySQL. Aplikaci tvoří šest hlavních pohledů pro správu oznámení, správu režimů promítání, správu uživatelů a promítání a tří kontrolerů (pro oznámení a režimy, pro uživatele a pro promítání).

Důležitá byla přenositelnost aplikace na různé systémy, které bylo dosaženo pomocí Dockeru. Aplikace byla testována pomocí automatických testů pro modely a pomocí scénářů na systémech Windows 10, Windows 11 a Ubuntu a na několika různých prohlížečích.

Aplikaci by bylo možné rozšířit například přidáním datumu/rozmezí datumů k režimům promítání nebo dalších parametrů promítání. Dále by bylo vhodné, aby kontejner se serverem nginx využíval místo HTTP protokolu zabezpečený protokol HTTPS, protože dochází k předávání nezašifrovaných dat mezi serverem a klientem.

Seznam zkratek

AES - Advanced Encryption Standard
API - Application Programming Interface
AVI - Audio Video Interleave
AWS - Amazon Web Services
BLOB - Binary Large Object
CSRF - Cross-site Request Forgery
DRY - Don't repeat yourself
GIF - Graphics Interchange Format
HTML - Hypertext Markup Language
HTTP - Hypertext Transfer Protocol
HTTPS - Hypertext Transfer Protocol Secure
IFrame - inline frame
JFIF - JPEG File Interchange Format
JPG/JPEG - Joint Photographic Experts Group
KISS - Keep it simple, silly
MB - megabyte
MIT - Massachusettský technologický institut (Massachusetts Institute of Technology)
MP4 - MPEG Layer-4 Audio (MPEG - Moving Picture Experts Group)
MVC - model-view-controller
ORM - Objektově relační mapování
PDF - Portable Document Format
PDO - PHP Data Objects
PHP - Hypertext Preprocessor
PNG - Portable Network Graphics
SOLID - Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion
SQL - Structured Query Language
TB - terabyte
TS - TypeScript
URL - Uniform Resource Locator

Literatura

[1] *The modern web developer's platform* [online]. Angular. [cit. 2021/6/15]. Dostupné z: `https://angular.io`.

[2] *Amazon Web Services* [online]. Wikipedia. [cit. 2021/1/6]. Dostupné z: `https://en.wikipedia.org/wiki/Amazon_Web_Services`.

[3] Bhadwal, A. *Best JavaScript IDE & Source Code Editors to Use* [online]. hackr.io. [cit. 2020/11/30]. Dostupné z: `https://hackr.io/blog/best-javascript-ide-source-code-editors`.

[4] *Binary large object* [online]. Wikipedia. [cit. 2021/1/5]. Dostupné z: `https://en.wikipedia.org/wiki/Binary_large_object`.

[5] *Build fast, responsive sites with Bootstrap* [online]. Bootstrap. [cit. 2021/6/15]. Dostupné z: `https://getbootstrap.com`.

[6] Brotherton, C. *The Most Popular PHP Frameworks to Use in 2021* [online]. Kinsta blog, 2021. [cit. 2021/5/14]. Dostupné z: `https://kinsta.com/blog/php-frameworks/`.

[7] *CakePHP Built fast, grow solid* [online]. CakePHP. [cit. 2021/5/27]. Dostupné z: `https://cakephp.org`.

[8] Chan, R. *The 10 most popular programming languages, according to the Microsoft-owned GitHub* [online]. Bussiness Insider, 2019/11/9. [cit. 2020/10/29]. Dostupné z: `https://www.businessinsider.com/most-popular-programming-languages-github-2019-11#1-javascript-10`.

[9] *Co je cloudové úložiště?* [online]. Microsoft Azure. [cit. 2021/6/19]. Dostupné z: `https://azure.microsoft.com/cs-cz/overview/what-is-cloud-storage/`.

[10] *Introduction* [online]. getcomposer.org. [cit. 2021/5/14]. Dostupné z: `https://getcomposer.org/doc/00-intro.md`.

[11] *Install Docker Compose* [online]. [cit. 2022/05/30]. Dostupné z: `https://docs.docker.com/compose/install/`.

[12] *Docker overview* [online]. https://docs.docker.com/. [cit. 2022/5/1]. Dostupné z: `https://docs.docker.com/get-started/overview/`.

[13] Dubey, B. K. *Difference between TypeScript and JavaScript* [online]. GeeksforGeeks, 2020/10/21. [cit. 2020/10/28]. Dostupné z: https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/.

