FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Modulární webový informační systém pro drobné podnikatele
Student:	Michal Junek
Vedoucí:	Ing. Marek Suchánek
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je vyvinout modulární webový informační systém podporující vybrané klíčové oblasti činnosti drobných podnikatelů.

1. Zmapujte a popište hlavní oblasti činnosti drobných podnikatelů a jejich souvislosti. Při popisu použijte formální konceptuální modelování.
2. Proveďte stručnou rešerši současných řešení podporujících tyto činnosti.
3. Navrhněte vlastní řešení ve formě modulární webové aplikace, kde moduly budou podporovat jednotlivé vybrané dílčí činnosti drobných podnikatelů. Při návrhu zohledněte základní principy a koncepty Normalized Systems Theory.
4. Implementujte a následně otestujte navržené řešení v programovacím jazyce PHP (verze 7.0 či vyšší). Předem zdůvodněte výběr případných dalších technologií, knihoven a frameworků.
5. Zhodnoťte přínosy vlastního řešení z pohledu softwarového inženýrství i vlivu na podnikatelské činnosti. Ve zhodnocení se mimo jiné zaměřte na evolvabilitu řešení ve vztahu k návrhu i k možnostem jazyka PHP a dalších zvolených technologií.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 24. listopadu 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Modulární webový informační systém pro drobné podnikatele

Michal Junek

Katedra softwarového inženýrství
Vedoucí práce: Ing. Marek Suchánek

14. května 2018

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Marku Suchánkovi za cenné rady a odborný dohled. Dále bych rád poděkoval své rodině a přítelkyni za jejich trpělivost a pomoc při opravě gramatických chyb.

Prohlášení

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

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

V Praze dne 14. května 2018 .....

Odkaz na tuto práci

Junek, Michal. *Modulární webový informační systém pro drobné podnikatele.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Práce se zabývá tvorbou modulárního systému pro drobné podnikatele s jednoduchou správou a možností rozšíření o další funkce. Řešení je realizováno pomocí webového frameworku Laravel postaveném na programovacím jazyku PHP, spolu s HTML, CSS a JavaScriptem. Výsledkem práce je zdarma dostupný open-source systém, který lze jednoduše nastavit či rozšířit k osobním potřebám. Bude sloužit hlavně drobným podnikatelům pro lepší prezentaci sama sebe na internetu.

Klíčová slova redakční systém, podnikání, modulární, php, laravel

Abstract

Aim of this work is to develop a modular system with easy management and possibility to expand intended for small businesses. The project is realized using web framework Laravel build on programming language PHP, along with HTML, CSS and JavaScript. Result of this work is a free open-source system, which can be easily set up and expanded for personal needs. It mainly benefits small businesses for better self presentation on the internet.

Keywords redaction system, bussiness, modular, php, laravel

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V dnešní moderní době, kdy internet využívá drtivá většina populace, je jedním z důležitých kroků pro začínajícího podnikatele sebeprezentace formou webových stránek. Pro zákazníky to znamená jednodušší přístup k informacím a vzbuzuje to pocit pilné práce ze strany vlastníka firmy. Jednoduše se sdílí, což poslouží jako reklama za zlomek ceny. Tento krok ale nemusí být pro laika jednoduchý. Disponuje-li peněžními zdroji, často si najímá programátora pro vytvoření jednoduchého redakčního systému a nemusí se tedy sám o nic starat. Na druhé straně, pokud nechce nebo si nemůže dovolit investovat peníze navíc, má na výběr z nepřeberného množství předpřipravených řešení.

Tyto systémy, někdy také „redakční systémy“ nebo „systémy pro správu obsahu“, jsou vybírány nejčastěji podle vzhledu, plnění požadovaných funkcí a hlavně podle možností lokalizace cílové skupiny zákazníků. Je ale třeba se zaměřit i na nefunkční kritéria těchto systémů, a to hlavně na bezpečnost, požadavky na zdroje a jednoduchost správy. Začínající podnikatel tedy musí zvážit, který systém pro něj bude nejvhodnější.

Problémy populárních systémů jsou hlavně uživatelská nepřehlednost nebo velké množství možností nastavení, kterým uživatel nerozumí. Také často nemusí vědět, co ovlivňují. To vše může výběr zkomplikovat. Další překážkou se jeví instalace. I když obsahuje pouze pár jednoduchých kroků, snadno může odradit uživatele, kteří se v tom nedokážou orientovat i přes popisky a vysvětlivky.

Kapitola 1

Cíl práce

Cílem práce je vyvinout modulární webový informační systém podporující vybrané klíčové oblasti drobných podnikatelů.

Výstupem je systém dostatečně jednoduchý na to, aby mohl být nastaven a používán běžnými uživateli, laiky nebo například drobnými podnikateli, kteří se teprve dostávají na trh nebo chtějí rozšířit své působení, aby se mohli prezentovat online. Bude stavět na výhodách populárních redakčních systémů, jako je modularita, rychlost a přehlednost, a bude eliminovat jejich nevýhody.

Uživatel bude mít možnost si vybírat moduly obsažené ve webové stránce, jejich pozici a jednoduchou personalizaci. Součástí systému bude přehledná administrace pro správu celého webu bez nutnosti zasahovat do souborů. Instalace bude jednoduchá a detailně popsaná, aby byla pochopitelná pro co nejširší skupinu uživatelů.

Provozovatel tedy může poskytnout všechny důležité informace, jako je seznam produktů, kontakt, obchodní údaje a smluvní podmínky nebo blog s aktualitami. Celý web bude fungovat na principu modulů, které budou od sebe navzájem odděleny a budou vykonávat výše uvedené funkce. Jednotlivé části bude možné libovolně aktivovat, deaktivovat nebo přidávat na určité pozice v rozložení webu.

Díky rozsáhlé dokumentaci je možné přidávat další moduly a rozšiřovat tím funkčnost webu, což ale nutně není požadavek na vlastníka webových stránek a tato možnost je tedy primárně určena programátorovi.

Kapitola 2

Analýza

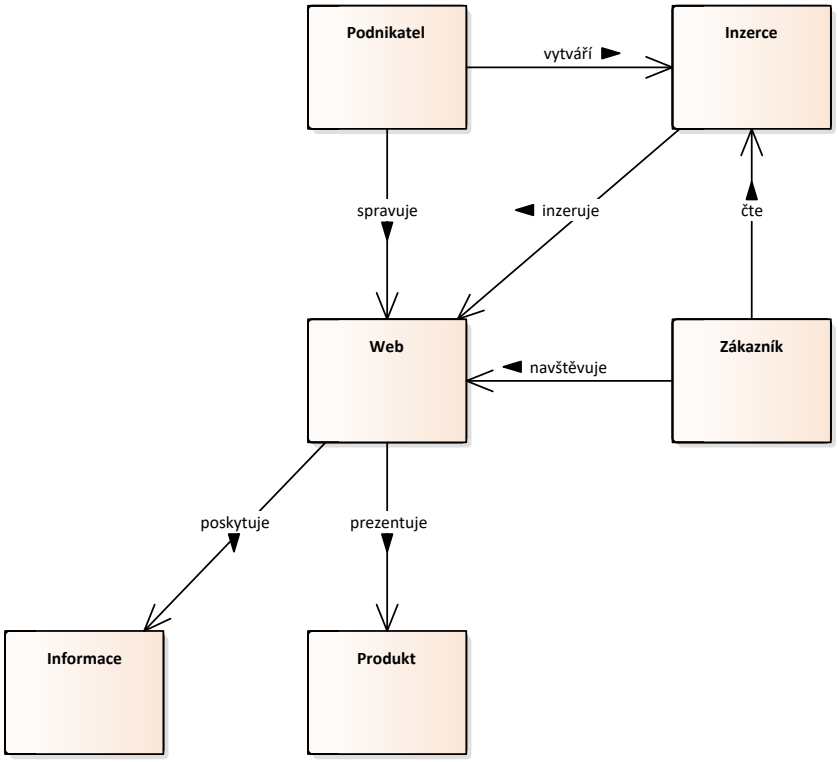
2.1 Problémová doména

Začít s podnikáním není nijak snadné. Nejdříve se musí člověk rozhodnout v jakém oboru bude podnikat. Často dělají chybu v tom, že se rozhodnou pro populární odvětví se spoustou již existujících podniků a pak se nedokáží prosadit, protože se jim nemusí dařit zaujmout potencionální zákazníky, popřípadě investory nebo budoucí zaměstnance.

Dále může pro ně být velkým problémem nedostatek financí. Pro začínajícího podnikatele to může znamenat velkou překážku. Nemůžou si např. dovolit plat pro případné zaměstnance, vylepšit technologii výroby nebo si zajistit reklamu, která jim přivede zákazníky. Marketingové firmy bývají mnohdy drahá záležitost.

V těchto situacích má začínající podnikatel tyto možnosti: buď si zkusí sehnat investora, nebo si půjčí od banky či od příbuzných, kamarádů atp. Pro reklamu může využít sociální sítě, což nebývá úplně dostačující.

Pro odstranění nedostatků reklamy pouze pomocí sociálních sítí je snadnou alternativou založení si webových stránek, které nabízejí mnohem víc možností, než sociální sítě. Pokud si začínající podnikatel neví rady, jak založit web, a nechce si jej nechat dělat úplně od základu, protože na to nemá prostředky, má možnost využít volně dostupné redakční systémy. Výběr vzhledu, typu systému a všech náležitostí s tím spojených se ale může jevit jako nezvládnutelný úkol. Proto stejně nejčastěji nezbývá, než si zaplatit programátora, který si s tím bude vědět rady. Toto řešení není už tolik finančně náročné, ale potencionální uživatelé určitě ocení možnost použití těchto systémů i bez nutnosti pomoci s instalací a nastavením.



Obrázek 2.1: Konceptuální model vztahu zákazníka a podnikatele

2.2 Současná řešení

V rámci této kapitoly jsem vybral několik nejpopulárnějších redakčních systémů na základě jejich způsobů fungování a skupiny uživatelů, na které jsou cíleny. Kritéria výběru byla především zaměřena na jednoduchost správy obsahu a instalace, rozšiřitelnost, používanost a osobní zkušenosti.

Hlavní motivací bylo se přiblížit začínajícímu podnikateli – laikovi – a zaměřit se na ty, které mají největší pravděpodobnost výběru. Prioritami analýzy bylo vytknout možné výhody a nedostatky těchto systémů.

Po důkladné analýze nynějšího trhu jsem se zaměřil na redakční systémy Joomla (viz. sekce 2.2.1.1) a WordPress (viz. sekce 2.2.1.2). Jakožto majitel webových stránek využívajících těchto systémů a autor rozšíření pro ně jsem schopen objektivně zhodnotit jejich klady i zápory. Také se s nimi často setkávám z pohledu návštěvníka.

