UNIVERZITA HRADEC KRÁLOVÉ
FAKULTA INFORMATIKY A MANAGEMENTU
KATEDRA INFORMATIKY A KVANTITATIVNÍCH METOD

BAKALÁŘSKÁ PRÁCE

Porovnání volně dostupných Open Source redakčních systémů

Comparison of freely available Open Source content management systems

Autor: Michal Horčičko

Vedoucí práce: Ing. Andrea Vokálová

Hradec Králové, 2017

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne Podpis autora

Poděkování

Za všechny cenné rady a hlavně neskutečnou trpělivost bych chtěl poděkovat vedoucí práce Ing. Andree Vokálové. Dále děkuji všem přátelům, kteří neváhali a při tvorbě této práce se podělili se svými zkušenostmi, vědomostmi a poznámkami. Nakonec bych rád poděkoval Michaelovi Kutému za jeho LaTeXovou šablonu[1], jejíž základ byl použit při tvorbě této práce.

Anotace

Cílem práce je popis a porovnání volně dostupných redakčních systémů a s nimi spojené technologie, uživatelská přívětivost těchto systémů, dostupnost, množství a kvalita rozšíření (pluginů) a šablon. Výstupem práce je celkový přehled o světě redakčních systémů, klady a zápory jednotlivých variant.

Praktická část práce je zaměřená na tvorbu vlastního jednoduchého redakčního systému s pomocí frameworku Django pro programovací jazyk Python. Výstupem této části je ukázkový projekt, základní seznámení s použitými technologiemi, architekturou MVC, strukturou a syntaxí těchto technologií.

Annotation

Main goal of this thesis is description and comparison of free-to-use content management systems and technologies associated with them, user friendliness of these systems, availability, quantity and quality of extensions (plugins) and templates. The result is a summary about the world of CMS, advantages and disadvantages of each different option.

Practical part of thesis is focused on making-of custom simple content management system using Django framework for Python programming language. The outcome of this part is a sample project, basic accustomization with technologies used, MVC architecture, structure and syntax of these technologies..

Klíčová slova

redakční systém, rozšíření, šablona, Python, Django

Keywords

content management system, extension, template, Python, Django

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek kódu

Seznam zkratek

AMP	Apache, MySQL, PHP
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CMS	Content Management System
CRUD	Operace Create, Read, Update, Delete
CSRF	Cross Site Request Forgery
CSS	Cascading Style Sheets
DTL	Django Template Language
FTP	File Transfer Protocol
FOSS	Free and Open Source Software
GNU	GNU is Not Unix
GPL	GNU General Public License
HTML	Hypertext Markup Language
MVC	Model View Controller architektura
PHP	PHP: Hypertext Preprocessor
PIP	PIP Installs Python
SEO	Search Engine Optimization
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator
WYSIWYG	What You See Is What You Get

Úvod

Již od počátku používání internetu byla potřeba publikovaná data nějakým způsobem spravovat, členit, graficky formátovat, měnit obsah bez znalosti kódu a mnohé jiné. Rozsáhlé statické weby bylo po nějaké době prakticky nemožné spravovat, ať už z časových, finančních anebo jiných důvodů. Díky těmto požadavkům na řízení webových dokumentů se začaly objevovat různé aplikace pro správu obsahu webových stránek - redakční systémy[1].

A právě redakční systémy jsou tématem této bakalářské práce. Autor práce se snaží takový systém popsat, jaké jsou jeho funkce, jaké má výhody a nevýhody. Další kapitola práce je zaměřena na historii a vývoj těchto systémů od počátku 90. let minulého století až po současnost.

Jelikož je v současné době k dispozici nepřeberné množství variant CMS, věnuje se další část práce porovnání nejpoužívanějších open source systémů. Hlavním důvodem pro porovnání pouze open source variant je veřejně dostupný zdrojový kód, lze tedy náhlednout "pod pokličku" systému a podívat se, v jakém jazyce je systém napsán. Toto by nebylo možné u proprietárních systémů. Všechny porovnávané systémy jsou také "free", tzn. zdarma dostupné ke stažení a používání. Pokud je software open source, ještě to neznamená, že je volně dostupný[2].

Třetí a čtvrtá část práce je věnována popisu, návrhu a implementaci vlastního jednoduchého redakčního systému, který slouží ke správě obsahu online kuchařky s prvky sociální sítě. Tento systém je postaven na frameworku pro tvorbu webových aplikací Django pro jazyk Python a databázi SQLite. Zdrojový kód je dostupný z gitového repozitáře `https://github.com/horca/bchlrr_cookbook`, případně zabalený v archivu v přílohách práce. Live verze kuchařky se nachází na webových stránkách `http://bchlrr.pythonanywhere.com/` a v době tvorby práce jsou stránky aktivní do 31. října 2017.

[1] angl. content management system, zkratka CMS

[2] software, který je open source a zároveň je volně dostupný, se nazývá FOSS

1 Redakční systémy a jejich historie

1.1 Co je to redakční systém

Redakční systém (anglicky Content management system, zkratka CMS) je internetová aplikace, která jednoduchým způsobem umožňuje vytvářet a upravovat webové stránky a hlavně jejich obsah. Redakční systém obsahuje administrační část, která se ovládá jednoduše pomocí internetového prohlížeče. K vytváření stránek pomocí redakčního systému vám stačí běžné znalosti práce s počítačem. Nemusíte znát programovací jazyk HTML, PHP, JavaScript, CSS a další. O to jak bude kód stránek vypadat se postará redakční systém. Redakční systémy se také označují jako publikační systémy, systémy pro správu obsahu apod. [2] [5]

1.2 Funkce redakčního systému

Jednotlivé systémy dostupné na dnešním trhu se liší např. složitostí nasazení, instalací, počtu dostupných rozšíření, komunitou, technickou podporou apod. Avšak základní funkce těchto systémů jsou v podstatě stejné.

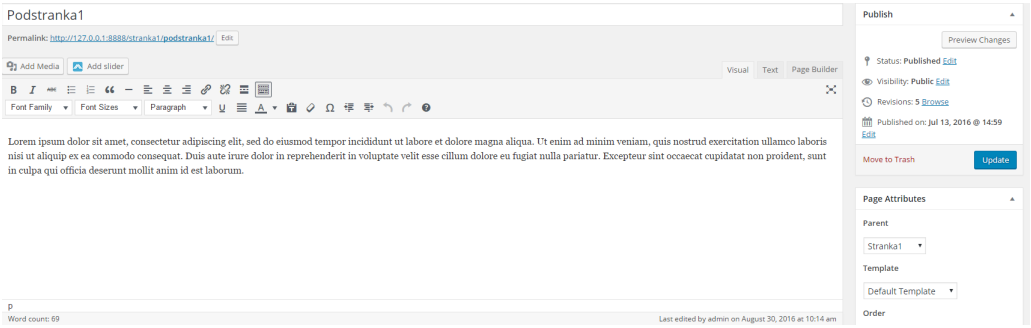
1.2.1 Správa obsahu

Ať už se jedná o vytváření, úpravu, mazání, vyhledávání, kategorizaci, práce s články, komentáři, médii a uživateli - správa obsahu je základní funkcí redakčních systémů. Práce s obsahem se také někdy nazývá CRUD operace. Tato zkratka plyne z anglického Create, Read (Retrieve), Update, Delete. Jedná se o 4 základní operaci při práci s daty. Pro práci s daty obsahuje většina redakčních systémů WYSIWYG[1] editor. Tento editor umožňuje člověku, který nezná HTML syntax,

[1] What You See Is What You Get

stylizovat text, zarovnávat obrázky, vytvářet nadpisy. Mezi programátory a kodéry není tento typ editoru velice oblíbený, ale počítačem nepolíbený uživatel jej zajisté ocení. [2]

Správa obsahu ve většině případů probíhá z backendu systému. Backend je prostředí pouze pro autorizované uživatele, kde se dá veškerý obsah stránek a veškeré nastavení upravit. Backend poskytuje práci s obsahem, nastavení šablony frontendu, správu pluginů, možnosti lokalizace, nástroje pro správu uživatelů a další nastavení. Na obrázku 1.1 je zobrazen WordPress backend, přesněji editace článku *Podstranka1*.



Obrázek 1.1: Backend systému WordPress, editace článku

Další již zmíněnou funkcí pro správu obsahu je kategorizace. Obsah je tříděn do kategorií, ať už dle druhu dat, nebo časového řazení. Zařazování pomáhá vyhledávání obsahu, optimalizaci vyhledávacích služeb a někdy je také použito jako navigace stránek.