[14] *Ecma International* [online]. Wikipedia. [cit. 2020/12/8]. Dostupné z: https://en.wikipedia.org/wiki/Ecma_International.

[15] *ECMAScript* [online]. Wikipedia. [cit. 2020/12/8]. Dostupné z: https://en.wikipedia.org/wiki/ECMAScript.

[16] Elizabeth, J. *Top 5 JavaScript IDEs* [online]. JAXenter. [cit. 2020/11/30]. Dostupné z: https://jaxenter.com/top-5-javascript-ide-146609.html.

[17] Graw, M. *Cloud storage vs external hard disk drive: Which one is better?* [online]. tom's guide, 2020/6/24. [cit. 2021/6/19]. Dostupné z: https://www.tomsguide.com/best-picks/cloud-storage-vs-external-hard-disk-drive-which-one-is-better.

[18] Jain, R. *Top 9 PHP Frameworks For Web Development In 2021* [online]. LambdaTest, 2021/3/4. [cit. 2021/6/3]. Dostupné z: https://www.lambdatest.com/blog/9-of-the-best-php-frameworks-for-web-development-2021/.

[19] *JavaScript Engine* [online]. Wikipedia. [cit. 2020/11/29]. Dostupné z: https://cs.xcv.wiki/wiki/JavaScript_engine.

[20] *JavaScript - Syntax* [online]. Tutorialspoint. [cit. 2020/11/27]. Dostupné z: https://www.tutorialspoint.com/javascript/javascript_syntax.htm.

[21] *What is Kiosk Mode?* [online]. KioWare. [cit. 2021/6/8]. Dostupné z: https://m.kioware.com/learn/what-is-kiosk-mode.

[22] Klimčík, J. *Co je Laravel?* [online]. laravelblog.cz. [cit. 2021/5/14]. Dostupné z: https://laravelblog.cz/co-je-laravel.

[23] Koďousková, B. *VUE JS: VÝHODY, NEVÝHODY A MOŽNOSTI VYUŽITÍ* [online]. Rascasone, 2021. [cit. 2020/11/19]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-framework-vuejs.

[24] *Introduction* [online]. Laminas. [cit. 2021/6/4]. Dostupné z: https://docs.laminas.dev/laminas-mvc/intro/.

[25] *Autentication* [online]. Laravel. [cit. 2021/6/5]. Dostupné z: https://laravel.com/docs/8.x/authentication.

[26] *Directory Structure* [online]. https://laravel.com/. [cit. 2022/5/2]. Dostupné z: https://laravel.com/docs/8.x/structure.

[27] *Eloquent ORM* [online]. laravel.com. [cit. 2021/5/23]. Dostupné z: https://laravel.com/docs/8.x/eloquent.

[28] *About Node.js* [online]. nodejs.org. [cit. 2020/12/1]. Dostupné z: https://nodejs.org/en/about/.

[29] *Objektově relační mapování* [online]. wikipedia.org. [cit. 2021/5/23]. Dostupné z: https://cs.wikipedia.org/wiki/Objektově_relační_mapování.

[30] Pavelka, J. *Kdy udělat jednostránkový web a jak ho naoptimalizovat pro vyhledávače* [online]. pavelkajan.cz, 2014/10/27. [cit. 2020/11/29]. Dostupné z: http://pavelkajan.cz/kdy-udelat-jednostrankovy-web-a-jak-ho-naoptimalizovat-pro-vyhledavace/.

[31] Pedamkar, P. *TypeScript vs JavaScript* [online]. eduCBA. [cit. 2020/11/5]. Dostupné z: https://www.educba.com/typescript-vs-javascript/.

[32] *What is PHP? Write your first PHP Program* [online]. Guru99. [cit. 2021/6/4]. Dostupné z: https://www.guru99.com/what-is-php-first-php-program.html.

[33] *React* [online]. ReactJS. [cit. 2021/6/15]. Dostupné z: https://reactjs.org.

[34] Salvet, P. *Proměnné v JavaScriptu* [online]. interval.cz, 2017/4/6. [cit. 2020/11/16]. Dostupné z: https://www.interval.cz/clanky/promenne-v-javascriptu/.

[35] *Storage Area Network* [online]. Wikipedia. [cit. 2020/12/29]. Dostupné z: https://cs.wikipedia.org/wiki/Storage_Area_Network.