2.2.1 Možnosti

2.2.1.1 Joomla

Oficiální web:	`https://www.joomla.org/`
Česká fan stránka:	`http://www.joomlaportal.cz/`
Rok vydání:	2005
Verze:	3.8.6
Licence:	GNU/GPLv2 [7]

Klady:

- Široký výběr nastavení, flexibilita
- Jednoduchý a přehledný design administrace
- Možnost e-commerce

Zápory:

- Pro běžného uživatele se může vše jevit komplikované
- Poměrně úzký výběr rozšíření
- Části dokumentace pro programátory jsou často zastaralé, chybné nebo neexistující

Tabulka 2.1: Technické požadavky (pro verze Joomla 3.x) [3]

Software	Minimální	Doporučená
PHP	5.3.10	5.6 nebo 7.0
Databáze		
MySQL	5.1	5.5.3
MS SQL	10.50.1600.1	10.50.1600.1
PostgreSQL	8.3.18	9.1
Webový server		
Apache	2.0	2.4
Ngnix	1.0	1.8
Microsoft IIS	7	7

Joomla pracuje na systému pluginů, modulů, komponent a témat. Komponenty jsou nejsložitější částí systému a poskytují komplexní řešení webu, jako je například e-shop nebo vytváření a editace článků.

Moduly obsahují několik šablon, které mohou být vloženy do pozic v tématu. Jsou méně složitější než komponenty a zobrazují jednoduché informace jako třeba statistiky uživatelů nebo výpis produktů.

Pluginy rozšiřují funkčnost systému a obsahují především malé vizuální komponenty. Používají se tedy primárně k rozšíření existujících modulů a komponent nebo k přeměně HTML elementů na funkční prvky (kalendáře, grafy atp.).

Témata jsou pouze šablony, které se nestarají o správu dat, ale pouze mění existující vzhled rozšíření.

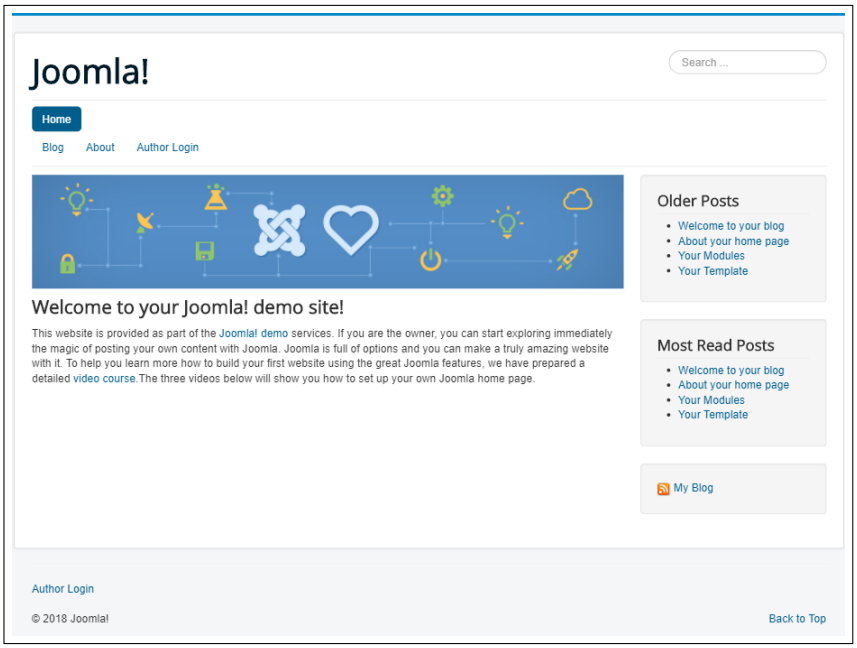
Instalace celého systému je poměrně jednoduchá - stačí stáhnout balíček z webových stránek a rozbalit do adresáře webového hostingu. Po otevření stránky se zobrazí formulář s několika kroky a detailním popisem všech polí pro základní nastavení aby stránky mohly fungovat. Rozšíření se přidávají do systému přesunutím staženého adresáře do příslušné složky pro komponenty, moduly nebo pluginy.

Díky velké a aktivní komunitě je Joomla jedním z preferovaných redakčních systémů pro mnoho velkých i malých firem. Online repozitář s rozšířeními čítá dohromady (dne 28. 3. 2018) 7 967 rozšíření a stále se rozrůstá. Je zde možnost si naprogramovat vlastní rozšíření, ale je nutno spoléhat také na jiné zdroje než oficiální dokumentaci, která místy obsahuje již nerelevantní informace.

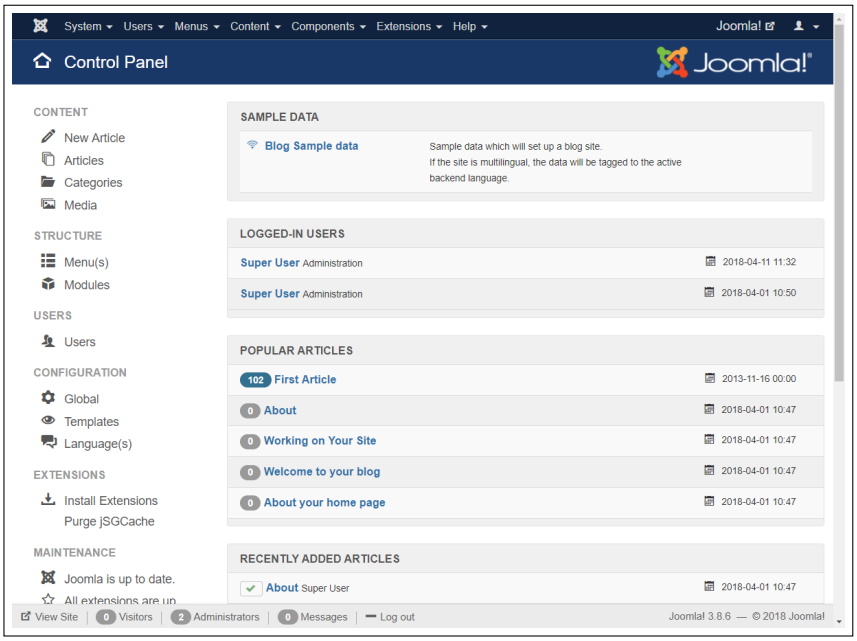
V základu je vše v anglickém jazyce, českou lokalizaci je tedy třeba nainstalovat ručně.

Výhodami CMS[1] Joomla jsou možnost si kompletně personalizovat svůj web a spravovat rozšíření tak, aby byl výsledek podle představ uživatele. Kdykoliv se dá přidat funkcionalita navíc s minimálním ovlivněním již existujícího řešení.

[1] Content Management System



Obrázek 2.2: Snímek obrazovky s úvodní stránkou základního webu Joomla



Obrázek 2.3: Snímek obrazovky administrace Joomla

2.2.1.2 WordPress

Oficiální web:	`https://wordpress.org/`
Česká fan stránka:	`https://cs.wordpress.org//`
Rok vydání:	2003
Verze:	4.9.4
Licence:	GNU GPLv2+ [8]

Klady:

- Jednoduchý na instalaci
- Široký ekosystém
  - Hodně rozšíření
  - Velká komunita
- Šetrný k systémovým zdrojům

Zápory:

- Náročnější úprava vzhledu
- K docílení vize je potřeba instalace pluginů
  - Příliš široký výběr
  - Většinou nepodporují českou lokalizaci
  - Často obsahují části uzamknuté za platební bránou
- Atraktivní pro hackery
  - Spousta rozšíření obsahuje bezpečnostní díry

Tabulka 2.2: Technické požadavky pro WordPress [4]

Software	Minimální	Doporučené
PHP	5.2.4	7.2
Databáze		
MySQL	5.6	5.0
MariaDB	10.0	10.0
Webový server		
Apache	Nespecifikováno	Nespecifikováno
Ngnix	Nespecifikováno	Nespecifikováno

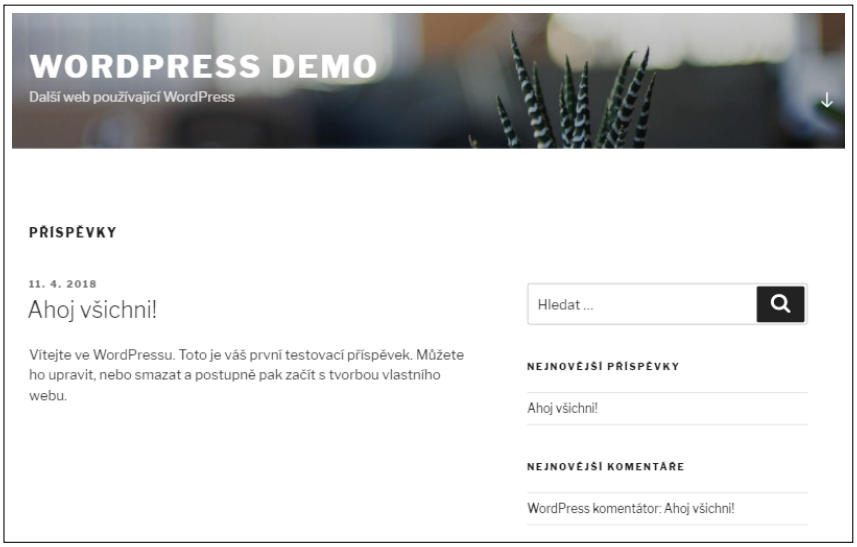
Systém pracuje pomocí pluginů, které jsou zpravidla umístěny na fixní pozice do šablony a nelze s nimi hýbat pokud na to neobsahují specializovaná nastavení. Proto je složité dělat větší změny v designu. Pluginů existuje spousta (54 786 k 1. 4. 2018), a je jednoduché najít ten, který zrovna uživatel potřebuje. Tento fakt ale přináší i svá rizika, protože část těchto pluginů je zastaralých a mohou obsahovat bezpečnostní skuliny, což je dělá atraktivními pro útoky hackerů.

Dle portálu w3techs.com je WordPress nejrozšířenější redakční systém, který používá 30 % webových stránek z nejlepších 10 miliónů seřazených podle Alexa ranking [9].

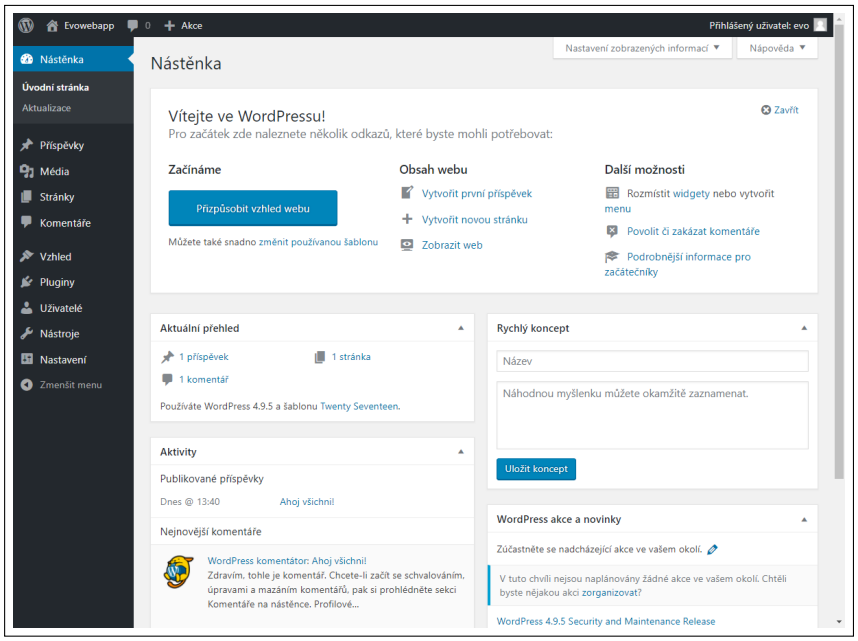
Hlavním důvodem, proč je WordPress velice populární, je jeho jednoduchá instalace a správa obsahu. Výhodou je jeho šetrnost k systémovým zdrojům a dobře funguje i na méně výkonných hostingových službách. Existuje mnoho vzhledů, které jsou dostupné zdarma, proto ve většině případů není nutný zásah externího grafického návrháře, a výdaje jsou tedy minimalizovány.