Zmíněné pluginy, nebo také zásuvné moduly, jsou samostatné kousky kódu, které se do systému dají "zasunout". Žadný redakční systém v defaultním nastavení neobsahuje všechny uživatelem potřebné funkce. Lze je tedy pomocí pluginů do systému přidat. Typický příklad pluginu je rozšíření obsluhující komentáře Disqus. Mezi další rozšíření by se daly zařadit ankety, galerie, nástroje proti botům, moduly pro optimalizaci vyhledávacích enginů, reklamní prvky, rozšíření pro podporu e-shopu a jiné.

1.2.2 Uživatelský model na bázi práv

Se správou obsahu jsou nevyhnutelně spojení uživatelé a uživatelské role. Ne každý návštěvník webových stránek smí obsah přidávat, či dokonce upravovat. Operace, které uživatel smí provést, ovlivňují jeho práva, které dědí podle příslušné skupiny,

do které je uživatel zařazen. V drtivé většině dnešních redakčních systému jsou již uživatelské práva a skupiny předem definované.

Nezaregistrovaný návštěvník webu smí většinou obsah pouze prohlížet. Autorizovaný uživatel smí obsah přidávat a také svůj obsah upravovat. Nad běžným uživatelem se nachází moderátor. Tento typ účtu má stejná práva jako normální uživatel - plus moderátorské nástroje. Mezi tyto nástroje patří moderování obsahu a uživatelů, hlášení nevhodného obsahu, "banování" nezpůsobilých uživatelů. Uživatel, který má k dispozici všechna práva se nazývá administrátor (někdy také pojmenovaný admin nebo superuser). Uživatel s právy administrátora má naprostou kontrolu nad obsahem stránek, které spravuje.

Většina na trhu dostupných redakčních systémů také nabízí tvorbu vlastních přístupových práv a skupin uživatelů, přesně odpovídající požadavkům provozovatele tohoto systému.

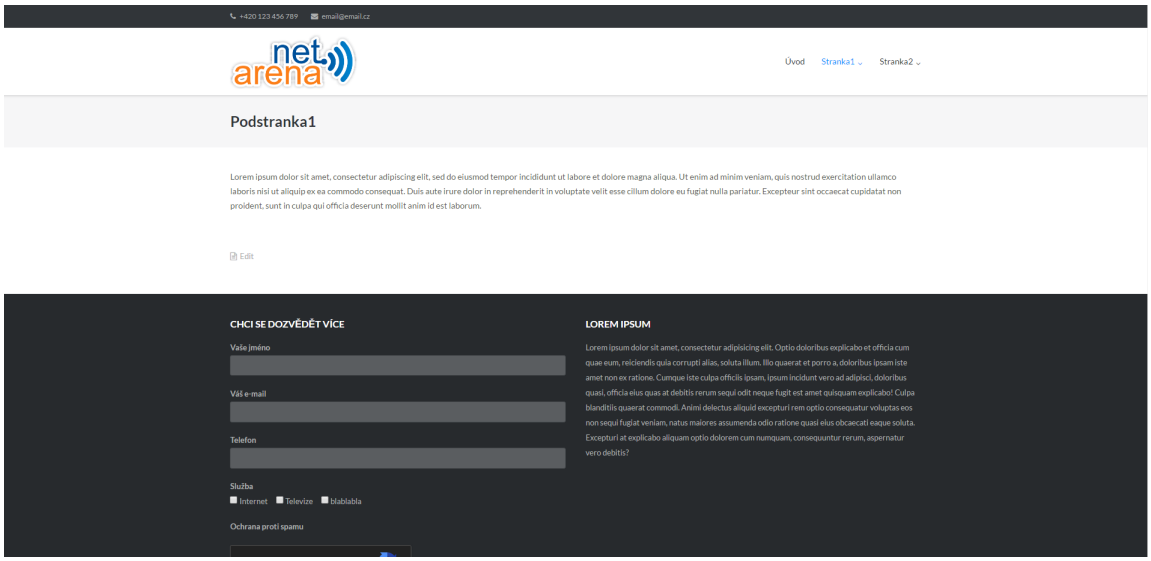
1.2.3 Frontend systému

Frontend systému je nejdůležitější prvek z pohledu návštěvníka. Nezajímá ho vzhled backendu systému, způsob, jakým jsou články přidávány, ani starosti, které řeší administrátor stránek. Jeho hlavním požadavkem je přehledné responzivní a vizuálně působivé zobrazení obsahu stránek. Tenhle problém řeší šablona systému. Tato šablona se stará o správné renderování obsahu stránek, má na starosti kaskádové styly, zajišťuje správné fungování javascriptu. Součástí šablony je také hlavička webu s logem a navigací. Ve spodu šablony se nachází patička. Na obrázku 1.2 lze vidět frontend článku *Podstranka1*.

Většina dnešních redakčních systémů je napsána v PHP a část z nich si osvojila šablonovací jazyk Twig pro jazyk PHP. Defaultní šablonovací syntaxe PHP je totiž velice krkolomná a pro designéra bez znalosti kódu nepřehledná.

1.3 Historie redakčních systémů

První CMS se začaly objevovat v polovině 90. let minulého století a není zcela jasné, který systém byl první. První systémy byly velice strohé a vyžadovaly uživatelovu znalost HTML. To ovšem nebyl problém, jelikož většinu těchto systému tvořili právě sami uživatelé. Na začátku milénia se do popředí dostaly komerční systémy. Většinou s rozpočtem přesahujícím šest nul. Tyto systémy využívaly



Obrázek 1.2: Frontend systému WordPress, detail článku

velké společnosti typu novinových portálů, e-shopů. V polovině předchozího desetiletí začaly práce na open source variantách, jmenovitě Mambo, Drupal a WordPress. V dnešní době jsou nejrozšířenější právě open source CMS, a to hlavně z důvodu financí a obrovské komunity, která tyto systémy obklopuje. [3] [2]

1.4 Výhody a nevýhody redakčních systémů

1.4.1 Výhody

Hlavním kladem je bezesporu možnost jednoduše upravovat obsah našich stránek a to bez znalosti kódu a zásahu externí osoby, což zajisté šetří čas a finance. Další výhodou je možnost použití externích pluginů. CMS také podporují lokalizace do více jazyků a mají implementovaný svoji formu verzovacího systému. Což znamená, že si systém pamatuje předchozí verze obsahu, a není problém starou verzi obnovit. Žádné pořizovací náklady volně dostupných variant jsou také příjemným kladem.

1.4.2 Nevýhody

Rozsáhlejší systémy mohou mít větší hardwarové požadavky než jednoduché statické stránky, a vyžadovat tak výkonnější hardware. V případě potřeby velkého

přizpůsobení vzhledu nebo funkcí stránek, se musí provozovatel webu stejně obrátit na programátora/kodéra, který má v daném systému předešlé zkušenosti.

2 Porovnání redakčních systémů

2.1 Seznámení s porovnávanými CMS

Dříve, než se pustíme do samotného porovnání systémů si je krátce představíme. Jejich stručný popis, nejsilnější stránky a nevýhody. Nahlédneme také krátce do historie těchto systémů.

WordPress

Nejznámější z nejznámějších. Redakční systém WordPress (obrázek 2.1), který k datu tvorby práce pohání 28,4 % [8] internetu, je nejrozšířenějším systémem pro správu obsahu na světě. Jedná se o opravdového giganta, který ovšem začínal poněkud skromně. WordPress vznikl jako fork[1] projektu b2 cafelog a jednou z hlavních novinek, která obsahovala první verze byla podpora XHTML 1.1. Plus pár šablon [4].

Systém WordPress, díky jeho jednoduchosti a přehlednosti, je vhodný pro uživatele, kteří se správou webu nemají žádné předchozí zkušenosti. V dnešní době je tento systém velice všestranným řešením, s perfektní správou obsahu a multimédií. Vestavěnou optimalizací vyhledávacích enginů. Obrovskou komunitou, nekonečným počtem volných i placeným šablon a rozšíření.

A právě velká komunita je jednou z WordPressových nevýhod. Větší počet uživatelů znamená větší počet útoků na stránky poháněné WordPressem.

Obrázek 2.1: Logo systému WordPress

[1] kopie, jejíž změny se neprojeví v původním projektu

Joomla

Dříve než byla Joomla (obrázek 2.2), bylo Mambo. Mambo byl CMS, jehož první verzi vydala v roce 2001 společnost Miro. Vedení této společnosti se chtělo, na rozdíl od vývojového týmu, zaměřit na komerční produkt, vývojáři na druhou stranu chtěli vyvíjet open source software. A tak se v roce 2005 většina tohoto týmu od Mira oddělila a vznikl nový projekt s názvem Joomla [6]. Její první verze byla identická se systémem Mambo, jelikož Mambo bylo do té doby vydávané pod licencí GPL. To umožnilo vývojářům Joomly jej použít jako základ pro svůj systém.

