Mendelova univerzita v Brně
Provozně ekonomická fakulta

Automatická konverze HTML šablony do redakčního systému

Bakalářská práce

Vedoucí práce:
RNDr. Zuzana Prišťáková, Ph.D.

Jan Krmela

Brno 2018

Děkuji především své vedoucí, RNDr. Zuzaně Priščákové, Ph.D., za velkou ochotu a množství cenných a důležitých informací, které mi poskytovala při vedení této práce. Také děkuji své rodině za podporu.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Automatická konverze HTML šablony do redakčního systému**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 27. dubna 2020 .....

Abstract

Krmela, J. Automatic converting HTML theme to content management system. Brno, 2018

The bachelor thesis deals with converting any HTML theme into content management system, where users can edit website in browser without knowledge of programming. The first part analyzes current options for creating websites with non-technical skills. The following defines requirements and vizualizes the design. The last part implements web application and algorithm for converting HTML templates, allowing easy editing in browser without coding.

Keywords

Web application, content managment system, CMS, theme, templates, Nette framework, HTML, Javascript, Doctrine ORM, Git, Less

Abstrakt

Krmela, J. Automatická konverze HTML šablony do redakčního systému. Brno, 2018

Bakalářská práce se věnuje konvertování jakékoliv HTML šablony do redakčního systému, pomocí kterého mohou uživatelé editovat webové stránky z prohlížeče bez znalosti programování. První část práce analyzuje současné možnosti vytvoření webových stránek bez nutnosti programování. Dále jsou specifikovány požadavky a provedena vizualizace aplikace. V poslední části je implementováno řešení s algoritmem pro konvertování HTML šablon a tím je umožněna snadná editace z prohlížeče.

Keywords

Webová aplikace, redakční systém, CMS, téma, šablony, Nette framework, HTML, Javascript, Doctrine ORM, Git, Less

Obsah

1 Úvod here

1.1 Úvod do problematiky

Na internetu je k dispozici tisíce šablon, napsaných v jazyce HTML, které není možné upravit k vlastní potřebě, průběžně je aktualizovat a publikovat na internetu, bez odborných počítačových znalostí. Ve většině případů, jsou tyto HTML šablony propracované a mohou sloužit jako elegantní webová prezentace [1]. Bohužel v současné době není řešení, jak tyto šablony mohou lidé se základními počítačovými znalostmi využít. Uživatelé tedy musí vybírat z omezeného množství webových šablon, naprogramovaných pro konkrétní redakční systém [2].

HTML šablona obsahuje obrázky, CSS soubory, Javascript soubory a zdrojové kódy napsané v jazyce HTML. Všechny tyto části dohromady tvoří webovou stránku, kterou zobrazují webové prohlížeče. Editace webové stránky je možná změnou zdrojového kódu. Tato činnost ovšem vyžaduje odborné znalosti. Z toho důvodu běžní uživatelé využívají redakční systémy, ve kterých zvládne úpravu webové stránky každý, kdo má základní zkušenosti s používáním počítače. Webová stránka se pak edituje podobně jako dokumenty v textovém editoru.

Úprava webové stránky v redakčním systému je tedy velmi pohodlnou záležitostí. Jádro problému je ovšem v obtížnosti implementace HTML šablony do redakčního systému. Každou HTML šablonu je nutné speciálně upravit, aby redakční systém rozuměl dané šabloně a dokázal nabídnout uživateli komfortní editaci. Programátor tedy musí použít další programovací jazyk, nejčastěji PHP, kterým šablonu upraví, rozdělí ji do samostaných částí, vytvoří databázi a provede další speciální kroky pro konkrétní redakční systém. Tento proces je bohužel povinný a zdlouhavý. V současné době neexistuje řešení, jak bez této úpravy editovat HTML šablony pohodlně z prohlížeče, bez znalosti programovacích jazyků.

1.2 Cíl práce

Navrhované řešení je vytvořit webovou aplikaci, která by byla schopna zpracovat libovolnou validní HTML šablonu, nabídnout uživateli pohodlnou možnost editace z prohlížeče a zveřejnění takové webové prezentace. Důležitou podmínkou je, aby tuto činnost zvládla i osoba bez pokročilých počitačových znalostí.

V současné době dokáží upravit a publikovat HTML šablonu pouze uživatelé se znalostí programování. Navrhované řešení by umožnilo používání všech HTML šablon i pro méně zdatné počítačové uživatele.

2 Použitá terminologie

V této kapitole jsou stručně popsány základní informace pro správné pochopení práce a ujasnění použitých termínů. Při psaní práce je vycházeno převážně z tištěné literatury, internetových zdrojů a oficiální dokumentace k jednotlivým nástrojům, které jsou v průběhu procesu použity. Oficiální dokumentace občas není ke všem nástrojům k dispozici. V takovém případě je jediným řešením oficiální API dokumentace, kterou už každý kvalitní nástroj zveřejněnou má.

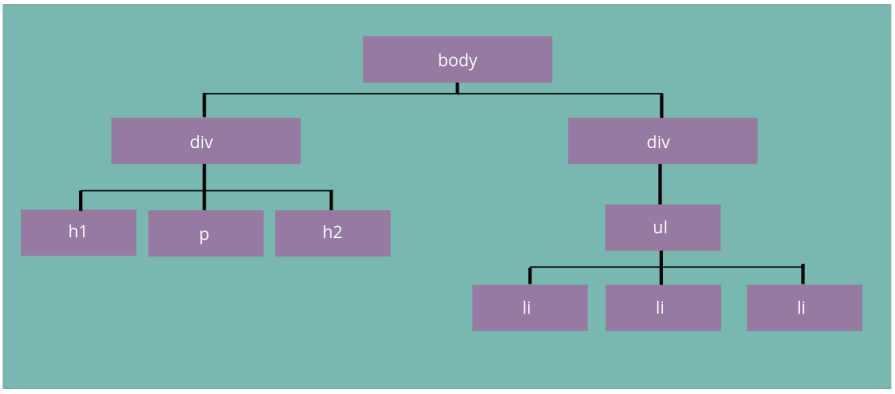
2.1 Definice principů HTML jazyka

HTML (Hypertext mark-up language) je sada daných značek, symbolů a kódů, vložených do dokumentu, který slouží pro zobrazování webových stránek. Každý element (tag), definuje strukturu webu, nebo její části [3]. Prohlížeče nezobrazují uživatelům tyto elementy, ale přímo prezentují obsah, zapsaný v tomto jazyce. HTML elementy jsou většinou párové. Tato vlastnost je i využita v této práci, při rozpoznávání struktury elementů [4].

Jednotlivým elementům lze přiřazovat atributy. Atributy se zapisují přímo dovnitř elementů, hned za úvodní název elementu. Oddělují se mezerou. Jeden z velmi důležitých atributů je unikátní identifikátor. Značí se pomocí klíčového slova id. Tento atribut hlavně využívá javascript a CSS pro komunikaci s daným elementem [5]. Velmi podobný je atribut třídy - class. Stejnou třídu lze přiřadit více elementům a jeden element může mít přiřazeno více tříd.

```
<div id="content" class="full-width">
    ...
</div>
```

Z pohledu datové struktury a teorie grafů jsou HTML elementy v dokumentu uspořádány jako strom [6]. Tomu je i přizpůsoben algoritmus procházení a analýzy HTML souborů, který je popsán v této práci.



Obrázek 1: Stromová HTML struktura

2.2 Konverze HTML jazyka

Konverze probíhá modifikováním HTML souborů do PHP podoby, která lze editovat z prohlížeče. Při konverzi je nutné dodržet několik zásad, aby výsledný soubor byl kompatibilní se všemi použitými nástroji zaručující požadovaný výsledek.

S ohledem na možnosti editoru v prohlížeči musí být části stránky rozdělené do jednotlivých editovatelných bloků. O správné rozpoznání editovatelných částí stránky a její rozdělení do jednotlivých bloků se stará konvertor. V konvertoru se obsah editovatelných bloků ukládá do databáze a nahrazuje se správnou syntaxí, která v případě potřeby, je schopna do stránky dosadit požadovaný obsah. Tyto bloky potřebují do svého nadřazeného elementu vložit identifikátor. Konkrétní identifikátor následně slouží pro uložení daného bloku k původní instanci.

Vzhledem ke stromové struktuře HTML je vhodné využít rekurzivní algoritmus pro analýzu celého dokumentu [7]. Takový algoritmus může být při velké a složité HTML struktuře časově náročný [8].

2.3 Porovnání konvertorů

V současné době není na internetu volně k dispozici aplikace, která by byla schopná automaticky, bez zásahu uživatele převést HTML šablonu do systému, kde by byla možná editace pouze z prohlížeče, bez znalosti programování.

Existují řešení, nabízející editaci z prohlížeče, ale jen s nutností předchozí úpravy HTML souborů. Uživatel bez znalostí programování není schopen zveřejnit a upravovat staženou HTML šablonu pomocí současných řešení. V této podkapitole jsou srovnány jednotlivé alternativy.

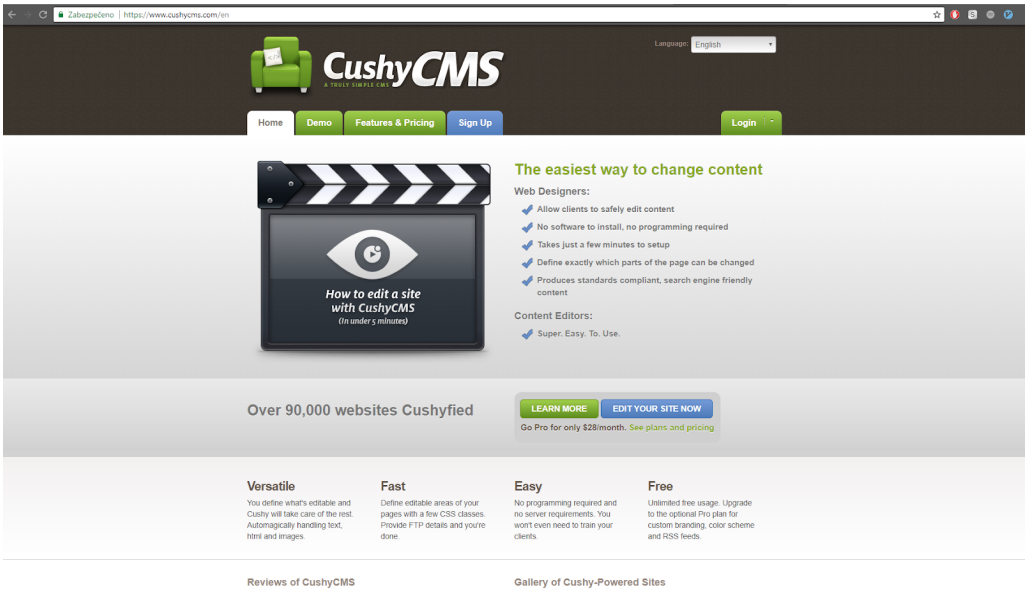
Surreal CMS

Tento redakční systém umožňuje editaci vlastní webové šablony z prohlížeče s WYSIWYG editorem. Před editací je ale nutné všechny zdrojové soubory šablony upravit. Každému elementu, který uživatel chce upravovat, musí přidat speciální třídu [9]. K této aplikaci není potřeba vlastnit hosting. Vše běží na serveru poskytovatele. Cena tohoto řešení přijde uživatele na 30 dolarů měsíčně, při středně velkých webových stránkách. Aplikace se nachází na adrese http://www.surrealcms.com.

Obrázek 2: Ukázka webu Surreal CMS

CushyCMS

Velmi podobná aplikace jako předchozí Surreal CMS. I zde je nutná editace HTML souborů pomocí přidání atributů třídy. Měsíční poplatek je 28 dolarů. Aplikace se nachází na adrese www.cushycms.com.



Obrázek 3: Ukázka webu CushyCMS

Běžné redakční systémy

Oblíbené redakční systémy jako Wordpress, Joomla, nebo Drupal nejsou vhodné

k porovnání. Tyto redakční systémy nesplňují hlavní požadavek a smysl této práce - možnost použití HTML šablony v neupravené podobě. Zmíněné redakční systémy podporují pouze speciálně upravené šablony pro konkrétní redakční systém. Takovou úpravu by běžný uživatel nezvládl. Jediná možnost je pak využít již vytvořené šablony pro tyto redakční systémy, kterých ale není mnoho. I z toho důvodu lze vidět spoustu stejných webových stránek pro různé firmy - jen s jiným obsahem.

Dalším problémem pro běžné uživatele může být instalace tohoto redakčního systému na vlastní hosting. I pro tyto činnosti méně zkušení uživatelé často kontaktují profesionální firmy, které jim pomohou.

V této práci je redakční systém velmi důležitý pojem. Cílová skupina aplikace jsou lidé, kteří nemají hluboké počítačové znalosti a právě redakční systém této skupině lidí velmi usnadňuje editování webových stránek. Redakční systém používá podobný princip úpravy webové stránky jako textové editory.