Tabulka 2.3: Nejpopulárnější systémy pro správu obsahu [5]

	použití	změna od 1. února 2018	podíl mezi CMS	změna od 1. února 2018
WordPress	30.5%	+0.6%	60.1%	−0.1%
Joomla	3.1%		6.1%	−0.1%
Drupal	2.2%		4.2%	−0.2%
Magento	1.1%	−0.1%	2.3%	−0.1%
Shopify	1.0%		2.1%	+0.2%



Obrázek 2.4: Snímek obrazovky s úvodní stránkou základního webu WordPress



Obrázek 2.5: Snímek obrazovky administrace WordPress

2.2.1.3 Magento

Oficiální web:	`https://magento.com/`
Česká fan stránka:	`http://magento.cz/`
Rok vydání:	2008
Verze:	2.2
Licence:	OSL v3, AFL v3 [10]

Klady:

- Velká a aktivní komunita
- Škálovatelný do budoucna
- Flexibilní

Zápory:

- Nákladné na zdroje
  - Spousta rozšíření je placených
  - Vyžaduje výkonný hosting
- Hodí se spíše na e-commerce než na osobní web
- Málo vývojářů, tedy pomalejší vývoj

Tabulka 2.4: Požadavky na technologie pro verzi 2.2 [6]

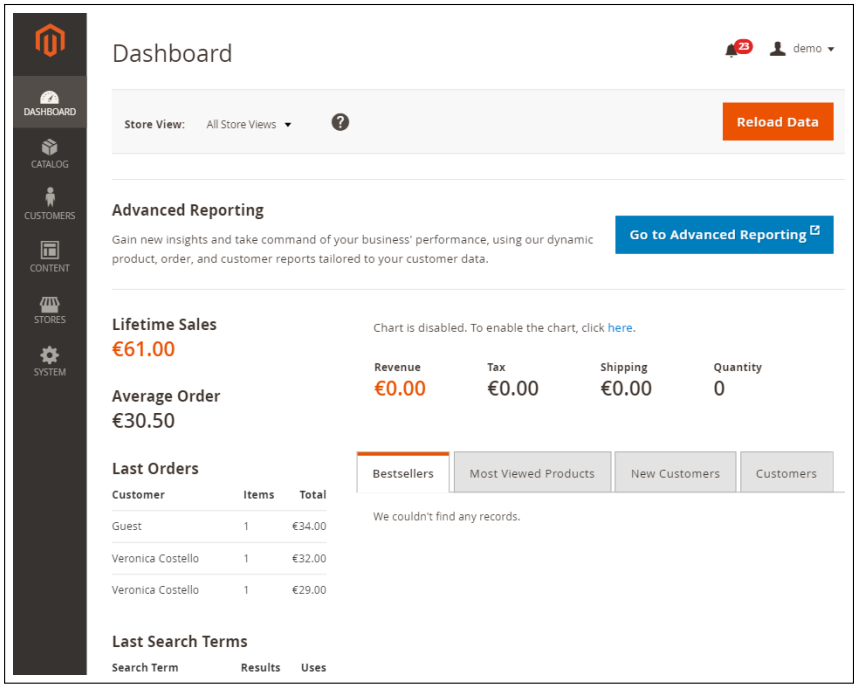
Software	Podporované verze
PHP	7.0.2, 7.0.4, 7.0.6 - 7.0.x, 7.1.x
Databáze	
MySQL	5.6, 5.7
MariaDB	10.0, 10.1, 10.2
Percona	5.7
Webový server	
Apache	2.2, 2.4
Ngnix	1.x

Magento je CMS postavený na PHP frameworku Zend a funguje na systému rozšíření a témat. Poskytuje profesionální řešení pro vytváření e-shopů, ale dá se použít i jako osobní web. Poté je možné povolit část s produkty a bez problému začít prodávat sortiment online. Na oficiálních stránkách je k dispozici několik placených verzí, ale nejvíce rozšířenou je Magento Open Source (dříve Magento Community Edition), která je dostupná zdarma a je libovolně upravitelná (v rámci licence).

Moduly je možné nainstalovat stažením balíčku z oficiálního obchodu s rozšířeními. Zde je nevýhodou, že většina z nich je placená. Začínající podnikatel ale tato rozšíření nepotřebuje, protože základní sada obsahuje vše potřebné. Magento je také stavěné na to, aby bylo používáno ne-programátorem, a proto má přehlednou administraci s možností si personalizovat celý web.

Hlavní nevýhodou jsou poměrně vysoké požadavky na výkon, tudíž na levnějším hostingu nemusí web běžet dostatečně rychle a mohl by snadno zákazníka dobou načítání odradit. Má také velice specifické požadavky na technologie, proto je těžké vybrat ten správný hosting, na kterém vše bude správně fungovat.

Obrázek 2.6: Snímek obrazovky s úvodní stránkou demo webu Magento



Obrázek 2.7: Snímek obrazovky administrace Magento

2.2.2 Shrnutí

V analýze existujících řešení je porovnáno několik vybraných systémů a byly zváženy jejich klady i zápory vůči ostatním.

2.3 Framework

V této kapitole bude analyzováno použití frameworku k realizaci projektu. Také se shrnou výhody a nevýhody nabízených možností a na jejich základě bude postaveno rozhodnutí.

Hlavní kritéria výběru:

- Rychlost
- Bezpečnost
- Míra usnadnění práce
- Možnosti pro programátory
- Jednoduchost nastavení a rozšíření funkcionality
- Dokumentace a podpora od komunity

2.3.1 Možnosti

Pro porovnání byly vybrány frameworky Nette, Laravel, Symfony a Zend. Všechny jsou vysoce modulární, řídí se hlavními best-practices pro vývoj webových stránek a jsou podrobně dokumentovány. Podporují vysoce přizpůsobitelný routing (přepisování URL a volání definovaných funkcí) a v základu využívají softwarové architektury MVC[2].

2.3.1.1 Zend

Framework Zend funguje na základě instalace komponent pomocí správce balíčků pro PHP `composer`. Prvotní tzv. „skeleton“ neumí prakticky nic, je proto potřeba přidat základní funkcionalitu, která zajistí minimální chod systému. Je ale možné použít příkazovou řádku, která programátora provede instalací a nabídne mu několik základních balíčků.

Hlavní výhodou je možnost si sám specifikovat funkce a robustnost. Firma Zend Technologies, která framework dlouhodobě vyvíjela jako interní systém, ho nyní aktivně spravuje pro veřejnost s licencí BSD 2.0. Nevýhodou pak je nutnost si vše nastavit sám. Dokumentace je pro začátečníka velice těžká na pochopení a místy ztrácí na rychlosti oproti konkurenci.

[2]Model–View–Controller

2.3.1.2 Nette

Oficiální stránky [11] uvádí, že „Nette je sada samostatných PHP 7 komponent tvořících dohromady framework, vyhodnocený jako 3. nejpopulárnější na světě. Nette klade mimořádný důraz na produktivitu, čistý kód a bezpečnost“. Využívá vlastního šablonovacího jazyka Latte. Také se zaměřuje na dodržování best practices, jako je DRY a KIS.

Vznikl jako český framework, jehož autorem je David Grudl. Nyní ho ale aktivně vyvíjí Nette Foundation.

Výhodou oproti ostatním frameworkům je vestavěná „laděnka“, oficiálně pojmenovaná Tracy, která odchytává a loguje chyby a proměnné, vedoucí k jednoduššímu debugování při běhu webu. Nevýhodou je malá celosvětová popularita, takže je občas těžké najít řešení problémů.

2.3.1.3 Symfony

Symfony je open source PHP framework fungující na architektuře MVC. K dosažení funkcionality využívá komponent a balíčků. Balíčky jsou hlavními stavebními kameny celého frameworku, protože obsahují jak jeho základní tak i rozšiřující funkce. Každý balíček je možné zapnou, vypnout nebo použít v jiném systému využívající Symfony bez nutnosti velkých změn.

Komponenty se starají o dílčí funkce nezávisle na systému, a proto je možné je využít i v systémech nepoužívajících Symfony.

2.3.1.4 Laravel

Laravel je open-source (licence MIT)[12] framework postaven na PHP 7.1.2, jehož první verze byla vydána na začátku roku 2012. Je spravován Taylorem Otwellem, který se projektu věnuje na plný úvazek. I přesto, že využívá komponenty z frameworku Symfony (verze 2), v nezávislých testech rychlosti vychází lépe.

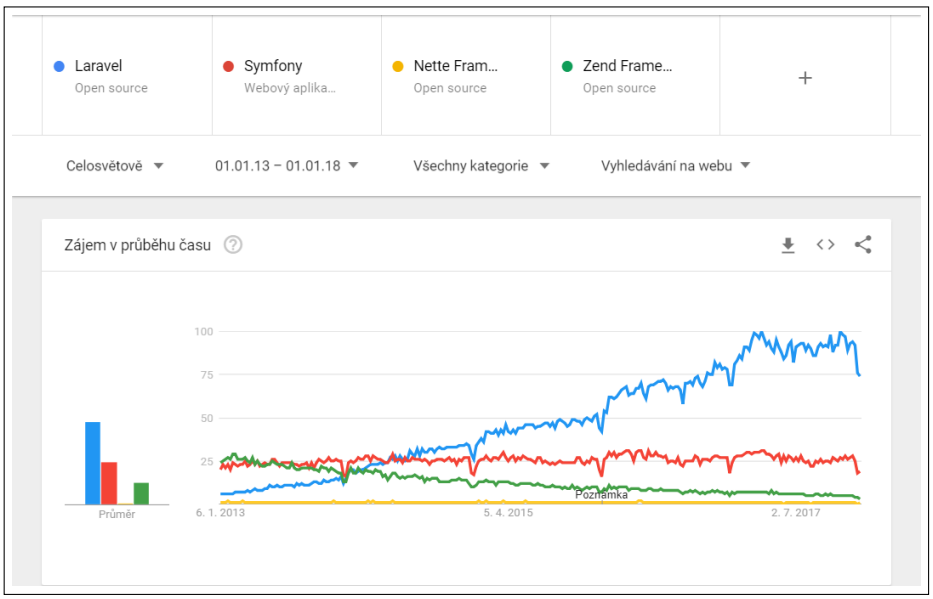
Pro šablony používá vlastní šablonovací jazyk Blade, který je jednoduše rozšiřitelný. Dále nabízí mnoho zjednodušení pro vývojáře, jako je generování komponent pomocí příkazové řádky. Všechny části jsou plně modulární a podrobně a přehledně dokumentované. Proto je vývoj svižný a snadný. Hlavní nevýhodou je, že framework v mnoha místech interně používá tzv. „magické“ metody, které dělají opravu případných chyb těžší. Problémem také může být časté dotazování na databázi, což zapříčiní, že větší weby nejsou optimální pro sdílené hostingy.

