
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ**
**DEPARTMENT OF COMPUTER SYSTEMS**


# WEBOVÁ PLATFORMA PRO SPOLUPRÁCI AKTIVISTŮ

**WEB COLLABORATION PLATFORM FOR ACTIVISTS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **JIŘÍ KABELKA**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. RADEK BURGET, Ph.D.**
**SUPERVISOR**



**BRNO 2019**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

# Zadání bakalářské práce


Student: **Kabelka Jiří**
Program: Informační technologie
Název: **Webová platforma pro spolupráci aktivistů**
**Web Collaboration Platform for Activists**
Kategorie: Web

Zadání:

1. Seznamte se s existujícími webovými nástroji pro skupinovou komunikaci a koordinaci.
2. Prostudujte aktuální technologie a architektury pro návrh webových aplikací se zaměřením na sociální sítě a aplikace pro týmovou spolupráci.
3. Analyzujte požadavky na webovou aplikaci pro spolupráci aktivistů na projektech. Navrhněte vhodnou architekturu aplikace.
4. Po dohodě s vedoucím implementujte navrženou aplikaci na vhodné platformě. Implementujte podporu týmové spolupráce na projektech a komunikace uživatelů v reálném čase.
5. Proveďte testování aplikace v reálném provozu.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 4. dubna 2019

# Abstrakt

Cílem této práce je vytvoření webové platformy pro sjednocování ekologicky zaměřených aktivistů obsahující nástroje pro vyhledávání vhodných projektových partnerů a následnou správu projektů. Obsahem práce jsou také nástroje pro komunikaci, hodnocení projektů a uživatelů. Platforma je postavena na architektonickém vzoru Model-view-controller. Pro implementaci komunikace se serverem používá platforma PHP Framework Laravel. Pro ukládání dat je použita databáze MySQL a pro vytvoření uživatelského rozhraní je použita CSS knihovna Bootstrap, Blade templating engine, Sass překompilovaný modulovým seskupovačem Webpack a nativní JavaScript s knihovnou jQuery.



# Abstract

The goal of this thesis is creating a web collaboration platform for connecting ecologically oriented activists containing tools for finding suitable collaborants and subsequent project management. The platform is built using the Model-view-controller architectural pattern. The platform is using the PHP Framework Laravel for server side communication and a MySQL database for data storage. The platform user interface is built using the CSS library Bootstrap, Blade templating engine, Sass compiled using the webpack module bundler and native Javascript with jQuery library.



# Klíčová slova

Projektová platforma, platforma pro spolupráci, sociální síť, správce projektů, PHP, Laravel, Blade, MySQL, Node.js, Bootstrap, CSS, Sass, Webpack, JavaScript, jQuery

# Keywords

Collaboration platform, social network, project management, PHP, Laravel, Blade, MySQL, Node.js, Bootstrap, CSS, Sass, Webpack, JavaScript, jQuery


# Citace

Kabelka Jiří: Webová platforma pro spolupráci aktivistů, bakalářská práce, Brno, FIT VUT v Brně, 2019

# Webová platforma pro spolupráci aktivistů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Kabelka
1.5.2019

## Poděkování

Děkuji svému vedoucímu Ing. Radku Burgetovi, Ph.D. za pomoc a odborné vedení při vypracování této bakalářské práce. Děkuji svému příteli Vojtěchu Adamovi za poradenství a pomoc s grafickými návrhy při vypracovávání designu platformy. Také děkuji celé akademické komisi a akademickému sboru FIT VUT za jejich čas a znalosti, které mi pomohly ve vypracovávání této bakalářské práce.

# Obsah

# 1 Úvod

Webové platformy a aplikace se stávají stále důležitějším aspektem moderního života. Díky své velké dostupnosti, přehlednosti a snadnému používání si webové aplikace efektivně udržují post nejpoužívanějšího prostředku pro sdílení informací, komunikaci, management financí, podniků a mezilidských vztahů. Cílem této práce je vytvoření webové platformy sjednocující uživatele zaměřené na ekologický aktivismus, která svým uživatelům poskytuje set funkcí a nástrojů pro správu a management jejich projektů.

Web v současnosti obsahuje velké množství blogů, sociálních sítí a fór, které se problematikou ochrany životního prostředí zabývají, charakter těchto webových stránek ale poskytuje svým uživatelům pouze webový prostor pro otevřenou diskuzi nad probíraným tématem bez motivace a nástrojů pro jejich okamžité řešení. Ve většině případů tento model webové stránky vede pouze k debatám, které nejenom neslouží k řešení dané problematiky, ale ve většině případů bývají spíše kontraproduktivní, jelikož všechny zúčastněné stojí velké množství času bez jakéhokoliv pozitivního konečného efektu. Vytvoření webové platformy Eco-Helper bylo motivováno sjednocováním lidí, kteří se chtějí zabývat řešením současných ekologických problematik, a následně jim poskytnout prostředí pro úspěšné provedení a dokončení jejich projektů.

Platforma je postavena na v současnosti nejpoužívanějším PHP aplikačním rámci Laravel, používá Model-view-controller návrh architektury a Sass preprocesor překompilovaný balíčkovacím nástrojem Webpack pro stylizaci vnějšího uživatelského rozhraní.

První část práce se zabývá popisem architektonického návrhu, popisem použitých nástrojů a technickými postupy použitými při vypracovávání aplikace. Druhá část dokumentu se bude zabývat designovými návrhy a nástroji určenými pro zajištění kvalitního uživatelského prostředí. Třetí část práce se zabývá výsledky webové platformy jako takové a úspěšností použitého designového a architektonického návrhu.

# 2 Analýza požadavků

Cílem této bakalářské práce je vytvoření webové platformy pro ekologicky zaměřené aktivisty ve věkovém rozmezí od osmnácti do třiceti let. Hlavním úkolem této platformy je kromě sjednocování uživatelů se stejným aktivistickým zaměřením i vytvoření jednoduchého projektového rozhraní, ve kterém mohou jednotliví účastníci projektu po jeho založení sdílet projektové informace a vzájemně spolupracovat. Samotná aplikace bude přitom rozdělena v rámci prvotního plánování do dvou sekcí. První sekce bude tvořena veřejnou stránkou, která bude volně dostupná všem uživatelům aplikace bez ohledu na vlastnictví uživatelského účtu. Druhá sekce bude tvořena vnitřní částí aplikace, která bude obsahovat hlavní funkcionality pro správu uživatelských projektů a jednotlivé nástroje, které budou tuto funkcionalitu rozšiřovat. Tato sekce bude uživatelům dostupná po založení uživatelského účtu a následném přihlášení. Celá platforma bude přitom vytvořena takovým způsobem, aby svými designovými prvky připomínala ostatní mainstreamové sociální sítě a aplikace zabývající se správou projektů [20].

**Finální specifikace požadavků pro webovou platformu určenou ke spolupráci aktivistů je tato:**

- Vytvoření veřejné části webové stránky, která bude prezentovat webovou platformu pro spolupráci aktivistů nezaregistrovaným návštěvníkům stránky.
- Vytvoření jednoduchého uživatelského menu, které bude fungovat jako rozcestník pro přihlášené uživatele.
- Podpora vyhledávání a přidávání uživatelů do existujících projektů.
- Systém pro hodnocení uživatelů a projektů.
- Zakládání nových projektů, vytváření projektových popisů a editace projektů jejich zakladatelem.
- Vytvoření projektového rozhraní, které bude jednotlivým členům poskytovat potřebné informace.
- Vytvoření jednoduché chatové aplikace umožňující komunikaci na úrovni jednotlivých členů projektu, v rámci všech uživatelů aplikace a pro soukromou komunikaci mezi dvěma uživateli.
- Vytvoření notifikačního systému pro usnadnění práce přihlášeného uživatele s platformou.
- Celá aplikace přitom musí být plně responzivní, přehledná a jednoduchá.

## 2.1 Srovnání s ostatními nástroji pro týmovou spolupráci

Na webu je v současnosti velké množství projektových a sociálních platforem. Tyto platformy se zaměřují na různé typy nástrojů pro týmovou spolupráci, jako je poskytování jednoduché komunikace a možnosti sdílení informací (např. Facebook[1]), přes nástroje pro rozdělování a organizaci jednotlivých projektových úkolů (Trello[2], Google Keep[3]), až po nástroje, které jsou specificky navrženy pro danou komunitu a typ projektového zaměření (GitHub[4]).

Všechny výše zmíněné platformy jsou zcela dostačující pro splnění svého účelů a jejich používání je velmi výhodné pro koncové uživatele. V případě vytváření webového prostoru pro aktivisty je nicméně častým problémem odchýlení platformy od původního cíle týmové spolupráce a z těchto platforem se stává uživatelské fórum, na kterém dochází k výměně informací o stávajících aktivistických problémech, přičemž tento dialog nevede k jejich řešení. Hlavním cílem webové platformy pro aktivisty bylo tedy vytvoření webové aplikace, která poskytuje svým uživatelům částečný sociální kontakt s ostatními uživateli na stránce a zároveň se silně zaměřuje na aktivní řešení jednotlivých projektů.

Dalším častým problémem výše zmíněných platforem bývá neflexibilní poskytování nástrojů pro organizaci a sdílení většího množství projektových informací. Typickým příkladem jsou například skupinové zdi v případě Facebooku nebo organizace příspěvků na internetovém fóru Reddit[5]. Po určitém množství přidaných informací se bohužel z důvodu nízké organizace jednotlivých příspěvků stávají skupina či fórum pro aktivní řešení projektů nepoužitelné. V případě platforem, které se zaměřují čistě na organizování jednotlivých projektových úkolů, je naopak častým problémem nedostatečná funkcionalita vzájemné interakce a tím způsobená potřeba používání dodatečných platforem a nástrojů.

**Řešení problémů týkajících se ostatních nástrojů pro týmovou spolupráci:**

- Na sociální platformě nebyly implementovány nástroje pro sdílení dat, které nejsou relevantní pro problematiku ekologie.
- Stránka na rozdíl od nástrojů jako jsou Trello nebo Google Keep poskytuje svým uživatelům jednoduchou chatovou aplikaci.
- Chatová aplikace stránky v případě komunikace většího množství uživatelů podporuje pouze globální a projektový chat.

[1] Facebook: https://www.facebook.com/
[2] Trello: https://trello.com/en
[3] Google Keep: https://keep.google.com/
[4] GitHub: https://github.com/
[5] Reddit: https://www.reddit.com/

- Webová platforma pro spolupráci aktivistů poskytuje svým uživatelům flexibilní nástroje pro spolupráci nehledě na hlavním zaměření projektu. Liší se tak od nástrojů pro týmy se specifickým zaměřením, které má například GitHub.
- Jednotlivé úkoly jsou v rámci projektového rozhraní rozděleny do tří hlavních sekcí, což zvyšuje přehlednost používání platformy a nepůsobí na nové členy projektu chaoticky, jak je tomu například v nástroji Trello.
- Na rozdíl od platforem Trello, Google Keep a GitHub umožňuje platforma pro spolupráci aktivistů jednoduché vyhledávací nástroje pro nábor nových členů projektu.