2.4 Návrh řešení webové aplikace

Analýza webové aplikace nebo informačního systému lze provést pomocí několika metod. Velmi často se používají strukturované a objektové analýzy. Strukturovaná analýza je starší než objektová a používá se z ní zejména funkční a datové modelování [10].

Poté, co se začalo programovat i objektově orientovaným způsobem, se začala používat i analýza, která pomohla osvětlit problém z toho objektově orientovaného hlediska. Tento nový pohled představil i nové metody pro analýzu [11]. Sloučení funkční a datové stránky aplikace je podstatou objektově orientovaného návrhu.

Pro kvalitní návrh je vhodné použít nástroje, které umožní správný a komplexní pohled na daný problém. Tyto požadavky splňuje jazyk UML. Jazyk UML definuje jasná pravidla pro vytváření a modelování diagramů. Různé diagramy přináší různé pohledy na problém. Jedině tak je možné dosáhnout optimálního návrhu. [12]

2.5 Redakční systém

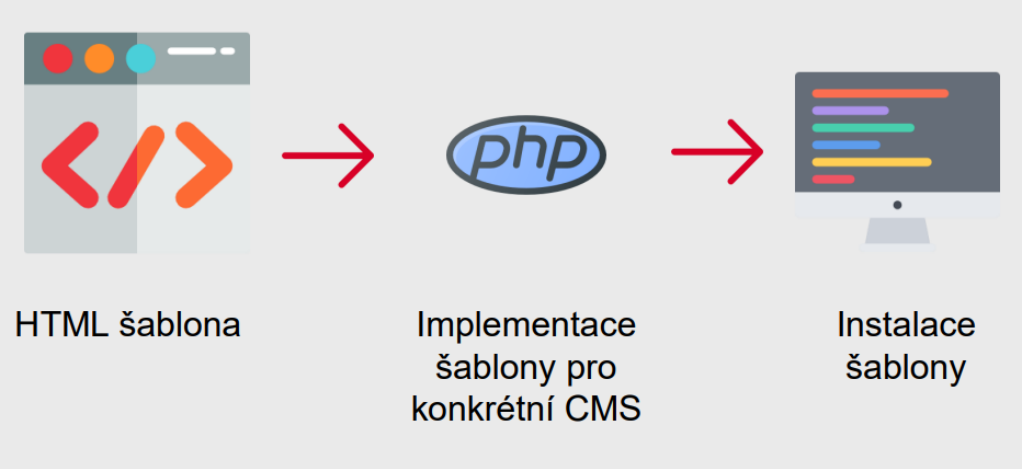
Redakční systém je webová aplikace, které má rozhraní pro správu, konfiguraci a editaci webové stránky. Její hlavní účel je, aby si každý uživatel mohl sám upravovat webovou stránku, bez nutnosti delegovat tuto práci programátorům [13]. I programátoři preferují editaci webových stránek z redakčního systému, protože často je taková editace komfortnější a rychlejší, než editace HTML souborů [14].

#	WEBSITES USING	MARKET SHARE %	ACTIVE SITES	# OF WEBSITES IN MILLION
1	WordPress	59.9 %	26,701,222	239,139
2	Joomla	6.6 %	2,009,717	13,480
3	Drupal	4.6 %	964,820	23,330
4	Magento	2.4 %	372,915	12,095
5	Blogger	1.9 %	758,571	15,779
6	Shopify	1.8 %	605,506	11,587
7	Bitrix	1.5 %	200,210	3,925
8	TYPO3	1.5 %	582,629	3,568
9	Squarespace	1.5 %	1,390,307	9,799
10	PrestaShop	1.3 %	262,342	2,099

Obrázek 4: Oblíbenost redakčních systémů (Převzato z websitesetup.org)

Nejpopulárnější redakční systém v roce 2017 byl Wordpress [15]. I přes technické nedokonalosti a bezpečnostní díry jej využívalo, dle statistik Google Trends, 239 milionů webových stránek.

Následující obrázek popisuje obvyklý princip implementace šablony do redakčního systému. Tuto metodiku využívají všechny výše uvedené redakční systémy, včetně nejpopulárnějšího Wordpressu.



Obrázek 5: Současný proces v redakčních systémech

3 Definice problému

3.1 HTML šablona

Webová šablona je univerzální vzhled webové stránky. Je vytvořená programátory a designéry. Důležitý parametr dobré šablony je, aby s minimální modifikací, vyhovovala co největšímu počtu uživatelů. Šablona musí být validní, dle posledních W3C standardů [2].

Šablony jsou programovány pomocí značkovacího jazyka HTML, je nutné pro jejich úpravu tento jazyk ovládat. Většina běžných počítačových uživatelů tento jazyk ovšem neovládá. V případě, že by nalezli na internetu vyhovující šablonu jejich účelu, nebudou schopni ji bez pomoci odborníků upravit pro svoje potřeby.

3.2 Publikování

K procesu vytváření a editování webové stránky jednoznačně patří i její zveřejnění na internetu [16]. Pro zveřejňování webových stránek se ve většině případů používá sdílený hosting. K zakoupenému hostingu uživatel dostane údaje k přístupu na úložiště přes FTP (File Transfer Protocol). Na základě těchto údajů je nutné připojit se k úložišti a přesunout tam webové stránky. Dále je nutné zakoupit a nakonfigurovat doménu. I přes poměrně složitý postup ještě není dosaženo optimálních výsledků. Je nutné splnit ještě několik kroků pro zabezpečení webových stránek.

Tato činnost může méně zkušeného uživatele odradit a raději tento problém řeší obrácením se na odbornou firmu. Proto je nutno i tuto činnost zjednodušit lidem, kteří si chtějí vytvořit, nebo editovat webové stránky a nemají potřebné znalosti. Mezi nejjednodušší řešení tedy patří, když uživatel nebude nucen mít vlastní hosting, který je nutný konfigurovat, přesouvat tam soubory a podobně, ale naopak, když bude všechno obsluhovat jeden společný server redakčního systému, o který se nebude muset uživatel starat.

3.3 Současné možnosti

Osoba bez odborných počítačových znalostí, která se chce prezentovat na internetu webovými stránkami, má v současné době jen několik možností. Jedna z nich je vytvořit webové stránky pomocí redakčních systému, jako je Wordpress nebo Joomla. Tato možnost je omezující z důvodu nutnosti využít šablony, které jsou přímo naprogramovány pro tento redakční systém. Takto upravených šablon není na internetu dostatek. I z toho důvodu se nezřídka stává, že na internetu lze nalézt stejné webové stránky, jen s odlišným textem. Další možností, finančně velmi nákladnou, je vytvořit poptávku u profesionální firmy, zabývající se tvorbou webových stránek na míru [17].

4 Metodika práce

Tato kapitola uvádí postup při řešení problému. Stručně jsou popsány jednotlivé postupy a důvody zvolení těchto postupů. Řešení problému lze rozdělit do tří hlavních skupin, jak již bylo naznačeno ve struktuře práce.

Před zahájením dalších kroků vedoucích k řešení problému, byl proveden sběr informací z literárních zdrojů, které pomohli pochopit následující důležité souvislosti. Nejdůležitější objekt této práce je redakční systém, proto byla velká část studia věnována právě redakčním systémům, jejich funkcím, dostupností a porovnáním jednotlivých vlastností redakčních systému.

Při studiu byla nutná velmi podrobná znalost o HTML, sémantice, struktuře i historii. V návaznosti na toto studium bylo prozkoumáno i velké množství HTML šablon, které byly staženy zdarma z internetu.

Po tomto studiu byla vypracována teoretická část, kde důležité poznatky, vázající se k této problematice, byly na základě literárních zdrojů shrnuty do jednotné podoby. Takto zpracovaná teoretická část má za cíl pochopit a dát do souvislostí informace nezbytné pro správné proniknutí do dalších částí této práce. Byly definovány principy HTML jazyka, konverze HTML jazyka a srovnána současné řešení, které jsou dostupné online.

Následně byl detailně zpracován a analyzován problém. S tím je vázáno i konkrétní definování požadavků. Při definování požadavků je velmi důležité věnovat zvýšenou pozornost, aby se jednotlivé požadavky shodovaly s definovanými cíli. Také je potřeba dodržet hlavní zásady jako jsou měřitelnost, proveditelnost a testovatelnost. Definice požadavků vychází primárně z analýzy problémů a snaží se jednotlivé problémy vyřešit. V tomto případě se snaží najít mezeru na trhu redakčních systémů. Zjišťování funkcí, vlastností a ceníku bylo prováděno na základě internetových stránek daných služeb.

V reakci na definici požadavků byl vytvořen návrh řešení, který se dělí na několik pohledů. Výsledná aplikace a její funkčnost byla shrnuta do neformální a formální specifikace, která se dále dělí na funkční a nefunkční požadavky. V nefunkčních požadavcích byl na základě předchozí analýze stanovena funkčnost, kterou by mělo výsledné řešení vlastnit a ve funkčních požadavcích byly stanoveny minimální požadavky na hardware. Konfigurace byla stanovena na základě aktuálních statistik náročnosti středně velkých webových aplikací.

Návrh modelu obsahoval propracovanou analýzu všech podstatných diagramů v rámci této aplikace, pomocí kterých byla vizualizována funkčnost a návaznost vzhledem k jednotlivým částem aplikace. Při návrhu se vycházelo z uvedených požadavků.

Design aplikace byl navržen dle současných trendů a nových metod. Jako první byly vytvořeny drátěné modely, které určí rozložení jednotlivých prvků na stránce. Při tvorbě rozhraní byly brány ohledy na omezené počítačové schopnosti cílové skupiny, proto bylo vše navrhnuto maximálně jednoduše a intuitivně.

Před zahájením implementační části se zjišťovali informace na výběr vhodných nástrojů a metod. Informace byly získány především z diskuzních fór odborníků a odborných článků na internetu. Na základě tohoto informačního přehledu byly zvoleny technologie nejvhodnější pro implementaci aplikace. Zásadní části implementace byly shrnuty v rámci podkapitoly řešení specifických problému. Jedná se převážně o ukázky a popis kódu, který je velmi důležitý k vyřešení daných problémů, nebo se jedná o úplně unikátní řešení, které ukazuje nový pohled na problém.

Při testování bylo zajištěno, aby veškeré testy proběhly v souladu s poučkami, dostupných v literárních zdrojích. Typy testů byly zvoleny na základě rozsahu aplikace a použitých nástrojů. Výsledky testů byly porovnány s odhadovanými hodnotami. Implementace a provedení testů velmi zjednoduší případné rozšíření aplikace.

5 Návrh modelu

5.1 Analýza problému

Na internetu se nachází k dispozici spoustu redakčních systému pro editaci webových stránek. Hlavní problém těchto redakčních systémů je nutnost upravovat HTML šablonu tak, aby redakční systém byl schopen nabídnout uživateli editaci této šablony.

Každý redakční systém vyžaduje jiné nároky na úpravu HTML šablony. V populárním redakčním systému Wordpress je nutné HTML šablonu vhodně rozdělit do samostatných PHP souborů a doplnit potřebný PHP kód. Nejsnadnější dosavadní řešení, které jsou k dispozici na internetu, jsou i redakční systémy, ve kterých stačí do šablony doplnit pouze speciální CSS třídy elementům, které mají být v redakčním systému editovatelné. Jako příklad takového redakčního systému může být uveden CushyCMS.

Protože každý redakční systém obnáší specifické upravování HTML šablon pro svou potřebu, a takto upravené šablony mezi sebou nejsou kompabilitní, vývojáři těchto šablon, kteří pracují na pozicích vývojářů se zaměřením na vytváření univerzálních šablon, proto většinou nevyvíjí šablonu pro konkrétní redakční systém, ale vytvoří ji čistou, bez dalších označení, pouze v HTML5 formě. Dále už nechají na konkrétním uživateli, aby si ji upravil pro redakční systém, který se chystá použít. Ve skutečnosti tento proces probíhá tak, že uživatel koupí, nebo zadarmo stáhne, takto univerzální HTML5 šablonu z různých webových portálů a vytvoří zakázku pro firmu, která mu tuto šablonu převede do vybraného redakčního systému.

Zveřejnění webových stránek je další problém pro méně zkušené uživatele. Redakční systém je většinou nutné nainstalovat na vlastní hosting. Pro některé může být i samotný nákup domény a hostingu problém, se kterým se obrací na odborníky.

Mezi další problémy patří velká náročnost editace. Současné redakční systémy mnohdy obsahují příliš mnoho funkcí, které běžní uživatelé nevyužívají. Celý systém jim pak neumožňuje dobrou orientaci a působí na tento typ uživatelů zmateně. Ke zvládnutí správné editace webových stránek pak potřebují speciální kurzy.