2.3.2 Statistiky

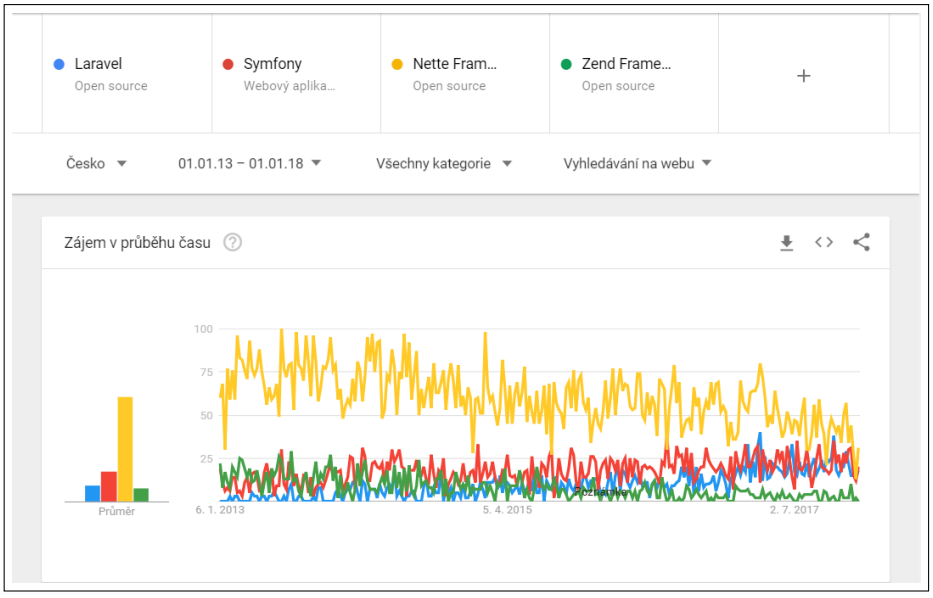
Dle statistik vyhledávání za období 1. 1. 2013 až 1. 1. 2018 na portálu Google Trends [13] je vidět vzestup popularity frameworku Laravel a pokles Zendu.

Symfony je stále stejně populární a proto si svoji pozici obhajuje. Na druhou stranu Nette je skoro neznámý.

Pokud jsou statistiky omezeny pouze na Česko, obsadí Nette první příčku.



Obrázek 2.8: Popularita vyhledávání vybraných webových frameworků celosvětově



Obrázek 2.9: Popularita vyhledávání vybraných webových frameworků pro Česko

2.3.3 Závěr

Bylo analyzováno několik populárních PHP frameworků, které by mohli přispět k vývoji zjednodušením částí systémů a zlepšením bezpečnosti. Rozhodnutí o použití bude stanoveno v kapitole Realizace.

2.4 Analýza požadavků

Tato kapitola se zabývá analýzou požadavků na systém, které by měl splňovat. Bylo analyzováno několik hlavních požadavků na funkce a vytvořeny případy užití pro konkrétní aktéry.

2.4.1 Aktéři

Základní struktura aplikace je rozdělena na dvě skupiny uživatelů. První skupinou jsou standardní uživatelé – návštěvníci – kteří mohou, ale nemusí, být registrováni a přihlášeni. Druhou skupinou jsou administrátoři, kteří mají přístup ke správě webu a jeho nastavením, například nastavení titulku webových stránek, spravování modulů a jejich nastavování.

Návštěvník Návštěvníkem je myšlen každý uživatel. Má přístup k veřejně dostupným položkám, ale nemá práva editovat žádný aspekt systému, pokud to modul výhradně nepovolí.

Administrátor Administrátor je přihlášený uživatel s rolí, která obsahuje příznak administrátora. Má přístup do administrace a ke všem funkcím, které má i návštěvník.

2.4.2 Funkční požadavky

Funkční požadavky udávají požadavky na funkce systému. Nepopisují kroky pro dosažení daného výsledku, ale udávají kostru procesu a výsledek, kterého bude dosaženo. Nehledí na konkrétní implementaci a jsou abstrahovány tak, aby mohli být aplikovány na obecné systémy.

F1 - Registrace Do systému bude možná registrace, která poskytne identifikaci jednotlivých uživatelů, na základě čehož se uživateli odemknou další možnosti, jako například komentáře.

F2 - Přihlášení Registrovaní uživatele se budou moci pomoci e-mailu a hesla přihlásit a využívat možnosti pouze pro přihlášené uživatele.

F3 - Správa modulů Systém bude podporovat správu modulů administrátorem. Patří sem hlavně jejich instalace, odinstalace a nastavení.

F4 - Správa instancí Spravování instancí spadá pod funkční požadavek F3 a zajišťuje umisťování modulů na pozice v šabloně.

F5 - Správa nastavení systému Administrátor webu může měnit základní nastavení systému.

F6 - Prvotní instalace Po stažení souborů se systémem bude muset uživatel vyplnit prvotní informace, aby byl systém funkční.

2.4.3 Nefunkční požadavky

Nefunkční požadavky mají za úkol zmapovat nezbytné vlastnosti systému, které nejsou obsluhovány uživatelem přímo, ale jsou plněny implicitně architekturou a implementací.

N1 - Stabilita Celý systém bude od základu stabilní a neobsahovat (v perfektním případě) žádné chyby, které by znemožnily nepřístupnost některé části webu. Bez zásahu vlastníka webu do systému souborů nenastane pád aplikace nebo nefunkčnost části webu. Instalace nových modulů, které nejsou součástí základní množiny, se nedá ochránit proti chybám, a je proto na vlastní nebezpečí.

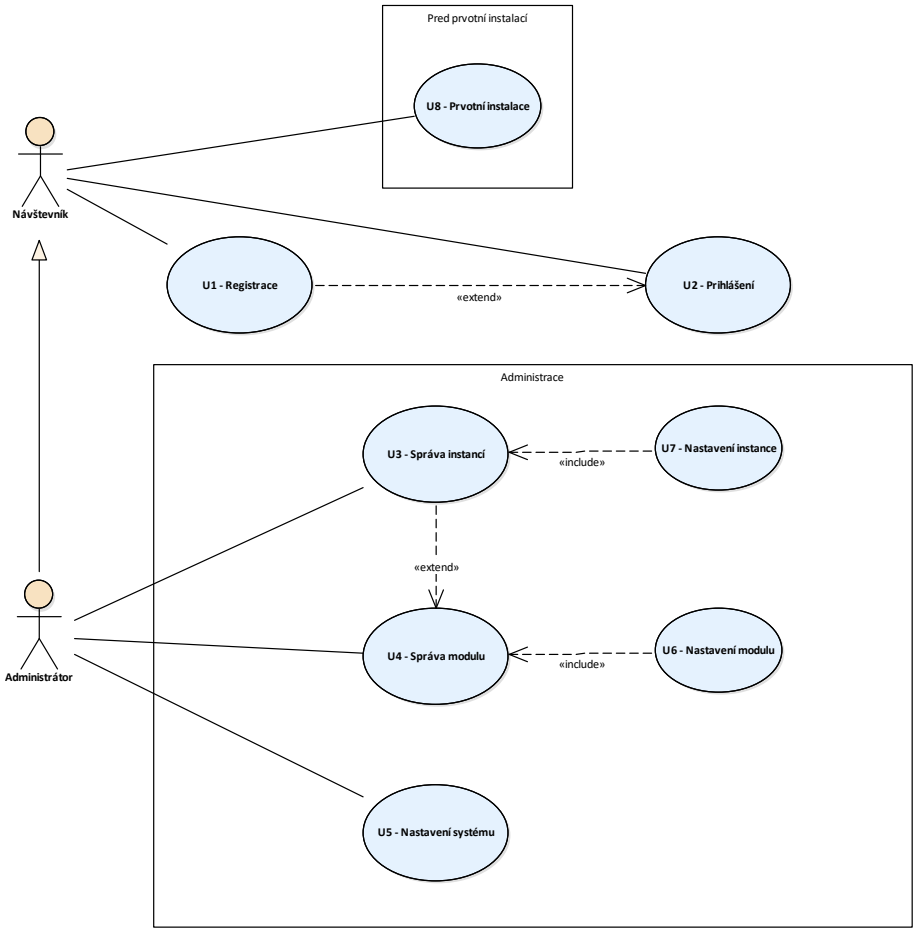
N2 - Bezpečnost Bezpečnost je pro všechny systémy nejvyšší priorita. Pokud uživatel nemá oprávnění dostat se k určitým zdrojům, nesmí mít tu možnost. Nepočítá se s tím, že si administrátor nastaví slabé heslo, které může uživatel náhodou uhodnout. V tomto případě nelze nijak zabránit neoprávněnému přístupu. Klade se důraz na ochranu proti využívání chyb v systému.

N3 - Rozšiřitelnost a modularita Funkce systému – kromě těch základních – budou obsluhovány moduly. Poskytují rozhraní pro vytváření prvků plnících určité role. Jsou nezávislé a každý modul je zapouzdřený tak, aby neovlivňoval nebo nelimitoval ostatní.

2.4.4 Případy užití

„Use case, česky případ užití, je termín označující v oborech jako Human–Computer Interaction (interakce člověka s počítačem) či interakční design metodu popisu sekvence kroků, který vykonává uživatel při plnění konkrétního úkolu za použití interagujícího systému (konkrétního softwarového či hardwarového produktu). Takovým úkolem může být například vytvoření určitého typu dokumentu v textovém editoru, nákup v internetovém obchodě, přečtení si článku na zpravodajském serveru i například výběr z bankomatu. Ve zvláštních případech může jít o interakci dvou strojů, z nichž jeden plní konkrétní úkol (například synchronizace zařízení).

Use case je složen ze sledu kroků, které uživatel vykonává, aby daný úkol splnil. Pro popis use case bývá užíván specializovaný jazyk Unified Modeling Language (UML), který formou diagramu popisuje jednotlivé akce uživatele a reakce systému, ale celou interakci je možné popsat i jednoduše slovně.“[14]



Obrázek 2.10: Diagram případů užití

U1 - Registrace Registrace začíná, když se nepřihlášený návštěvník rozhodne, že si vytvoří účet. Nutnou podmínkou jsou povolené veřejné registrace od administrátora. Návštěvník klikne na tlačítko `Registrace`, vyplní e-mail, heslo a odešle formulář. Pokud vše proběhne v pořádku, je registrován a může se přihlásit.