[36] Shankar, R. *Typescript vs Javascript* [online]. hackr.io, 2020/8/7. [cit. 2020/12/4]. Dostupné z: https://hackr.io/blog/typescript-vs-javascript.

[37] *Typescript vs JavaScript: What's the Difference?* [online]. Guru99. [cit. 2020/11/20]. Dostupné z: https://www.guru99.com/typescript-vs-javascript.html.

[38] Villanueva, J. C. *Amazon S3 vs Local Storage - Where Should You Store Files Uploaded to Your File Transfer Server?* [online]. JSCAPE, 2017/12/17. [cit. 2021/1/6]. Dostupné z: https://www.jscape.com/blog/amazon-s3-vs-local-storage-where-should-you-store-files-uploaded-to-your-file-tra

[39] *The Progressive JavaScript Framework* [online]. VueJS. [cit. 2021/6/15]. Dostupné z: `https://vuejs.org`.

[40] VÍTEK, R. *Symfony po krůčkách - Twig* [online]. zdroják.cz, 2016. [cit. 2021/5/13]. Dostupné z: `https://zdrojak.cz/clanky/symfony-po-kruckach-twig/`.

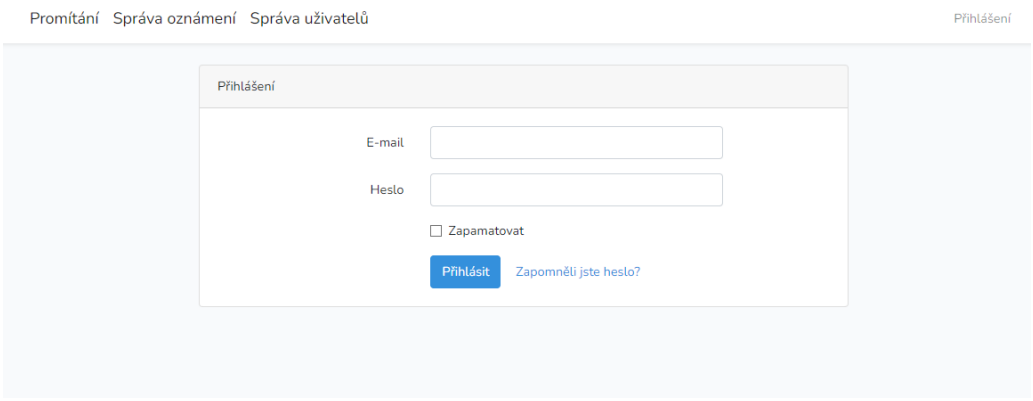
[41] YUSOV, K. *Top 15 JavaScript IDEs and JS Editors for Frontend Development 2021* [online]. Jelvix. [cit. 2020/11/30]. Dostupné z: `https://jelvix.com/blog/best-javascript-ides`.