## 2.2 Technická specifikace

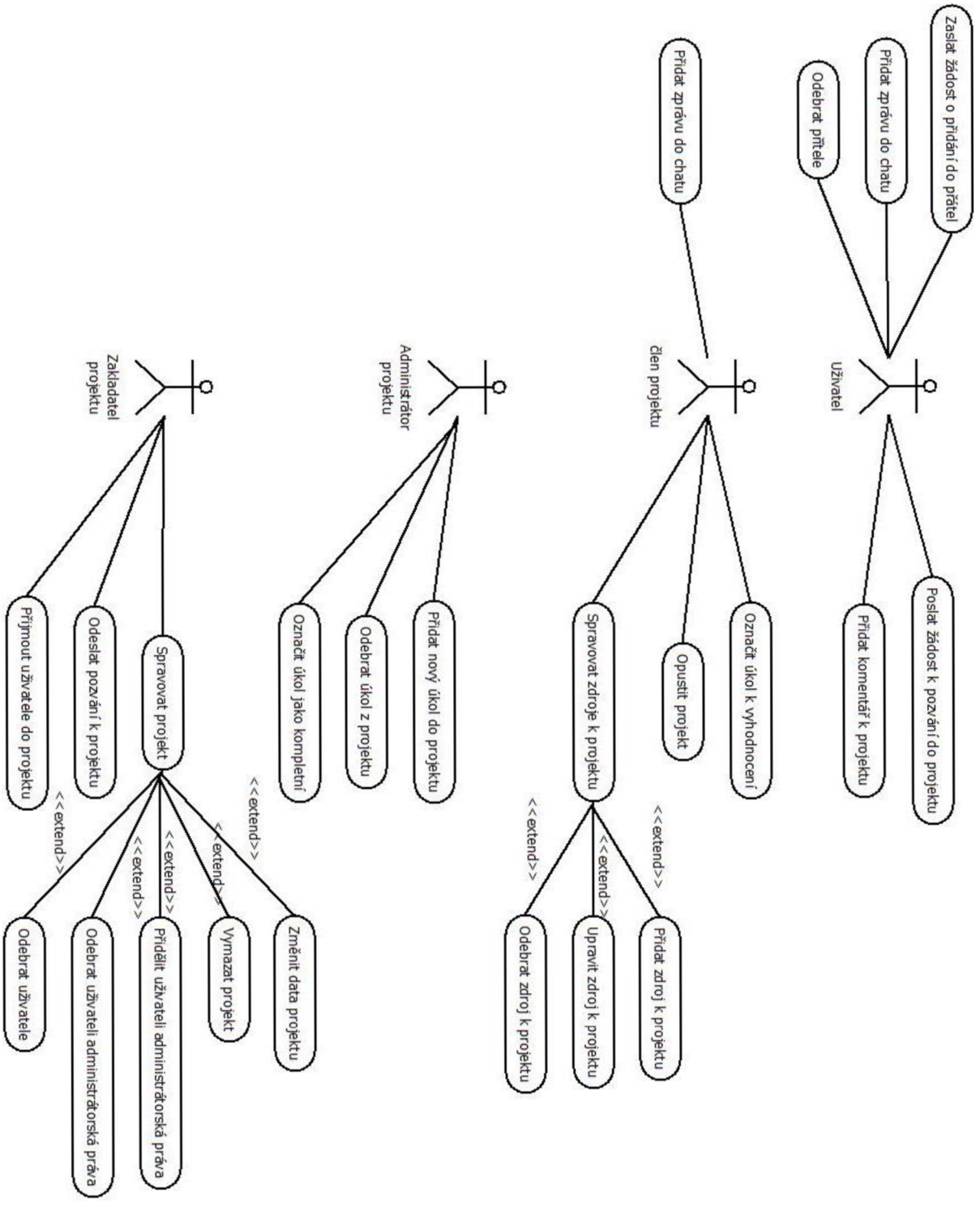
Celá platforma bude naprogramována za použití PHP aplikačního rámce pro vytváření webových aplikací Laravel. Aplikační rámec Laravel bude použit z důvodu jednoduchého používání, velkého množství aplikačních balíčků a pro dobré přizpůsobení metodice pro vytváření aplikací Rapid application development[6]. Pro vytvoření klientské části bude použit nativní systém pro vytváření webových šablon aplikačního rámce Laravel - Blade templating engine v kombinaci s kaskádovými styly definovanými za použití preprocesoru Sass. Pro přeložení Sassu do standardních kaskádových stylů bude použit balíčkovací a kompilační nástroj Webpack. Skripty a dynamické prvky klientské části aplikace budou napsány ve standardním programovacím jazyce JavaScript za použití knihovny jQuery a JavaScript doplňků, které jsou součástí knihovny Bootstrap [20]. Serverová část aplikace bude vytvořena s pomocí nativních nástrojů a postupů, které jsou součástí aplikačního rámce Laravel za použití modelu návrhu aplikací Model-View-Controller. Jednotlivé databázové tabulky budou nadefinovány s pomocí modelu Schema, který je předdefinovaným modelem aplikačního rámce Laravel v kombinaci s databázovým systémem MySQL. Součástí platformy bude také zajištění plně responzivního designu, korektního zobrazovaní aplikace na mobilních zařízeních a optimalizace pro snadnější vyhledávání platformy ve webových vyhledávacích nástrojích.

Tato technická specifikace platformy pro spolupráci aktivistů byla rozhodnuta v první plánovací iteraci a kromě designových aktualizací a obměn nebyla v dalších fázích vývoje razantnějším způsobem změněna

[6] Rapid application development: https://en.wikipedia.org/wiki/Rapid_application_development

## 2.3 Use Case diagram

Dominantou celé platformy pro spolupráci aktivistů je jednoduché projektové rozhraní, které vytváří přehledné prostředí pro sdílení informací mezi jednotlivými členy projektu. Kromě sdílení projektových informací umožňuje správu projektových dat administrátory a zakladatelem projektu. Jednotlivé funkce projektového rozhraní jsou popsány na obrázku 2.1.

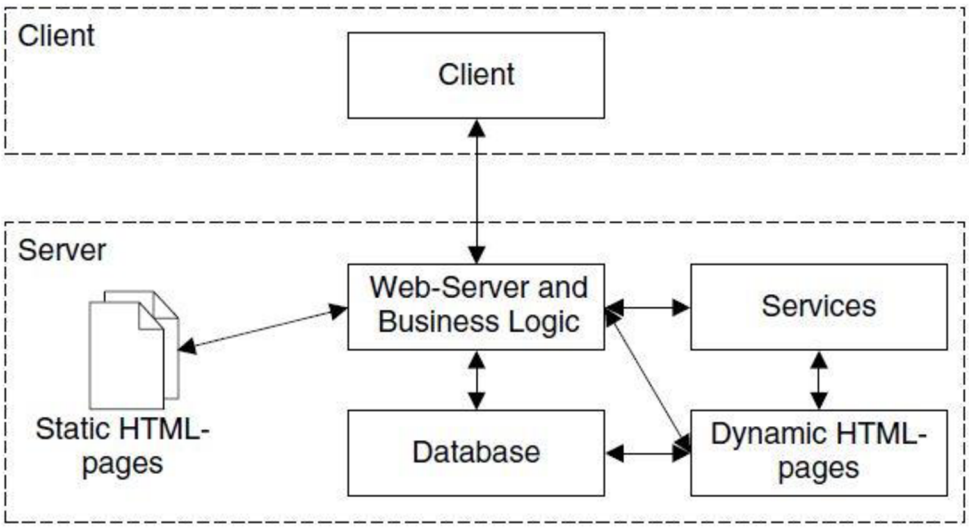



Obrázek 2.1: Use Case diagram webové platformy pro spolupráci aktivistů

# 3 Technologie pro tvorbu webových aplikací

V historii vývoje webových aplikací se vystřídalo několik dominantních architektonických návrhů. Z generálního hlediska lze ale rozložit všechny webové aplikace na tři základní složky. První složkou webových aplikací je klientská část webového prohlížeče. Tato část webových aplikací slouží k prezentování dat uživateli a funguje také jako základní interakční prostředek mezi uživatelem a zbytkem aplikace. Druhou částí webových aplikací je databázová vrstva tvořena strukturovaným systémem pro dlouhodobé přechovávání dat. Poslední složkou webových aplikací, která je ve většině případů značně dominantní, tvoří serverová komunikace. Ta je tvořena hlavní aplikační logikou a slouží jako přechod mezi databázovým systémem a prezentační částí webu. Standardní architektura webových aplikací je popsána na obrázku 3.1.

Pro vytvoření aplikační vrstvy webu jsou současnými vývojáři používány dva hlavní návrhy aplikační architektury. Ty bývají často obohaceny jednotlivými developery o jejich vlastí postupy, šablony aplikačních objektů a recyklovatelné knihovny zajišťující opakující se požadavky na funkce webových aplikací. Z těchto aplikačních návrhů jsou nejrozšířenější návrhy Model-1 a Model-2, který je známý pod názvem Model-view-controller nebo také MVC. Hlavním rozdílem mezi těmito modely je hlavně počet menších abstrakčních částí, na které tyto architektonické návrhy celou aplikační vrstvu rozdělují [4].




Obrázek 3.1: Diagram standardního architektonického návrhu webové stránky převzatý z [4]

## 3.1 Webové aplikační rámce

Pro oba modely aplikační vrstvy existuje velké množství aplikačních rámců, které umožňují vývojářům kromě rychlejšího a čistšího procesu programování také zajištění základních struktur webové aplikace, které se v architektuře často opakují. Tyto základní postupy by bylo při vývoji aplikace časově neefektivní samostatně programovat. Základem těchto rámců bývá často rozložení aplikační vrstvy, přichystání konstantního spojení s databázovým systémem, vytvoření abstraktních tříd jednotlivých složek aplikačních objektů nebo například ochrana webových formulářů proti základním útokům typu SQL Injection[7].

Kromě rámců, které se zabývají aplikační a databázovou vrstvou, existuje také velké množství knihoven, které se zaměřují na klientskou část webu a vytvářejí pro koncového uživatele dynamičtější a flexibilnější vjem z interakce s webovou stránkou. V současnosti jsou velmi rozšířené knihovny, které se zabývají možností vytváření tzv. „jednostránkových aplikací." Tyto knihovny umožňují webovým prohlížečům načtení celé klientské části webu do jednoho dokumentového modelu. Tento model poté umožňuje svým uživatelům téměř okamžitý přechod mezi jednotlivými stránkami webu, protože se při přechodu mezi jednotlivými pohledy nemusí vysílat další požadavek na webový server. Jedná se ale o relativní novinku, proto může být použití těchto nástrojů v současnosti neefektivní hlavně z hlediska nedostatečné optimalizace pro webové vyhledávací nástroje. Další nevýhodou může být také zvýšená obtížnost implementace stránky webovým vývojářem. Nejpoužívanější knihovny klientské části webu tvoří v současnosti Angluar.js, React.js a nejnovější knihovna Vue.js [5], která se pokouší spojit nejlepší prvky obou předchozích nástrojů dohromady.

**Nejrozšířenější webové aplikační rámce k roku 2017 [5]:**

1. ASP.NET
2. AngularJS
3. Ruby on Rails
4. ASP.NET MVC
5. React
6. Django
7. Laravel
8. Angular
9. Spring
10. Express

[7] SQL Injection: https://cs.wikipedia.org/wiki/SQL_injection

## 3.2 Srovnání webových aplikačních rámců

### 3.2.1 Laravel

Laravel je webový aplikační rámec založený na programovacím jazyce PHP, který byl použit při implementaci webové platformy pro spolupráci aktivistů, jež je předmětem této bakalářské práce. Laravel je v současnosti nejpoužívanějším webovým PHP aplikačním rámcem [5], který slouží k implementaci dynamických webových aplikací založených na architektuře MVC – Model-view-controller. Specifičností Laravelu je jednoduché používání a velmi rychlá učební křivka. Laravel řeší všechny základní problémy, které je potřeba řešit na začátku vývoje většiny komplexních webových aplikací. Poskytuje také velké množství dodatečných balíčků a funkcí, které jsou zahrnuty v kořenovém adresáři. K jejich užívání slouží jednoduchá změna jednořádkového příkazu v souboru nastavení.

Kromě základních funkcí v adresáři Laravelu vytváří PHP komunita velké množství sekundárních balíčků, které lze jednoduše naimportovat stažením a zahrnutím v souborech aplikačního rámce. Hlavní výhodou Laravelu je ale extrémní jednoduchost používání a samotná implementace aplikace se pohybuje pouze v rozmezí navrhování pohledů, kontrolerů a modelů s jednoduchými návrhy databázových tabulek a kaskádových stylů. Problémem Laravelu je jeho komplexnost a objem základního projektového adresáře společně s obtížnějším nasazováním aplikace do produkce.