U2 - Přihlášení Uživatel se přihlašuje pomocí tlačítka `Přihlášení`, které ho přesměruje na formulář pro vyplnění jeho e-mailu a hesla. Po odeslání formuláře a ověření údajů je přihlášen.

U3 - Správa instancí Administrátor spravuje instance modulů po kliknutí na tlačítko `Detail` v **U4**.

U4 - Správa modulů Administrátor klikne na tlačítko `Moduly` a poté může pomocí dalších tlačítek instaloval a odinstalovat moduly nebo si zobrazit jejich detail a instance.

U5 - Správa nastavení systému Administrátor edituje nastavení systému z hlavní stránky administrace. Změní požadované hodnoty a poté klikne na tlačítko uložení.

U6 - Nastavení modulu Administrátor klikne na tlačítko `Detail` z **U4**. Zobrazí se detail modulu kde je možné změnit nastavení pomocí formuláře.

U7 - Nastavení instance Administrátor klikne na tlačítko `Detail` z **U3**. Zobrazí se detail instance modulu kde je možné změnit nastavení pomocí formuláře.

U8 - Prvotní instalace První uživatel, který si zobrazí web, vyplní formulář základním nastavením pro web.

2.4.5 Pokrytí funkčních požadavků

Tabulka 2.5: Pokrytí funkčních požadavků případy užití

	F1	F2	F3	F4	F5	F6
U1	✓					
U2		✓				
U3				✓		
U4			✓			
U5					✓	
U6			✓			
U7				✓		
U8						✓

Kapitola 3

Návrh

3.1 Moduly

Moduly jsou jedním z hlavních stavebních kamenů systému. Jejich funkce budou navzájem distinktivní. V základu bude dostupných několik modulů. Každý modul bude moci být umístěn do libovolné předdefinované pozice na webu dle uvážení administrátora.

Každý modul se bude moci nacházet ve dvou stavech – nainstalovaný a neinstalovaný – kde ten druhý bude počáteční. Po úspěšné instalaci může být dále umístěn do webu.

Základní moduly:

- Seznam produktů
- Informace o obchodníkovi

3.2 Architektura

Každá dobře navržená webová aplikace se řídí předem určenými způsoby, jak mezi sebou vnitřní komponenty komunikují. Zlepšují přehlednost kódu, zjednodušují opravu chyb a zrychlují celý systém i vývoj, pokud jsou správně aplikovány a používány.

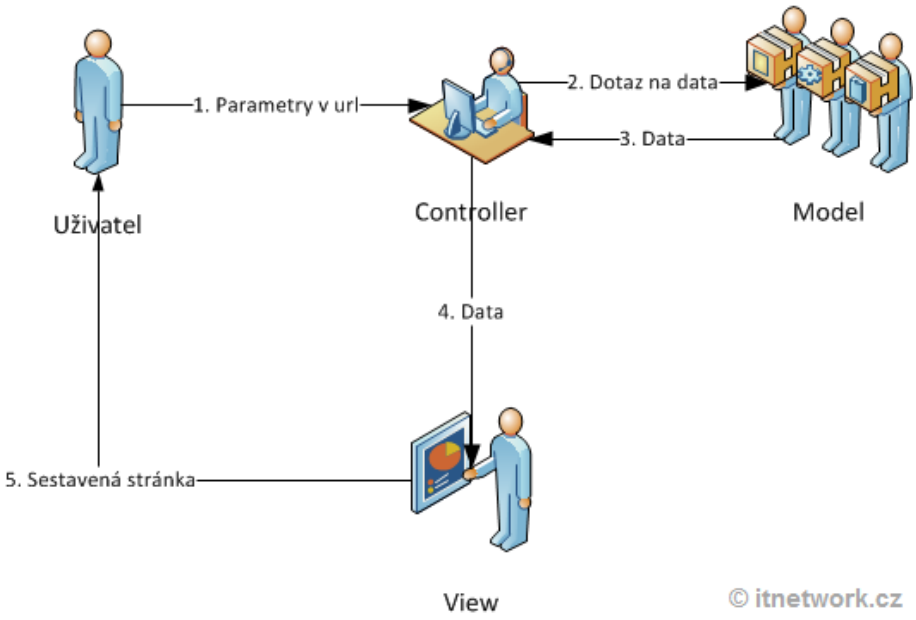
3.2.1 MVC

`Model-View-Controller`[3] se používá především v serverové části systémů. `Model` Značí data, která využívá `Controller`, reaguje na požadavky od uživatele a poté výsledek pošle do `View`, který vše zpracuje a odešle uživateli.

Tento model využívají frameworky Nette, Laravel, Symfony, CakePHP, Django a další. Některé z nich ho podporují a je možné je použít k vytvoření

[3] Více o architektuře v [1]

webu s MVC architekturou, ale jsou mnohem komplikovanější a není nutné se jí striktně řídit.

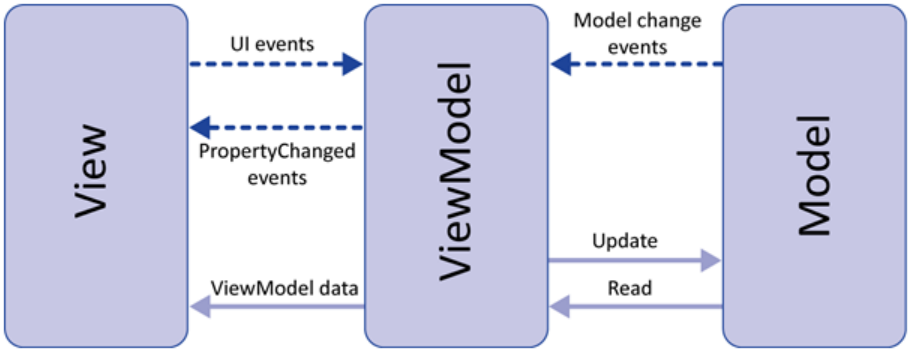


Obrázek 3.1: Model softwarové architektury MVC[1]

3.2.2 MVVM

Architektura `Model-View-View-Model`[4] využívá `Model`, který obstarává data a bussiness logiku, `View`, starající se o zobrazování dat a propagování událostí do `View-Model`, který volá příslušné metody na modelu. Tato architektura se spíše používá u reaktivních aplikacích často využívajících asynchronní načítání dat pomocí JavaScriptu.

Na bázi tohoto modelu fungují například Angular, AngularJS, Vue.js a Aurelia.



Obrázek 3.2: Model softwarové architektury MVVM[2]

3.3 Administrace

Administrace bude dostupná pomocí administrátorského účtu. Zde bude možné nastavit základní informace o webu, jako je například název, a spravovat samostatné moduly.

3.4 Prvotní instalace

Když se uživatel rozhodne pro instalaci systému, nesmí být odrazen komplikací prvotní instalace. Je počítáno s tím, že doména je již zaplacená, uživatel má přístup k souborovému systému a e-mailem, nebo pomocí jiného komunikačního kanálu, obdržel přihlašovací údaje databázi.

Bude se skládat z formuláře s políčky k vyplnění, která budou co nejdetailněji popsána, aby i laik pochopil, co má vyplnit.

3.5 Bezpečnost

Systém bude splňovat hlavní bezpečnostní kritéria. Uživatel, který není oprávněný k přístupu ke zdrojům, nesmí mít možnost je číst nebo s nimi mani-

[4]Podrobný popis architektury MVVM v [15]

pulovat. Je důležité zamezit hlavním typům bezpečnostních děr, které se ve vývoji webových stránek vyskytují.

3.5.1 Cross-site Scripting

Jedním z nejrozšířenějších bezpečnostních rizik je Cross-site Scripting, tzv. XSS[5], kdy díky chybě může útočník podstrčit nic netušícímu uživateli skript, a tím manipulovat s obsahem, který se mu zobrazí. Nejčastěji se používá soubor typu JavasScript, který může libovolně měnit, jak web vypadá, přesměrovat uživatele nebo posílat informace aplikacím třetích stran. V mnoha případech si chyby nevšimne ani samotný administrátor, a pokud je útočník zkušený, chyba v systému může přetrvat i měsíce nebo roky.

Chyba nastává především u vypisování obsahu vloženého jiným uživatelem – útočníkem – například u článků, komentářů, popisků nebo uživatelských jmen. Na druhou stranu je velice jednoduché se proti této chybě bránit, a to jednoduše všechen uživatelský obsah filtrovat při vypisování nebo ukládání do databáze uživateli.

3.5.2 SQL Injection

Tato bezpečnostní chyba se vyskytuje především u webových aplikací, které nejsou stavěny na frameworku, který automaticky vytváří dotazy do databáze. Funguje na principu neošetření uživatelem zadaných dat a umožňuje útočníkovi manipulovat s interními daty systému. Ve vstupu použije speciální znaky pro daný databázový server, který web používá, a změní tak čistý příkaz do databáze, který server posílá, na vlastní.

Ochránit se programátor může pouze tím, že bude filtrovat všechna příchozí data tak, aby neobsahovala speciální znaky používané databází. Jazyk PHP také podporuje tzv. „prepared statements“, které místo otazníků dosazují data a automaticky filtrují speciální znaky.

Více informací o SQL Injection a jak se bránit v [17].

3.5.3 Cross-Site Request Forgery

CSRF[6] je méně častým, ale stále důležitým bezpečnostním rizikem. Zpravidla neumožňuje získat přístupové údaje od uživatele. Jedná se především o odesílání požadavků na změnu či vypsání zdrojů na úkor uživatele, který je již přihlášen do aplikace. Uživatel pak otevře stránku, kterou útočník předem připravil. Nutnou podmínkou je, aby byl uživatel přihlášen do systému s dostatečným oprávněním – například administrátor – a aby útočník znal strukturu dat, které je nutno odeslat.

[5] Další informace o útoku jsou dostupné na [16]

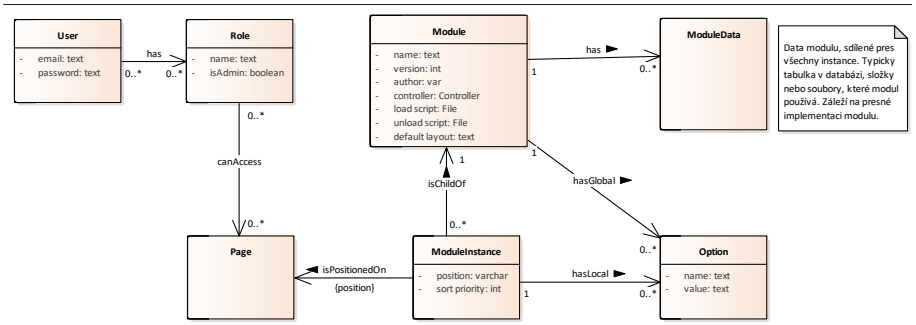
[6] Více informací o tomto typu útoku je dostupných v [18]

Nejjednodušším způsobem, jak se proti útoku bránit, je používat tzv. „CSRF token“, který bude součástí každého formuláře vygenerovaného systémem a bude se ověřovat vždy při odeslání požadavku. Tento token je zpravidla náhodně generovaný řetězec nebo číslo a je uložený v samostatném skrytém poli.