V dnešní době je Joomla druhým nejrozšířenějším redakčním systémem. Jednou z výhod Joomly je ochotná uživatelská komunita, která vždy ráda poradí. Nevýhodou je méně přehledné UI, větší technické požadavky na provozovatele a větší velikost a HW nároky redakčního systému.

Obrázek 2.2: Logo systému Joomla

Drupal

Drupal (obrázek 2.3) je všestranný systém pro správu obsahu, který odmění technicky zručnějšího provozovatele. Pro správný provoz stránek se od správce systému očekává, že bude zvládat základy PHP, HTML a CSS. Pokud správce tyto požadavky splňuje, je odměněn velice mocným nástrojem pro správu obsahu. Technická náročnost systému je tedy jeho výhodou i nevýhodou.

Historie Drupalu začíná v roce 2000, kdy dva studenti Antverpské univerzity sdíleli své internetové připojení s dalšími osmi přáteli. Ke správě této malé sítě potřebovali nějaký nástroj, pomocí něhož by mohli mezi sebou komunikovat a sdílet problémy, které se na síti objeví. Jeden z těchto studentů se chopil příležitosti a stvořil systém novinkového portálu - první verzi Drupalu. [7]

Obrázek 2.3: Logo systému Drupal

Django CMS

Systém Django CMS (obrázek 2.4) byl pro porovnání zvolen z důvodů využití technologie Django v praktické části práce. Jako jediný z porovnávaných systému není napsaný v PHP, ale v jazyce Python za použití frameworku Django.

Musíme si dát pozor s rozlišením jednotlivých technologií. Django je Pythonový framework pro tvorbu webových aplikací, zatímco Django CMS je systém určený pro správu obsahu na Djangu postavený.

Narozdíl od WordPressu, Joomly a Drupalu je Django CMS systém velice strohý, určený pro pokročilé uživatele, který umožňuje správci stránek absolutní kontrolu nad jejich obsahem a vzhledem.

Bližší seznámení s Pythonem a Djangem se nachází v další části práce.

Obrázek 2.4: Logo systému Django CMS

Shrnutí

Přirovnejme tyto čtyři redakční systémy ke skládačkám. WordPress je již hotový produkt, který je použitelný přímo z produkce. Systémy Joomla a Drupal by se daly přirovnat k nábytku obchodního domu Ikea. Musíte si sami poskládat požadované funkce, aby byl systém podle vašich požadavků. Django CMS je skládačka LEGO, ze které jde vybudovat téměř cokoliv, jediným omezením je vaše fantazie, potřebný čas a úsilí.

2.2 Komunita a podíl na trhu

Při výběru redakčního systému nás zajímá jeho komunita. Větší komunita systému značí větší uživatelskou podporu, větší počet uživatelů, kteří se setkali se stejným problémem, větší počet tvůrců rozšíření a šablon, rychlejší řešení bugů a bezpečnostních děr. Na druhou stranu větší počet uživatelů znamená větší počet hackerů, kteří se se systémem setkají.

Pro porovnání komunit jednotlivých variant bylo zvoleno několik hledisek. Výsledky tohoto porovnání se nachází v následujících tabulkách. Veškeré uvedené data jsou aktuální ke květnu 2017

CMS	% podíl veškerého internetu	% podíl mezi CMS
WordPress	28,4 %	59,3 %
Joomla	3,3 %	6,8 %
Drupal	2,3 %	4,7 %
Django CMS	zanedbatelný	zanedbatelný

Tabulka 2.1: Podíl redakčních systémů na trhu [8]

CMS	Facebook	Twitter	Google+	Reddit
WordPress	1,1 mil.	600 tis.	101 tis.	39 tis.
Joomla	170 tis.	63 tis.	5 tis.	1,8 tis.
Drupal	90 tis.	69 tis.	1,8 tis.	7 tis.
Django CMS	1,2 tis.	4,9 tis.	763	47

Tabulka 2.2: Počet uživatelů komunit CMS na sociálních sítích

CMS	Počet otázek	Počet otázek bez odpovědi
WordPress	126 tis.	57 tis. (45 %)
Joomla	14 tis.	5 tis. (35 %)
Drupal	18 tis.	6 tis. (33 %)
Django CMS	1,1 tis.	373 (33 %)

Tabulka 2.3: Počet StackOverflow témat označných tagem redakčního systému

CMS	Commity	Přispěvatelé	Issues	Vyřešené
WordPress	37 tis.	40	neuvedeno	neuvedeno
Joomla	29 tis.	553	5,4 tis.	4,9 tis.
Drupal	33 tis.	42	neuvedeno	neuvedeno
Django CMS	15 tis.	365	2,9 tis.	2,7 tis.

Tabulka 2.4: Statistiky GitHub repozitáře redakčnícho systému

V podílu na trhu (tabulka 2.1) jednoznačně vede WordPress. Tento gigant pohání 28,4 % [8] veškerého internetu[2]. U sociálních sítí (tabulka 2.2) nás zajímá velikost komunit Google+ a Reddit. Google+ je sociální síť od Googlu, která měla před několika lety ukončit éru Facebooku. Bohužel se tak nestalo, a Facebook stále nutí své uživatelé, kteří kvůli svým známým nezmění platformu, číst neustálé clickbait články, falešné zprávy a všelijaké nepodstatné věci. Zpátky k tématu. V dnešní době tvoří většinu sítě Google+ nadšenci výpočetních technologií. A právě z tohoto důvodu je dobré se na Google+ zaměřit. Reddit je sociální sítí založenou na sdílení odkazů, médií a textových příspěvků. Reddit se dále dělí na tzv. subreddity, které se od sebe liší tématem sdíleného obsahu. A stejně jako na Google+ se na Redditu nachází početná komunita se zalíbením v počítačové problematice.

Autor tyto dvě zmíněné sítě používá pro konzultaci a sdílění věcí okolo linuxových distribucí, ať už se jedná o funkcionalitu, řešení problémů nebo vzhled desktopového prostředí.

Statistiky z webu StackOverflow (tabulka 2.3) jsou skvělým ukazatelem fungující a ochotné komunity. Tento portál je již několik let středem veškerého řešení problémů v oblasti programování a celkově veškerých věcí týkajících se počítačů. WordPress i přes 100 000 přidaných otázek má 45% podíl otázek nezodpovězených, což značí své. U GitHubových statistik (tabulka 2.4) je dobrým ukazatelem celkový počet commitů[3], počet issues[4] a počet jich vyřešených. Například Django CMS má přes 93 % problémů vyřešených. Bohužel WordPress a Drupal tyto statistiky neuvádějí.

2.3 Proces instalace

Systémy WordPress, Joomla a Drupal jsou napsány v jazyce PHP a k jejich lokální instalaci je zapotřebí lokálního vývojového prostředí. LAMP pro systémy GNU/Linux, WAMP pro systémy Windows a MAMP pro systémy macOS. Toto vývojové prostředí obsahuje webový server Apache, dále MySQL, MariaDB databázové systémy a PHP, Perl, Python programovací jazyky. Před instalací redakčního systému s databází MySQL je třeba předem databázi vytvořit. Jednou z variant, jak databázi založit je pomocí nástroje PHPMyAdmin. Pro spuštění

[2] nějakou formu redakčního systému používá 48 % internetových stránek

[3] nahrání změn do repozitáře

[4] nalezených bugů

instalace je třeba rozbalený archiv redakčního systému zkopírovat do kořenové složky vývojového prostředí a v prohlížeči navštívit lokální adresu. Po splnění všech těchto podmínek se lze pustit do instalace systému.

WordPress

Prvním krokem instalace WordPressu je zvolení lokalizace. Systém již v základu obsahuje češtinu. V dalším kroku WordPress vyžaduje specifikovat detaily databáze. Jedná se o databázi, kterou lze vytvořit pomocí PHPMyAdmin. Pokud vše proběhne v pořádku a systém se připojí k databází, následuje krok zadání osobních údajů uživatele. Jedná se o název stránky, uživatelské jméno, heslo, email a souhlas s indexováním stránek vyhledávacími enginy. Po zadání těchto údajů je instalace kompletní a stránky lze začít používat.

Joomla

V prvním kroku lze vybrat jazyk instalačního průvodce a nastavení stránek. Lze zde vyplnit název a popis stránek, plus vytvoření administrátorského účtu. Dalším krokem je uvedení detailů databáze. V posledním kroku lze na stránky nahrát testovací obsah (v nabídce jsou i prázdné stránky). Na stejné stránce se nachází souhrn informací systému, který bude nainstalován. Pokud uživatel se vším souhlasí, může spustit instalaci. Poté, co je instalace dokončena, je zapotřebí odstranit instalační složku z kořenové složky vývojového prostředí. Toto lze provést přímo v instalačním průvodci. Po odstranění této složky je instalace kompletní a uživatel může Joomlu začít používat.