### 3.2.2 CodeIgniter

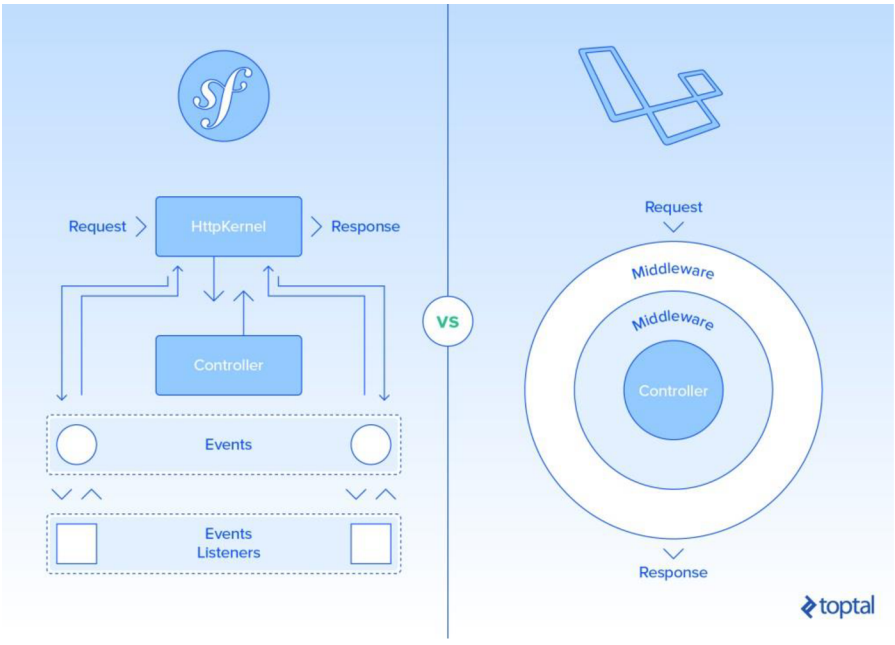
CodeIgniter je stejně jako Laravel webový aplikační rámec napsaný v jazyce PHP. Na rozdíl od Laravelu, který je zaměřený na flexibilní vytváření webových aplikací bez ohledu na náročnost, množství uživatelů a zaměření aplikace, se CodeIgniter zaměřuje v první řadě na rychlost implementace webové stránky. CodeIgniter se nespecializuje na spolupráci v týmu a podporu komplexnějších projektů. Je tedy výhodnější pro vývojáře, kteří pracují na jednodušších individuálních projektech. Zaměření na jednodušší projekty nicméně snižuje komplexnost samotného aplikačního rámce a dává mu rychlejší učební křivku [12].

Kromě rychlejší učební křivky umožňuje jednodušší způsob aktualizování aplikačního kódu v případě přechodu na novější verzi. Výhodou je také jednoduchá zpětná kompatibilita s dřívějšími verzemi PHP, což umožňuje vývojáři snadnou práci s vytvářením aplikace pro starší servery a technická prostředí. Hlavní nevýhodou ale vyšší obtížnost rozšiřování aplikačního kódu v komplexnějších aplikacích a snížená flexibilita při práci ve větších týmech [12].

### 3.2.3 Symfony

Dalším programátorským aplikačním rámcem podskupiny jazyka PHP je Symfony. Na rozdíl od Laravelu, který přistupuje k celé aplikační logice jako k jedinému celku napsanému s pomocí návrhu architektury MVC, Symfony rozděluje všechny jednotlivé části své aplikační logiky do uzavřených, samostatných, mezi sebou komunikujících komponent. Jednotlivá komponenta napsána v rámci Symfony přitom nejsou omezena pouze na recyklaci a znovupoužití v dalších aplikacích napsaných v totožném aplikačním rámci, ale lze je snadno importovat do jiných aplikací a projektů, které používají pro svou implementaci programovací jazyk PHP (samotný aplikační rámec Laravel obsahuje v současnosti 14 komponent napsaných za použití Symfony). Samostatné naučení práce se Symfony a použití programátorských praktik, které tvoří jeho základ, může být pro vývojáře přínosné i v případě používání jiných programovacích jazyků a aplikačních rámců.

Symfony také používá pro zpracovávání webových požadavků před vstupem do řadiče svůj vlastní generátor aplikačních událostí. Laravel pro stejnou funkcionalitu používá Middleware, který požadavek zpracuje a může vytvořit specializovanou odpověď, nicméně generátor událostí může na jednotlivé požadavky reagovat komplexnějším způsobem [13]. Používání generátoru je tedy zpravidla výhodnější pro zpracovávání požadavků složitějších aplikací, nicméně pro většinu webových stránek postačí vývojáři mechanismu middleware. Zpracovávání požadavků v rámcích Laravel a Symfony lze vidět na obrázku 3.2. Kromě těchto dvou zásadních rozdílů jsou si ale Laravel a Symfony velmi podobné [13]. Oba aplikační rámce používají model architektury MVC, obě architektury umožňují dědičnost jednotlivých aplikačních pohledů a v případě obou rámců používají systém přidávání dodatečných závislostí do projektu.




Obrázek 3.2: Diagram komunikace s API v aplikačním rámci Symfony převzatý z [13]

### 3.2.4 Nette

V množině světově používaných webových aplikačních rámců se v nedávné době objevil také český PHP Framework Nette, který je v současnosti nejpoužívanějším webovým aplikačním rámcem v České republice. Stejně jako Laravel i Nette Framework používá model architektury MVC. Také ke správě svých komponent a přidávání dodatečných funkcí do celého projektu používá Composer a Node.js balíčkových systémů. S Laravelem sdílí i stejný přístup k vytváření a správě webových formulářů oba aplikační rámce mají pro tuto funkcí vlastní systém, který se dá volitelně využít.

Velkou výhodu Nette frameworku představuje dynamický přístup k webovým formulářům [15], který obsahuje jednoduchá, jednořádková ověření s jasně definovanými reakcemi, které mohou být přizpůsobeny vyplnění volitelnému množství jednotlivých textových input oken. Kromě dynamičnosti formulářů obsahuje Nette framework také vlastní systém pro generování forem za použití souborů obsahujících data k definování konkrétních znovupoužitelných pravidel [15]. Nette framework se na rozdíl od Laravelu v základu zabývá čistě řešením základních implementačních problémů, které se opakují v každém projektu [15]. Neobsahuje tolik defaultně vložených funkcí, jako jsou například ochrana proti cross-site skriptům[8], poskytování hašovacích funkcí[9], automatické ověřování uživatelů, automatické generování kódu, integrace platebních bran apod. [15], což se vývojářům, kteří se pohybují hlavně v Laravelu, může zdát jako jasná nevýhoda. Omezení množství základních funkcí, které obsahuje každý nově vygenerovaný projekt Nette frameworku, nicméně vede k mnohonásobně menšímu objemu projektového adresáře a zrychlení aplikačního rámce jako takového. Nette se také drží základních vývojářských praktik pro programování PHP webových aplikací, což umožňuje snadnější uvádění aplikace do produkce. Nette framework je z velké části založen na továrnách kódu, které jsou podkladem pro rychlé vytváření jednotlivých aplikačních komponent, a vlastním komponentovém modelu, který umožňuje snadné rozdělování celé aplikace do libovolného množství jednotlivých prvků.

Mezi hlavní nevýhody může patřit již dříve zmíněný nedostatek pokročilých funkcí v čerstvě vytvořených projektech, větší komplexnost aplikačního kódu, delší čas potřebný k implementaci projektu, delší učební křivka a celkově složitější proces implementace. Zvýšená složitost implementačních procesů a nedostatek základních funkcí může také v případě začínajících programátorů vést ke zvýšené náchylnosti webové aplikace k různým formám útoků.

[8] Cross-site scripting: https://cs.wikipedia.org/wiki/Cross-site_scripting
[9] Hašovací funkce: https://cs.wikipedia.org/wiki/Ha%C5%A1ovac%C3%AD_funkce

### 3.2.5 Wordpress

Spolu se vznikem velkého množství různých typů aplikačních rámců pro programování webových aplikací začala také vznikat aplikační rámce s primárním zaměřením na udržovatelnost a editaci stránky jejím vlastníkem. V případě Wordpressu se standardně nepoužívá pojem aplikační rámec, protože se zabývá více udržovatelností webových stránek, než samotnou implementací. V praxi se nicméně často využívají šablony jednotlivých programátorů, které se dají označit za jednoduché aplikační rámce pro opakovanou implementaci již v minulosti řešených technických detailů jiných stránek. Mezi nejznámější webové aplikační rámce zaměřené na údržbu vlastníkem patří Joomla[10], Drupal[11] a v současnosti nejpoužívanější systém pro správu obsahu Wordpress.

Vývojářská komunita vytváří pro aplikační rámec Wordpress množství šablon, zásuvných modulů a jiných funkcí na rychlé přidávání vlastností do webové aplikace. Tato vlastnost je specifická pro systémy pro rozšířené systémy pro správu obsahu a je ohromnou výhodou oproti standardním webovým aplikačním rámcům. Tyto moduly jsou lehko udržovatelné jak vývojářem, tak samotným vlastníkem stránky. Z pohledu vývojáře se hlavní část implementace stránky skládá ze dvou částí. První částí je vytvoření vlastní šablony, která je poté editovatelná zákazníkem na produkci. Druhou částí je zajišťování komunikace mezi vlastnoručně naprogramovanou šablonou a ostatními zásuvnými moduly [14]. Velkou výhodou Wordpressu je rychlá implementace aplikací a platforem, které umožňují svým vlastníkům snadné a rychlé upravování stránky, a to ve velkém detailu, bez potřeby jakýchkoliv programátorských znalostí. Kromě toho jsou stránky ve Wordpressu s pomocí vlastního systému pro správu verzí snadno udržovatelné a poskytují vlastníkům stránky vlastní systém optimalizace pro vyhledávací nástroje.

Kvůli důrazu na předpřipravené zásuvné moduly a postupy často vývojáře samotný charakter Wordpressu značně omezuje a proto se ve většině případů Wordpress používá pouze na jednodušší stránky, které systém pro správu obsahu vyžadují. Mezi tyto stránky patří například blogy, jednoduché stránky a internetové obchody. V případě složitějších projektů a aplikací se většina vývojářů uchyluje k používání alternativních webových aplikačních rámců, které jim umožňují větší množství vlastních úprav a vytváření systému od základu. Mezi nevýhody patří častá chybovost a potřeba podpory klienta po vytvoření webové stránky. Kvůli své komplexní charakteristice a velkému množství úprav vede používání Wordpressu vlastníkem stránky k nechtěným úpravám a jiným chybám

[10] Joomla: https://www.joomla.org/
[11] Drupal: https://www.drupal.org/

### 3.2.6 ASP.NET

ASP.NET je serverové webový aplikační rámec pro vytváření dynamických webových stránek a aplikací vyvíjený společností Microsoft. Pro ASP.NET se často používá alternativní název „webové formuláře," který naráží na specializovaný styl XHTML programovacího jazyka, jež se v aplikačním rámci ASP.NET používá [11].

Mezi hlavní výhody používání ASP.NET patří hlavně modulární přístup k vývoji jednotlivých komponent, který umožňuje jednoduché znovupoužití kódu při modifikacích nebo novějších iteracích aplikačního rámce [11]. ASP.NET obsahuje také velké množství knihoven, které podporují použití aplikačního rámce pro aplikace s velkou náročností a velkým užitím hardwarových prostředků. Proto má ASP.NET ve většině případů lepší výkon a využití zdrojů.

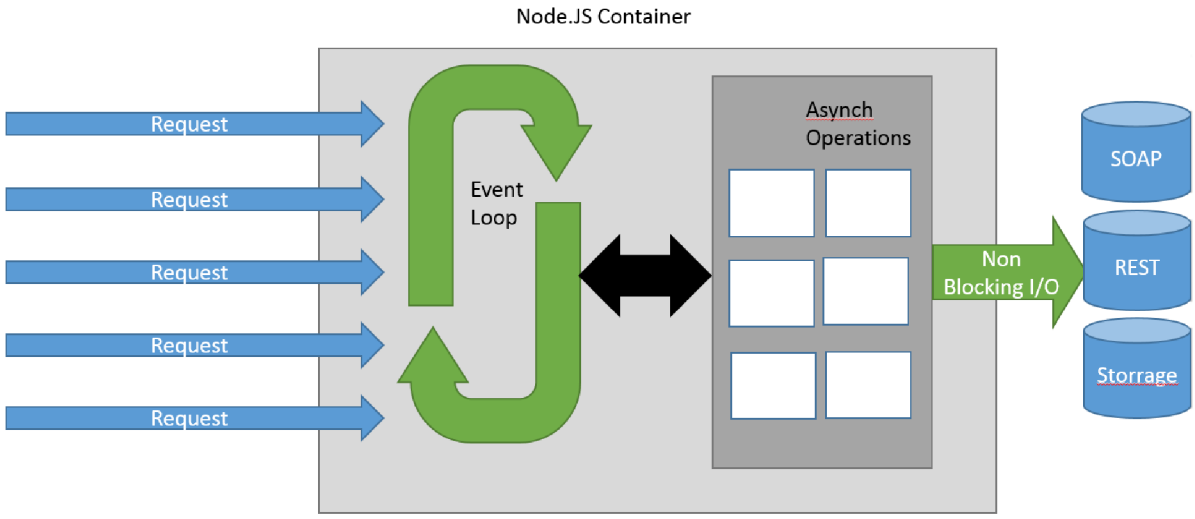
### 3.2.7 Java

Java se kromě vytváření webových stránek používá také k vývoji nativních aplikací, které jsou kompatibilní na mobilních i desktopových zařízeních napříč různými typy operačních systémů. Stejně jako jazyk PHP, i jazyk Java je ve webovém prostředí používán velkým množstvím webových aplikačních rámců. Ve většině případů se jedná o čisté, přímo zaměřené aplikační rámce. Na rozdíl od aplikačních rámců napsaných v jazyce PHP, které ve většině případů používají velké množství balíčků a dodatečných funkcí, což může částečně redukovat aplikační stabilitu. Velkou výhodou oproti PHP je také nepotřebnost restartování virtuálního modelu v případě obdržení webového požadavku od uživatele stránky, což značně napomáhá celkovému výkonu webových aplikací naprogramovaných v jazyce Java. Mezi další klady tohoto jazyka patří také lépe napsaný kód, který je díky vývojářským nástrojům a modulárnímu přístupu v Javě snadno udržovatelný a spolehlivý. Modulární vývoj je sice běžný i pro jazyk PHP, avšak větší komunitní vlivy a velká popularita vytváření balíčků může v některých modulech vést k nekompatibilitám a jiným chybám. Nevýhodou Javy oproti aplikačním rámcům ASP.NET a PHP je nedostatečná možnost řešení problém s jinými developery. Mezi další problémy se dá zařadit i horší kompatibilita se serverovými zařízeními, nebo delší učební křivka, která je způsobena nižší komunitní podporou. Komunitní podpora je horší z důvodu menší oblíbenosti tohoto jazyka a z toho vyplývající menší developerskou základnou.