Framework Laravel ochranu proti CSRF obsahuje už od základu, proto odmítne jakýkoliv odeslaný formulář, pokud neobstahuje daný token. Každý formulář musí v šabloně obsahovat makro `@csrf`, které automaticky vloží token do formuláře [19].

3.6 Návrhový model

Návrhový model tříd je konceptuální model popisující relace mezi objekty v systému. Je nejčastěji reprezentován pomocí UML[7] diagramu.



Obrázek 3.3: Návrhový model tříd

[7]Unified Modeling Language

Kapitola 4

Realizace

4.1 Framework

Po analýze současných PHP frameworků v kapitole 2.3 byl vybrán Laravel (viz. 2.3.1.4). Díky předchozím zkušenostem na osobních projektech a množství návrhových vzorů, které usnadňují práci a zlepšují bezpečnost i stabilitu, je velice přitažlivý pro nové a moderní projekty.

Pro vzhled (nebo také front-end) webu bude použit framework Bootstrap 4. Je to responzivní CSS[8] a JS[9] šablona poskytující sadu komponent vhodných pro jednoduché a rychlé stylování webů. Je přibalen v základní instalaci frameworku, proto není nutné ho ručně přidávat. Tento framework používá i celosvětově známý web Twitter.

4.1.1 Šablony

Blade je jednoduchý, a přesto účinný šablonovací jazyk, který je obsažen přímo v Laravelu. Díky tomu, že se překládá do čistého PHP jen tehdy, když je zaznamenána změna v souboru, je jeho používání volitelné a na rychlosti webové stránky neubírá. Soubory používají příponu `.blade.php` a jsou standardně uloženy ve složce `/resources/views` [20].

Výpis proměnné nebo zavoláni funkce a výpis jejího výsledku je možný pomocí obalení kódu do dvojitých složených závorek „`{{ ... }}`“.

Speciální funkce (nebo také makra) poté používají znak zavináč `@` a často přijímají složené argumenty. Plní jednoduché funkce, například podmíněné bloky, nebo o dost složitější funkce, jako je rozšíření existující šablony a následného vložení vlastního obsahu.

Díky vlastnostem Laravelu je možné rozšířit šablony o vlastní makra. V této práci je vytvořeno makro pro zobrazování modulů v šabloně pomocí `position`.

[8] Cascading StyleSheet
[9] JavaScript

Zdrojový kód 4.1: Registrace makra `@position` pro šablony

```
<?php
Blade::directive('position', function ($position) {
  return '<?php
    if($__modules->has(' . $position . ')) {
      foreach($__modules->get(' . $position . ') as $__module) {
        echo view(
        "Module_" . $__module->module->name . "::default",
        array_merge(
          [ "module" => $__module ],
          $__module->controller->getTemplateVariables()
        )
        );
      }
    }
  ?>';
});
```

Všechny šablony, kromě těch specifických pro moduly, jsou uloženy ve složce `/resources/views`. Laravel používá řetězce k určení cesty šablon, která je výčet názvů složek končící jménem souboru bez přípony. Šablony pro moduly se nachází v podsložce modulu `layout`, proto je nutné zaregistrovat šablonový `namespace` určující alias složky, ve které se má vyhledávat. Ten je poté možné specifikovat tak, že cesta k šabloně začíná jménem namespace, dvěma dvojtečkami `::` a zbytkem cesty. Všechny moduly mají automaticky registrován namespace `Module_<název modulu>`.

4.1.2 Routing

Routing je pojem u webových stránek, kdy jsou URL zpravidla spravovány jedním nebo více tzv. „routery“, do kterých jsou programově vloženy routes. Po dotazu na adresu odpovídající jedné route je požadavek zpracován, následně je zavolána s ní spojená funkce a vrácen výsledek. Typicky je složena šablona, které příslušný controller dodá data, které jsou pouze zobrazeny v čisté podobě jako JSON[10], XML[11] nebo jiná datová struktura, nebo vloženy do šablony a odeslána uživateli.

[10] JavaScript Object Notation
[11] eXtensible Markup Language

Zdrojový kód 4.2: Příklad registrování routes

```php
<?php
Route::get('welcome', function(Request $request) {
  return view('pages.welcome', [ 'test' => $request->input('test') ]);
});

Route::post('api/post/{id}', 'PostController@createPost');

Route::resource('comment', 'CommentController'); //CRUD operations
```

4.2 Administrace

Pro přístup do administrace je nutné se přihlásit pod administrátorským účtem. Ten je vytvořen automaticky při prvotní instalaci a je mu přiděleno náhodně vygenerované bezpečné heslo, které se poté dá změnit. Z administrátorské sekce je možné spravovat moduly jako takové, nebo přistupovat k routes, které modul registruje.

4.2.1 Challenges

V průběhu vývoje systému se vyskytlo několik technologických překážek, které občas nebylo možné vyřešit tradičním způsobem podle konvencí frameworku nebo nových funkcí jazyka PHP 7.1.

4.2.1.1 Controller modulu

Ovládací třídu modulu – controller – není možné získat nezávisle na jménu třídy. Je tedy vždy nutné znát jméno třídy, ze které chceme konstruovat instanci – objekt. Případný autor modulu musí pojmenovat třídu přesně podle konvence, tedy název složky modulu konvertovaný na StudlyCase[12] ve formátu `Module<název složky modulu>Controller`, například pro modul s názvem „`product-list`“ bude výsledná třída pojmenována „ModuleProductListController“. Systém bude hlásit chybu pokud tak nebude učiněno. PHP verze 5.3+ podporuje klíčová slova „`namespace`“ a „`use`“ pro členění tříd do skupin. Pokud je pomocí „`use`“ importována třída, která do té doby nebyla importována jinak (například přes “include” nebo “require”), pokusí se autoloader načíst třídu pomocí cesty ze stejnojmenného souboru. Protože třída se nachází v souboru “controller.php” a konvence jazyka vyžaduje klíčové slovo „`use`“ v kořenu třídy nebo souboru, je třeba třídu modulu uvnitř funkce importovat pomocí „`require`“.

[12] slovní spojení bez mezer s prvním písmenem každého slova velkým

4.2.1.2 Moduly v šablonách

Při specifikování šablony je třeba dodat proměnné, které budou viditelné z kódu. Zde byl problém zaručit, že v každé šabloně bude globálně dostupné pole s moduly, ze kterého bude čteno při zobrazování modulů na pozici. Laravel ale nabízí možnost sdílení proměnné se všemi šablonami, které vzniknou.

4.3 Moduly

Moduly jsou složky se všemi potřebnými částmi pro jejich funkci. Každý má dva stavy: nainstalovaný a neinstalovaný. Instalace probíhá z administrátorské sekce přímo na webu. Vlastník si tedy může vybrat, které moduly potřebuje, a které může ponechat vypnuté.

4.3.1 Struktura

Složky modulů se nachází v „`/app/Modules`“, kde jsou jednotlivé složky pojmenovány způsobem „kebab-case“ (slova mají malá písmena, bez diakritiky, a jsou odděleny spojovníkem „`-`“). U složek nepojmenovaných podle konvence není zaručena funkčnost.

Adresářová struktura modulu:

```
Modules
├── example-module
│   ├── module.json
│   ├── load.sql
│   ├── unload.sql
│   ├── controller.php
│   ├── layout
│   │   └── default.blade.php
│   └── lang
│       └── cs
│           └── settings.php
└── ...
```

- **module.json**
  - Soubor ve formátu JSON obsahující informace o modulu, jako je například autor, verze, celé jméno modulu a soubor nastavení, které lze nastavit v administraci.
- **load.sql**
  - Nepovinný soubor ve formátu SQL který se spustí vždy při instalaci modulu. Slouží k inicializaci databázových tabulek, které modul využívá.

- **unload.sql**
  - Nepovinný soubor ve formátu SQL který se spustí vždy při odinstalaci modulu. Slouží k typicky k uklizení databáze a smazání dat a tabulek.
- **controller.php**
  - Soubor obsahující kontrolní třídu modulu. Je zde možnost registrování routes a rozšiřování funkčnosti.
- **layout**
  - Složka obsahující views
  - **default.blade.php**
    - Základní šablona pro modul, kterou používá notace @position
- **lang**
  - Složka obsahující lokalizační složky
  - **cs**
    - Čeština
    - **settings.php**
      - Lokalizační řetězce pro nastavení v modulu a jeho instancí

4.3.2 Načítání

Všechny moduly a jejich informace jsou vždy načítány po složkách. Pokud by se stalo, že je modul nainstalován a jeho složka smazána, prakticky zaniká a jeho existující záznamy v databázi budou ignorovány. Nejprve je načítána složka a informace z "module.json", poté modul z databáze obsahující dodatečné informace, jako je datum instalace, jeho identifikátor a v poslední řadě jsou získány instance tohoto modulu. Vše je spojeno třídou ModuleBundle pro jednotný přístup a sdíleno do všech šablon.

4.3.3 Základní moduly

V základu bude systém obsahovat několik předpřipravených, ale neinstalovaných, modulů.

4.3.3.1 Seznam produktů

Seznam produktů bude listovat produkty, které vlastník webu nabízí. Samotný modul vložený na pozici zobrazuje produkty jako karty s tlačítkem pro zobrazení detailu. V administraci je možné nastavit povolení stránkování a po kolika produktech bude stránkováno. Modul také přidává kartu Produkty do hlavního navigačního menu, které je samostatně nastavitelné.

4.3.3.2 Informace o obchodníkovi

Jednoduchý modul zobrazující hlavní informace o obchodníkovi, musí každá OSVČ nebo právnická osoba na svém webu uvést. Dle [21] se pro OSVČ jedná o:

- Jméno a příjmení
- Identifikační číslo
- Místo podnikání
- Údaj o zápisu do živnostenského rejstříku
- Podnikatel zapsaný v obchodním rejstříku uvede také údaj o tomto zápisu včetně oddílu a vložky

... pro právnickou osobu poté:

- Název společnosti
- Identifikační číslo
- Adresa sídla
- Údaj o zápisu do obchodního rejstříku včetně oddílu a vložky
- Popřípadě údaj o zápisu organizační složky podniku nebo podniku zahraniční osoby do obchodního rejstříku

4.4 Použité návrhové vzory a principy

4.4.1 Normalized Systems theory

Teorie Normalizovaných Systémů[13] se snaží vytvořit vysoce evolvabilní softwarové systémy dle doporučeného svazku teorémů, které byly formálně dokázány jako nutné podmínky pro dosažení evolvability. Tato teorie ukazuje několik působivých a unikátních charakteristik pro řešení problému agility: je zakořeněna v konceptech z teorie systémů. Tyto odvozené teorémy poskytují specifické řízení programování a u toho sjednocení nejlepší praktiky. Svojí proveditelnost dokázali v praxi v mnoha implementacích pro různorodé typy systémů [23].

V práci se uplatňují všechny 4 následující teorémy do nejvyšší možné míry. Použitý framework některé z nich následuje, je proto jednoduché zaručit jejich správné užití.