Drupal

Stejně jako u předchozích instalací je prvním krokem zvolení jazyka. Další položka instalace nabízí buďto doporučenou instalaci, která nainstaluje doporučené komponenty systému. Nebo instalaci minimální, která nainstaluje samotný systém. Tato volba je určena pro pokročilé uživatele. V dalším kroku instalační průvodce zkontrolu požadavky na PHP a databázi. Další krok se zabývá vyplněním detailů databáze. Lze použít MySQL, MariaDB anebo SQLite, která nepotřebuje žádnou konfiguraci. Pokud vše proběhlo v pořádku systém se nainstaluje. Poslední krok instalace obsahuje vyplnění informací o stránce a vytvoření správcovského účtu. Po zadání těchto údaju je systém Drupal úspěšně nainstalován.

Django CMS

Instalace Django CMS se od ostatních porovnávaných systémů velice liší. Pro zprovoznění nepotřebuje xAMP prostředí, nýbrž virtuální prostředí poskytované Pythonem. Před samotnou instalací se musíme ujistit, že máme na počítači nainstalovaný Python verze 3 nebo novější a manažer Python balíčků pip.

Vytvoříme nový adresář projektu a aktivujeme v něm virtuální prostředí Python pomocí příkazu *python -m venv myvenv* a *source myvenv/bin/activate*.
Po úspěšné aktivaci virtuálního prostředí pomocí nástroje pip nainstalujeme balíček djangocms-installer - *pip install djangocms-installer* a příkazem *djangocms MY_SITE* spustíme jeho instalaci. Instalace probíhá v okně terminálu a během této operace se instalační průvodce ptá na několik věcí. Jako první krok lze nastavit databázi (defaultně SQLite), v dalším kroku specifikujeme verzi Django CMS, dále verzi Djanga. Poté nastavíme jazykovou lokalizaci, časové pásmo, použití Twitter Bootstrapu, nahrání ukázkových dat a nastavení defaultní šablony. Po úspěšné specifikace veškerých informace se spustí instalace systému. Po dokončení instalace je zapotřebí vytvořit nový správcovský účet. Po jeho vytvoření je Django CMS úspěšně nainstalováno. Pro spuštění serveru je třeba přejít do adresáře instalace a pomocí příkazu *python manage.py runserver* lokální server spustit. [5]

2.4 Funkčnost “out of the box”

Jedno z hlavních kritérií při výběru redakčního systémů je jeho funkčnost ihned po instalaci. Nezkušený uživatel ocení kompletní funkčnost systému, bez potřeby dalšího zásahu a úpravy. Někdo zase požaduje funkční základ systému a jeho specifické funkce si zajistí sám.

Pro porovnání byly zvoleny nejčastěji požadované funkce systému a jejich dostupnost v nové instalaci. Jak lze z tabulky 2.5 vidět, nejobsáhlejším redakčním systémem je WordPress.

Jedinou vlastnost, kterou WordPress nenabízí je šablonovací jazyk. Šablony jsou tedy psány v čistém PHP a to značně znepříjemňuje práci designéra šablony. Systémy Joomla a Drupal používájí šablonovací jazyk Twig, který tvorbu šablon velice zjednodušuje. Django CMS používá Django Template Language, jehož syntax je Twigu velice podobný.

Funkčnost	**WordPress**	**Joomla**	**Drupal**	**Django CMS**
správa článků (blog)	ano	ano	ano	plugin
WYSIWYG editor	ano	ano	plugin	ano
korekce pravopisu	ano	plugin	plugin	ne
obnovení smazaného obsahu	ano	ano	ne	ne
správce multimédií	ano	plugin	plugin	ne
práce s obrázky	ano	ano	plugin	ano
vestavěná galerie	ano	plugin	plugin	plugin
vyhledávání	ano	ano	ano	plugin
správa URL	ano	ano	ano	ano
zabezpečení CAPTCHA	plugin	plugin	plugin	plugin
Drag-n-Drop podpora	ano	ne	plugin	ne
ověření e-mailu	ano	ano	ano	plugin
historie přihlášení	plugin	ano	ano	plugin
historie verzí	plugin	plugin	ano	plugin
webové statistiky	plugin	ano	ano	plugin
podpora FTP	plugin	ano	omezená	ne
ankety	plugin	ano	ano	ne
podpora metadat	ano	ano	ano	ano
lokalizace	ano	ano	ano	ano
mapa stránek	plugin	plugin	plugin	ano
šablonovací jazyk	ne	ano	ano	ano

Tabulka 2.5: Funkčnost systému ve výchozím nastavení [9]

Z porovnání je zřejmé, že Django CMS je systém velice základní a moc toho "out of the box" neumí. Ne vždy je toto negativní vlastnost. Jednoduchost systému ocení autoři malých stránek a tvůrci vlastních CMS, kteří pro svůj projekt potřebují pevný základ.

Joomla a Drupal se nachází v zóně zlaté střední cesty. Ve výchozím nastavení poskytují základní funkce, a potřebné doplňkové služby lze pomocí pluginů bez problémů doinstalovat.

2.5 Pluginy

Pluginy třetích stran jsou samostatné doplňky, které rozšiřují funkcionalitu systému. Některé z těchto pluginů jsou dostupné zdarma, některé jsou placené. Rozšíření se také velice liší kvalitou.

WordPress

Pluginy systému WordPress jsou spravovány a distribuovány z portálu `https://wordpress.org/plugins/`, kde lze rozšíření přidávat, stahovat, hodnotit a komentovat. V květnu 2017 tento portál nabízí ke stažení 51 tisíc volně dostupných WordPressových rozšíření třetích stran. Placené pluginy se zde nenachází, a vývojář je tedy musí sdílet jiným způsobem.

Mezi nejpopulárnější pluginy se řadí Contact Form 7, který obstarává práci s formuláři. Nástroj Akismet, který v obsahu stránek vyhledává nevhodný a spam obsah. Dalším oblíbeným pluginem je WooComerce. Tento plugin změní stránky WordPressu do podoby e-shopu. Plugin Yoast SEO spravuje optimalizaci vyhledávačů. Top 7 rozšíření se pyšní počtem stažení přesahující 3 milióny.

Celková kvalita dostupných nejpopulárnějších WordPress pluginů je chvalitebná. Tyto rozšíření mají pěkně zpracované grafické prvky a detailní technickou dokumentaci.

Pluginy lze nainstalovat přímo z vestavěného backendového prohlížeče, nebo pomocí instalace zipového souboru pluginu. Nastavení jednotlivých pluginů lze nalézt v backendu systému pod příslušnou položkou.

Joomla

Rozšíření systému Joomla jsou dostupné z `https://extensions.joomla.org/`. Tento portál nabízí volně dostupné i placené pluginy v celkovém počtu 7905 rozšíření. Mezi nejoblíbenější položky portálu patří Akeeba Backup, nástroj určený k záloze stránek. Další oblíbenou položkou je Breezing Forms, placený správce formulářů. Bohužel Joomla neuvádí počty stažení.

Grafické zpracování jednotlivých pluginů není na takové urovni jako WordPress. Neurazí, ale ani nenadchne. Technické dokumentace jsou rozsáhlé, ale méně přehledné než u WordPressových rozšíření.

Tyto pluginy lze instalovat v backendové části stránek pomocí instalace zipového souboru.

Drupal

Drupal na svých oficiálních stránkách nabízí přes 38 tisíc volně dostupných pluginů. Bohužel jednotlivé pluginy neposkytují žádné logo, pouze jejich snímek, a díky tomu je portál lehce nepřehledný. Avšak grafické zpracování pluginy nahrazují svoji funkčností. Rozšíření Drupalu jsou totiž velice účinné nástroje pro správu systému.

Nejstahovanější plugin se nazývá Chaos tool suite. Tato souprava nástrojů a API pomáhá lépe spravovat systém Drupal. Mezi jeho funkce patří nástroj pro práci s AJAX formuláři, práci s modálními okny, cachování objektů a další. Tento balíček si už stáhlo přes 8 milionů uživatelů, a aktivně jej využívá 880 tisíc.

Instalace rozšíření probíhá pomocí archivu pluginu v backendové části aplikace.

Django CMS

Instalace pluginů v Django CMS je odlišná od zbylých porovnávaných systémů. Pluginy lze instalovat přímo z adresy `https://marketplace.django-cms.org/en/addons/`, ale stránky musejí být hostované u společnosti Divio (společnost, která stojí za Django CMS). Pokud chceme rozšíření instalovat lokálně, nebo u jiného hostingu, musíme ho do projektu manuálně importovat. Tyto rozšíření jsou v podstatě Django aplikace, o kterých bude řeč v praktické části práce.