5.1.1 Definice požadavků

Pro kvalitní implementaci aplikace je nutná konkrétní definice požadavků. Definici je nutno provést v souladu s hlavními cíli. Tímto způsobem je možno vytvořit úspěšnou aplikaci [18]. Bez této analýzy by implementace mohla stát větší množství financí a vynaloženého času na implementaci. Požadavky je nutné stanovit reálně, aby je opravdu bylo možno splnit. Navzájem by se požadavky neměly vylučovat. Konkrétnost požadavků zaručí, že požadavky budou splněny tak, jak je určila analýza.

Jednoduchá instalace šablony

Základ aplikace tvoří intuitivní a jednoduchá instalace šablony, která uživatele zbaví všech složitých činností. Za složité činnosti při instalaci šablony je považována

například úprava zdrojových kódů, vytváření databáze, konfigurace routování, nebo nastavování FTP. Uživatel nahrává do aplikace svůj, nebo stažený ZIP s webovou šablonou, ve které se nachází všechny potřebné soubory, jako jsou HTML soubory, CSS soubory, obrázky a další. O zbytek se stará aplikace a poté uživateli nabízí editaci této šablony.

Hostování aplikace na společném serveru

Aplikace dostupná prostřednictvím webové adresy. Pro uživatele odpadne nutnost ji kdekoliv instalovat.

Snadná editace

Editace umožněna z pohodlí prohlížeče. Mezi nejjednodušší úpravy dokumentů patří WYSIWYG editor. Na tento typ editorů je každý uživatel zvyklý i s minimální počítačovou zručností. Je totiž běžné, že téměř každý umí upravovat dokumenty v programu Office Word. Tento program používá stejný princip úpravy - WYSIWYG. Uživateli má možnost i snadné organizace stránek a SEO vlastností.

5.2 Návrh řešení

5.2.1 Neformální specifikace

Aplikace má splňovat velmi jednoduchou, rychlou a univerzální editaci libovolné HTML5 šablony. Od toho je také nutné odvodit specifikace.

Uživatel není zdržován složitými operacemi, ale na pár kliknutí je mu dostupná registrace a vytvoření webové stránky. Bez náročného studování návodu je možnost nahrát do aplikace balíček s HTML5 šablonou, kterou získá například stažením na jednom ze spousty webů, kde nabízí tyto šablony zdarma.

Takto nahraná šablona se automaticky nainstaluje, rozpozná editovatelné části stránky a přímo nabídne uživateli editovat text, vytvářet nové stránky, měnit URL adresy stránek a publikovat výsledný web. Cílení těchto funkcí je především na méně zdatné uživatele v technické oblasti, nebo na jednoduché vytváření a editaci webu pro události, osobní nebo firemní prezentace. Není cílem konkurovat s touto aplikací na přeplněném poli komplexních systémů pro vytváření složitých a propracovaných webových aplikací.

5.2.2 Formální specifikace

Funkční požadavky

Při vytváření funkčních požadavků je vycházeno z definice problému. Dané požadavky mají za úkol tyto problémy vyřešit v souladu s vytyčenými cíly práce. Požadavky se přímo vztahují na daný problém a cílovou skupinu, kterou v tomto případě tvoří méně zdatní počítačoví uživatelé. Také jsou tyto požadavky vybrány v reakcí na konkurenci. Snaží se najít chyby v konkurenčních řešeních, a tím zaplnit mezeru na trhu, ale také použít již funkce, na které jsou uživatelé zvyklí z jiných řešení.

- **Vytvoření webové stránky**
  Důležitá je rychlá možnost vytvoření nové webové stránky. Po uživateli je požadováno co nejméně kroků, aby se zachovala jednoduchost. V případě, že uživatel nemá svou vlastní webovou adresu, bude mít možnost zvolit subdoménu, na které bude jeho web dostupný.
- **Instalace HTML5 šablony**
  Šablonu (Theme), kterou si například stáhne zdarma z internetu, nainstaluje jednoduchým vložením ZIP balíčku do systému. Systém automaticky nainstaluje šablonu a připraví vše pro její editaci.
- **Editace**
  Za editaci se považují funkce editování stránky, editování textu, editace obrázků a uložení.
- **Publikace**
  Další požadavky jsou spojeny s publikací - to znamená uložení stránky, změna URL adresy a smazání stránky. A vše musí být zabezpečeno autentizací uživatelů.

Nefunkční požadavky

Mezi požadavky pro fungování aplikace patří počítač s jakýmkoliv operačním systémem, kde fungují internetové prohlížeče. Bezproblémový chod je zaručen v prohlížeči Google Chrome. Dále je nutné internetové připojení.

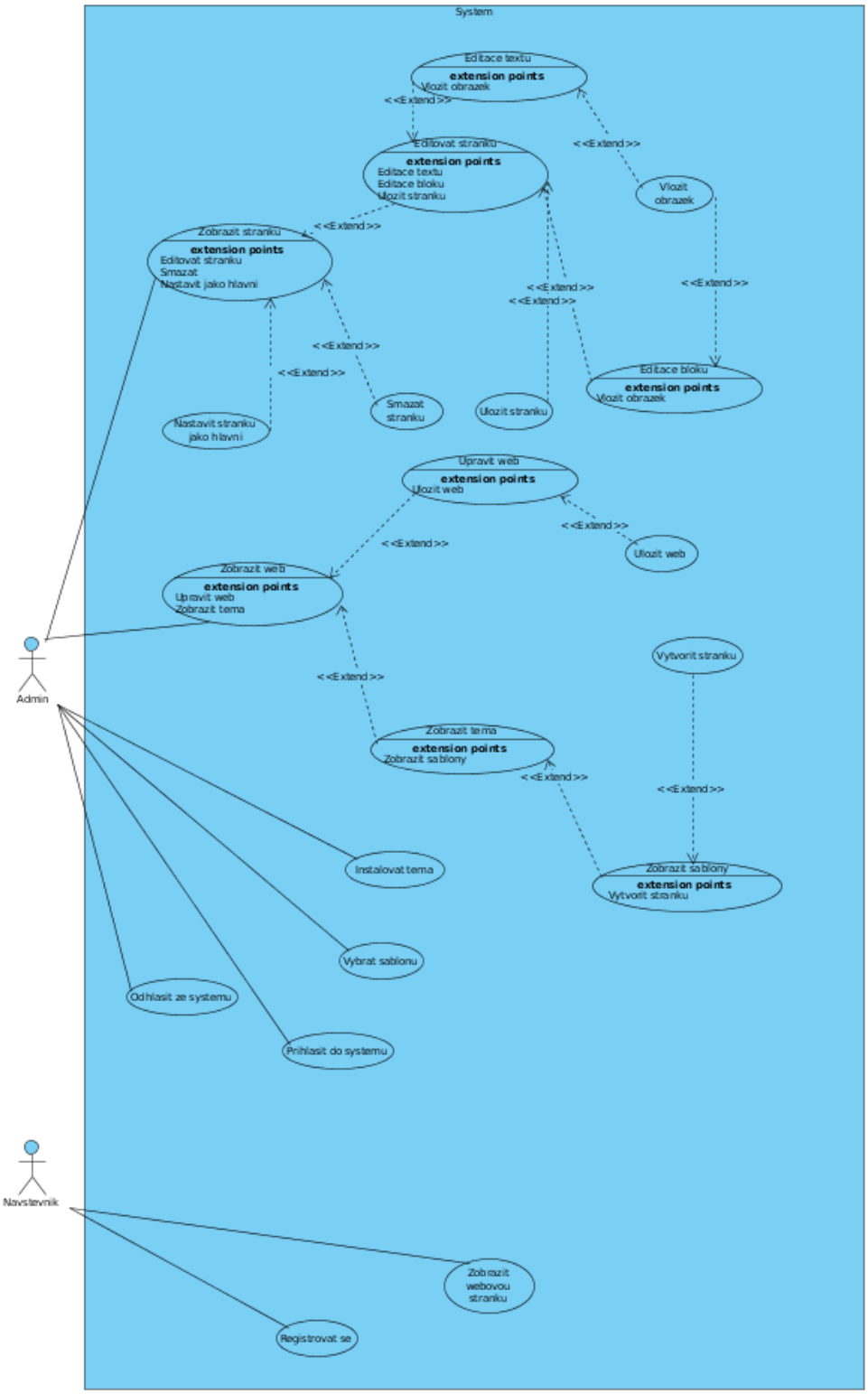
5.2.3 Návrh modelu

Návrh modelu slouží pro zobrazení požadavků a zobrazení dat. Požadavky jsou analyzovány v první části návrhu informačního systému. Popsány jsou i informace uložené v databázi. Na stukturované data byl použit entitně-relační diagram.

Pomocí diagramů UML jazyka byla specifikována a vizualizována aplikace. Tento návrh pomohl pochopení aplikace a umožní kvalitní implementaci aplikace. [19]

Diagram případů použiti (Use case diagram)

Aplikace rozlišuje aktory na dvě skupiny. Na Administrátory a Návštěvníky. Rozdíl mezi těmito skupinami tvoří stav registrace. Návštěvník je koncový uživatel, který může pouze prohlížet vytvořené webové stránky. Registrace udělá z Návštěvníka Administrátora, který už má podstatně rozsáhlejší možnosti činností v aplikaci.



Obrázek 6: Use case diagram

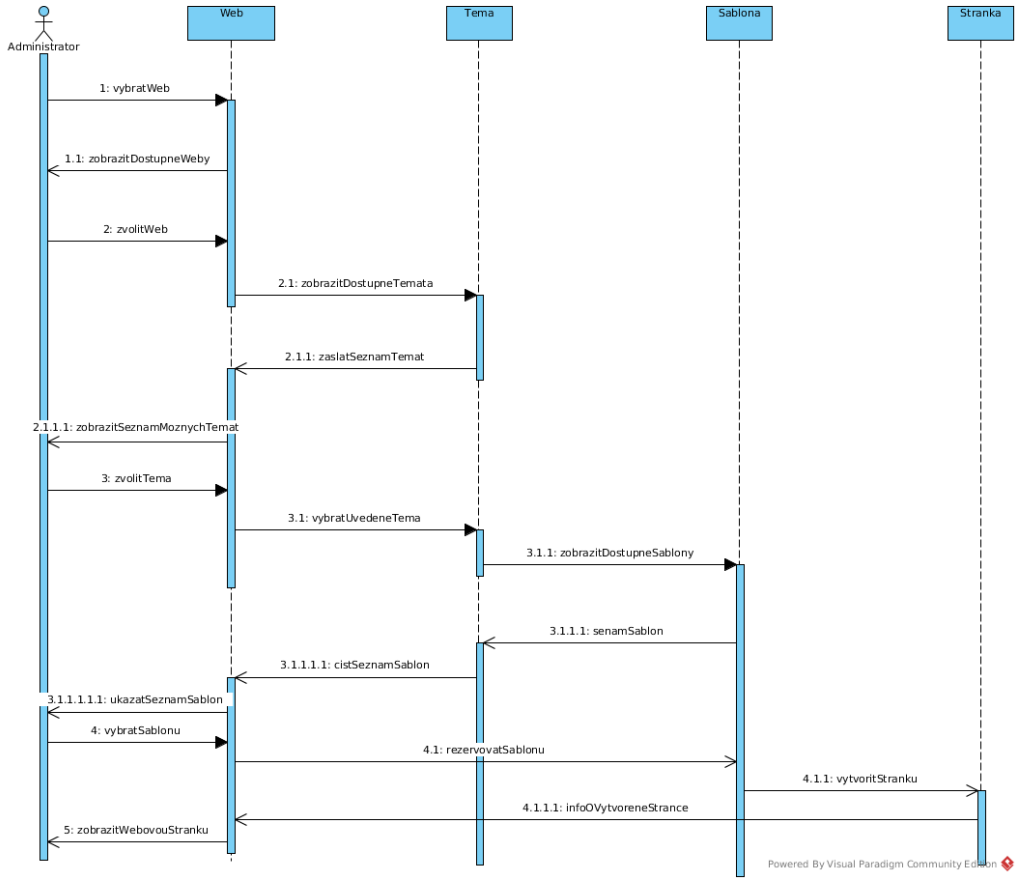
Registrovat se - každý návštěvník má možnost registrace do systému. Pro registraci je nutné vyplnit údaje. Registrace je co nejvíce zjednodušena. Po volbě přihlašovacího jména, případně emailu, je vyžadováno ještě heslo. Aplikace toto heslo zašifruje pomocí bezpečného algoritmu Bcrypt, který přidává k heslu i sůl, která podstatně zvyšuje bezpečnost.

Ihned po úspěšné registraci je uživatel automaticky přihlášen. Stává se z něj aktor Administrátor ve stavu přihlášen.

Instalovat téma - pro webovou stránku je nezbytné nainstalovat téma, podle kterého se web bude vykreslovat. Alternativně lze také vybrat z již nainstalovaných témat. Při instalaci tématu je nutné nahrát ZIP s vhodnými soubory. Dále je nutné zvolit název tohoto tématu. Volitelně lze zvolit zveřejnění šablony i pro ostatní uživatele. V tomto případě by si mohl téma vybrat a používat jakýkoliv jiný administátor pro svoji webovou stránku.