[13] Více o NS v [22]

4.4.1.1 Separation of Concerns

Tento vzor, česky "oddělení zodpovědnosti", je pojem softwarového inženýrství používaný pro strukturování programu tak, aby se jeho části co nejméně překrývali a neplnili ty samé funkce. Úkolem je zapouzdřit tyto části do dobře strukturovaných rozhraní s předem známou implementací.

V práci je SoC základním stavebním kamenem celého systému. Jednoduše se implementuje, protože ho použitý framework podporuje. Vše je tedy strukturováno tak, aby se funkce jednotlivých částí nepřekrývali, což vede k čitelnějšímu kódu a snadnější opravě případných chyb.

4.4.1.2 Data Version Transparency

Transparence verze dat zaručuje, že změna datové struktury modelu nezmění nebo nenaruší existující funkčnost systému.

DVT je v práci použito při nastavování defaultních hodnot pro nastavení a použití pojmenovaných funkcí pro jejich získávání. Pokud je tedy některá část změněna, nevyhodí systém chybu.

4.4.1.3 Action Version Transparency

Tento vzor zaručuje, že pokud je existující funkce systému změněna, neovlivní to systém do míry že by musela být změněn způsob použití této funkce na více něž jednom místě.

4.4.1.4 Separation of States

Rozdělené stavů zaručuje zapamatování si stavu po každém kroku v nějakém souboru kroků, vedoucí k rozdělení těchto kroků v čase.

4.4.2 Framework

Použitý framework využívá mnoho návrhových vzorů, ale všechny nemusí nutně být použity.

Dle [24] podporuje Laravel následující návrhové vzory:

- Builder (Manager)
- Factory
- Repository
- Strategy
- Provider
- Facade

Navíc je možnost využít Dependency Injection, Singleton nebo Lazy Initialization.

4.4.3 Singleton

Návrhový vzor Singleton – jedináček – v OOP[14] zaručuje, že v celém programu může při běhu být pouze jedna instance určité třídy. To je většinou zaručeno privátním konstruktorem a metodou na získání instance (často statická metoda `getInstance()`), která vytvoří nový objekt, pokud je zavolána poprvé, nebo vrátí jeho instance při dalších volání.

Systém toto využívá při načítání modulů a jejich informací. Instance třídy `ModuleProvider` bude pro každý požadavek zkonstruována nanejvýše jednou, a to pouze tehdy, když ji bude jiná třída potřebovat pomocí DI[15]. Je zaručena rychlost, protože se data nemusejí načítat vícekrát, a jejich identita, protože je vše čteno přes jedno rozhraní.

Zdrojový kód 4.3: Registrace singletonu pro `ModuleProvider`

```
<?php
$this->app->singleton(ModuleProvider::class, function ($app) {
  $mp = new ModuleProvider($app);
  $mp->init();
  return $mp;
});
```

4.5 Testování

Testování je důležitá část každého většího projektu. Odchytává chyby, na které by mohl uživatel narazit náhodou a být odrazen od dalšího používání webu. Existuje mnoho typů testování, z těch nejznámějších například Unit, Smoke, Monkey nebo Vulnerability.

Pro otestování systému byly použity dva hlavní typy testů popsané níže. Všechny testy byly provedeny ručně, tedy nebyly použity žádné testovací nástroje. Systém se podrobil testům a bylo detekováno několik chyb, kde některé byly následně opraveny.

4.5.1 Happy-Path

Tento typ testů kontroluje, zda při zadání správných hodnot systém nevyhodí chybu. Nepokouší se posílat špatné informace nebo mezní hodnoty, a proto pokrývá většinu případů uživatelů, kdy jsou vyplněné informace validní.

[14] Objektově Orientované Programování
[15] Dependency Injection

4.5.2 Ad-Hoc

Ad-Hoc se zaměřuje především na testování bez použití předpřipravených případů a jakékoliv dokumentace. Spoléhá na vlastní uvážení subjektu, který testy provádí. To je výhodné hlavně k odchycení případů, které formální testování nepokrývá.

4.5.3 Testovací scénáře

4.5.3.1 Registrace / Přihlášení

V tomto testu se zaručuje, že se uživatel nemůže přihlásit bez registrace, ani po registraci při zadání špatného e-mailu nebo hesla.

Kroky:

1. Uživatel klikne na tlačítko `Přihlásit se` v pravé horní části obrazovky
2. Zobrazí se formulář k vyplnění e-mailu a hesla
   a) Uživatel není registrovaný, nemůže se tedy přihlásit
   b) Pokusí se zadat nějaký e-mail a heslo, ale systém hlásí, že jsou údaje chybné
3. Klikne na tlačítko `Registrovat se`, se taktéž nachází v pravém horním rohu
   a) Při zadání nesprávného e-mailu, prázdného uživatelského jména nebo hesla kratšího než 5 znaků je zobrazena příslušná chybová hláška
   b) Zadáním správných údajů je uživatel registrován a může se přihlásit
4. Vrátí se na obrazovku s přihlášením
   a) Zadáním chybného e-mailu nebo hesla se zobrazí stejná chybová hláška. To zaručuje, že útočník nemůže použít hrubou sílu pro uhodnutí uživatelského jména.
5. Zadá správné údaje (stejné, které zadal při registraci)
6. Uživatel je úspěšně přihlášen

Pokryté případy užití: **U1**, **U2**
Zde nebyla zjištěna chyba a vše funguje, jak má.

4.5.3.2 Životní cyklus modulu

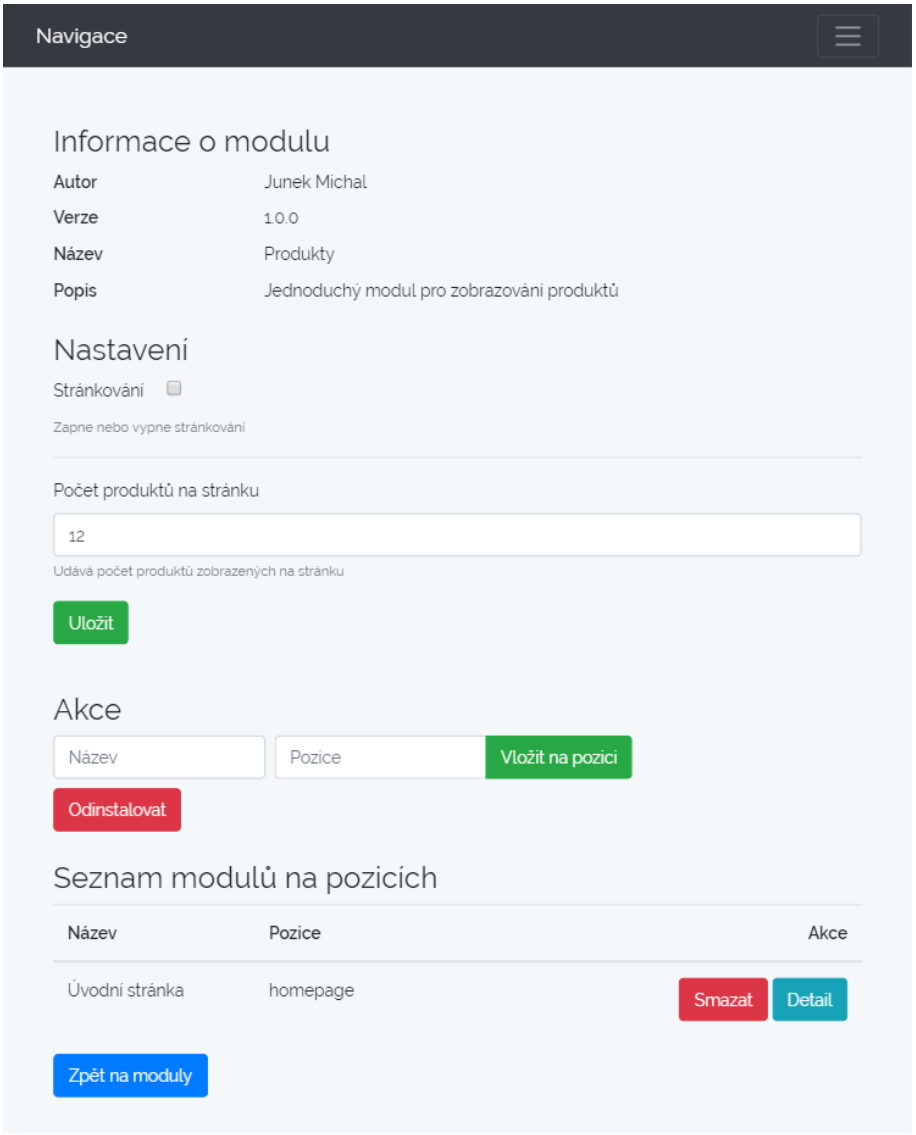
Tento test popisuje základní kroky pro ověření životního cyklu modulu. Součástí je instalace modulu, změna nastavení a následná odinstalace.

Pro tento scénář je zaručeno, že je přihlášen administrátorský účet a uživatel se nachází v administraci v sekci `Moduly`.

Kroky:

1. Uživatel vybere modul, který chce nainstalovat
2. Klikne na zelené tlačítko `Instalovat` u vybraného modulu
   a) Modul nahraje do databáze všechna potřebná data a spustí svůj instalační soubor
   b) V horní části se zobrazí hláška, že byl modul úspěšně nainstalován
3. Instalační tlačítko se změní na červené tlačítko `Odinstalovat` a přibude tlačítko `Detail`
4. Po stisknutí tlačítka `Detail` je uživatel přesměrován na detail modulu
5. Nyní jsou vidět informace o modulu, lze měnit nastavení nebo umisťovat modul na pozice
   a) Změna nastavení je probíhá vyplněním příslušných polí a stisknutím zeleného tlačítka `Uložit`
   b) Po odeslání formuláře s nastavením se zobrazí hláška s úspěchem
6. Uživatel klikne na červené tlačítko `Odinstalovat` pro odinstalaci modulu
   a) Spustí se odinstalační soubor, aby po sobě modul uklidil všechny uložené informace (kromě samotné složky modulu, ta zůstává)
   b) Zde je možnost použít tlačítko `Zpět na moduly` a odinstalovat modul stejným tlačítkem vedle modulu

Pokryté případy užití: **U4**, **U6** Tento test neprokázal žádné nedostatky a všechny kroky prošly bez problému.



Obrázek 4.1: Stránka detailu modulu

4.5.3.3 Životní cyklus instance modulu

Tento testovací scénář je rozšířením předchozího scénáře a cílem je ověřit správnou funkčnost vytvoření, smazání a editaci nastavení instancí modulu.

Zde je opět zaručeno, že je přihlášen účet s administrátorskými právy. Navíc se nyní uživatel nachází v detailu nainstalovaného modulu (krok 4. v 4.5.3.2).