Prvním krokem instalace je zkopírování aplikace do adresáře projektu. Většina Django CMS aplikací je dostupných na GitHubu a jednoduchým příkazem *git clone REPOZITAR* ji do projektu zkopírujeme. V případě, že autor aplikace vydá

novou verzi, pomocí příkazu *git pull* (v adresáři aplikace) aplikaci zaktualizujeme. V nastavení je nutné aplikaci přidat do *INSTALLED_APPS*, aktualizovat soubor *urls.py* a provést migraci databáze. Poté můžeme rozšíření na jednotlivé stránky manuálně přidat. [5] Této problematice se dále věnuje praktická část práce.

2.6 Šablony

Šablona systému má na starosti vzhled frontendu stránek a jedná se o jednu z nejdůležitějších částí systému. Vizuální podoba stránek je totiž první věc, kterou návštěvník hodnotí. Dobře navrhnutá a implementovaná šablona zajistí přehlednost obsahu a lehkou navigaci stránek.

Jak již bylo řečeno Joomla a Drupal využívají šablonovací jazyk Twig, Django CMS využívá Django Template Language (ukázka 2.2) a WordPress čisté PHP (ukázka 2.1).

```
<!--WordPress-->
<ul>
    <?php $query = new WP_Query( array( 'post_type' => 'post'
       ) ); ?>
    <?php if ( $query->have_posts() ) : ?>
        <?php while ( $query->have_posts() ) :
           $query->the_post(); ?>
            <li><a href="<?php the_permalink() ?>"> <?php
               the_title() ?> </a></li>
        <?php endwhile; ?>
    <?php else : ?>
        <li><?php _e( 'Sorry, no posts matched your criteria.'
           ) ?></li>
    <?php endif; ?>
    <?php wp_reset_postdata(); ?>
</ul>
```

Ukázka kódu 2.1: Cyklus FOR v PHP [10]

```
<!--Twig, DTL-->
<ul>
    {% for post in posts %}
        <li><a href="{{ post.permalink }}">{{ post.title
           }}</a></li>
    {% else %}
        <li>Sorry, no posts matched your criteria</li>
    {% endfor %}
</ul>
```

Ukázka kódu 2.2: Cyklus FOR v Twig, DTL [10]

WordPress

Šablon dostupných pro systém WordPress je opravdu nepřeberné množství. A právě to je největší problém při výběru šablony pro systém WordPress. V angličtině existuje krásné slovní spojení, které takovou situaci perfektně vystihuje. Paralysis by analysis, neboli paralýza analýzou. K dispozici je takové kvantum šablon, že je problém najít opravdu kvalitní variantu, a která provozovatele stránek finančně nepřivede na mizinu. K dispozici jsou varianty od jednoduchých volně dostupných šablon až po profesionální šablony za několik stovek dolarů, plus výdajů na technickou podporu.

Oficiální repozitář šablon `https://wordpress.org/themes/` nabízí necelých 20 tisíc volně dostupných variant ke stažení. Šablony z tohoto portálu lze nainstalovat pomocí backendového prohlížeče šablon. Pokud však chce uživatel šablonu profesionálnějšího rázu, musí se pustit do jejího hledání. Externí šablony se instalují nahráním archivu šablony v backendu systému. Celková kvalita WordPressových šablon je na výborné úrovni. I jednoduché volně dostupné šablony vypadají pěkně a moderně (tj. žádné hrozné gradienty, přehnané stíny).

Joomla

Joomla trpí podobným problémem jako WordPress. K dispozici je obrovské množství šablon, a absence oficiálního Joomla repozitáře moc nepomáhá. Šablony lze získat volně dostupné i placené, oproti WordPressu jsou však o maličký stupínek hůře zpracované. Instalace probíhá podobně jako u WordPressu nahráním archivu šablony v backendu systému.

Drupal

Šablony dostupné z `https://www.drupal.org/project/project_theme` jsou spíše základy pro šablonu. Od uživatele se očekává, že daný základ vezme a postaví na něm šablonu podle svého vkusu. Uživatelé, kteří chtějí hotový produkt, musí šablonu s požadovanými vlastnostmi na internetu najít. Různé varianty lze najít volně dostupné i placené a svým zpracováním jsou lehce pod úrovní Joomly. Instalace je stejná jako u WordPressu a Joomly, tedy nahráním archivu se šablonou do backendu systému.

Django CMS

Django CMS žádné oficiální šablony nenabízí, systém ani funkci pro instalaci a změny šablon nemá implementovanou. Jedním z důvodů je uživatelova absolutní volnost a kontrola nad frontendem stránek. Pro tento systém lze použít v podstatě jakoukoliv šablonu, stačí pouze dummy text[5] nahradit DTL[6] značkami.

2.7 Optimalizace vyhledávačů

Search engine optimalization je velice důležitou operací při správě webu, která může náš web zviditelnit. Vyhledávače používají tzv. crawlers[7], což jsou automatizované programy, které jednotlivé stránky prolézají, skenují a indexují. Mezi skenované data patří v podstatě veškerý obsah stránek, ale nejdůležitější pro proces indexování jsou nadpisy, obrázky, klíčová slova a odkazy na jiné stránky. Indexování je proces, ve kterém se tato data uloží do obrovských databází a následně setřídí pomocí hodnotícího algoritmu. Tento algoritmus vyhodnocuje, která data jsou nejvíce relevantní s námi požadovanou vyhledávanou frází [11]. Pomocí optimalizace vyhledávačů obsah našich stránek crawlerům a jejich požadavkům přizpůsobujeme.

SEO je tedy klíč ke zviditelnění našich stránek pomocí vyhledávacího enginu. Samozřejmě pokud nepočítáme zaplacené přední příčky.

[5] ukázkový obsah šablony
[6] Django Template Language
[7] česky žížala, šplhoun, "prolézávač"

WordPress

I přes svoji rozšířenost není WordPress v základním nastavení velice SEO přátelský. Systém se musí optimalizovat pomocí externích pluginů. Například SEO Yoast je volně dostupný plugin, který zvládá funkce typu náhledu stránek ve vyhledávači, analýzy stránek, správy metadat, usnadnění práce crawlerům, XML mapu stránek a mnohé další operace, které pomůžou dostat stránky na vyšší pozice ve vyhledávači. Tento plugin aktuálně používá přes 3 milióny stránek.

Joomla

Joomla, stejně jako WordPress, je bez správcova zásahu v podstatě nevyhledatelná. Tento problém, stejně jako u WordPressu, lze řešit externími pluginy. Nejpopulárnější Joomla SEO plugin v oficiálním repozitáři je sh404SEF. Tento plugin upravuje obsah stránek tak, aby vyhovovaly požadavkům vyhledávacích enginů.

Drupal

Drupal nabízí zkušenějšímu provozovateli stránek větší kontrolu nad obsahem, a tudíž i nad SEO. Díky tomu není třeba žádné další rozšíření. Provozovatel však musí optimalizaci provést sám. V případě, že by radši tento proces přenechal zkušenějším, existuje několik pluginů ke správě optimalizace (např. Drupal SEO Tools).

Django CMS

Django CMS je v podobné situaci jako Drupal. Počítá se totiž s tím, že si SEO zajistí sám provozovatel stránek. Případně lze použít SEO aplikaci přímo určenou pro webové stránky postavené na Djangu - Django SEO Framework.

3 Analýza a návrh redakčního systému

Jelikož se redakčním systémem dá nazvat jakýkoliv nástroj, pomocí kterého spravujeme obsah webu, je třeba vhodně zvolit rozsah ukázkového systému. Základní blog pro přidávání příspěvků jedním autorem není dostatečně komplikovaný na to, aby poukázal na problémy spojené s uživatelskými účty, právy, kontrolou obsahu, interakci mezi uživateli a další. Na druhou stranu složitější projekt typu WordPress je programátorská záležitost na několik měsíců.

3.1 Popis ukázkového systému

Autor práce si jako dostatečně komplexní ukázku redakčního systému zvolil systém pro správu obsahu online kuchařky s prvky sociální sítě. Na této ukázce lze demonstrovat práci s uživatelsky generovaným obsahem, uživateli, práci se šablonou, šablonovacím jazykem, modelování aplikace, práce s obrázky a médii.

3.1.1 Volba názvu projektu

Přestože se jedná pouze o ukázkový projekt, tak i tento jednoduchý redakční systém si zaslouží svůj název.

Moderní start-upy a aplikace mají všelijaké názvy, a velká část z těchto názvů končí na -er. Následně tvůrci těchto projektů pár souhlásek z názvu vypustí, ať už z důvodu obsazenosti domény, nebo z jiných důvodů. Můžeme se tedy setkat s názvy jako Tumblr, Flickr, Dopplr, Grindr a mnohé další.