A Uživatelská příručka

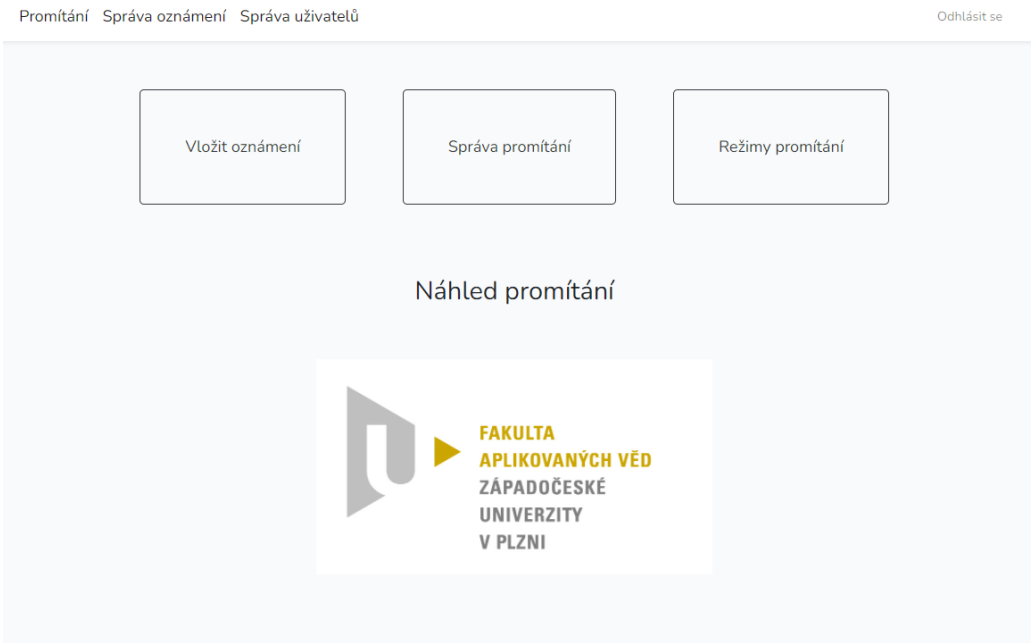
A.1 Přihlášení

Přejděte na webovou adresu https://nazev-serveru/login. Zobrazí se Vám webová stránka zobrazená na obrázku A.1 a).

Pro přihlášení zadejte e-mailovou adresu a heslo k účtu a stiskněte tlačítko *Přihlásit se*. Po úspěšném přihlášení se zobrazí obrazovka A.1 b).



Obrázek A.1 a) Přihlášení

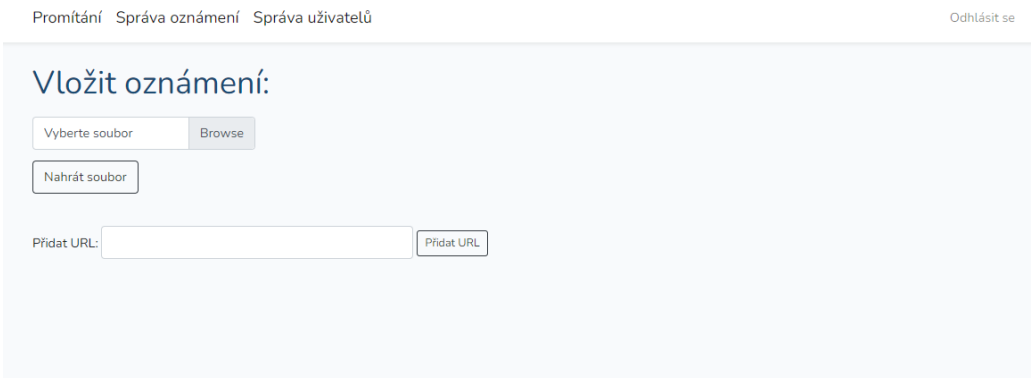


Obrázek A.1 b) Správa promítání - rozcestí

A.2 Vložení souboru

Přejděte na webovou adresu https://nazev-serveru/files/insert. Pro přihlášeného uživatele se zobrazí pohled A.2.

Klikněte na tlačítko *Browse* nebo *Vyberte soubor*. Zobrazí se vám dialogové okno s průzkumníkem souborů. Vyberte oznámení které chcete vložit a klikněte na tlačítko *Otevřít*. Okno se zavře a stiskněte tlačítko *Nahrát soubor*. Pokud se soubor podaří vložit vyskočí v horní části obrazovky zelené upozornění *Soubor byl úspěšně vložen!*.