### 3.2.8 Node.js

Systém Node.js, který obsahuje svůj vlastní balíčkovací NPM systém, obsahuje také množství aplikačních rámců zabývajících se poskytnutím serverové komunikace mezi webovými aplikacemi a hardwarem serveru. Mezi nejznámější z těchto balíčků patří v současnosti balíček Express[12]. Hlavní výhodou Node.js je použití asynchronní technologie, která vytváří pro každý požadavek stránky separátní proces, což značně urychluje celkový provoz webového serveru. Přístup k požadavkům v rámci aplikačního rámce Node.js je zobrazen na obrázku 3.3. V současnosti se ukazuje, že je optimalizace práce s jednotlivými webovými požadavky oproti standardním PHP aplikačním webovým rámcům dvojnásobně efektivnější. Velkou výhodou je také lepší komunitní podpora a množství aplikačních rozšíření, která usnadňují a urychlují vývoj aplikací. Kromě optimalizace rychlosti zpracování webových požadavků a práce s hardwarem umožňuje Node.js také efektivnější správu aplikací, které pro svůj provoz potřebují využívat větší množství serverových procesů [21]. Mezi tyto aplikace se řadí např. sociální sítě, stránky obsahující dynamická média apod.

Používání Node.js je vhodné jen u větších aplikací, v jiných případech je nevhodné hlavně z důvodu častých nekompatibilit se serverovými zařízeními. Dalším problémem je zhoršený výkon při zpracovávání objemnějších, celistvých dat [21] a relativní novost nástroje spojená s nedostatečnou komunitní podporou a absencí stabilních, spolehlivých aplikačních rámců.




Obrázek 3.3: Diagram přijímání klientských požadavků v aplikačním rámci Node.js převzatý z [21]

[12] Express: https://expressjs.com/

## 3.3 Balíčky pro webové aplikace

V rámci webových platforem je běžné používání balíčků, které obsahují často se opakující úseky kódu jednotlivých webových aplikací. Mezi tyto požadavky mohou patřit například implementace webových nákupních košíků, chatových rozhraní, platebních bran apod. V dnešní době již většina webových aplikačních rámců nativně podporuje snadný import těchto balíčků. Ke snadnějšímu používání a kombinování importovaných balíčků se vyvinulo množství balíčkovacích systémů, které s pomocí přednastavených souborů importují, propojují a kontrolují správnou komunikaci jednotlivých aplikačních modulů. Webová platforma pro ekologické aktivisty používá jedny z nejznámějších balíčkovacích systémů NPM pro import klientských modulů a systém Composer pro správu balíčků umožňujících korektní funkčnost serverové části aplikačního rámce Laravel.

### 3.3.1 NPM

Npm je balíčkovací systém JavaScriptových knihoven, který kromě knihoven pro klientskou část aplikace poskytuje řadu vedlejších funkcionalit, jako je například používání lokálních vývojářských serverů, překládání modernějších webových jazyků do podoby, ve které může být kód přečten současnými webovými prohlížeči a používání alternativních nástrojů pro vývoj webových aplikací, které nejsou současnými prohlížeči podporovány [16]. V současnosti je také balíčkovací systém NPM používán k importu knihoven umožňujících vývoj serverových části aplikací za použití nástroje Node.js. Npm umožňuje lokální import běžně používaných knihoven, jako jsou například Bootstrap, JavaScript nebo jQuery a může tak svým vývojářům poskytnout lehkou kontrolu nad jednotlivými aplikačními knihovnami. Npm také usnadňuje programování aplikace v případě, kdy programátor nemá v některých obdobích vývoje přístup k internetu.

### 3.3.2 Composer

Stejně jako NPM, i Composer je balíčkovací systém, který umožňuje komunikaci a import jednotlivých aplikačních balíčků [17]. Hlavním rozdílem mezi NPM a Composerem je fakt, že balíčkovací systém Composer slouží k importování a správě balíčků programovacího jazyka PHP. V případě platformy pro spolupráci aktivistů se Composer používá k importování modulů použitých k zajištění správné serverové funkčnosti aplikačního rámce Laravel. Volitelně importuje také nástroj Laravel Tinker[13] pro správu databázových objektů z příkazové řádky, dodatečné balíčky sloužící ke správě sekundárních modulů a nástroj PHPUnit[14] pro jednoduché vytváření unit testů v jazyce PHP.

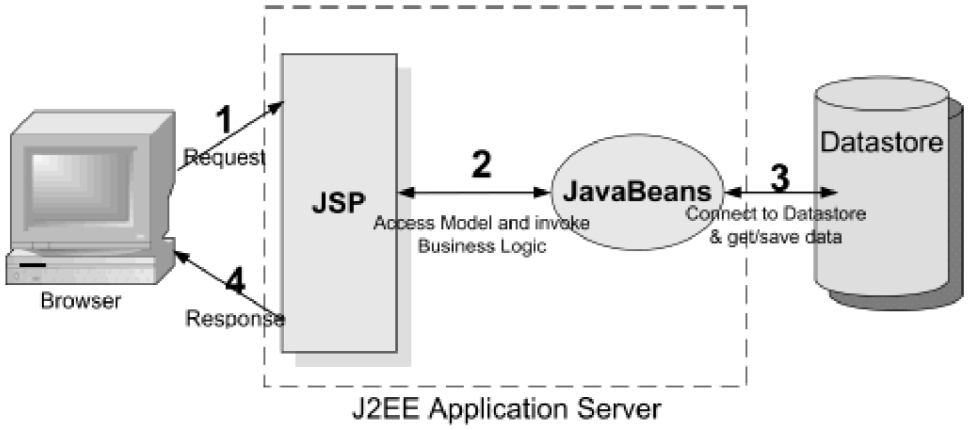
[13] Laravel Tinker: https://laravel-news.com/laravel-tinker
[14] PHPUnit: https://phpunit.de/

## 3.4 Architektonické návrhy webových aplikací

Na platformě webů se stejně jako v mobilních a desktopových aplikacích opakují často řešené problémy a jednotliví vývojáři se snaží pro své aplikace vyvíjet časově efektivnější, rozšiřitelnější a snadno udržovatelné formy architektonického návrhu pro své aplikace. V současnosti převládají dvě formy architektonického návrhu Model-1 a Model-view-controller, který je v současnosti zejména pro svou snadnou rozšiřitelnost, jednoduchost používání a obecnou přehlednost nejpoužívanějším webovým návrhem architektury [4].

### 3.4.1 Model-1

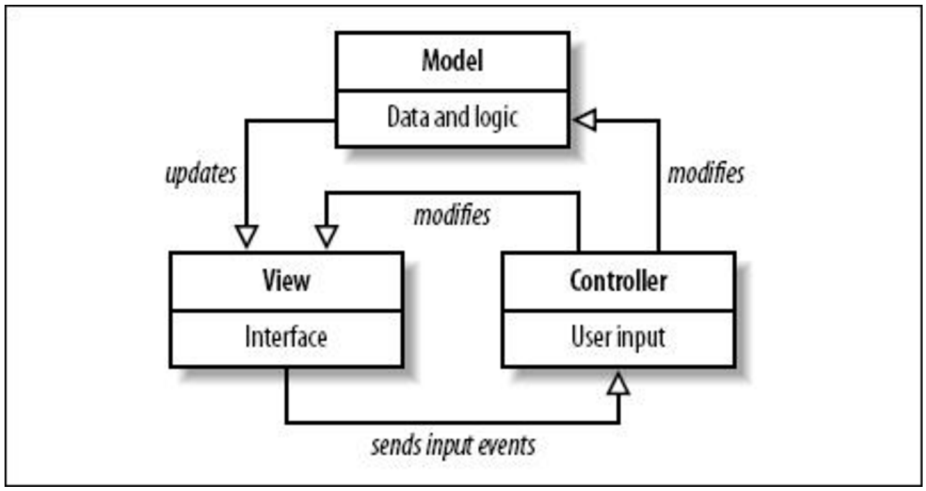
Model-1 rozděluje aplikační vrstvu na dvě hlavní složky prezentační a modelové části. Prezentační část přitom tvoří hlavní komunikační prostředek mezi uživatelem stránky a aplikační vrstvou, část modelová obsahuje hlavní aplikační logiku a komunikaci s databázovým systémem [3]. Architektura modelu-1 je zobrazena na obrázku 3.4. Jednoduchost modelu-1 nabízí vývojářům rychlejší a přehlednější vytváření menších aplikací. Ve větším měřítku se ale ukazuje být jednoduché rozdělení aplikační vrstvy na pouze dvě podskupiny problémové hlavně z důvodu rychlého nárůstu kódu v prezentační části aplikace a následným omezeným možnostem přidávání nových funkcí a správy stávajících úseků projektu. Proto se postupem času přešlo na model architektury Model-view-controller, který nabízí podrobnější rozdělení aplikační vrstvy na menší podskupiny a na rozdíl od modelu-1 vývojářům jasně definuje použití svých jednotlivých podsložek [4].




Obrázek 3.4: Diagram zpracovávání klientských požadavků v návrhu aplikační architektury Model-1převzatý z [3]

### 3.4.2 Model-view-controller

Jak už napovídá název návrhu aplikační vrstvy MVC, architektura Model-view-controller rozděluje celou webovou stránku na tři oddělitelné prvky, které mají jasně nastavenou roli a účel v rámci celé webové aplikace. První část architektury tvoří model, který představuje základní objektovou a aplikační logiku. Druhou část tvoří pohled, který slouží k prezentování dat uživateli. Třetí částí je kontroler (nebo česky řadič), který vytváří hlavní komunikační prostředek mezi jednotlivými modely a pohledy. MVC architektura je v současnosti jedním z nejpoužívanějších návrhových vzorů pro budování webových aplikací. Je tomu tak hlavně z důvodu lehké udržovatelnosti, přehlednosti a možnosti lehkého pochopení abstrakce výsledného kódu [6]. Mezi aplikační rámce, které používají MVC architekturu, patří například Symfony, ASP.NET, Ruby on Rails[15] nebo Laravel, který platforma pro spolupráci aktivistů využívá. Diagram zpracování klientských požadavků v návrhu MVC je zobrazen na obrázku 3.5.




Obrázek 3.5: Diagram zpracovávání klientských požadavků v návrhu aplikační architektury Model-view-controller převzatý z [6]

#### 3.4.2.1 Ukázka použití

Uživatel se dostane na stránku s pomocí odkazu URL, tento odkaz je předán příslušnému řadiči, který tento požadavek volitelně ověří. Řadič poté přistupuje k příslušnému modelu nebo skupině modelů, které slouží k provedení uživatelem specifikovaného požadavku. Modely poté provedou uživatelem specifikované aktualizace dat, popř. vrátí řadiči již existující data požadovaná uživatelem. Řadič poté předá požadovaná data do pohledu, který společně s daty tvoří nové zobrazení stránky uživateli. Aplikace poté čeká na novou uživatelskou aktivitu, která celý proces započne znova.