Sekvenční diagram - vytvoření stránky

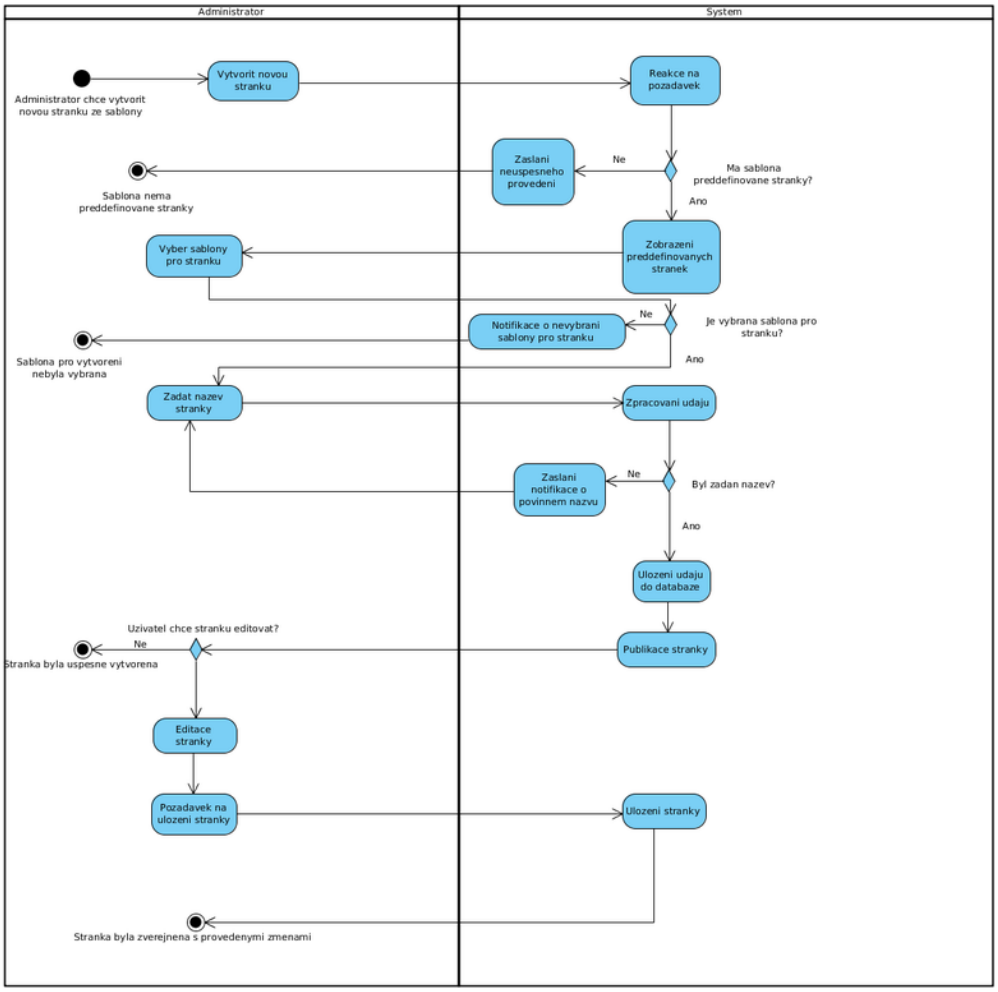
Následující sekvenční diagram popisuje use case vytvoření stránky, který obsahuje posloupnost procesů v čase při vytváření nové stránky (Page). Pro vytvoření stránky je nutné zvolit web, pro který se má stránka vytvořit. Každý web má předdefinované šablony (Templates), ze kterých lze vytvářet instance stránek. Z jedné šablony je možné vytvořit mnoho stránek. Uživatel musí vybrat danou šablonu pro novou stránku. Šablony jsou omezeny vzhledem k tématu (Theme). Nelze nabídnout jiné šablony, než které byly nainstalované společně s celým balíkem tématu. Například index.html tvoří jednu šablonu, about-us.html tvoří druhou šablonu a podobně. Při instalaci je uchován výchozí obsah těchto. Při vytvoření nové stránky se tento obsah vždy načte jako výchozí bod pro editaci.



Obrázek 7: Sekvenční diagram - vytvoření stránky

Diagram aktivit

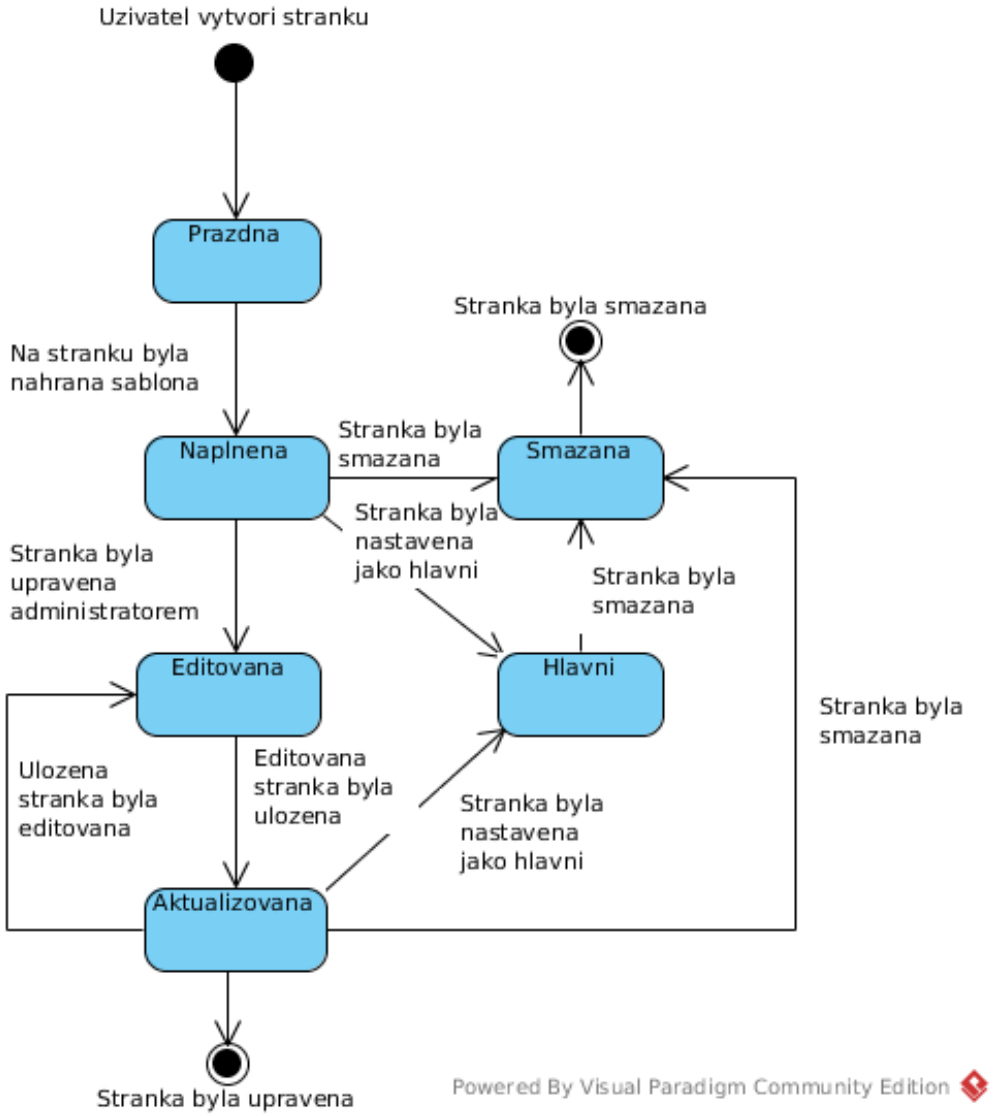
Další diagram popisuje rovněž vytvoření nové stránky, ovšem z pohledu diagramu aktivit. Tento proces začíná požadavkem uživatele Administrátora na vytvoření nové stránky. V další části zpracuje aplikace Administrátorův požadavek a nastávají dvě možnosti pokračování. Web potřebuje k vytvoření nové stránky nějakou šablonu. Pokud téma u daného webu není nastaveno nebo nainstalováno, nelze vytvořit stránku, protože nemá předlohu. Uživatel bude o tomto problému informován v podobě notifikace. V opačném případě pokračuje uživatel ve výběru šablony pro stránku. V dalším kroku musí uživatel zvolit název stránky. Tyto údaje pak aplikace uloží do databáze. Takto uložená stránka se přímo publikuje a nabídne se uživateli její editace. Editovanou stránku aplikace uloží a nastává konec procesu.



Obrázek 8: Diagram aktivit

Stavový diagram - vytvoření stránky

Vytvoření stránky z pohledu stavového diagramu zobrazuje všechny stavy, které mohou nastat. První stav po vytvoření požadavku na novou stránku je Prázdná. Tento stav znázorňuje, pokud na nově vytvořené stránce se nenacházejí žádná data. Na stránku je následně třeba nahrát šablonu. Po takové činnosti se změní stav z Prázdná na Naplněná. V tomto stavu se na stránce nachází výchozí hodnoty šablony, které se přenesly na tuto stránku, která byla původně prázdná. Dále je možné stránku smazat nebo editovat. Editovaná stránka lze uložit nebo nastavit jako hlavní. Po uložení je možno ji znovu editovat. Odlišnost mezi editovanou a naplněnou stránkou tvoří změna dat - naplněná stránka má pouze výchozí data, ale editovaná stránka má tyto původní data modifikované.



Obrázek 9: Stavový diagram - vytvoření stránky

Entitně relační diagram

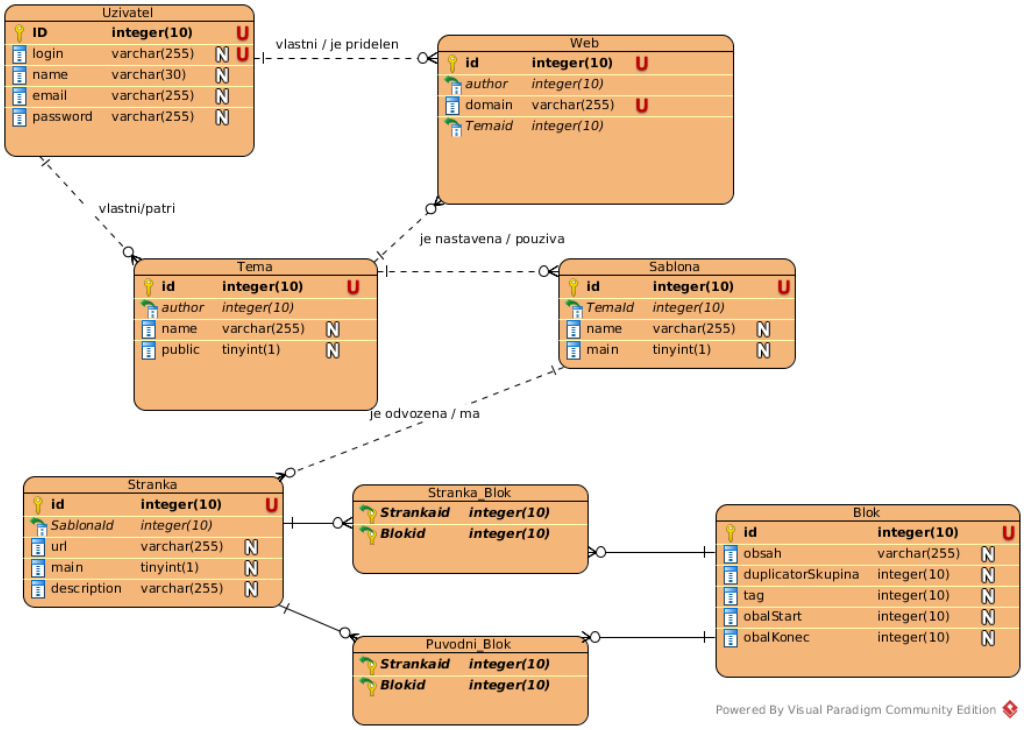
V diagramu se nacházejí tyto entity:

- Uživatel - obsahuje informace o uživatelích a jejich údajích. K uživatelům evidujeme jméno, přijmení, email a heslo.
- Web - web je tvořen doménou a jménem.
- Téma - tvoří každý nainstalovaný ZIP obsahující šablony. K tématu ukládáme i jeho jméno a vlastnost, která určuje, zda téma mohou použít i jiní uživatelé.
- Šablona - za šablonu je považován každý HTML soubor, který se nacházel v tématu. Má svůj název a informaci o tom, jestli je daná stránka vzhledem

k tématu hlavní.

- Stránka - stránka má svoji URL adresu a popis sloužící pro SEO optimalizaci. Dále ukazatel, zda je hlavní pro daný web.
- Blok - zaznamenává svůj obsah, který se vykresluje na stránce. Na stránce je rozlišován dle svého tagu. HTML kód, který obaluje blok je uložen v obalStart a obalKonec.