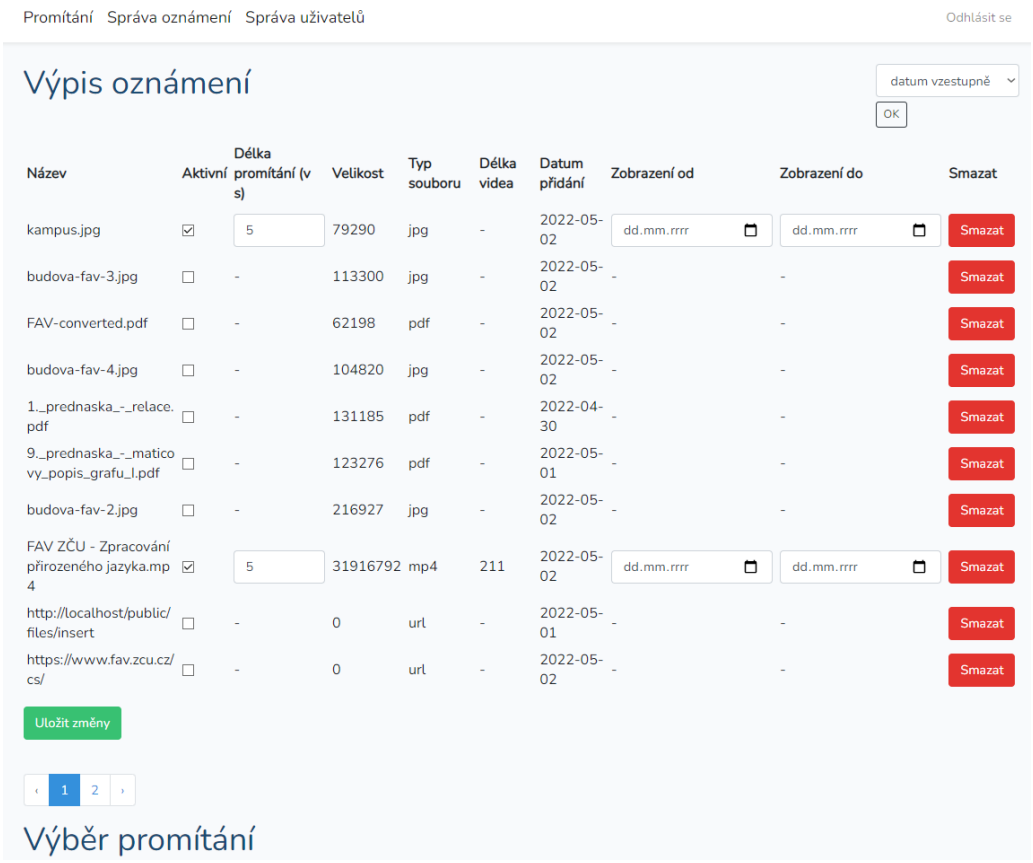


Obrázek A.2 Správa oznámení - vložení oznámení

A.3 Zařadit soubor mezi promítané

Přejděte na webovou adresu https://nazev-serveru/files/manage. Pro přihlášeného uživatele se zobrazí pohled A.3 (pro role 'admin' a 'r/w', pro uživatele s rolí 'r/o' se nezobrazí červená tlačítka pro smazání).

V sekci *Výpis oznámení* se nachází tabulka s oznámeními. Druhý sloupec tabulky obsahuje žaškrtávací políčka. Pokud chcete oznámení přidat mezi promítané (nebo-li aktivní), zašktněte toto políčko a potvrďte stisknutím tlačítka pod tabulkou *Uložit změny*. Po uložení změn se v řádku nově aktivního oznámení zobrazí pole pro zadání délky promítání oznámení (v sekundách) a pole pro zadání dvou datumů. První datum určuje od kdy se bude oznámení promítat, druhé datum naopak do kdy. Upravte tedy tyto parametry podle potřeby, opět uložte změny a pokud jsou parametry promítání nastaveny tak, že by se teď oznámení mělo promítat, můžete si to ověřit kliknutím na nápis *Promítání* v horní liště.

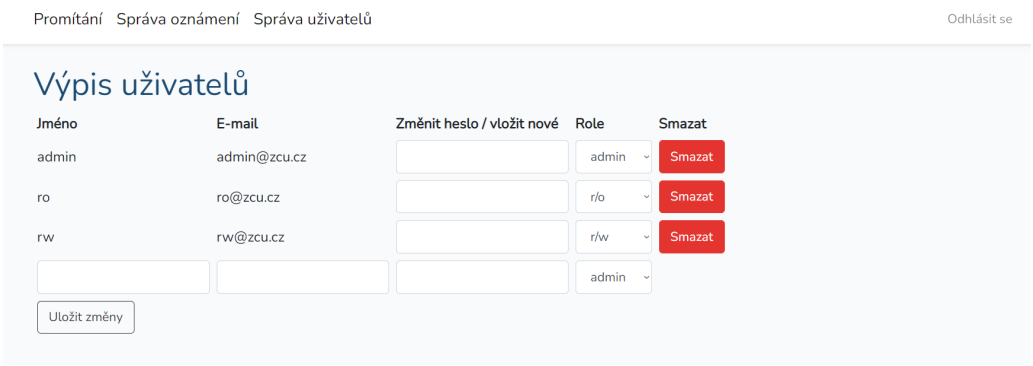


Obrázek A.3 Správa oznámení - správa promítání

A.4 Přidání nového uživatele

Tato akce je možná pouze pro administrátory aplikace. Přejděte na webovou adresu https://nazev-serveru/usersadmin. Pro přihlášeného uživatele se zobrazí pohled A.4.