[15] Ruby on Rails: https://rubyonrails.org/

#### 3.4.2.2 Model

První částí MVC návrhu architektury je tzv. „model," který v rámci webu reprezentuje jednotlivé samostatně fungující objekty, se kterými aplikace pracuje. V praxi obsahují modely hlavní CRUD[16] logiku pro zobrazování, modifikaci a mazání aplikačních dat. Ve většině případů je každý model napojen na vlastní databázovou tabulku, kterou disponuje. Modely ale mohou svobodně přistupovat k metodám a atributům jiných objektů a stejně tak je tedy mohou vytvářet, editovat, či odstraňovat. V modelech platí stejná pravidla, která platí pro objektově orientované programování. Každý model může disponovat privátními, statickými i veřejnými atributy a metodami podle použitého programovacího jazyka. Objektově orientovaný charakter modelů také umožňuje jednoduché vytváření modelových hierarchií tvořících vztahy mezi jednotlivými objekty.

#### 3.4.2.3 View

Druhou část MVC návrhu architektury tvoří tzv. „view," (nebo česky pohled), který slouží k prezentování dat uživateli a dává mu možnost dynamické interakce s daty aplikace. Pohledová část MVC modelu v základu splňuje celou úlohu klientské části webu, hlavním rozdílem je ale způsob, jakým pohled komunikuje se zbytkem aplikace. Při příchodu uživatele na stránku je prvně s pomocí URL odkazu nalezen řadič, který danému požadavku v rámci aplikace odpovídá, poté jsou zavolány modely, které z databázového systému získají požadovaná data, ta jsou poté načtena s požadovanou šablonou zpět uživateli. V případě potřeby můžou být do šablony importovány ukazatele na další požadované modely, které umožňují větší flexibilitu a více možností v rámci zobrazování dat uživateli. V případě Laravelu jsou pohledy tvořeny Blade templating enginem. Blade templating enginem obsahuje vlastní syntaxi PHP výrazů a balíček nástrojů umožňující jednodušší zobrazení a práci s PHP syntaxí. Také obsahuje nástroje pro přímý přístup k aplikačním modelům.

#### 3.4.2.4 Controller

Poslední část MVC návrhu architektury tvoří řadič, který slouží jako komunikační prostředek mezi jednotlivými modely a šablonami pohledů. Řadiče obsahují prostředky pro zajištění odpovědi na odkazy URL a validátory pro zajištění správnosti dat po vyplnění a odeslání požadavku uživatelem. Kromě zajištění odpovědí a ověření správnosti požadavků slouží řadiče také k získání dat požadovaných uživatelem. Dosahují toho pomocí volání příslušných modelů, které daná data zpracovávají. V praxi tedy řadiče obsahují hlavně validaci vstupů a připravují data pro modely, které obsahují hlavní aplikační logiku zpracovávající operace požadované uživatelem, změny v databázovém systému a získávání požadovaných dat. Při vývoji je typicky velká snaha o zajištění co nejmenší složitosti řadičů, zatímco hlavní algoritmizační část aplikace je převáděna do modelů. V případě chyby nebo úspěšného zpracování požadavku je součástí řadiče také systém vrácení chybových a potvrzovacích zpráv.

[16] Create, read, update, delete logika: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

# 4 Návrh platformy

Platforma pro spolupráci aktivistů je naprogramována s pomocí PHP aplikačního rámce Laravel 5.4. Laravel používá MVC aplikační architekturu a k práci s jednotlivými pohledy používá Blade templating engine. Aplikace používá k přechovávání dat databázový systém MySQL. Pro svou klientskou část používá aplikace nativní JavaScript, jQuery, Bootstrap a Sass překompilovaný s pomocí balíčkovacího systému Webpack do CSS. CSS kód je rozdělený do dvou zdrojových souborů dělených pro veřejnou a privátní část aplikace. Jednotlivé skripty jsou s pomocí Blade templating enginu importovány přímo do jednotlivých pohledů.

## 4.1 Laravel Framework

Laravel je v současnosti díky svému lehkému používaní, krátké učební křivce a možností jednoduchého importu dodatečných uživatelských balíčků nejpoužívanějším webovým vývojářským aplikačním rámcem napsaném v jazyce PHP [5]. Zároveň je také jedním z deseti nejpoužívanějších webových aplikačních rámců na trhu [5].

Kromě uživatelských balíčků obsahuje Laravel také velké množství volitelných knihoven a rozšíření, která jsou nativně zabudovaná do základní struktury a k jejich použití stačí jednoduché importování nástrojů do předpřipraveného objektu zajišťujícího načítání jednotlivých aplikačních tříd aktivních balíčků. Mezi nejpoužívanější z těchto nástrojů patří například Laravel Horizon[17] zajišťující podporu Redis[18] databází, Laravel Dusk[19] zabývající se zjednodušeným testováním a Laravel Echo[20], který aplikaci umožňuje jednoduché vytváření komponent fungujících v reálném čase [9].

[17] Laravel Horizon: https://laravel.com/docs/5.8/horizon
[18] Redis: https://redis.io/
[19] Laravel Dusk: https://laravel.com/docs/5.8/dusk
[20] Laravel Echo: https://laravel.com/docs/5.8/broadcasting

## 4.2 Blade templating engine

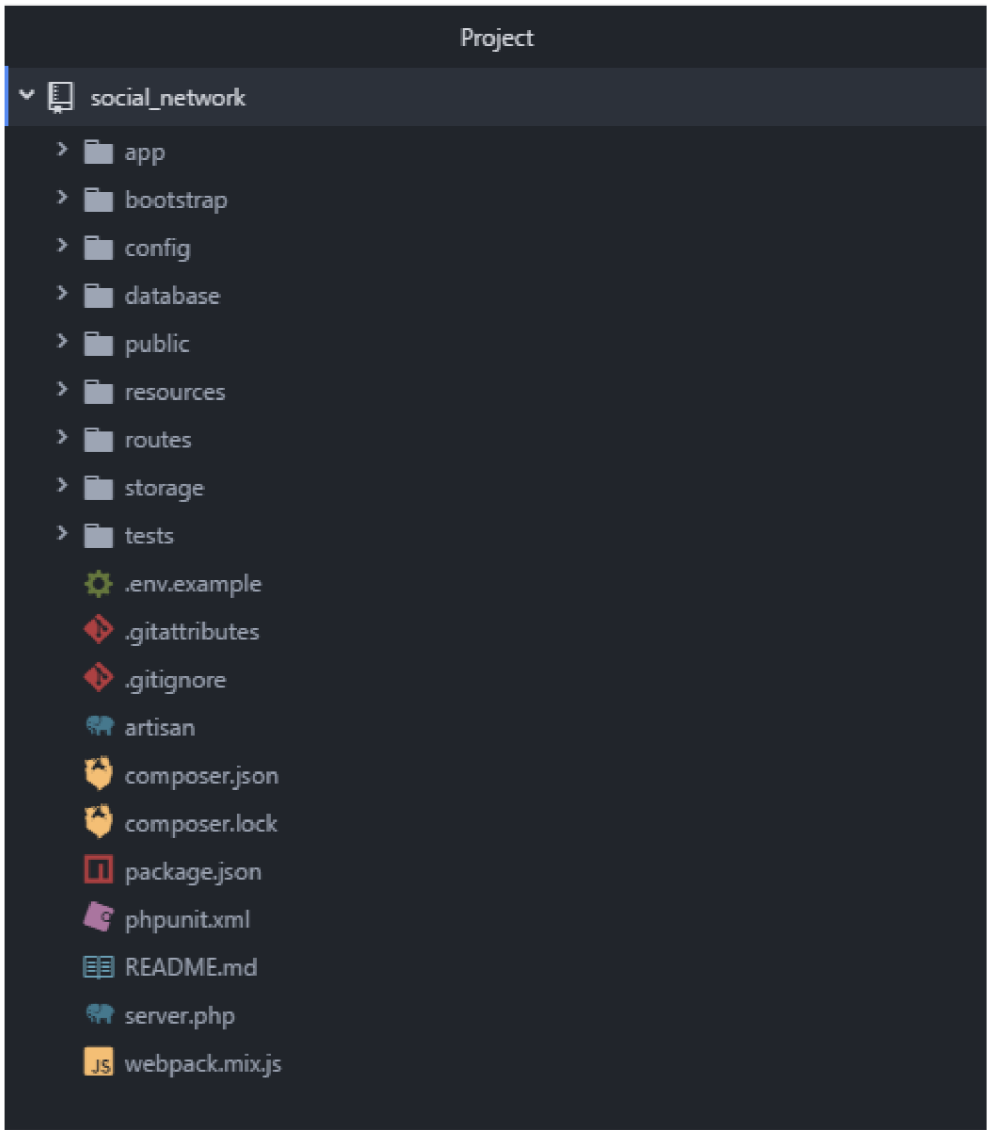
Moderní webové aplikační rámce obsahují ve většině případů vlastní systém pro vytváření pohledů. Laravel používá k vytváření svých pohledů nativní Blade templating engine [7]. Blade umožňuje kromě vlastní syntaxe PHP sloužící k psaní efektivnějšího kódu také vytváření hierarchie a dědičnosti mezi jednotlivými pohledy aplikace. Dále se zaměřuje na používání klasického PHP kódu, v pozadí aplikace totiž dochází při změnách kódu pohledu ke konstantnímu překladu pohledu do klasického PHP [7]. Na obrázku 4.1 je ukázka použití Blade templating enginu v rámci této bakalářské práce.

```
<h4>Members</h4>
<div class="member-dashboard">
  @foreach ($post->getUsers() as $user)
    <div class="">
      @if (strlen($user->image) == 0)
        <img src="/img/user-image.png" alt="">
      @else
        <a href="#"><img src="{{ $user->getImage() }}" alt=""></a>
      @endif
      <h5>
        {{ $user->name }} <br>
        @if ($user->isOnline())
          Online
        @else
          Offline
        @endif
        | {{ $user->getRole($post->id) }}
      </h5>
    </div>
  @endforeach
  <div class="add-member">
    <a href="/user/browse">
      <div class="add-member-icon">+</div>
      <h5>Add a new member</h5>
    </a>
  </div>
</div>
```

Obrázek 4.1: Ukázka Blade templating enginu v rámci webové platformy

## 4.3 Rozložení projektu

Aplikační rámec Laravel rozděluje svou souborovou strukturu na několik hlavních podsekcí rozdělených dle úlohy, kterou v projektu plní. V základu lze tyto sekce rozdělit do čtyř hlavních skupin. Tyto skupiny jsou soubory spravující aplikační část webu, které jsou mezi sebou koordinovány návrhem architektury MVC [9]. Druhou částí jsou soubory balíčků a přídavných funkcí webu. Mezi tyto balíčky patří například přídavné funkce zajištěné s pomocí nástroje na správce závislostí Composer a balíčky systému pro psaní webových aplikací Node.js. Třetí sekci tvoří soubory přídavných tříd, které poskytují možnost urychlení vývoje a modulování celého programovacího procesu. Poslední sekce je tvořena soubory, které jsou určeny pro změny v nastavení aplikace, a přídavnými funkcemi např. pro správu souborového systému aj. [9] Kořenový adresář projektu je zobrazen na obrázku 4.2.




Obrázek 4.2: Rozložení projektu v rámci aplikačního rámce Laravel

## 4.4 Souborová struktura Laravelu

Kořenová adresářová struktura aplikačního rámce Laravel rozděluje své zdrojové kódy na několik hlavních úseků. Soubory jsou děleny do jednotlivých podsložek, které obsahují kód k zajištění jednotlivých úseků aplikační logiky [9].

Složka „app“ obsahuje základní aplikační logiku, jednotlivé modely, řadiče, notifikace a middleware, který slouží např. k ověření aktuálního přihlášení jednotlivých uživatelů [9].

Složka „bootstrap“ slouží k přechovávání kódu, který aplikaci načítá. Přechovává taky dočasně uložené proměnné a soubory pro zlepšení optimalizace stránky ukládáním repetitivně používaných zdrojů [9].

Složka „config“ obsahuje soubory upřesňující nastavení aplikace a nastavení jednotlivých aplikačních modulů [9]. Kromě nastavení jednotlivých částí aplikace má Laravel také množství dalších sekcí kódu zabývajících se změnou chování balíčků, objektů a modulů aplikace. Složka config nicméně obsahuje všechna generální nastavení, která nemusí být detailněji specifikována používanou třídou objektu.

Složka „database“ obsahuje jednotlivé tabulky aplikační databázové vrstvy a soubory pro doplnění dat, která jsou definována modely, do aplikační databáze (ekvivalent série SQL příkazů INSERT INTO TABLE) [9].

Složka „public“ obsahuje všechny veřejně přístupné aplikační zdroje, jako jsou kaskádové styly, fonty, obrázky, použité skripty a jiné zdroje [9].