Vztahy mezi daty jsou popsány pomocí entitně relačního modelu. Uživatel může vlastnit více Webů. Následně web používá Téma. Téma má vztah s uživatelem, aby bylo jasné, kdo Téma nainstaloval. Každé téma může mít více Šablon. Z těchto Šablon je možné vytvořit několik stránek. Na stránkách leží bloky. Je nutné rozlišovat bloky a původní bloky - to jsou ty, které byly na stránce při její instalaci. Dále jsou používány při vytváření nových stránek, aby stránka nebyla prázdná. Vztah mezi Blok a Stránka je udáván pomocí Stranka_Blok a Puvodni_Blok.



Obrázek 10: Entitně relační diagram

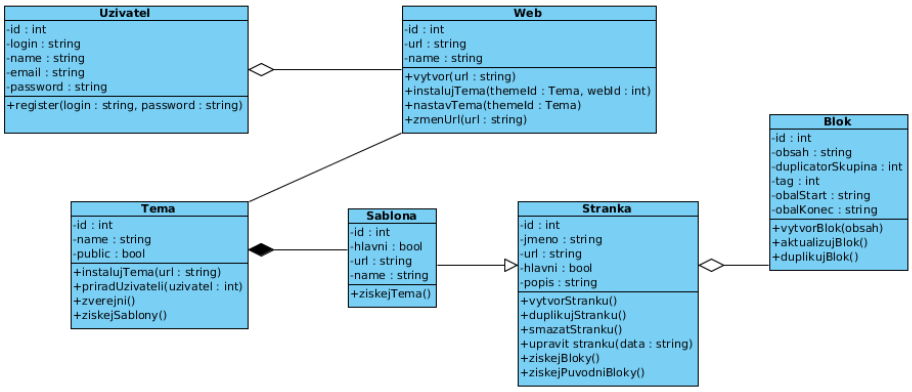
Diagram tříd

Aplikace byla také navrhnuta pomocí diagramu tříd. Statický pohled na systém umožňuje zobrazení souvislosti mezi třídami, obsahem tříd a vztahy.

- Web - třída zajišťuje definici názvu webové stránky a adresu, kde bude web dostupný. Metoda *vytvor(url:string)* přijímá jako parametr URL adresu, kte-

rá je nutná pro každou nově vytvořenou webovou stránku. Metoda *instalujTema(themeId:Tema, webid:int)* zajišťuje instalaci a přiřazení instalovaného tématu k danému webu. K přiřazení již nainstalovaného tématu je použita metoda *nastavTema(themeId:Tema)* a pro změnu URL *zmenaUrl(url:string)*.

- Uživatel - je definován pomocí identifikátoru, přihlašovácího jména, přijmení, emailu a hesla. Při registraci je používána metoda *register(login:string, password:string)*.
- Téma - mimo jména a identifikátoru obsahuje také stav, zda je téma veřejné. V případě veřejného tématu je dovoleno ostatním uživatelům použít toto téma. Instalace prorbíhá metodou *instalujTema(url:string)* a jako parametr obsahuje cestu k tématu. Téma lze přiřadit k danému uživateli, specifikováného v parametru metody *priradUzivatele(uzivatel:int)*.
- Šablona - jednoduchá třída obsahující informace o šabloně, která patří danému tématu. Téma, ke kterému šablona patří, je možno získat zavoláním metody *ziskejTema()*. Třída obsahuje také atribut, který definuje, jestli je šablona považována za hlavní. Taková šablona se načte při zobrazení webu jako první, pokud uživatel toto nastavení nezmění. Dále šablonu definují atributy jméno, URL a unikátní identifikátor.
- Stránka - třída obsahuje podobné atributy jako Šablona. Navíc je přidán atribut popis. Specifikovanými metodami lze vytvořit novou stránku, nebo stávající stránku duplikovat. Také lze stránku smazat a upravit. Metodami *ziskejBloky()* a *ziskejPuvodníBloky()* lze získat Bloky, které dále slouží pro editaci obsahu stránky.
- Blok - obsahuje editovatelnou část stránky. Určuje atributy *tag* a *duplicatorSkupina*, které zařazují Blok na vhodnou pozici ve stránce. Atributy popisující obalovací HTML elementy bloku se nazývají *obalStart* a *obalKonec*. Obsah Bloku je uložen v atributy *obsah*. Bloky se při instalaci vytváří metodou *vytvorBlok(obsah)*. Při editaci se používá metoda *aktualizujBlok()* a *duplikujBlok()*.



Obrázek 11: Diagram tříd

5.2.4 Design aplikace

Persony

Jakub Omáčka - je studentem střední průmyslové školy. Je mu 17 let. Bydlí na vesnici v blízkosti Brna. Rád se věnuje sportu a četbě. Má výborné znalosti cizích jazyků - zejména angličtiny a italštiny. Nejvíc času tráví tréninkem na fotbalovém hřišti. Má rád nové technologie, které ve volném čase zkoumá. Sleduje všechny velké konference, kde se ukazují nové produkty velkých technologických firem. Své počítačové znalosti považuje za dostatečné - zvládá i pokročilé činnosti, jako je například úprava fotografií. Programovat však neumí v žádném programovacím jazyku. Ve svém fotbalovém klubu je také pověřen správou a aktualizací webových stránek.

Veronika Krupičná - je matka na mateřské dovolené. Je jí 29 let. Žije v centru města Brna se svým manželem a svou dcerou. Před mateřskou dovolenou pracovala jako daňová poradkyně ve velmi malé firmě. Svůj volný čas tráví turistikou s rodinou po českých a slovenských horách. Její schopnosti práce na počítačích se omezují na základní používání. Dokáže komunikovat přes email s kolegy a svůj domácí počítač používá zejména k vyhledávání destinací na turistické výlety. Její telefon patří do kategorie smartphone, ovšem nevyužívá všechen jeho potencionál. Používá ho pouze k telefonování a pořizování fotografií.

Adam Drobný - 38letý programátor webových aplikací. Pracuje jako OSVČ a klienty si převážně hledá sám, ale má i klientelu, která už zná Adamovu spolehlivost a často se k němu obrací znovu. Žije sám v Praze, kde vlastní i malý byt. Pokud zrovna nepracuje, tak tráví svůj čas také na počítači, ovšem jinými činnostmi. Zkoumá nové technologie, čte technicky zaměřené články, nebo sleduje filmy. Vzhledem k tomu, že ho oslovují hlavně klienti, kteří požadují jen osobní web, či webovou prezentaci pro svou firmu, stává se jeho práce velmi monotónní. Pořád opakuje stejné činnosti k vytváření takových webů, pouze obměňuje vzhled.

Radek Nováček - má 62 let. Pracuje jako učitel zeměpisu na gymnáziu v malém městě u Olomouce. Jeho rodina s ním žije v rodinném domě v blízkosti školy. Ve volném čase se věnuje křížovkám a houbaření. S počítačem se naučil pracovat až v posledních letech při kurzu, který nabízela škola pro své pracovníky. Rád by vytvořil webové stránky pro své studenty, kde by mohl zveřejňovat informace a materiály pro své studenty. Ale s tvorbou webů nemá vůbec žádnou zkušenost.

UX aplikace

Při tvorbě jakékoliv webové aplikace je nutnost přizpůsobit náročnost ovládání aplikace očekávaných počítačových schopností cílové skupině [20]. V případě tohoto systému, má cílová skupina převážně základní počítačové znalosti. Proto interface tohoto systému byl navržen velmi jednoduše. V rámci zachování přehlednosti layout neobsahuje příliš mnoho tlačítek, odkazů a informací. Je nutné spíše předvídat a nabídnout uživateli takové informace, které právě hledá a potřebuje. V některých případech je dokonce i vhodné zvýšit uživatelský prožitek na úkor funkčnosti [21]. Příliš mnoho funkcí, tlačítek a informací může uživatele zmást na tolik, že nebude schopen aplikaci použít.

Abychom usnadnili orientaci je nutnost, aby uživatel pořád věděl, kde se nachází. K tomuto účelu slouží drobečková navigace nad obsahovou částí každé stránky [22]. Jako primární navigace slouží panel umístěný v pravé části stránky. Tento panel se nebude se skrývat pod hamburger-menu. Bude vždy viditelný.

Ke zrychlení orientace poslouží časté používání tematických ikonek. Ty mají za následek, že uživatel získá představu o dané informaci, tlačítku nebo akci během velmi krátkého času, aniž by musel text číst.

Zároveň je dodrženo pravidlo o konzistenci. Názvy tlačítek, akcí i objektů jsou v aplikaci pojmenovány vždy stejným názvem. Nepůsobí tedy na uživatele zmateně a nenutí ho přemýšlet.

6 Implementace modelu

6.1 Použité technologie

Pro kvalitní webové aplikace bývá ve většině případů použita kombinace mnoha technologií, jazyků a frameworků. Vhodný výběr je nutný provést na základě požadavků, jak má výsledná aplikace fungovat a vypadat. Je podmínkou vybrat takový jazyk, který má dobrou dokumentaci, nebo podporu komunity. Na základě toho je pak možno rychle řešit problémy. Vhodné je i myslet na budoucí rozšiřovaní aplikace. Jazyk nebo framework, o kterém je známo, že již nebude dále vyvíjen a podporován není dobrou volbou pro nové aplikace. Dále je také velmi důležité myslet na bezpečnost.

PHP 7

O back-endovou část aplikace se stará jazyk PHP, v aktuálně nejnovější verzi 7. Tento jazyk je skriptovací a při dodržovaní daných pravidel i velmi bezpečný. Fungování tohoto jazyka zajišťuje server pomocí Apache. Až výsledek práce PHP skriptů je přenášen k uživateli. PHP umožňuje i práci s databázovými systémy, například s MySQL. Jazyk není nutné předem kompilovat, což je jednoznačně výhodou [23]. Na internetu se nachází kvalitní dokumentace a spoustu řešení obvyklých problémů. Tento jazyk však může svádět k používání k tzv. špagetového kódu. Za tuto vlastnost na internetu sklízí hodně kritiky.

Framework Nette

Výše zmíněný špagetový kód řeší právě tento framework. Rozšiřuje PHP o mnoho funkcí, šablonovací systém, komponenty, filtry a mnoho dalšího. Nette používá MVC architekturu, která splňuje požadavky na kvalitní aplikaci. Mezi tyto požadavky patří znovupoužitelnost, bezpečnost a přehlednost. Nette programátora nutí psát kód správně a bezpečně.

Vývoj aplikace usnadňuje i díky přehlednému zobrazování chyb. Tento modul frameworku označuje Laděnka. Díky ní je možné přehledně zjistit, jak aplikace probíhá, kde nastala chyba, jaké funkce se volaly, apod. [24]

ORM Doctrine 2

Technologie ORM používá objektově-orientovaný přistup při komunikaci s relační databází. Je tedy možné manipulovat se záznamy z databáze jako s objekty, což je obrovská výhoda a usnadnění pro programátora. Dále programátor nemusí trávit tolik času psaní SQL dotazů. Je možné používat integrované funkce pro jednoduché činnosti. V kombinaci s Nette frameworkem nastává výborná symbióza, protože obě technologie přistupují ke kódu stejným objektově-orientovaným přístupem. Následná výměna dat je tedy velmi snadná a intuitivní [25]. Konkrétní implementaci pro Nette framework vytvořil v rámci své bakalářské práce Filip Procházka.

LESS

LESS je nástavba na CSS, pomocí které je možné používat v CSS proměnné, vnořený kód, dědičnost, výpočty a funkce. Musí se kompilovat při vývoji. Kompilace převede LESS na CSS. V tomto nástroji jsou napsány například i zdrojové kódy oblíbeného Bootstrap 3. Alternativa na LESS je SASS. Rozdíly mezi těmito nástroji jsou minimální. [26]

Git

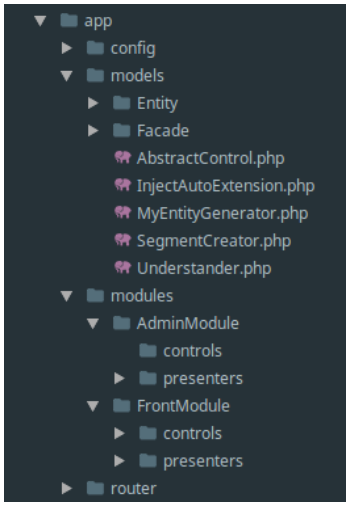
Verzovací nástroj Git umožňuje přehledné spravování verzí kódu. Mezi jeho hlavní výhody patří synchronizace kódu při práci v týmu, zálohování kódu, možnost vrátit se na starší verze a větvení kódu. Jelikož na této práci pracuji sám, tak nemohu ocenit hlavní výhodu synchronizace kódu. Ale zálohování kódu na serveru bitbucket.org, který nabízí vytvoření repozitáře zdarma velmi ocením.