Kroky:

1. Uživatel si vybere pozici, na kterou chce umístit modul
2. Vyplní vlastní pojmenování
3. Klikne na zelené tlačítko `Vložit na pozici`
    a) V horní části obrazovky ukáže hláška o úspěchu akce
4. Ve spodní části obrazovky se přidá řádek zobrazující informace o vytvořené instanci
    a) Součástí je vlastní název, pozice a tlačítka pro odstranění, detail a posunutí modulu
        i. Tlačítka pro posunutí se zobrazí pouze pokud existuje více modulů na stejné pozici
5. Klikne na tlačítko `Detail` pro navigaci na detail
    a) Zde je možnost upravit nastavení instance, pokud nějaké nabízí
        i. Po vyplnění nastavení a kliknutí na `Uložit` se nastavení uloží
6. Klikne na tlačítko `Zpět na modul`
7. Klikne na tlačítko `Smazat` u příslušné instance
    a) Ukáže se hláška o úspěšném smazání

Pokryté případy užití: **U3**, **U7**

V tomto testu byla objevena chyba. Instance modulů nejdou posouvat, nejsouli ve stejném modulu, protože se nezobrazí příslušná tlačítka. Dále zde chybí zobrazit, kde se instance modulu právě nachází, proto si uživatel nedokáže představit, před a za kterými instancemi bude seřazena. Oprava těchto drobných chyb je ponechána na budoucí rozvoj.

Kapitola 5

Zhodnocení

Výsledek práce, i přes to, že je na něm hodně možností, jak ho vylepšit, je na dobré cestě stát bok po boku s ostatními modulárními redakčními systémy. Tvorba modulů je velice jednoduchá i přes prozatím nerealizovanou dokumentaci, na rozdíl od existujících řešení, jako například u CMS Joomla, kde dokumentace sice je, ale je nedostatečná a proto tvorba modulů trvá déle než u ostatních systémů.

5.1 Evolvabilita a modularita

Evolvabilitu zaručuje především systém modulů, které jsou mezi sebou nezávislé a jednoduše se díky použitému frameworku upravují nebo rozšiřují. Tyto moduly je možné volně vkládat na pozice a každou takovou instanci dále nastavit. Je proto možné mít například na jedné stránce informace o více podnikatelích najednou a následně je nezávisle na sobě upravovat nebo mazat. Výhodu mají programátoři, kteří použitému frameworku rozumí, protože se budou v kódu rychleji a jednodušeji orientovat. Doplnění funkcionality do modulu nebo vytvoření nového je snadné, proto nebude drahé si na to najmout programátora.

5.2 Technologie

Systém funguje – až na pár chyb – dle navrženého konceptu a realizace hlavně díky šablonovacímu engine Blade, který je rychlý, bezchybný, jednoduše ovladatelný a umožňuje funkce, které by jinak zabraly spoustu času naprogramovat tak, aby fungovaly. Dále je hodně využit princip routeru starající se o přepis URL, zobrazování dat a v neposlední řadě dependency injection poskytující data všem strukturám, které si je vyžadují. Všechna funkcionalita ale není zajištěna pouze frameworkem. Dynamické nahrávání controlleru fun-

guje na principu jazyka PHP instanciovat třídy, jejichž jména jsou uložena v proměnných.

5.3 Technické požadavky

Náklady na provoz webu, na kterém bude hostován tento systém, mohou být minimální. Pokud vlastník nezamýšlí hostovat například opravdu velké (více jak 10 000) produktů, nebude problém web spustit na serveru s 128 MB RAM. To je o mnoho méně než konkurenční CMS, kde WordPress po instalaci několika rozšíření bez problému spotřebuje 512 MB RAM, a Magento potřebuje minimálně 2 GB.

Systém podporuje databázové servery MySQL, PostgreSQL, SQL Server a SQLite, které podporuje většina známých hostingových služeb, a nejeví se tedy jako překážka. SQLite je souborový databázový server, není proto potřeba se připojovat na externí.

5.4 Dopad na podnikatele

Podnikatelé využívající tento systém by měli zaznamenat rozšíření povědomí o jejich podnikání díky možnosti umístit na webové stránky informace o svém podnikání, jako například seznam produktů.

Na sociálních sítích se často můžeme setkat s poptávkou po firmách ohledně potencionální zakázky, kde lidé pak doporučují vhodně firmy pomocí odkazu na webové stránky. Potencionální zákazník si pak může sám zhodnotit, jestli bude firma vyhovovat jeho požadavkům na cenu a případnou kvalitu podle referencí, které si většinou firmy na své weby umisťují.

Naopak pokud se jedná o kamenný obchod, lidé určitě ocení možnost online procházení produktů. Hlavně pokud se jedná o obchod, který se nachází v jiném městě.

Když se podnikatel rozhodne pro živnost zabývající se chovatelstvím, může mu webová stránka poskytnout prostor, kde může snadno lidem odpovídat na jejich dotazy ohledně chovu. Zároveň může mít na webu rubriky s různými zajímavými články a radami, které ocení i lidé, kteří si od tohoto podnikatele nic nekoupili a narazili na tyto články náhodou. Tímto si může takový podnikatel získat na popularitě a může si získat možné budoucí zákazníky.

Závěr

Práce splňuje požadavky zadání na samotný systém. Je nutné vylepšit uživatelské prostředí, aby se jím lépe navigovalo, a přidat podpůrné funkcionality. Dalším krokem by poté byla online dokumentace a návody ve formě videa.

Rozhodnutí využít framework se osvědčil díky jednoduchosti implementace a výsledné rychlosti celého systému, který bez problému zvládá načítat i větší množství dat.

Systém se nachází ve stavu funkčního prototypu, kde bylo vytvořeno několik základních modulů pro demonstraci jejich implementace a rozhraní. Po inspekci stávajících modulů je možné jednoduše vytvořit další se specifickou funkčností. Následným rozvojovým milníkem je pak implementace systému prvotní instalace, proto je v tuto chvíli nutná alternativní, pro uživatele méně příjemná, instalace editací konfiguračních souborů. Vylepšit do budoucna by šlo i efektivnější využití frameworku, například předělat instalační soubory modulů do tzv. `Migration`.

Literatura

[1] Čápka, D.: 1. díl - Popis MVC architektury. [online], [cit. 5. 5. 2018]. Dostupné z: https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury

[2] Saleh, H.: MVVM architecture, ViewModel and LiveData (Part 1). [online], [cit. 12. 5. 2018]. Dostupné z: https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1

[3] Joomla: Requirements for Joomla! 3.x. [online], [cit. 17. 4. 2018]. Dostupné z: https://downloads.joomla.org/technical-requirements

[4] WordPress: System requirements. [online], [cit. 17. 4. 2018]. Dostupné z: https://wordpress.org/about/requirements/

[5] W3Techs: Most popular content management systems. [online], [cit. 17. 4. 2018]. Dostupné z: https://w3techs.com/

[6] Magento: Technology stack requiremenrs. [online], [cit. 5. 5. 2018]. Dostupné z: https://devdocs.magento.com/guides/v2.2/install-gde/system-requirements-tech.html

[7] Joomla: License. [online], [cit. 17. 4. 2018]. Dostupné z: https://tm.joomla.org/joomla-license-faq.html

[8] WordPress: License. [online], [cit. 17. 4. 2018]. Dostupné z: https://wordpress.org/about/license/

[9] Sawers, P.: Wordpress now powers 30% of websites. *VentureBeat*, březen 2018, [cit. 17. 4. 2018]. Dostupné z: https://venturebeat.com/2018/03/05/wordpress-now-powers-30-of-websites/

[10] Magento: License / Trademarks. [online], [cit 5. 5. 2018]. Dostupné z: https://magento.com/legal/licensing

[11] Nette: Homepage. [online], [cit. 5. 5. 2018]. Dostupné z: https://nette.org/cs/

[12] Otwell, T.: laravel / laravel. [online], [cit 17. 4. 2018]. Dostupné z: https://github.com/laravel/laravel/blob/master/readme.md

[13] Trends, G.: Framework Search Popularity. [online], [cit. 5. 5. 2018]. Dostupné z: https://trends.google.com/trends/explore?date=2013-01-01%202018-01-01&q=%2Fm%2F0jwy148,%2Fm%2F09cjcl,%2Fm%2F04n23m7,%2Fm%2F0cdvjh

[14] Martinek, J.: Use Case. [online], [cit. 11. 5. 2018]. Dostupné z: http://wiki.knihovna.cz/index.php/Use_Case

[15] TutorialsPoint: MVVM – Introduction. [online], [cit. 11. 5. 2018]. Dostupné z: https://www.tutorialspoint.com/mvvm/mvvm_introduction.htm

[16] Spett, K.: Cross-Site Scripting. [online], [cit. 11. 5. 2018]. Dostupné z: http://people.cs.ksu.edu/~hankley/d764/Topics/SPIcross-sitescripting.pdf

[17] Halfond, W. G. J.; Orso, A.: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. [online], [cit. 11. 5. 2018]. Dostupné z: https://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.orso.ASE05.pdf

[18] Burns, J.: Cross Site Request Forgery - An introduction to a common web application weakness. [online], 2007, [cit. 11. 5. 2018]. Dostupné z: http://www.icir.org/vern/cs161-sp17/notes/CSRF_Paper.pdf

[19] Laravel: CSRF Protection. [online], [cit 12. 5. 2018]. Dostupné z: https://laravel.com/docs/5.6/csrf

[20] Laravel: Blade Templates. [online], [cit 17. 4. 2018]. Dostupné z: https://laravel.com/docs/5.6/blade

[21] Marešová, M. K.: POVINNÉ ÚDAJE NA WEBOVÝCH STRÁNKÁCH PODNIKATELŮ. [online], [cit. 12. 5. 2018]. Dostupné z: https://www.maceklegal.cz/povinne-udaje-na-webovych-strankach-podnikatelu.html

[22] Mannaert, H.; Verelst, J.; Bruyn, P. D.: *Normalized Systems Theory: From Foundations for Evolvable Software Toward a General Theory for Evolvable Design*. Kermt: Koppa, 2016, ISBN 978-90-77160-091.

[23] Bruyn, P. D.; Mannaert, H.; Verelst, J.; aj.: Enabling Normalized Systems in Practice – Exploring a Modeling Approach. [online], prosinec 2017, [cit. 3. 5. 2018]. Dostupné z: https://link.springer.com/content/pdf/10.1007%2Fs12599-017-0510-4.pdf

[24] Kilicdagi, A.; YILMAZ, H. I.: *Laravel Design Patterns and Best Practices*. Packt Publishing, červenec 2014, ISBN 978-1-78328-798-7.

Příloha **A**

Seznam použitých zkratek

URL Universal Resource Locator

MVC Model–View–Controller

MVVM Model–View–View-Model

CMS Content Management System

JSON JavaScript Object Notation

NS Normalized Systems

SoC Separation of Concerns

DVT Data Version Transparency

AVT Action Version Transparency

SoS Separation of States

DRY Don't Repeat Yourself

KIS Keep It Simple

OOP Objektově Orientované Programování

DI Dependency Injection

CSS Cascading StyleSheet

JS JavaScript

HTML HyperText Transfer Protocol

UML Unified Modeling Language

Příloha B

Obsah přiloženého CD

/
- `src` ..... zdrojové kódy implementace
- `uml` ..... zdrojový projekt pro Enterprise Architect
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `tex` ..... zdrojové LaTeX soubory textu práce
- `readme.md` ..... stručný popis obsahu CD a instalační informace