Složka „resources“ obsahuje soubory, které musí být před vložením do stránky překompilované do standardního CSS nebo JavaScriptu. Standardně jde o Sass, Less nebo Stylus soubory v případě CSS a v případě klientských skriptů soubory JavaScriptových knihoven React, Vue.js nebo Angular [9]. Po překompilování jsou nativní CSS, popř. JavaScript verze těchto souborů vloženy do dříve zmíněné složky „public,“ kde jsou veřejně přístupné uživatelům stránky.

Složka „routes“ přechovává definice aplikačních cest URL a volání odpovědí řadičů obsahujících požadované reakce na volanou cestu (získání požadovaného pohledu a s ním spojených dat, změny v databázi aj.) [9].

Složka „storage“ zodpovídá za přechovávání souborů generovaných aplikačním rámcem Laravel, samotnou aplikací a aplikačními log soubory [9]. V praxi se tento adresář používá pro přechovávání uživatelských souborů, jako jsou např. profilové obrázky, soubory uživatelského fotoalba, obrázky sloužící k ilustraci uživatelského profilu (jako jsou například obrázky na pozadí nebo v jiných částech profilu) apod.

Složka „tests“ obsahuje aplikační testy [9]. Aplikační rámec Laravel poskytuje vývojáři ve svém základním balíčku nástroje pro vytváření PHP testů PHPUnit, které mohou být použity k vytváření jednoduchých unit testů pro kontrolu funkcionalit jednotlivých částí aplikace.

Ostatní soubory v kořenovém adresáři slouží k základnímu nastavení aplikace, zajištění použitelnosti defaultních správcovských příkazů s pomocí nativního Laravel systému php artisan[21], správu Node.js a Composer balíčků, spouštění testů, překladu klientských skriptů, kaskádových stylů a README souboru obsahujícího popisek pro git repozitář[22].

## 4.5 Rozložení pohledů

Jednotlivé pohledy v platformě jsou rozděleny do dvou hlavní sekcí. První sekci představují pohledy, které jsou veřejně dostupné všem uživatelům stránky bez potřeby registrace a vytváření uživatelských účtů. Druhou sekci představují pohledy definující prezentační část webu pro registrované uživatele. Tato sekce obsahuje také hlavní aplikační sekci umožňující interakci se všemi hlavními funkcemi platformy registrovaným uživatelům. Mezi tyto funkce patří vytváření a správa projektů, vyhledávání a přizvání jiných uživatelů do již existujících projektů, využívání systému pro zasílání zpráv nebo hodnocení aktivních projektů systémem přidávání komentářů. Každá sekce pohledů používá svůj vlastní individuálně specifikovaný balíček JavaScriptových knihoven a kaskádových stylů, každý jednotlivý pohled má specifikovaný svůj vlastní kontejnerový id atribut, který je totožný s id atributem kontejneru v jednotlivých kaskádových stylech. Id atribut umožňuje organizování kódu CSS do individuálních sekcí patřících specifickým pohledům.

[21] PHP Artisan: https://laravel.com/docs/5.8/artisan
[22] Git repozitář: https://git-scm.com/book/cs/v1/Z%C3%A1klady-pr%C3%A1ce-se-syst%C3%A9mem-Git-Z%C3%ADsk%C3%A1n%C3%AD-repozit%C3%A1%C5%99e-Git

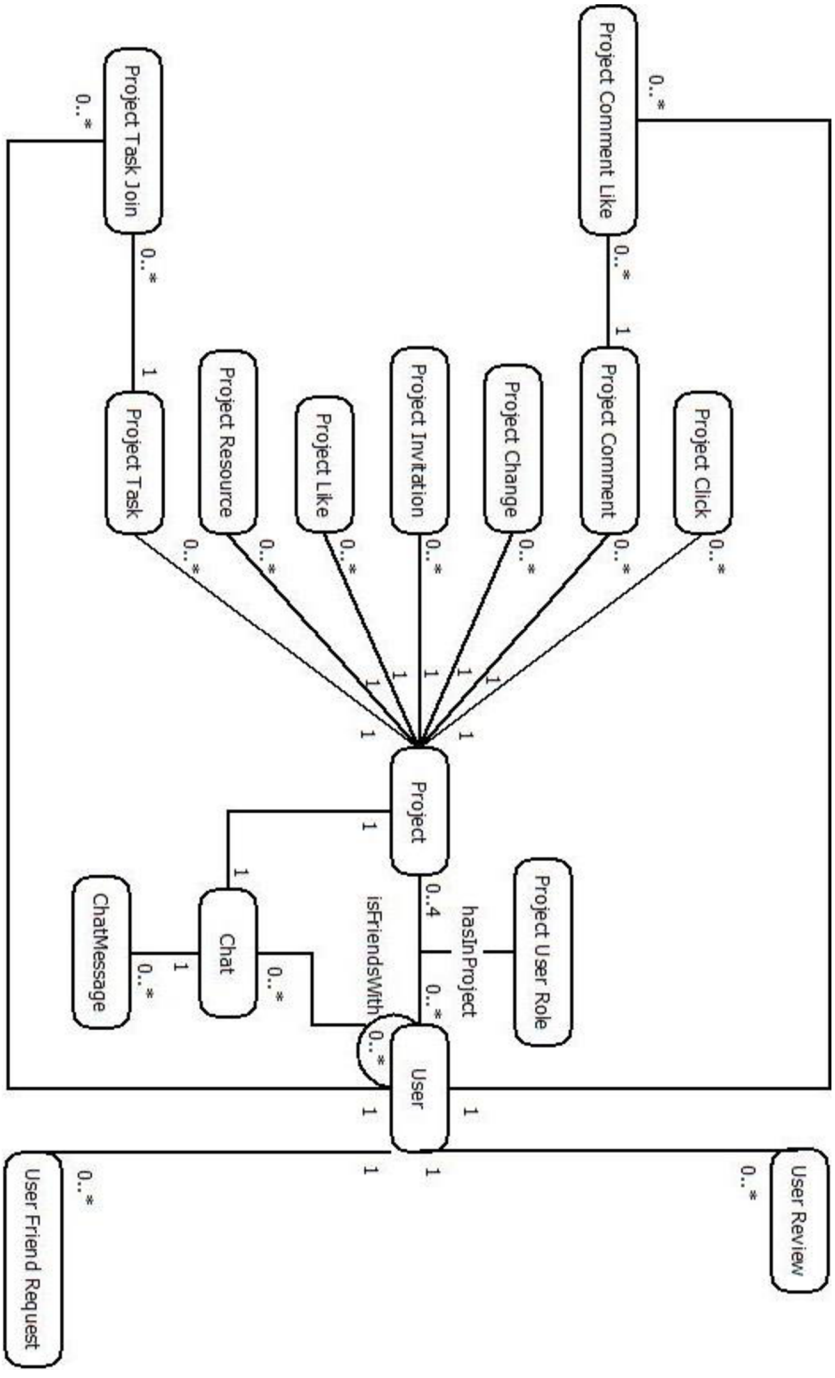
Jednotlivé sekce pohledů jsou dále rozděleny na hlavní layout, pohledy jednotlivých webových cest a pohledy komponentů importovaných volitelně do ostatních pohledů aplikace pro zajištění snadného znovupoužití kódu. Layout pohledy slouží jako šablonová vrstva obsahující import knihoven a nastavení meta dat v hlavičkách obou hlavních webových sekcí. Také slouží k zobrazování dodatečného JavaScriptového kódu ve spodní části pohledu a k další specifikaci dokumentového objektového modelu. Sekční pohledy poté určují strukturu jednotlivých stránek, rozložení elementů a použité klientské skripty. Sekční pohledy si dále mohou volitelně naimportovat další komponenty HTML kódu, čímž dochází ke zredukování množství kódu v projektu a opakované webové komponenty se stávají recyklovatelnými [7]. Ukázka strukturování jednotlivých pohledů v rámci webové platformy je zobrazena na obrázku 4.3.

```
@extends('layouts.app')

@inject('user', 'App\User')

@section('content')

  <div id="project-dashboard">

    <div class="container">
      <div class="row">
        <div class="col-xs-12 col-sm-8 col-sm-offset-2 text-center">
          @include('app.components._alert')
        </div>
        <div class="col-xs-12 col-sm-8">
          @include('app.components.project._tasks')
        </div>
        <div class="col-xs-12 col-sm-4">
          @include('app.components.project._leftmenu')
        </div>
      </div>
    </div>

  </div>

@endsection

```

Obrázek 4.3: Ukázka strukturování pohledů v rámci webové platformy

## 4.6 ER diagram

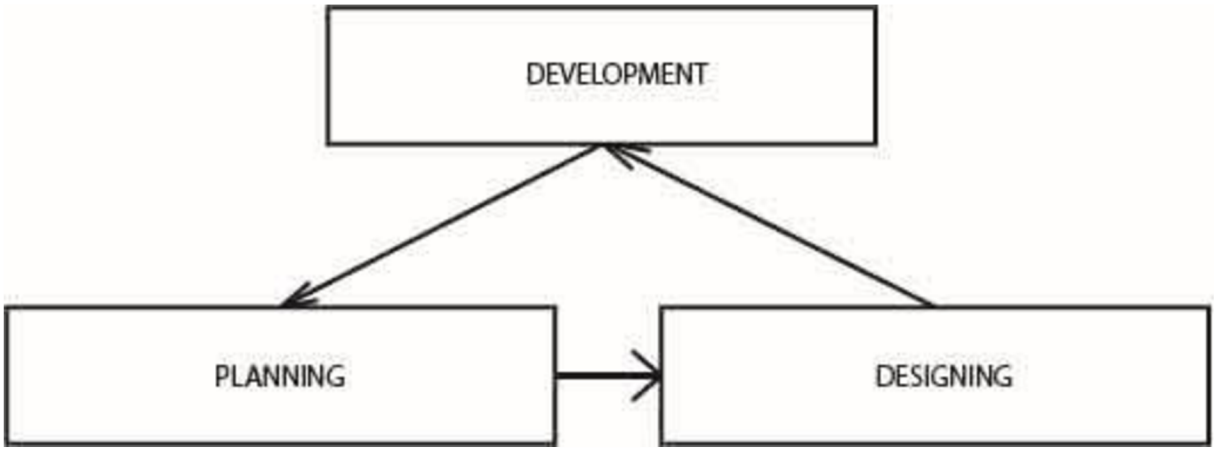
Z důvodu komplexnosti aplikace jako celku bylo potřeba během vývoje udržovat informace o vztazích mezi jednotlivými třídami platformy. Tyto informace bylo zapotřebí udržovat v rámci implementace v ER diagramu, a to především kvůli zachování konzistence databáze a kvůli korektnímu definování vztahů mezi jednotlivými aplikačními modely. Výsledný ER diagram webové platformy pro spolupráci aktivistů je zobrazen na obrázku 4.4.




Obrázek 4.4: ER diagram webové platformy pro spolupráci aktivistů

# 5 Implementace Systému

Celá aplikace byla navržena za použití metodiky vývoje softwaru Rapid application development. Jednotlivé vlastnosti stránky byly postupně programovány po funkcích, které byly následně testovány na iterace. Po ukončení první iterační vlny a zprovoznění všech základních funkcí stránky byl k aplikaci přizván profesionální grafický designér a uživatelský tester, se kterými bylo provedeno ohodnocení stránky a prvotní uživatelské testování. Po ukončení prvotního testování probíhal další vývoj platformy v agilních iteračních kruzích. V první řadě došlo k navržení jednotlivé sekce nebo pohledu aplikace a následně byly vytvořeny grafické podklady, s pomocí kterých byl vytvořen hlavní kódový návrh aplikace. Po ukončení vývoje byla sekce v další vývojové iteraci upravena za použití stejného postupu nebo se přešlo na vývoj další funkce, popř. pohledu platformy (viz. obrázek 5.1). Na konci vývoje platformy bylo provedeno poslední rozsáhlejší uživatelské testování, po kterém byly provedeny finální úpravy aplikace a následné uvedení do produkce.




Obrázek 5.1: Diagram postupu při implementaci webové platformy

## 5.1 Použité technologie

Celá platforma byla naprogramována v jednom z nejpoužívanějších serverových skriptovacích jazyků PHP. Tento jazyk byl vybrán hlavně z důvodu kompatibility s většinou serverů, které se v dnešní době používají pro hostování webových aplikací. Kromě toho je v jazyce PHP naprogramováno velké množství aplikačních rámců pro programování webových aplikací, které výrazně zjednodušují celý vývojářský proces. Mezi tyto nástroje patří také Laravel, které je v současnosti nejen jedním z nejpoužívanějších webových aplikačních rámců [5], ale také je to jeden z aplikačních rámců, které jsou nejjednodušší z hlediska používání vývojářem. Platforma používá pro svůj vývoj databázový systém MySQL z důvodu velké rozšířenosti a snadné manipulaci s daty. Aplikace používá místo klasických kaskádových stylů preprocesor SASS, který má jednodušší syntaxi a umožňuje používání širšího množství funkcí. Z důvodu komercializace pokročilejších nástrojů se pro podporu komponent fungujících v reálném čase používá pouze standardní nástroj Ajax.