6.2 Řešení specifických problémů

Struktura aplikace

Aplikace je uspořádána do dvou modulů. Dělí se na Front modul, který obsluhuje zobrazování aplikace pro nepřihlášené uživatele a zajišťuje vykreslování webových stránek vytvořených v této aplikaci pro návštěvníky. Druhý modul obsahuje kompletní administraci, včetně instalací nových šablon, vytváření webů a editování webových stránek.

Aplikace se drží doporučované struktury webových aplikací pro MVC architekturu. Následující obrázek zobrazuje adresářovou strukturu důležité složky app, která zajišťuje hlavní jádro aplikace.



Obrázek 12: Adresářová struktura aplikace

Datová vrstva

S využitím frameworku Doctrine 2, který implementuje ORM přistup nad relační MySQL databází byly sestaveny jednotlivé entity. Každou entitu reprezentuje jeden soubor s názvem entity a příponou .php. Doctrine používá velmi specifický zápis. Rozšiřuje základní použití atributů a getterů a setterů o anotace. Při správném použití anotací a atributů vygeneruje po zadání daných příkazu přes terminál v databází příslušné záznamy.

Mezi základní příkazy v terminálu pro Doctrine patří příkaz pro validaci:

```
php index.php orm:validate-schema
```

Příkaz pro vytvoření struktury databáze:

```
php index.php orm:schema-tool:create
```

Vytváření vazeb mezi entitami je umožněn pomocí oboustranné specifikace vztahu. Je nutné na obou entitách specifikovat inverzní a vlastní stranu.

```
/**
   * @ORM\ManyToMany(targetEntity="Block")
   * @ORM\JoinTable(name="pages_blocks",
   *      joinColumns={@ORM\JoinColumn(name="page_id",
↪  referencedColumnName="id", onDelete="cascade")},
   *      inverseJoinColumns={@ORM\JoinColumn(name="block_id",
↪  referencedColumnName="id", onDelete="cascade")}
   *      )
   * @ORM\OrderBy({"rank" = "ASC"})
   */
   protected $blocks;
```

Výše uvedený příklad zobrazuje anotaci atributu Blocks z entity Stránka (Page). Daný vztah je dle typu ManyToMany. Každá stránka má více bloků a blok patří na více stránek. V anotaci se nastavují veškeré podrobnosti tohoto vztahu, jako například název atributu, pro který má nastat sloučení, nebo co se má odehrát po smazání jedné ze stran tohoto vztahu. Pro následné zjednodušeni práce byla nastavena i anotace pro seřazení při výpisu. Instance se budou řadit dle atributu rank. Stejného výsledku lze dosáhnou i přímo pomocí SQL dotazu, ale zaznačení přímo v entitě v tomto případě značně zrychlí práci.

Instalace tématu

Za instalaci komprimovaného souboru ZIP, obsahující téma, který nahrál uživatel, je zodpovědný InstallPresenter.php. Po nahrání na server probíhá následující proces:

Vytvoření nové složky na serveru - název složky je generován dle původního názvu a přidáním aktuálního času.

```
$nameForZip = Strings::webalize($values->name.time());
```

Bezpečností kontrola - potencionálně nebezpečné soubory jsou odstraněny před zahájením dalších kroků. Rozbalení archivu do nové složky a cyklus reagující na nalezené HTML soubory.

```
foreach (Finder::findFiles('*.html')->in($this->pathToTemplate .
↪  $nameForZip) as $key => $fileInfo) {
  …
}
```

Pro každý nalezený HTML soubor je provedena skupina kroků, která zpracovává tento soubor. Cesta k souboru je použita jako parametr k volání metody loadTemplate z třídy Understander. Tato třída analyzuje HTML soubor, rozpoznává editovatelné části, nahrazuje HTML kód tak, aby bylo možné šablonu editovat z prohlížeče. Do InstallPresenteru vrací pole s daty o proběhlé analýze. Tyto data jsou následně uloženy do databáze. Detailní implementace třídy Understander je popsán níže.

Vytvoření záznamů v databázi - z dat vzniklých při analýze a údaji od uživatele je nově vytvořen záznam o tématu (Theme), šablonách (Templates) a je také vytvořena jedna nová stránka (Page) ze souboru index.html, která je přímo označena jako hlavní.

Přesměrování - pokud uživatel instaloval toto téma z již vytvořeného webu, je přímo přesměrován na administrativní stránku, kde může editovat tento web.

V případě, že nebyl zvolen web, pro který je téma instalováno, je přesměrován na hlavní stránku administrace. Vybrat web, který bude používat toto téma může až později.

Získání všech informací k webu - aby administrátoři dostali ke svým webových stránkám kompletní informace pro snadnou editaci webu, slouží WebsitePresenter.php. Jeho činností je získat informace o dané webové stránce a zpracovat je. Následně tyto informace odesílá do šablony.

Získání těchto informací je velmi snadné, diky správnému návrhu a použití ORM Doctrine 2. Následující ukázka demonstruje proces práce s daty napříč všemi vrstvami.

```
public function renderDefault($websiteId)
{
   $this->website = $websiteId;
   /** @var Website $website */
   $website = $this->entityManager->getRepository(Website::class)-
   ↪  ->findOneBy(['id =' =>
   ↪  $websiteId]);
   if ($website->getUser()->getId() !== $this->user->id){
```

```
            throw new \Error('No permission to do this!', 403);
        }
        $this->template->website = $website;
        if ($website->getTheme()){
            $this->template->templates =
            ↪  $website->getTheme()->getTemplates();
            foreach ($website->getTheme()->getTemplates() as $template){
                $templatesFromThisTheme[] = $template->getId();
            }
            $this->template->pages =
            ↪  $this->entityManager->getRepository(Page::class)->findBy(
                [
                    'website =' => $website->getId(),
                    'template' => $templatesFromThisTheme
                ]
            );
        }
        $this->template->themes =
        ↪  $this->entityManager->getRepository(Theme::class)->findAll();
    }
```

V ukázce vidíme metodu renderDefault(), která přijímá ID webu, ke kterému je třeba zobrazit informace. Zobrazují se kompletní data k webu. Mezi tyto data patří základní informace o webové stránce, její šablony a stránky. Navíc jsou přidány i informace o dalších dostupných šablonách, které je možné použít pro tento web.

Získané data jsou postupně předávány do šablony. Je také ověřováno, zda daný uživatel má právo tyto data získat.

Vytváření nové stránky - uživatel při vytváření nové stránky (Page) vybírá z předdefinových šablon (Template). Takto vybraná šablona použije svoje původní bloky a vloží je na nově vytvořenou stránku. O tento proces se stará metoda handleNewPage s parametrem ID šablony.

```
public function handleNewPage($templateId)
{
    $website = $this->entityManager->getRepository(Website::class)-
    ↪  ->findOneBy(['id =' =>
    ↪  $this->website]);
    /** @var Template $template */
    $template = $this->entityManager->getRepository(Template::class)-
    ↪  ->findOneBy(['id =' =>
    ↪  $templateId]);
    foreach ($website->getTheme()->getTemplates() as $themeTemplate){
        $templatesFromThisTheme[] = $themeTemplate->getId();
    }
```

```
    $mainPages =
    ↪  $this->entityManager->getRepository(Page::class)->findBy(
        [
            'website =' => $website->getId(),
            'template' => $templatesFromThisTheme,
            'isMain' => true
        ]
    );
    /** @var Page $page */
    $page = new Page();
    $page->setWebsite($website);
    $page->setName($template->getName());
    $page->setUrl($template->getPath());
    if (count($mainPages) == 0){
        $page->setIsMain(true);
    }else{
        $page->setIsMain(false);
    }
    $page->setTemplate($template);
    $defaultBlocks = $template->getDefaultBlocks();
    foreach ($defaultBlocks as $defaultBlock){
        $block = clone $defaultBlock;
        $this->entityManager->persist($block);
        $page->addBlock($block);
        $this->entityManager->flush();
    }
    $this->entityManager->persist($page);
    $this->entityManager->flush();
    $this->redirect('Website:default',[$website->getId()]);
}
```

Nejdříve je nutné v tomto procesu získat data z databáze o všech entitách, které se týkají této operace. Problematickou část tvoří rozhodnutí algoritmu, zda nově vytvořená stránka má být nastavena jako hlavní, neboli indexní. Každý web totiž musí obsahovat jednu hlavni stranu. V opačném případě by návštěvník nemohl web zobrazit.

V posledním foreach cyklu se duplikují původní data z šablony (Template) na novou stránku. Administrátor pak při první editaci nezačíná s prázdnou stránkou, ale rozložení základních prvků má již předdefinované a značně mu to zjednoduší editaci.

Algoritmus pro zpracování šablony

Jednotlivé HTML soubory v tématu zpracovává třída Understander. PHP má pomocnou třídu na práci s HTML elementy. Lze tedy procházet HTML jako objekty. Existuje spoustu pomocných metod pro identifikaci elementů, atributů a hodnot.

Také je možné tyto atributy editovat a přidávat nové. Celý dokument lze procházet jako datový typ strom. Cyklus prochází jednotlivé prvky a podrobuje je analýze.

Načtení struktury HTML dokumentu probíhá hlavně pomocí těchto příkazů. Uvedeny jsou zde pouze základní příkazy. V konečném výsledku jich je mnohem víc, aby bylo dosaženo optimálního řešení.

```
$this->doc = new DomDocument("1.0", "utf-8");
$this->doc->loadHTML($html);
```

Dále je nutné pro průchod všech prvků iterovat pomocí cyklu while.

```
$elements = $this->doc->documentElement;
$first = $elements->firstChild;
$this->inspectElement($first);
while ($first->nextSibling){
   $first = $first->nextSibling;
   $this->inspectElement($first);
}
```

Tento cyklus prochází pouze všechny prvky na stejné úrovní (sourozence). Dále se zjišťuje, zda daný prvek nemá potomky. Rekurzivně se postupuje dokud není dosaženo posledního prvku - listu. Analyzování prvku se dále rozděluje dle informacích o prvku a jeho okolí. Toto rozhodování je ponecháno na metodě inspectElement, kde se před analyzováním provede jeho formální úprava zavoláním funkce editPaths.

Dále je nutné zjistit, zda prvek patří do skupiny editovatelných elementů pomocí funkce isEditable. Ta vrací true nebo false, v závislosti, zda daný prvek splňuje předem definovaný seznam editovatelných prvků. Do něj patří například elementy všech nadpisů, odstavce <p>, obrázky <img>, odkazy <a> a podobně. Prvek splňuje test optimality i v případě, že sám editovatelný není, ale má potomky, kteří do pole editovatelných elementů patří.

Dále se zjišťuje na základě sourozenců, zda má element podobnou strukturu jako ostatní sourozenci. V tomto případě je zpracován jinak, než běžné prvky. Tyto typy prvků jsou vysvětleny níže.

```
private function inspectElement($element)
{
   $this->editPaths($element);
   if ($this->isEditable($element)) {
       if (($element->nextSibling) and ($this->compareChilds($element,
       ↪  $element->nextSibling))) {
           /** is duplicator (same as his siblings) */
           $this->processBlock($element, true);
```

```
        } elseif (($element->previousSibling) and
        ↪  ($this->compareChilds($element,
        ↪  $element->previousSibling))) {
            /** is last duplicator (same as his previous siblings) */
            $this->processBlock($element, true, true);
            $this->counter++;
            $this->duplicatorGroup = 0;
        } else {
            /** is normal block, replaced with variable */
            $this->processBlock($element, false);
        }
        /** if not, dive in to his childes recursively */
    }elseif ($element->hasChildNodes()){
        /** hey kids, i-will check you too, just for sure */
        foreach ($element->childNodes as $child){
            $this->inspectElement($child);
        }
    }
}
```

Editor vyžaduje vhodné zaznačení prvků, které má nabídnout k editaci. Regiony, ve kterých jsou editovatelné prvky musí mít speciální třídu a identifikační značku. Algoritmus tedy obsahuje velké množství kódu, který se snaží vhodně rozdělit šablonu do editovatelných regionů. V tomto textu jsou uvedeny pouze nejdůležitější části kódu. Další z těchto podstatných částí je nahrazení elementu PHP kódem (respektive Latte syntaxí pro Nette). Editovatelný obsah HTML elementů je uložen do databáze. Elementům obalující tento obsah je přiřazen atribut data-editable a data-name. Unikátní hodnotu získáváme z proměnné counter.

```
$element->setAttribute('data-editable',$this->counter);
$element->setAttribute('data-name',$this->counter);
$element->nodeValue =
↪  '{$blocks['.$this->counter.'][0]->getContent()}';
$this->counter++;
```

Data z původních elementů se získali pomocí následujících příkazů. Vše je ukládáno do pole. Po skončení se toto pole rozdělí do záznamů databáze a uloží.

```
$this->data[] = [
    'tag'=> $this->counter,
    'content' => $duplicator ? $this->doc->saveHTML($element) :
    ↪  $this->getInnerContent($element),
    'duplicatorGroup' => $duplicator ? $this->duplicatorGroup : NULL,
    'duplicatorWrapStart' => isset($tmpElement) ?  $tmpElement[0].'>' :
    ↪  NULL,
```

```
    'duplicatorWrapEnd' => isset($tmpElement) ?
    ↪  $tmpElement[count($tmpElement)-2].'>' : NULL,
    'rank' => $duplicator ? $this->duplicatorGroup : null
];
```

Rozpoznání duplikátorů

Duplikátory jsou podobné skupiny HTML elementů, ale s jiným obsahem. Na firemních webových stránkách to může být například sekce Naši zaměstnanci. Obvykle má zde každá osoba fotku, jméno a pracovní pozici. Administrátorovi, který edituje obsah webu, by usnadnilo práci, kdyby mohl přidat další osobu, změnit pořadí těchto osob, nebo ji smazat. Pokud algoritmus vhodně rozpozná tyto části je možné nabídnout administrátorovi tyto možnosti.