Autor si pro svůj ukázkový projekt zvolil základ jména Bachelor[1], pár souhlásek vypustil, jednu samohlásku přidal a vznikl moderní název Bchlrr (obrázek 3.1).

[1] Bachelor - angl. bakalář

Obrázek 3.1: Logo kuchařky Bchlrr

3.1.2 Funkce projektu

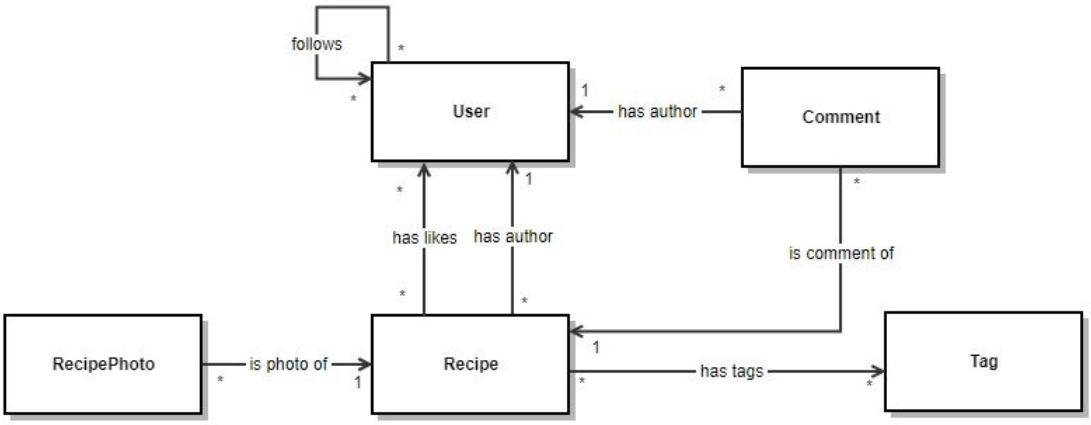
Základem kuchařky jsou recepty, které obsahují informace potřebné k úspěšné zhotovení jídel, např. ingredience, doba přípravy, samotný recept. Tyto recepty může jakýkoliv návštěvník i uživatel procházet a za pomoci dostupných filtrů třídit. Recepty může přidávat pouze registrovaný uživatel. Tento uživatel také smí své recepty upravovat a mazat. K receptu lze přiřadit několik fotek a vybrat si hlavní fotku, která bude zobrazena jako úvodní fotka receptu. Recept mohou registrovaní uživatelé komentovat. Mezi sociální prvky patří možnost přidat recept mezi oblíbené, neboli "lajknout". Všechny své oblíbené recepty poté uživatel nalezne na svém profilu, nebo úvodní stránce, pod položkou Oblíbené. V případě, že uživatel objeví dalšího uživatele, jehož recepty se mu líbí, nebo často přidává velice dobré recepty, může začít tohoto uživatele sledovat. Pomocí filtru sledovaných uživatelů[2] nalezne recepty pouze od uživatelů, které sleduje. Nepříliš sociální komponentou, ale na dnešních sociálních sítí velice rozšířenou, jsou tzv. značky (také známé jako tagy, mezi mladými lidmi hlavně jako hashtagy). Uživatel při vytváření nového receptu může tento recept "označkovat" klíčovými slovy, které vystihují daný recept. Díky těmto tagům jsou recepty kategorizované a velice dobře vyhledatelné. Při vyhledání požadovaného tagu se zobrazí veškeré recepty, které jsou tímto tagem označené. Předběžný diagram modelu je zobrazen v obrázku 3.2.

Filtrování receptů je rozděleno do několika skupin. Všechny recepty, Sledované recepty, Oblíbené recepty a Moje Recepty. Všechny recepty jsou přístupné všem návštěvníkům, zbylé tři skupiny pouze přihlášeným uživatelům. Ve všech těchto skupinách lze vyhledávat recepty podle zadaných frází a následně třídit - od nejnovějších, nejlepších nebo abecedně. Vyhledávání tagů je oddělené, při vyhledání tagu se zobrazí seznam receptů, které tento tag obsahují. Tento seznam lze také seřadit.

Administrátor má přístup do backendu, ze kterého může celou kuchařku spravovat. Z backendu lze všechny recepty a uživatele upravovat či mazat.

[2] obdoba follow feedu

3.1.3 Návrh aplikačního modelu



Obrázek 3.2: Předběžný aplikační model

User

Třída *User* reprezentuje jednotlivého registrovaného uživatele. Hlavními atributy jsou přihlašovací jméno a heslo, těmito údaji se uživatel přihlašuje do systému. Booleanovská hodnota *is_superuser* rozhoduje, zda-li je uživatel administrátorem, a má k dispozici nástroje s administrací spojené. Atribut jméno představuje uživatelovo zobrazované jméno. Uživatel má možnost na svůj profil nahrát profilovou a úvodní fotku. Pokud tak neučiní, jsou k dispozici výchozí fotky. Uživatel má možnost přidávat nové recepty, editovat vlastní, sledovat ostatní uživatele a přidávat recepty do svých seznamu oblíbených.

Recipe

Hlavní atribut třídy *Recipe* je jeho název. Při vytváření receptu má uživatel možnost přidat ingredience, dobu přípravy, počet porcí, značky, fotky a samotný recept. Atribut datum přidání je automaticky nastaven na datum vytvoření receptu. Atributy datum úpravy a booleanovská hodnota *edited* jsou určené pro editaci receptu. Autor receptu je automaticky nastaven při vytváření receptu.

Comment

Třída *Comment* obsahuje pouze jediný modifikovatelný atribut, a to obsah komentáře. Autor komentáře a recept, ke kterému komentář patří jsou automaticky nastaveny při vytváření objektu komentáře.

3.2 Použité technologie

3.2.1 Python

Python je jazyk ideální pro začátečníky. Má jednoduchou a čistou syntaxi - k odsazování se používá tabulátor nebo mezery. V Pythonu se pro bloky kódu nepoužívají závorky. Podporuje tři programovací paradigma - procedurální, funkcionální a objektové paradigma. Z tohoto důvodu není potřeba programy balit do tříd, nebo importovat nějaké knihovny. Navíc jsou jeho jádro a náročné funkce napsány v jazyku C, čímž tento jazyk zvládá výpočty velice rychle. Má mnoho datových typů, dokáže například počítat i s komplexními čísly. [12]

První implementace jazyka Python spatřila světlo světa v prosinci roku 1989. První stabilní verze poté v lednu 1994. Od té doby se Python neustále vyvíjí. Aktuální verze jazyka 3.6 byla vydaná 23. prosince 2016. V této verzi byly představeny nové syntaxové vlastnosti, vylepšení bezpečnosti, nové formátování stringových literálů a mnohé další.[13]

3.2.2 Django[3]

Django je vyspělý webový framework napsaný v jazyce Python, který podporuje rychlý vývoj a čisté, pragmatické konstrukce.

Historie frameworku Django začíná na podzim roku 2003, když programátoři novinového výtisku Lawrence Journal-World, Adrian Holovaty a Simon Willison, začali používat Python pro vývoj systému dedikovanému novinovému portálu. V létě 2005, kdy se tento systém stal plnohodnotným frameworkem pro správu obsahu, byl Django vydán jako open source software. Framework byl pojmenován po jazzovém kytaristovi Djangovi Reinhardtovi. [14]

A právě díky vzniku v novinářském prostředí je framework Django velice vhodný pro "obsahové"stránky. Stránky, které nabízí dynamický, databází poháněný ob-

[3] v projektu byl použit pouze samotný framework Django **bez** systému Django CMS

sah.

Aktuální verze v době vzniku této práce je 1.11. Tato verze představila SubQuery, neboli skládání QuerySetů. Renderování formulářových prvků přímo šablonou, místo dříve používaných Pythonových komponent. Class-based indexovaný model, který usnadňuje práci s daty přímo nad databází (např. řažení).

Mezi nejznámější společnosti využívající Django patří Instagram, sociální síť pro sdílení fotek. Mozilla, nadace bojující za otevřený internet. Disqus, plugin pro správu komentářů a Pinterest, někdy také nazývaný veřejná nástěnka internetu.

Django funguje na principu MVC[4] architektury [15], která aplikaci rozděluje do tří logických komponent.