Pro používání a správu některých z výše zmíněných technologií bylo potřeba použití dodatečných balíčkovacích nástrojů. Mezi tyto balíčkovací nástroje patří balíčkovací systém NPM určený pro překlad SASS souborů do standardních kaskádových stylů a systém Composer, který je implicitní součástí aplikačního rámce pro programování webových aplikací Laravel.

### 5.1.1 PHP

V současnosti nejpoužívanějším programátorským jazykem [10], který se používá pro serverový vývoj webových aplikací, je skriptovací jazyk PHP. Stejně jako ostatní skriptovací jazyky, i jazyk PHP může být použit lokálně pro vytváření aplikací bez nutnosti připojení k webu. V drtivé většině případů se ale používá pro vytváření webů, kde umožňuje buďto vytváření samostatný skriptů, které jsou externě importovány do jednotlivých pohledů, nebo tvoří součást pohledů, kde může být jazyk volně kombinován s klasickým značkovacím jazykem HTML [10].

Hlavní výhodou používání jazyka PHP pro implementaci webových aplikací je jeho rozšířenost a jednoduchá použitelnost v širokém prostředí hostujících serverů. Jazyk PHP je velmi nenáročný [10] a je implementačně přizpůsoben pro funkčnost na nejpoužívanějších linuxových a Apache[23] serverech. PHP má velmi rychlou učební křivku, přehledné metodiky snadné k pochopení, dobrou přehlednost a rozsáhlou komunitu vývojářů, kteří poskytují novým programátorům dodatečnou výpomoc při řešení implementačních problémů. PHP je nejpoužívanějším programátorským jazykem pro vytváření webových aplikací a v současnosti tvoří zhruba 80% celého webového prostředí [10]. Mezi nejznámější webové aplikace naprogramované v jazyce PHP patří například sociální síť Facebook, internetová encyklopedie Wikipedia.org[24], čínský webový prohlížeč Baidu.com[25] nebo čínský instant-messaging-systém Qq.com[26]. [10]

Mezi nevýhody jazyka PHP patří příliš velká volnost při implementaci jednotlivých jazykových modulů a nestálost použitých praktik, což vede k velkým rozdílům mezi jednotlivými iteracemi jazykových verzí a následně k problémům při převádění již existujících webových aplikací na novější jazykové verze [10]. Často se také vyskytují problémy způsobené neefektivním používáním hardwarových zdrojů, které bývají často lépe vyřešeny například v aplikačních rámcích ASP.NET nebo v aplikačních rámcích programovacího jazyka Java [10].

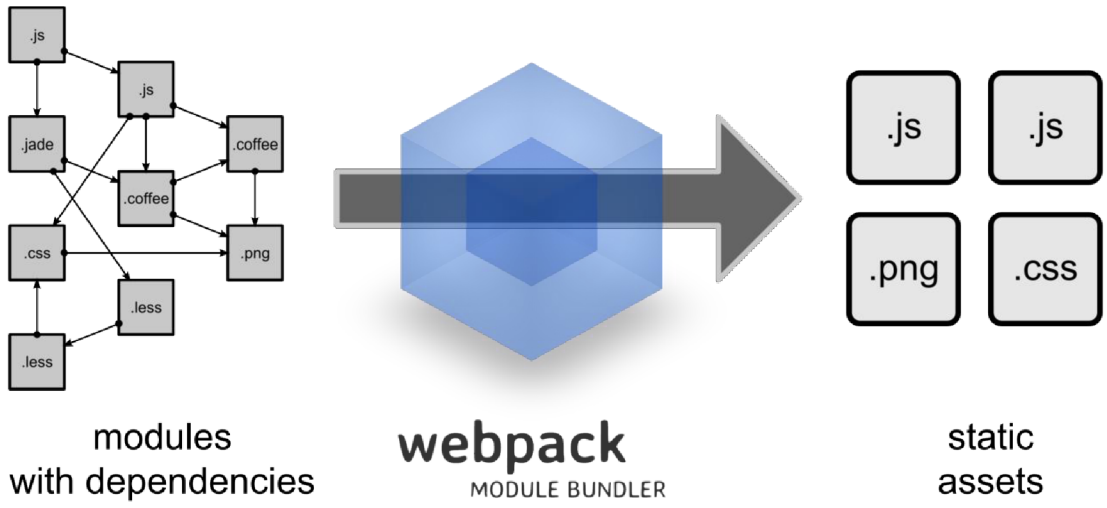
[23] Apache server: https://en.wikipedia.org/wiki/Apache_HTTP_Server
[24] Wikipedia: https://www.wikipedia.org/
[25] Baidu.com: https://www.baidu.com/
[26] Qq.com: https://www.qq.com/?fromdefault

### 5.1.2 MySQL

Webová platforma pro spolupráci aktivistů používá pro přechovávání dat databázový systém MySQL založený na jazyku SQL. MySQL je centrální komponent skupiny webových aplikací LAMP[27] a je i součástí jiných AMP balíčků (pro vytváření webové platformy byl použit nástrojový stack XAMPP[28]) [18]. Pro vytváření a úpravy dat v databázových tabulkách používá Laravel svůj vlastní systém modelů schémat a svůj vlastní nástroj Query Builder[29] pro přístup a úpravy jednotlivých databázových dat.

### 5.1.3 Webpack

V současnosti většina moderních technologií překračuje technologické hranice moderních webových prohlížečů, a proto se pro používání velkého množství technologií používají kompilační a balíčkovací systémy, které překládají jednotlivé zdrojové soubory nepodporovaných programovacích jazyků a nástrojů do podoby, ve které s nimi prohlížeče mohou pracovat (viz obrázek 5.2). Mezi nejpoužívanější technologie pro překládání a spojování jednotlivých balíčků se v současnosti řadí technologie Gulp a Webpack, který je použit i k přeložení SASSu na platformě pro spolupráci aktivistů. Kromě překompilování jednotlivých nástrojů obsahuje Webpack také nástroje pro minimalizaci a jednodušší vývojový proces [19]. Mezi tyto nástroje se řadí například specializované překladače s možností vlastních chybových hlášení a lokální developerské servery, které automaticky restartují vyvíjenou webovou aplikaci při změně v souborovém systému projektu (změny kódu, projektových zdrojů aj.).




Obrázek 5.2: Diagram kompilace aplikačních modulů nástrojem Webpack

převzatý z [19]

[27] LAMP: https://en.wikipedia.org/wiki/LAMP_(software_bundle)
[28] XAMPP: https://www.apachefriends.org/index.html
[29] Query Builder: https://laravel.com/docs/5.8/queries

### 5.1.4 SASS

Platforma používá pro definování svých kaskádových stylů preprocesor SASS. SASS je skriptovací jazyk, který je s pomocí balíčkových kompilačních systémů překládán do kaskádových stylů. Skriptovací jazyk SASS obsahuje množinu použitelných syntaktických variací. Mezi tyto variace patří jazyk SCSS, SASS a Stylus. Jazyk SCSS je definovaný používáním podobné blokové struktury, jako kaskádové styly, nicméně je obohacen o velké množství dodatečných funkcí umožňující přesnější a dynamičtější vytváření CSS. Jazyk SASS, ve kterém jsou nadefinovány kaskádové styly aktivistické platformy, je vyznačen hlavně používáním prázdných znaků pro definování kódových bloků (viz. obrázek 5.3). Posledním jazykem SassScriptů je jazyk Stylus, který velmi zjednodušuje použitou syntaxi a umožňuje tak ještě větší přehlednost celku a jednoduchost kódu. Všechny jazyky SassScriptu se vyznačují především přídavnými funkcemi, které poskytují webovým vývojářům možnost použít proměnné, mixiny (znovupoužitelných úseků kódu), zanořování a dědičnosti.

```
#user-browseusers
  padding-bottom: 40px
  h2
    margin-top: 3vh
  .breadcrumbs
    margin-top: 2vh
  .input-group
    margin-bottom: 10px
    input
      border: 1px solid $eco-green
      z-index: 0
    button
      background: $eco-green
      border: 1px solid $eco-green
      z-index: 0
      color: white
      font-weight: bold
  .dropdown
    margin-bottom: 10px
    cursor: pointer
  .advanced-search-dropdown
    .dropdown-menu
      width: 100%
      p
        padding: 12px
        float: left
        input
          margin: 2px
```

Obrázek 5.3: Ukázka preprocesoru kaskádových stylů SASS

#### 5.1.4.1 OOCSS

Platforma pro uspořádání svých kaskádových stylů používá postup objektově orientovaného programování CSS. OOCSS je koncept psaní kódu založený na principu komponentového psaní CSS [8]. OOCSS uzavírá skupiny elementů do uzavřených objektů, které jsou charakterizovány sjednoceným seskupením několika HTML elementů množinou CSS selektorů upravujících tyto HTML elementy. Objektově orientované psaní CSS zajišťuje větší recyklovatelnost, přehlednost a udržovatelnost CSS kódu. Hlavní výhodou OOCSS je nezávislost CSS na hierarchii elementů v HTML [8].

Celý koncept OOCSS je postaven na třech základních principech. Prvním principem OCSS je již zmíněná nezávislost na hierarchii [8]. V programátorské praxi se často vyskytuje zvyšování nepřehlednosti kódu kvůli použití CSS závislého na hierarchii. Problematické je také použití různé hierarchie HTML znaků mezi použitými selektory. Ve většině případů je poté programátor nucen pracovat s nepřehledným kódem, popř. dochází k vytváření dalších selektorů pro úpravy již upravovaných elementů, což vede k velkým časovým ztrátám a vystupňování nepřehlednosti kódu.

Druhým principem OOCSS je snaha o co největší možné snížení specifičnosti jednotlivých CSS objektů, resp. komponentů, pro zajištění znovupoužití jednotlivých komponent v jednotlivých částech aplikace [8]. Objektově orientované programování pro zajištění principu snížení specifičnosti zakazuje používání selektorů potomka a kombinovaných selektorů.

Třetí princip OOCSS spočívá v rozložení jednotlivých komponent aplikace na tři základní typy objektu, elementu a modifikátoru [8]. Objekt přitom představuje samostatně uzavíraný kořenový element celého komponentu. Element představuje jednotlivé subjekty kořenového elementu se samostatným stylem a modifikátor představuje optimální dodatečné úpravy kořenového elementu nebo jeho subjektů [8].

**Ukázka SASS OOCSS na komponentu veřejné projektové karty (obrázek 5.4):**

```
.project
  margin-top: 1vh
  background-color: white
  padding-bottom: 0.5vh
  word-wrap: break-word
  box-shadow: 0px 5px 5px rgba(0, 0, 0, 0.3)
  &-image
    width: 100%
    height: 200px
    background-image: url('/img/user_bck.jpg')
    background-size: cover
    background-position: center
  &-imageicon
    float: right
    margin: 8px
    padding: 2px 8px
    color: white
    background-color: $eco-blue
    border-radius: 3px
    cursor: pointer
  &-header
    margin: 0
    padding: 12px 0px
    background-color: $eco-blue
    box-shadow: 0px 5px 5px rgba(0, 0, 0, 0.3)
    color: white
    text-align: center
    font-size: 18px
  &-section
    padding: 24px
```

Obrázek 5.4: Ukázka objektově orientovaných kaskádových stylů

### 5.1.5 Ajax

V moderních webových platformách nastává častá nutnost používání dynamických komponent fungujících v reálném čase. Tyto komponenty se používají k vytváření uživatelských upozornění, varování, zasílání zpráv apod. Na tyto komponenty nelze používat technologie, které jsou již dostupné na klasické webové stránce, protože zaslání požadavku na stránku obvykle vede k načtení nového pohledu, popř. znovunačtení aktuální stránky. Pro zajištění funkcí v reálném čase se v platformě pro spolupráci aktivistů používá nástroj Ajax.