Skupiny duplikátorů rozpoznává metoda comapeChilds. Porovnává, zda uzly mají stejné potomky. Nebere se ohled na HTML atributy, nebo obsah. Rozhoduje struktura HTML uzlu.

```
private function compareChilds(DOMNode $a, DOMNode $b)
{
   $result = false;
   if (($a->nodeName == $b->nodeName) && ($a->hasChildNodes()) &&
   ↪  ($b->hasChildNodes())) {
       $a~= new DOMNodeList($a);
       $b = new DOMNodeList($b);
       $result = true;
       $aCount = $a->length;
       for($pos=0; $pos<$aCount; $pos++) {
           if ($a->item($pos)->nodeName != $b->item($pos)->nodeName){

               $result = false;
           }
       }
   }
   return $result;
}
```

Duplikátory se nahrazují do šablony jiným řetězcem, než klasické editovatelné bloky. Je nutné je nahradit foreach cyklem, protože obyčejné vypsání proměnné by zde nestačilo. Musí se vypsat všechny duplikátory, které mají stejnou identifikační značku. Nahrazení se provádí pouze u posledního duplikátoru. Všechny předchozí jsou smazány.

```
if($last == true){
   $element->parentNode->setAttribute('id', 'duplicated');
   $element->parentNode->nodeValue = '
```

```
        {ifset $blocks[' . $this->counter . ']}{foreach $blocks[' .
↪   $this->counter . '] as $content}
            {$content->getContent()}
        {/foreach}{/ifset}
    ';
}

$this->duplicatorGroup++;
```

6.3 Shrnutí implementačních kroků

Pro efektivní splnění požadavků na aplikaci byly provedeny tyto kroky:

- zvolení vhodných technologiích
- navrhnutí struktury projektu
- vytvoření entit
- vygenerováni databáze
- naprogramování administrace a autentizace
- naimplementování algoritmu pro zpracování HTML šablon
- sestrojení zobrazování webu pro návštěvníky

Předchozí analýza značně urychlila a usnadnila celou implementaci a nebylo nutné upravovat implementační části z důvodu rozrůstajícího se aplikace.

6.4 Přístup k aplikaci

Aplikace je dostupná na webové adrese **http://jankrmela.cz/**. Přihlášení je možné po rychlé registraci. Po úspěšné registraci následuje průvodce, který uživateli pomůže se základním nastavením adresy nového webu, instalací šablony a její editací.

7 Testování modelu

Testování je téměř nutnost každé větší webové aplikace. Pomocí správně zvolených a provedených testů je možné odhalit chyby v kódu mnohem dříve. Značně zlepšují kód a usnadňují vývoj. V některých případech dokáží odhalit závažné chyby a bezpečnostní díry, které by bez testů nebyli včas odhaleny. Mohla by tak vzniknout značná finanční ztráta, při publikování aplikace.

Důvody zvolení konkrétních testů jsou objasněny v následující podkapitole u jednotlivých testů. Rovněž jsou popsány očekávané výsledky a jejich vliv na ověření řešení aplikace.

7.1 Použité testy

Jednotkové automatické testy

Ověřování funkčnosti kódu zajišťují jednotkové testy. Používají se pro testování malých separovatelných částí. V objektovém programování se za tyto části dají považovat třídy, nebo funkce a metody. Průběh testu tvoří porovnávání očekávaného výsledku s reálným výsledkem.

Je vhodné tyto testy psát průběžně ke každé dokončené části a vždy po přidání nové funkcionality, nebo provedení změny všechny dosavadní jednotlivé testy spustit. Na základě toho je docíleno ověření, že provedené změny neměly negativní vliv na původní kód.

Tento typ testu byl zvolen z důvodu kontroly správnosti kódu a komunikace mezi jednotlivými objekty. V běžné programátorské praxi komplexních webových aplikací patří jednotkové automatizované testy jako základ při sestavování testů. O tomto faktu svědčí i integrování testovací knihovny do frameworku Nette.

Funkčnost konvertoru na různých typech šablon

Ověření správnosti implementace konvertoru šablon je stěžejní funkce této webové aplikace. Je proto velmi důležité ověřit funkčnost tohoto modulu u kterého je korektní funkčnost nepostradatelná.

Vzhledem ke skutečnosti, že HTML šablony jsou velmi různorodé, je nutné otestovat funkčnost na velkém množství typů webových stránek. Výsledkem tohoto testu bude označení, zda konvertovaní šablony umožnilo pohodlnou editaci z prohlížeče. U testované šablony může nastat problém v několika případech. Lze očekávat, že rozpoznávání všechn editovatelných částí nebude fungovat spolehlivě na všech typů šablon. Další problém může způsobit špatná konstrukce původní HTML šablony, nebo nevalidní kód. Kaskádové styly často velmi ovlivňují skrývání různých elementů, protože čekají na interakci uživatele - například kliknutí na tlačítko. Tato vlastnost může způsobit komplikace při editaci.

Tento test tedy může upozornit na nestandartní případy u šablon a poskytne zpětnou vazbu a data, v případě, že by v budoucnu byla plánována úprava nebo rozšíření konvertoru.

Test rychlosti

V rámci Nette Frameworku je značně usnadněn průběh vývoje i z hlediska testování a hledání chyb. Běh aplikace, monitorování procesů a detailní zaznamenávání chyb je umožněno pomocí Tracy Debuggeru. Rychlost, která je velmi důležitý faktor webové aplikace, je rovněž k dispozici v tomto nástroji.

Tento test rychlosti byl vybrán a proveden pro ověření výkonu aplikace, která přímo ovlivňuje uživatele. Rychlost načítání jednotlivých stránek je v této webové aplikaci velmi důležitý faktor. Značně zpříjemňuje práci uživatelům, kteří tuto aplikaci používají. Při zdlouhavém načítání může nastat tendence webovou službu opustit a najít jiné, rychlejší řešení. Pro tuto aplikaci tedy bylo využito i tato metrika rychlosti jako stěžejní. Výsledek tohoto testu ukáže, zda je naimplementované řešení dostačující pro svižný běh aplikace a pohodlnou interakci ve vztahu uživatele s webem.

7.2 Provedení testů

Jednotkové automatické testy

V rámci této aplikace bylo vytvořeno několik jednotkových testů, které ověřovali správnost naimplementovaných tříd pro komunikaci s databází a prezencí dat. Testy porovnávaly, zda dochází k očekávaným výsledkům při procesu s jednotlivými daty. [27]

Spouštění testů se provádí přes konzoli. Ve vývojovém prostředí byl vytvořen alias pro jednoduché spuštění testů. Běžně se Nette tester spouští dlouhým příkazem uvedeným níže.

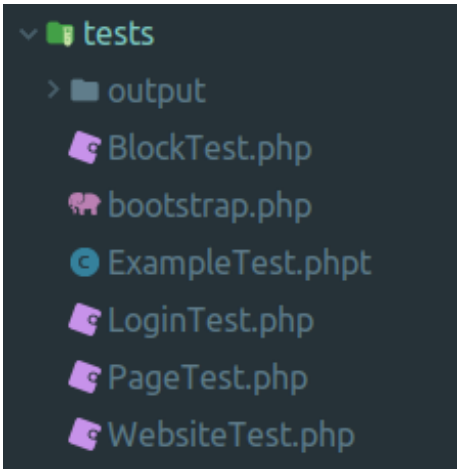
```
vendor/bin/tester
```

V testovaném vývojovém prostředí ale stačí použít jen následující příkaz.

```
tester
```

Poté by se v případě úspěchu testů měl na výstupu zobrazit počet úspěšných testů se zkratkou **OK**.

Struktura souborů pro testování je zobrazena na následujícím obrázku.



Obrázek 13: Struktura testovacích tříd

Funkčnost konvertoru na různých typech šablon

Pro nejpřesnější výsledky tohoto testu je nejvhodnější provést tento test ručně. Ruční testování umožní objevit problémy, se kterými se může setkat i uživatel. Napodobení jeho procesu je tedy nejvhodnější metodou pro tento test.

Uživatel instaluje šablonu následujícím procesem po sobě jdoucích kroků. Při testování se bude postupovat přesně podle tohoto procesu, který kopíruje očekávaný postup uživatele.

1. Výběr šablony z internetu
   Uživatel si vybírá v katalogu šablon dostupných volně na internetu z různých webových portálů. Soustředí se na šablony zdarma, které odpovídají zvolenému účelu webové stránky.
2. Stažení vybrané šablony
   Uživatel zvolenou šablonu stáhne do svého počítače.
3. Instalace šablony
   Uživatel v aplikaci uskuteční nezbytné kroky pro instalaci šablony. Je nutné zvolit web, pro který bude šablona nainstalována a nahrát šablonu do aplikace.
4. Přepnutí do editačního módu
5. Editace textu a obrázků na webové stránce
   Na stránce se většinou před editací vyskytuje univerzální text, který usnadňuje editaci a naznačuje rozložení. Uživatel začne postupným přepisováním původního textu na vlastní text, který chce prezentovat na stránce. Pro účely testování se editují všechny textové a obrázkové oblasti.
6. Kontrola výsledku úprav
   Následné je nutné ověřit, zda dané úpravy se úspěšně uložily a změny jsou vidět na webové stránce pro veřejnost.

Při testování lze prohlásit daná šablona za úspěšně konvertovanou, v případě dosáhnutí očekávaného výsledku. Při tomto typu testu je vhodné otestovat velké množství šablon z důvodu již zmíněné různorodosti každé webové šablony.

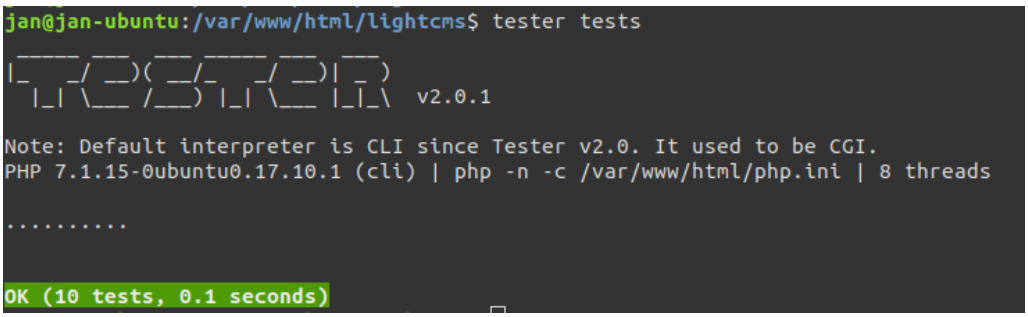
Test rychlosti

Nette Tracy debuggeru zobrazuje dobu načítání stránky v milisekundách. Tyto hodnoty jsou zaznamenávány a následně porovnány. Hodnoty se navzájem od sebe většinou velmi liší. Je nutné tedy stanovit meze, které určí maximální možnou dobu načítání stránky. Odborná literatura UX designu definuje maximální časovou jednotku načítání první stránky webu třemi vteřinami. Po tomto časovém úseku uživatel raději opouští webovou stránku, než aby déle čekal [28].

7.3 Výsledky testů

Jednotkové automatické testy

Výsledek všech deseti testovacích tříd a jejich metod proběhli úspěšně. Všechny testy byly vykonány za sto milisekund.



Obrázek 14: Výsledek jednotkových testů

Dle zobrazeného úspěchu je možné usuzovat, že jednotlivé testované prvky fungují správně. V případě rozšíření nebo úpravy kódu je doporučováno spustit opět tyto testy pro ověření. V případě neúspěchů se zobrazují přehledné informace, proč testování selhalo.

Funkčnost konvertoru na různých typech šablon

Celkem bylo testováno 86 webových šablon stažených zdarma z různých portálů, nabízející tyto webové šablony. Nejvíce vypovídající způsob prezentace výsledků je slovní popis z důvodu komplexnosti testů.