Model layer

Modelová vrstva (ukázka 3.1) zajišťuje data aplikačního modelu a práci s databází. Dotazy na databázi ve frameworku Django probíhají pomocí tzv. QuerySetů (ukázka 3.2).

```
#model receptu
class Recipe(models.Model):
    # FOREIGN KEY autor receptu
    recipe_author = models.ForeignKey(User)
    # VARCHAR(100) název receptu
    title = models.CharField(max_length=100)
    # VARCHAR(MAX) samotný recept
    content = models.TextArea()
```

Ukázka kódu 3.1: Práce s aplikačním modelem

```
# QuerySet ekvivalentu SELECT * FROM Recipes
Recipe.objects.all()

# QuerySet ekvivalentu SELECT * FROM Recipes WHERE
   recipe_id='id'
Recipe.objects.filter(recipe_id=id)
```

Ukázka kódu 3.2: Django QuerySet

[4]Model-View-Controller

View layer

Pohledová vrstva je Controller částí MVC architektury. Tato vrstva zajišťuje požadavky uživatele a vrací na ně příslušné odpovědi. Prvek pohledové vrstvy je pohled, což může být metoda, která přijímá požadavek a vrací příslušnou odpověď, nebo generická třída, která již v sobě má zakomponované metody pro různé požadavky (práci s objekty, GET a POST metody, kontextová data a další). Ukázka 3.3 znázorňuje URL požadavek a s ním spojenou pohledovou metodu, která vrací detail receptu.

```
# urls.py
url(r'^recipe/(?P<id>\d+)/$', 'recipe_detail_view',
   name='recipe_detail')
# views.py
def recipe_detail_view(request, recipe_id):
    qs = Recipe.objects.get(id=recipe_id)
    return render(request, 'detail.html', {'recipe': qs})
```

Ukázka kódu 3.3: Příklad pohledu

Template layer

Šablonová vrstva je ekvivalentem View části MVC architektury. Tato vrstva zajišťuje renderování frontendových stránek. Jazyk, který umožňuje oddělení HTML a Pythonu se nazývá DTL[5]. Tento jazyk používá velice podobný syntax jako šablonovací jazyk Twig pro PHP. Pythonové operace se uvozují složenými závorkami a procentem, proměnné dvojitými složenými závorkami (ukázka 3.4 a 3.5).

```
<!--detail.html-->
<!--/recipe/segedin/-->
{% if recipe %}
    <h1>{{recipe.title}}</h1>
    <h2>Publikoval: {{recipe.recipe_author}} </h2>
    <p>{{recipe.content}}<p>
{% else %}
    <h1>Recept nenalezen.</h1>
{% endif %}
```

Ukázka kódu 3.4: Šablona detailu receptu, podmínka IF

[5]Django Template Language

```
<!--recipe_list.html-->
<!--/recipes/-->
<ul>
{% for recipe in recipe_list %}
    <li>
        <!--url metoda zjištění absolutní URL adresy-->
        <a href="{% url 'recipe_detail' recipe.id %}">
            {{ recipe.title }}
        </a>
    </li>
{% empty %}
    <li>Žádný recept nenalezen.</li>
{% endfor %}
</ul>
```

Ukázka kódu 3.5: Šablona seznamu receptů, cyklus FOR

Zpracování požadavku

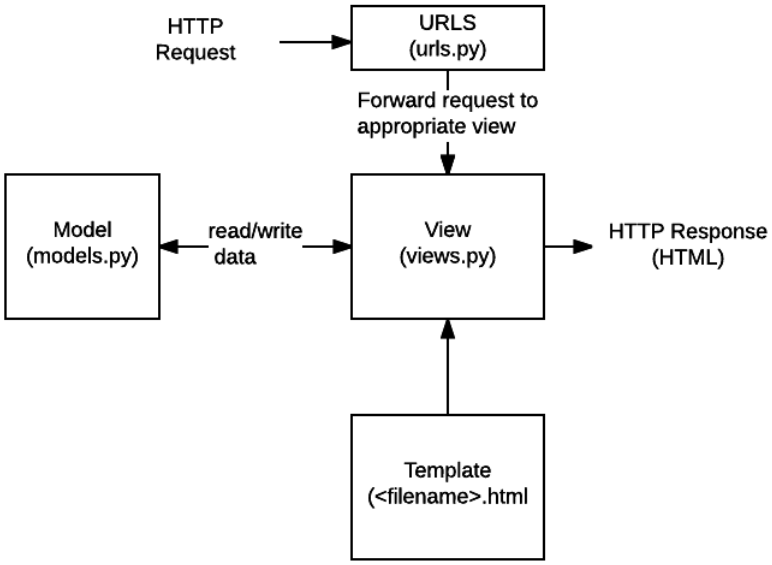
Při vzniku požadavku porovná Django požadovanou URL adresu s adresami v souboru *urls.py*. S každou jednotlivou URL adresou v tomto souboru je spojen právě jeden pohled. Tento pohled se postará o dotaz na databázi a požadovaná data vrací spolu s příslušnou šablonou. Šablona je poté za pomoci DTL převedena do běžného HTML a vrácena prohlížeči uživatele. Tento cyklus požadavku je znázorněn na obrázku 3.3.

Další funkce frameworku Django

Django nabízí nespočet dalších funkcí a velká část jich bude dopodrobna popsána v implementační části práce spolu s příslušnou ukázkou na konkrétním problému.

3.2.3 Bootstrap

Bootstrap je sada nástrojů pro vytváření webových stránek a aplikací. Obsahuje připravené HTML a CSS upravitelné šablony pro typografii, tlačítka, navigace, tabulky, formuláře a další komponenty uživatelského rozhraní webu. Obsahuje rovněž rozšíření pro využití JavaScriptu.



Obrázek 3.3: Cyklus požadavku na systém Django [16]

Hlavní předností Bootstrapu je grid layout. Jedná se o responzivní mobile-first fluidní mřížkový systém, který se skládá ze 12ti sloupců, a který vhodně škáluje na základě šířky viewportu. V ukázce 3.6 je nastíněn layout tří divů, které jsou široké čtyři sloupce. Tento layout na středních a velkých displejích (786px a více) zobrazí tři položky vedle sebe. Na zařízeních s obrazovkou menší než 768px se tyto položky zobrazí pod sebou. Čtenáře, které Bootstrap problematika zajímá se mohou více dočíst na adrese `http://getbootstrap.com/css/`. Bohužel Bootstrap není tématem této práce a dále se jím již zabývat nebude.

```
<div class="row">
  <div class="col-sm-4">První položka</div>
  <div class="col-sm-4">Druhá položka</div>
  <div class="col-sm-4">Třetí položka</div>
</div>
```

Ukázka kódu 3.6: Ukázka Bootstrap grid layoutu

Základ šablony, ze které kuchařka Bchlrr vychází, je postavená na Bootstrapu. Jedná se o šablonu Clean Blog [17] od autora Davida Millera.

3.2.4 Git a GitHub

I když je tato záležitost mimo rozsah tématu práce, rozhodl se autor letmý popis Gitu do práce zařadit. Jedná se totiž o velice versatilní nástroj s velkým množstvím využití.

Git je moderní verzovací systém poprvé vyvinut v roce 2005 Linusem Torvaldsem, tvůrcem Linuxového jádra, a to z důvodu zpřehlednění verzí a změn při vývoji kernelu. Git vede lokální historii vývoje zdrojového kódu, ke kterému lze kdykoliv přistoupit. Git dále umožňuje vytváření tzv. branchí[6] pro různé fáze vývoje, např. produkční branch, vývojová branch. Změny v jedné branchi neovlivní ostatní branche. Lokální repozitář lze založit pomocí příkazu *git init*, soubory se do repozitáře přidají příkazem *git add xyz* a jakékoliv změny se uloží příkazem *commit -m 'popis změny kódu'*. Pomocí funkce *push* lze repozitář nahrát do vzdáleného úložiště, odkud si ho mohou ostatní uživatelé stáhnout (*clone, pull*) a provádět vlastní změny. Tyto funkce dovolují, a velice zjednodušují práci více autorů na jednom projektu. [18]

GitHub je neznámější a nejrozšířenější služba pro hostování Gitových repozitářů. Alternativy k GitHubu jsou BitBucket, GitLab, SourceForge aj.

[6] větví

4 Implementace systému

Kapitola implementace systému se věnuje samotnému vývoji kuchařky Bchlrr. Autor se snažil co nejvíce vývoj aplikace popsat, avšak není možné zachytit vše. Jak již bylo řečeno v úvodu práce, veškerý zdrojový kód a rozšiřující informace jsou dostupné z autorova GitHubu `https://github.com/horca/bchlrr_cookbook/`, případně v přílohách práce. Je možné, že GitHubová verze bude oproti verzi v této práci obsahovat nějaké funkce navíc. Z důvodu překvapivého zalíbení v projektu je autor rozhodnutý ve vývoji kuchařky pokračovat a nové funkce neustále přidávat.

Veškeré návody potřebné ke zprovoznění projektu se nachází v přílohách práce.