Vyplňte prázdná pole v posledním řádku tabulky jménem, unikátním e-mailem pro daného uživatele, heslem a vyberte roli uživatele v systému (admin - administrátor, r/w - správce oznámení, r/o - správce promítání) a klikněte na tlačítko *Uložit změny*.



Obrázek A.4 Správa uživatelů

A.5 Promítání

Pro zobrazení promítaných oznámení přejděte na adresu https://nazev-serveru/ nebo https://nazev-serveru/show. Zobrazí se Vám přes celou obrazovku právě promítané oznámení na velmi tmavě šedém pozadí.

A.6 Odhlášení

Na kterékoliv stránce aplikace kromě promítání, se v horní části aplikace nachází lišta s odkazy na jednotlivé stránky na levé straně a nápisem *Odhlásit se* na pravé. Klikněte na něj a aplikace Vás odhlásí a přesměruje na promítání aktivních oznámení.

B Obsah přiloženého CD

CD obsahuje dva adresáře **program** a **dokumentace**.

Adresář **dokumentace** obsahuje dokumentaci bakalářské práce.

Adresář **program** obsahuje soubor *docker-compose.yml* a tři podadresáře - nginx, php a bakalarka. `Nginx` obsahuje Dockerfile pro kontejner nginx-container a soubor *default.conf* s nastavením serveru. `Php` obsahuje pouze Dockerfile pro kontejner php-container. `Bakalarka` obsahuje zdrojové kódy vlastní aplikace (struktura aplikace je popsaná v sekci 6.3), Laravelu a SQL soubor pro vytvoření databáze, který se nachází v podadresáři `database` s názvem *bakalarka-23-6-2021.sql*.

Pro spuštění aplikace je třeba mít nainstalovaný Docker verze 20.10.13 a vyšší a Docker Compose verze 3.3, který je na Windows součástí Docker Desktop [11]. Na Windows doporučuji kvůli rychlosti odezvy překopírovat obsah adresáře **program** do \\wsl$ (například \\wsl$\docker-desktop-data\mnt\wsl), nicméně aplikaci lze spustit bez ohledu na použité úložiště. K přístupu do úložiště \\wsl$ použijte PowerShell, do kterého zadejte příkaz 'wsl'. Pomocí příkazu 'cd cesta-ke-slozce' pak změňte aktuální cestu do adresáře nad adresářem **program**. Následně změňte v souboru *program/bakalarka/config/app.php* hodnotu 'path_to_project'.

Pokud aplikaci spouštíte pomocí WSL nebo na Linuxu, spusťte příkaz pro nastavení práv:

```
sudo chmod -R 777 program/bakalarka/storage
```

A následně spusťte ve složce **program** příkaz pro vytvoření kontejnerů Dockeru a vytvoření prostředí, ve kterém poběží aplikace:

```
docker-compose up
```

Server se spustí na portu 80, databáze na 3310 a phpmyadmin na portu 8081. Nahrané soubory se ukládají do složky *"cesta ke kořenovému adresáři serveru"/storage/app/files*. Pokud server poběží jinde než na localhostu, je třeba změnit hodnotu proměnné *APP_URL* v souboru `.env` (v adresáři program) na IP adresu serveru a v souboru */program/database/config.php* u spojení 'mysql'. A dále v souboru */program/config/app.php* je třeba upravit proměnné *server_name* na danou IP adresu a *folder* na cestu k adresáři, do kterého jste soubory zkopírovali. Pokud byste měnili i port, stačí ho připojit k IP adrese (IP:port).

Výchozí nastavení aplikace poskytuje tři uživatele s jednotlivými rolemi. Přístupové údaje pro roli `admin` jsou e-mail - 'admin@zcu.cz' a heslo '123456', což doporučuji co nejrychleji změnit (nejjednodušeji asi přímo v aplikaci v sekci *Správa uživatelů*). Stejně tak pro role `r/o` a `r/w` - e-mail - ro@zcu.cz, respektive rw@zcu.cz a heslo 'ro', respektive 'rw'. Aplikace obsahuje také několik ukázkových souborů. V již zmiňovaném souboru */program/config/app.php* je možné změnit i další výchozí hodnoty, mimo jiné délku promítání nebo počet položek u stránkování.

Pro ukončení aplikace slouží příkaz `docker-compose down`. A pro odstranění nepotřebných kontejnerů aplikace atd. slouží příkaz `docker system prune --volumes`.