U všech šablon se podařilo dosáhnout úspěšné instalace, která nezpůsobila žádnou chybu. Z toho výsledků lze usoudit, že instalace dokáže reagovat na různé chyby, které můžou nastat při instalaci.

Nejobávanější část konvertováni bylo rozpoznání editovatelných částí. I tuto metriku lze prohlásit za úspěšnou. U 12ti šablon se vyskytla částečná chyba. Algoritmus při konverzi u těchto 12ti případů nerozpoznal některou z editovatelných částí. Uživatel by tedy nemohl editovat některou část stránky. Obvykle se jednalo o patičky webu, které měli nestandardní HTML zaznačení.

Největší komplikace nastala u šablon, využívajících javascript a kaskádové styly pro skrývání některých elementů. Šablony se například snaží při namíření kurzoru na elementy zobrazovat, nebo skrývat a přesouvat obrázky nebo text. Konvertor v takových případech správně rozpoznal, že jde o editovatelnou část, ale následně nebylo možno ji z prohlížeče upravit. Úpravě brání složitá nutnost interakce uživatele a zobrazování možnosti upravit text v daných oblastech.

Výsledkem tohoto testu je tedy skutečnost, že konvertor funguje dle očekávání. Současné řešení ale i tak nabízí prostor pro zlepšení algoritmu rozpoznávání editovatelných částí. Dále bylo zjištěno, že konvertor není vhodný pro šablony s javascriptovým kódem, který ovlivňuje vlastnosti a rozložení HTML elementů. Při řešení tohoto problému by se mělo dbát na zachování jednoduchosti editace. Celková úspěšnost konvertoru byla více než 70 procent. V případě testování šablon bez nevhodného javascriptu, se dosáhlo úspěšnosti 85 procent.

Test rychlosti

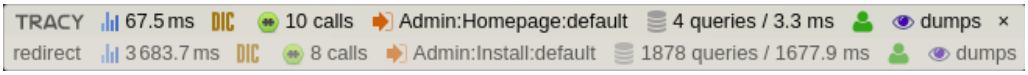
Rychlost načtení úvodní strany se pohybovalo obvykle mezi 30ti až 90ti milivteřinami. Takto rychlé načítání je možné, protože se na backendové části nevykonávají dotazy do databáze, nebo složitější operace. Úvodní strana pouze zobrazuje statický obsah.

Obrázek 15: Výsledek testu rychlosi - úvodní stránka

Následné činnosti a operace se v administraci pohybovali v časech maximálně do 200 milivteřin.

Obrázek 16: Výsledek testu rychlosi - administrace

Při běžných činnostech lze tvrdit, že načítání webových stránek splňuje stanovený limit tří vteřin. Jediný případ, kdy test nesplňoval limit byla akce instalace šablony. Tento proces zahrnoval spoustu událostí, včetně nahrávání souboru, analýzy všech HTML šablon, instalací tématu a obsáhlých zápisů do databáze. Celková doba instalace trvala 3,7 vteřiny.



Obrázek 17: Výsledek testu rychlosi - instalace šablony

V tomto případě lze akceptovat takto dlouhou dobu vykonávání činnosti. Tato akce probíhá jednorázově a nezatěžuje uživatele několikrát denně. V běžných případech uživatel nainstaluje téma jednou a následně jej pouze edituje.

8 Diskuze

8.1 Zhodnocení řešení

Cílem práce bylo vytvořit webovou aplikaci, která by umožnila uživateli snadnou editaci HTML šablon, bez nutnosti předchozí implementace šablony do redakčního systému pomocí programovacích jazyků. Na současném trhu takové řešení zatím neexistuje. Všechny redakční systémy podporují pouze šablony, které jsou upraveny dle specifického způsobu pro konkrétní redakční systém.

Při implementaci bylo nejtěžším úkolem naprogramovat algoritmus, který by dokázal univerzálně zpracovat všechny druhy šablon. Nejlepších výsledků dosahuje konverze u šablon, kde není přiliš javascriptového kódu, který ovlivňuje zobrazování a funkci stránky. Redakční systém je tedy nejvhodnější pro typy stránek jako jsou osobní webové stránky, blog, firemní prezentace, nebo web pro události a akce.

Jeho výhoda spočívá v obrovské jednoduchosti a rychlosti editace. Uživatel se nemusí omezovat výběrem mezi pár vzhledy, které většinou nabízí jiné redakční systémy, ale může si vybrat jakýkoliv z tisíce vzhledů, které najde kdekoliv na internetu.

Naopak méně vhodný je na webové stránky, kde se očekává rozsáhlejší funkčnost a interakce s uživatelem pomocí dotazníků, anket a přihlašování. Algoritmus totiž není schopen rozpoznat plánovanou funkčnost a tedy nelze nabídnout uživateli editaci těchto interaktivních prvků.

8.2 Budoucí rozšíření

Prostoru pro další rozšíření je tu nespočet. Základní prvek, na kterém stojí celá aplikace, je algoritmus pro zpracování HTML šablon. Ten by byl vhodný vylepšit na základě analýzy velkého množství různých typů HTML šablon.

Dále by bylo užitečné ještě více zpříjemnit uživateli editaci stránky, nabídnout mu více možností. V tomto kroku je ale nutné postupovat opatrně, aby více možností nepůsobilo na uživatele zmateně. Je důležité i nadále zachovat jednoduchost, protože cílovou skupinou této aplikace jsou méně zdatní počítačoví uživatelé.

9 Závěr

Cílem práce bylo vytvořit redakční systém, který by umožnil uživateli editovat jakoukoliv HTML šablonu z pohodlí prohlížeče, bez znalostí programování. V práci bylo postupováno systematicky od teoretické části, která vysvětlila problematické pojmy a pomohla pochopení principu problému.

Analýzou současných řešení bylo zjištěno, že navrhovaná metodika je úplně nová a podobnou myšlenku zatím nikdo neuskutečnil. V současných řešení je vždy nutné upravit zdrojový kód HTML šablony pro konkrétní redakční systém.

Byly stanoveny konkrétní požadavky, které by měla aplikace splňovat. Všechny byly vytvořené v souladu s očekáváním od cílové skupiny. Touto skupinou jsou především lidé bez hlubokých počítačových znalostí. Všechny požadavky tedy měly za cíl velké zjednodušení editace a publikace webové stránky.

Na základě stanovených požadavků byla aplikace modelována za použití diagramů a designových návrhů. Tento postup velmi zjednodušil následnou implementaci a díky tomuto návrhu nebylo nutné kód výrazně přepracovávat.

Po této analýze a ujasnění si funkčnosti, bylo nutné zvolit vhodné nástroje. Bylo vybráno mnoho technologií a nástrojů, které ve správné kombinaci umožnily dosáhnout optimálních výsledků.

Následně byla aplikace naimplementována. Implementace obsahuje nové řešení problémů a nové algoritmy navržené pro konkrétní problém. Nebylo možné využít současných řešení, protože tuto problematiku ještě nikdo veřejně neřešil. Výsledná aplikace byla otestována automatickými testy i manuálním hodnocením.

Ukázalo se, že aplikace je vhodná na jednoduché weby jako jsou osobní prezentace, blog nebo webové stránky událostí a akcí. Na těchto typech webu přináší velké zrychlení tvorby, zjednodušení editace a teoreticky neomezené množství šablon, které lze použít. Uživatel se již nemusí vázat na omezený počet šablon, připravených pro konkrétní redakční systém.

Tato práce přinesla novou myšlenku při tvorbě webů, kdy se na proces vytváření webových stránek podívala přesně z opačné strany. To přineslo velké zjednodušení pro koncového uživatele, ale také bylo vykoupeno složitou implementací.

10 Literatura

1. KRUG, Steve. *Don't make me think, revisited : a common sense approach to Web usability*. San Francisco, California: New Riders, Peachpit, Pearson Education, 2014. ISBN 978-0-321-96551-6.
2. CAMERON, Dane. *A software engineer learns HTML5, JavaScript & jQuery*. 2nd. U.S: Cisdal Publishing, 2014. ISBN 978-14-936-9261-3.
3. DUCKETT, Jon. *HTML & CSS : design and build websites*. 1st. Indianapolis, IN Chichester: Wiley Wiley distributor, 2011. ISBN 978-1-118-00818-8.
4. BOEHM, Anne. *Murach's HTML5 and CSS3*. 4th. Fresno, CA: Mike Murach & Associates Inc, 2018. ISBN 1943872260.
5. GAUCHAT, J. D. *HTML5 for masterminds : how to take advantage of HTML5 to create responsive websites and revolutionary applications*. 3th. Place of publication not identified: Mink Books, 2017. ISBN 154292331X.
6. REFSNES DATA, W3SCHOOLS. *JavaScript HTML DOM* [online]. 2018 [visited on 2018-02-07]. Available from: `https://www.w3schools.com/js/js_htmldom.asp`.
7. LORENTZ, Richard. *Recursive algorithms*. 1st. Norwood, N.J: Ablex Pub. Corp, 1994. ISBN 1567500374.
8. HYLMAR, Radek. *Programování pro úplné začátečníky*. 1st. Brno: Computer Press, 2009. ISBN 978-80-251-2129-0.
9. ENVATO. *A 15 Minute Surreal CMS Integration* [online]. 2010 [visited on 2018-04-07]. Available from: `https://code.tutsplus.com/articles/a-15-minute-surreal-cms-integration--net-10413`.
10. LARMAN, Craig. *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*. 1st. Upper Saddle River, N.J: Prentice Hall PTR, 2005. ISBN 978-0131489066.
11. BUCHALCEVOVÁ, Alena. *Příklady modelů analýzy a návrhu aplikace v UML*. 1st. Praha: Oeconomica, 2013. ISBN 9788024519227.
12. PODESWA, Howard. *UML for the IT business analyst : a practical guide to object-oriented requirements gathering*. 2th. Australia United States: Course Technology/Cengage Learning, 2010. ISBN 978-1598638684.
13. VERENS, Kae. *CMS design using PHP and jQuery : build and improve your in-house PHP CMS by enhancing it with jQuery*. 1st. Birmingham: Packt Pub, 2010. ISBN 978-1849512527.

14. BROECK, Martijn van den. *How to decide which CMS to use for your Online Portfolio* [online]. 2016 [visited on 2018-03-17]. Available from: `https://medium.com/portfolio-principles/how-to-decide-which-cms-to-use-for-your-online-portfolio-3451ed8763ba`.

15. MENING, ROBERT. *Popular CMS by Market Share* [online]. 2017 [visited on 2018-03-20]. Available from: `https://websitesetup.org/popular-cms/`.

16. ZANDSTRA, Matt. *PHP objects, patterns, and practice*. 3th. Berkeley, Calif. New York: Apress Distributed to the Book trade worldwide by Springer-Verlag New York, 2010. ISBN 978-1-430-22925-4.

17. ARLOW, Jim. *UML 2 a unifikovaný proces vývoje aplikací : objektově orientovaná analýza a návrh prakticky*. Brno: Computer Press, 2007. ISBN 978-80--251-1503-9.

18. T, David. *Undreamt Requirements* [online]. 2007 [visited on 2018-04-01]. Available from: `http://www.techmachina.com/2007/03/undreamt-requirements.html`.

19. CHONOLES, Michael. *UML 2 for dummies*. 2th. New York: Wiley Pub, 2003. ISBN 978-0764526145.

20. BULEY, Leah. *The user experience team of one : a research and design survival guide*. 1st. Brooklyn, New York: Rosenfeld Media, 2013. ISBN 978-1933820187.

21. BROWN, Daniel. *Designing together the collaboration and conflict management handbook for creative professionals*. 1st. Place of publication not identified: New Riders, 2013. ISBN 978-0321918635.

22. YOUNG, Indi. *Practical empathy : for collaboration and creativity in your work*. 1st. Brooklyn, New York: Rosenfeld Media, 2015. ISBN 978-1933820484.

23. THE PHP GROUP. *PHP Manual* [online]. 2018 [visited on 2018-04-11]. Available from: `http://php.net/manual/en/`.

24. NETTE FOUNDATION. *Nette Framework dokumentace* [online]. 2018 [cit. 2018-04-11]. Dostupné z: `https://doc.nette.org/cs/2.4/`.

25. TROSTER, František. *ORM frameworky pro PHP5: Obecný úvod* [online]. 2010 [cit. 2018-04-11]. Dostupné z: `https://www.zdrojak.cz/clanky/orm-frameworky-pro-php5-obecny-uvod/`.

26. LESS TEAM. *It's CSS, with just a little more.* [Online]. 2017 [visited on 2018-04-11]. Available from: `http://lesscss.org/`.

27. SOFTWARE TESTING FUNDAMENTALS. *Unit Testing* [online]. 2017 [visited on 2018-04-25]. Available from: `http://softwaretestingfundamentals.com/unit-testing/`.

28. MACHMETRICS. *Unit Testing* [online]. 2018 [visited on 2018-05-02]. Available from: `https://www.machmetrics.com/speed-blog/average-page-load-times-websites-2018/`.