Ajax je sám o sobě programátorskou JavaScriptovou technikou pro získávání serverových dat bez znovunačtení pohledu a umožňuje stránce reagovat na libovolnou uživatelskou aktivitu zasláním webového požadavku serverové straně aplikace. Serverová strana aplikace ho poté zpracovává standardním způsobem a vrací odpověď v libovolném formátu specifikovaném v serverové části aplikace. Ukázka použití Ajaxu v rámci webové platformy pro spolupráci aktivistů je zobrazena na obrázku 5.5.

```
var arr = [];
var newArr = [];
function requestNotifications() {
  var userid = <?php echo json_encode(Auth::user()->id); ?>;
  $.ajax({
    type: "GET",
    url: "/user/"+userid+"/notifications",
    success: function(data) {
      newArr = data;
      if (newArr.length > arr.length) {
        arr = data;
        // reset the container
        var container = document.getElementById("home-app-notifications")
        container.innerHTML = '';
        // create new notifications
        for (var i = 0; i < newArr.length; i++) {
          newNotification(newArr[i].data[1], newArr[i].id);
        }
        $("#home-app-notifications-counter").text(newArr.length);
      }
    },
    error: function() {
      console.log("notification database request failed");
    }
  });
}
```

Obrázek 5.5: Ukázka používání nástroje Ajax pro získávání uživatelských upozornění v rámci webové platformy pro spolupráci aktivistů

# 6 Testování

## 6.1.1 Průběžné testování aplikace

V rámci vývoje aplikace bylo po dokončení implementace jednotlivých částí projektu průběžně provedeno několik testovacích iterací. Hlavními složkami testování byla hlavně práce s projektovým rozhraním, používání aplikačního chatovacího systému a operace nad uživatelskými profily a jejich správou [20]. Kromě serverových částí aplikace byly také otestovány složky přední části aplikace – v přední klientské části aplikace byly otestovány hlavně možnosti responzivního zobrazování na mobilních zařízeních, snadná orientace v rámci celé aplikace a celkový pocit z aplikačního prostředí [20].

## 6.1.2 Testování projektového rozhraní

V rámci charakteru stránky byl kladen dodatečný důraz na testování projektového rozhraní. Hlavními aspekty testování bylo ověřování celkové přehlednosti, jednoduchosti používání a základních funkcí rozhraní uživatelskými testery. Před samotným uživatelským testováním[30] bylo prvně znovu otestováno, že všechny základní funkce rozhraní fungují korektně a nedochází k žádným neočekávaným chováním a chybám. V první fázi kromě ručního otestování funkčnosti všech hlavních částí projektového rozhraní došlo také k otestování responzivního designu stránky na mobilních zařízeních a jednotlivých způsobů ochrany projektového rozhraní před nepovoleným přístupem uživateli, kteří nemají dostatečná oprávnění pro provádění jednotlivých úkonů nad projektem nebo nejsou členové projektu [20].

**Mezi hlavní funkce rozhraní, které byly předmětem testování, byly zejména:**

- Přidání a odebírání projektových úkolů
- Přidávání uživatelů k jednotlivým úkolům
- Korektní zobrazování projektových úkolů v rámci rozhraní
- Označování úkolů pro ohodnocení a zkompletování
- Korektní online zobrazování uživatelů v projektovém menu
- Přidávání a odebírání alternativních projektových zdrojů
- Korektní výpis všech uživatelů projektu
- Přidání a odebírání nových členů do projektu a změna práv uživatelů v rámci projektu

[30] *Google Developers. In:* Youtube *[online]. 22.06.2015 [cit. 2017-10-15]. Dostupné z:* https://www.youtube.com/watch?v=0YL0xoSmyZI *Kanál uživatele Google Developers*

Po ověření výše zmíněných základních funkcí s pomocí ručního testování přešlo testování rozhraní do fáze uživatelského testování, které bylo zaměřeno hlavně na rychlost a přehlednost používání jednotlivých funkcí. Hlavním hodnotícím aspektem testování byla přitom rychlost zpracování jednotlivých předpřipravených úkonů a uživatelský komentář. Komentář byl obdržen s pomocí jednoduchého předpřipraveného testovacího protokolu, který byl obdržen uživatelskými testery v rámci testování [20].

V první části uživatelského testování došlo k opětovnému ověření základních funkcí projektového rozhraní, v druhé fázi testování uživatelé v rámci celé aplikace zpracovávali komplexnější sety úkonů, které na sebe navazovaly [20].

**Mezi hlavní úkony v rámci uživatelského testování patřilo např.:**

- Přidání předpřipraveného setu alternativních odkazů do projektu a jejich následná úprava
- Přidávání předpřipraveného setu úkolů do projektu a jejich označování jako „připravené k ohodnocení“ a „kompletní“
- Změny práv ostatních uživatelů v rámci projektu
- Přidávání komentářů ve veřejném zobrazení projektu

Klíčové prvky byly hlavně sekce projektového rozhraní s jednotlivými projektovými úkoly, přidanými zdroji a správou uživatelů, nicméně charakter těchto funkcí a projektového rozhraní jako celku nevyžadoval vytvoření alternativních způsobů testování pro podrobnější ověření, že všechny klíčové funkce pracují korektně. Funkce správy projektových úkolů, uživatelů a alternativních odkazů byly otestovány stejným způsobem, jako zbytek projektového rozhraní [20].

Výslednými daty testování byly rychlosti zpracování jednotlivých úkonů a komentáře uživatelů k vypracovávání těchto úkonů, které napomohly při optimalizování uživatelského prostředí projektového rozhraní. Hlavním přínosem byla možnost opravení často zmiňovaných nedostatků projektového rozhraní a jejich opětovné testování v rámci další testovací iterace. Komentáře uživatelů pomohly v upravování designu projektu jako celku a v jeho změnách pro zachování co pokud možno nejjednoduššího používání a přehlednosti vůči uživateli [20].

### 6.1.3 Závěr testování

Obě fáze testování proběhly úspěšně, uživatelští testeři dosahovali dobrých časů při splňování přednastavených úkolů v rámci celé aplikace. Uživatelům se líbila zejména jednoduchost snadné používání aplikace. Mezi další pozitivní názory uživatelů patřil například kvalitní design a „dobrý pocit“ z používání aplikace [20]. Mezi často zmiňované nedostatky patřily časté používání ikon s nepřímo jasnou funkcionalitou a absence některých funkcí, které jsou běžné na jiných stránkách. Mezi tyto funkce patřily například filtry proti vulgarismům [20]. Testování bylo po naměření časových hodnot a po převážně pozitivním hodnocení testerů vyhodnoceno jako velmi úspěšné a aplikace byla připravena na převod do produkce [20].

# 7 Závěr

Webová platforma pro spolupráci aktivistů Eco helper byla naprogramována v celém plánovaném rozsahu, s přidáním dodatečných funkcí během implementace pro zajištění širších možností používání aplikace jako celku. Vytvoření aplikace trvalo celkově čtyři měsíce a zabralo kolem pěti set hodin čisté programátorské práce. Aplikace byla vytvořena za použití doporučených praktik pro psaní kódu v rámci modelu MVP, doporučeného návrhu architektury pro psaní kódu HTML a s pomocí objektově orientovaných kaskádových stylů pro zajištění snadné expanzivnosti a udržovatelnosti aplikace. Vývoj aplikace byl završen uživatelským testováním, které dosáhlo velmi dobrých výsledků.

Vývoj platformy pro spolupráci aktivistů byl zdaleka nejkomplexnější projekt, který jsem prozatím jako programátor vytvořil a kromě základních programátorských praktik mě studium jednotlivých teoretických aspektů při vypracovávání této bakalářské práce naučilo řadu sekundárních programátorských praktik, které mě posunuly na vyšší úroveň programování a navýšily mé znalosti v oboru vývoje webových aplikací. Kromě toho mě rozsah aplikace donutil vyhledat nové, výhodnější praktiky pro psaní rozsáhlého, udržovatelného kódu, a také rozšířil mé znalosti o používání nových nástrojů, jako jsou databázové seedery, aplikace pro správu databáze a nástroje pro vytváření komponent fungujících v reálném čase.

Projekt v současnosti čeká nahrání do produkce a následná etapa vylepšování v rámci komunikace s aktivními uživateli. Projekt bude uveden do produkce po vytvoření stránek na sociálních sítích a příspěvkových účtů, aby bylo možné získat příspěvky od jednotlivých uživatelů pro snadnější udržovatelnost aplikace. Očekávaná doba nahrání projektu do produkce je listopad 2018 až leden 2019. Očekává se, že v případě kvalitního provedení marketingové kampaně bude mít aplikace relativně velký pozitivní vliv a v případě velkého úspěchu bude propojena s dalšími platformami a firmami, které se zabývají ekologickou problematikou.

Platforma bude dále vyvíjena na produkci. V současné době je již naplánována prvotní várka změn, které budou v aplikaci provedeny. Mezi tyto změny patří hlavně přepracování ochranných prvků pro neoprávněný přístup v rámci aplikačních pohledů, přidání jednoduchého systému správy pro manipulaci s vulgárními a jinými uživateli, kteří porušují zásady užívání aplikace, ladění optimalizace pro vyhledávací nástroje a potenciální přidání správcovského systému pro snazší správu aplikace. Hlavní náplň dodatečných změn určí zpětný ohlas aktivních uživatelů po nahrání platformy na živou produkci.

# Literatura


[1] Rábová, Z., Hanáček, P., Peringer, P., Přikryl, P., Křena, B.: Užitečné rady pro psaní odborného textu [online]. Brno: c1993-2008, aktualizováno 2008-11-01 [cit. 2008-11-28]. Dostupné na URL: <http://www.fit.vutbr.cz/info/statnice/psani_textu.html>

[2] Kolektiv autorů: *Pravidla českého pravopisu*. Academia, Praha, 1993. ISBN 80-200-0475-0

[3] Model 1 Architecture. *Java samples - Programming tutorials on Java, C, C++, PHP, ASP, J2ME, Struts, Hibernate, VB.net, JSP, JSP, Javascript* [online]. Dostupné z: http://www.java-samples.com/showtutorial.php?tutorialid=349

[4] Model 1 and Model 2 (MVC) Architecture - javatpoint. *Tutorials - Javatpoint* [online]. Copyright © Copyright 2011 [cit. 19.10.2017]. Dostupné z: https://www.javatpoint.com/model-1-and-model-2-mvc-architecture

[5] Web framework rankings | HotFrameworks. *Web framework rankings | HotFrameworks* [online]. Dostupné z: https://hotframeworks.com/

[6] Model-view-controller – Wikipedie. [online]. Dostupné z: https://cs.wikipedia.org/wiki/Model-view-controller

[7] Blade Templates - Laravel - The PHP Framework For Web Artisans. *Laravel - The PHP Framework For Web Artisans* [online]. Copyright © Taylor Otwell. [cit. 19.10.2017]. Dostupné z: https://laravel.com/docs/5.5/blade

[8] OOCSS: objektové psaní CSS. *Vzhůru dolů — webový frontend ze všech stran* [online]. Dostupné z: https://www.vzhurudolu.cz/prirucka/oocss

[9] Laravel - The PHP Framework For Web Artisans. *Laravel - The PHP Framework For Web Artisans* [online]. Copyright © Taylor Otwell. [cit. 19.10.2017]. Dostupné z: https://laravel.com/docs/5.5/structure#the-bootstrap-directory

[10] PHP - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/PHP

[11] ASP.NET - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/ASP.NET

[12] Laravel vs Codeigniter (2017 update) - YouTube. *YouTube* [online]. Dostupné z: https://www.youtube.com/watch?v=U3JWFgTjq94

[13] The PHP Duel: Symfony vs. Laravel | Toptal. *Toptal - Hire Freelance Talent from the Top 3%* [online]. Copyright © Copyright 2010 [cit. 19.10.2017]. Dostupné z: https://www.toptal.com/php/choosing-between-symfony-and-laravel-frameworks

[14] WordPress - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/WordPress

[15] 302 Found. *302 Found* [online]. Dostupné z: https://blog.ikw.cz/laravel-vs-nette-t%C5%99i-m%C4%9Bs%C3%ADce-pot%C3%A9-78f1200497c9

[16] npm. *npm* [online]. Dostupné z: https://www.npmjs.com/

[17] Introduction - Composer. *Composer* [online]. Dostupné z: https://getcomposer.org/doc/00-intro.md

[18] MySQL - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/MySQL

[19] Webpack - Webpack. [online]. Dostupné z: https://webpack.github.io/assets/what-is-webpack.png

[20] Jiří, K., Technická dokumentace – VUT FIT [ITU] 2017.

[21] Oracle Mobile Cloud Service: Designed for Performance, Scalability and Productivity | A-Team Chronicles. [online]. Dostupné z: http://www.ateam-oracle.com/scalability-and-productivity-for-your-mobile-applications-with-mobile-cloud-solutions

# Příloha A

## Obsah CD

- Zdrojové soubory (source_files.zip)
- Popis ke spuštění a instalaci aplikaci (README.md)