4.1 Zprovoznění webové aplikace

4.1.1 Virtuální prostředí

Dříve, než se pustíme do samotného programování, je třeba si připravit vývojové prostředí. Tento krok není nutný, ale velice usnadní práci s projektem a jeho celkovou přehlednost. Python pro jednotlivé projekty nabízí tzv. virtual environment, neboli virtuální prostředí. Virtuálnímu prostředí můžeme nastavit verzi Pythonu, která bude specifická právě pro toto prostředí. Na mnohých linuxových distribucí můžeme nalézt instalaci jak Pythonu verze 2, tak verze 3. Pomocí virtuálního prostředí specifikujeme verzi právě pro náš projekt. Stažení a instalace externího balíčku při používání virtuálního prostředí se projeví pouze v jeho rámci, a neovlivní ostatní balíčky nainstalované na počítači. V případě, že už na počítači programátor jiné projekty vyvíjel a virtuální prostředí nepoužil, může se stát, že jsou tyto projekty závislé přímo na verzi nainstalovaného balíčku v počítači. Avšak ve svém novém projektu by rád použil novější verzi balíčku. Toto je typický příklad výhody těchto "lokálních" balíčků virtuálního prostředí.

Instalace a aktivace v unixovém prostředí

```
$ mkdir DJANGOPROJECT
$ cd DJANGOPROJECT
$ python -m venv myvenv
$ source myvenv/bin/activate
```

Ukázka kódu 4.1: Virtualenv pod unixovou platformou

Instalace a aktivace v systému Windows

```
> mkdir DJANGOPROJECT
> cd DJANGOPROJECT
> python -m venv myvenv
> myvenv/Scripts/activate
```

Ukázka kódu 4.2: Virtualenv v systému Windows

Po úspěšné aktivaci virtuální prostředí je třeba ověřit pár věcí. Zda-li máme správnou verzi Pythonu a zda-li je nainstalovaný pip. Pip, stejně jako GNU, je rekurzivní zkratka, která znamená "Pip Install Python". Jedná se o package manager Python balíčků, obdoba známých Ruby Gems.

```
(myvenv)$ python --version
Python X.X.X
(myvenv)$ pip --version
pip X.X.X from PIP_PATH
```

Ukázka kódu 4.3: Ověření instalace Pythonu a pipu

Pokud kontrola proběhla v pořádku, můžeme se vrhnout na další krok. [5]

4.1.2 Vytvoření nového projektu Django

Před vytvořením nového Django projektu musíme Django nainstalovat.

```
(myvenv)$ pip install Django
```

Ukázka kódu 4.4: Instalace Django

Nyní máme vše připraveno a můžeme se pustit do nového Django projektu.

```
# založení nového projektu Django
(myvenv)$ django-admin startproject NAZEVPROJEKTU

# zjištění změn mezi modelem a databází
(myvenv)$ python manage.py makemigrations

# převedení modelu aplikace do databáze
(myvenv)$ python manage.py migrate

# vytvoření administrátora
(myvenv)$ python manage.py createsuperuser

# spuštění webové aplikace
(myvenv)$ python manage.py runserver
```

Ukázka kódu 4.5: Založení nového Django projektu

Pokud vše proběhlo úspěšně, na adrese 127.0.0.1:8000 (nebo localhost:8000) běží funkční Django aplikace.

4.1.3 Nastavení databáze

Ve výchozím nastavení pracuje Django s databází SQLite, která je obsažena v Pythonu, a která se automaticky vygeneruje při první migraci. Tato databáze je perfektní pro vývoj aplikace, avšak ve větším projektu reálného světa je doporučeno použít jinou databázi. Django podporuje SQLite, PostgreSQL, MySQL a Oracle Database. Pro účely tohoto projektu a práce se autor rozhodl pro databázi SQLite, která funguje bez dalších potřebných nastavení.

Při jakékoliv změně modelu aplikace je třeba provést operaci *makemigrations*, která zjistí změny v modelu a operaci *migrate*, která změny do databáze zapíše.

Aby Django třídu modelu registroval jako databázovou entitu, musí tato třída dědit z třídy *Model* z Django balíčku *models*.

4.2 Struktura projektu Django

V kořenové složce projektu se nachází soubor *manage.py*, pomocí kterého se celá aplikace spravuje. Dále se zde nachází složka se stejným názvem jako projekt.

V této složce nalezneme soubor *settings.py*, který obsahuje veškeré nastavení týkající se projektu. Další soubor, který nás bude zajímat, je *urls.py*. Jedná se o soubor, ve kterém je seznam všech možných URL požadavků.

Veškeré další soubory, týkající se modelů a práce s modely, se člení na tzv. aplikace. Pro projekt kuchařka Bchlrr byly zvoleny dvě aplikace. Aplikace *recipes*, která řeší veškerou práci s recepty. A aplikace *profiles*, která řeší veškeré operace spojené s uživatelskými účty.

V každé takové aplikace nalezneme několik důležitých souborů. Soubor *admin.py* řeší registrace vlastního modelu do backendu aplikace, *forms.py* se stará o formuláře, které se využívají při operacích s objekty. Soubor *models.py* obsahuje veškeré modely spojené s danou aplikací. Pomocí souboru *urls.py* lze rozšířit hlavní *urls.py* soubor, který se nachází ve složce s nastavením, a zpřehlednit tak URL spojené právě s danou aplikací. Poslední soubor se nazývá *views.py* a má na starosti všechny pohledy spojené s modelem dané aplikace. Aplikaci založíme pomocí příkazu *python manage.py startapp nazev_aplikace* v kořenovém adresáři projektu. Aplikaci je poté nutno přidat do seznamu nainstalovaných aplikací *INSTALLED_APPS* v souboru *settings.py*. Ukázka 4.6 obsahuje strukturu nově vygenerovaného projektu s aplikací *exampleapp*.

```
HelloWorld
|   manage.py
|---exampleapp
|   |   admin.py
|   |   apps.py
|   |   models.py
|   |   tests.py
|   |   views.py
|   |   __init__.py
|   |---migrations
|           __init__.py
|---HelloWorld
    |   settings.py
    |   urls.py
    |   wsgi.py
    |   __init__.py
    |---__pycache__
```

Ukázka kódu 4.6: Struktura projektu HelloWorld s aplikací *exampleapp*

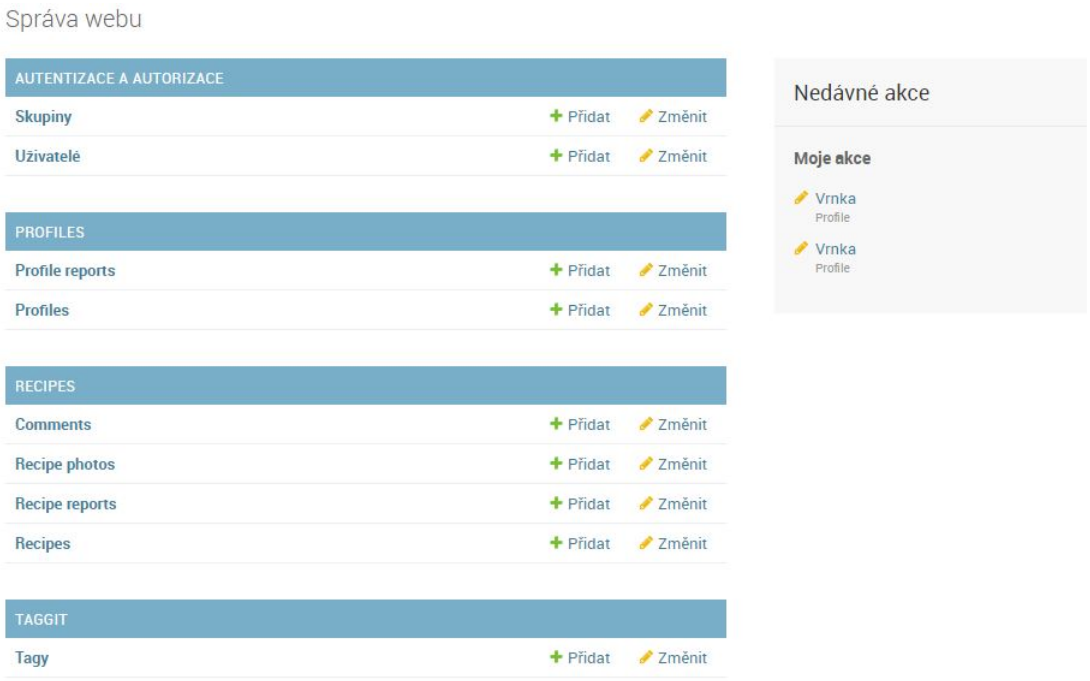
4.3 Backend aplikace

O backend aplikace se stará Django. Již ve výchozím nastavení je bez jakýkoliv externích balíčků k dispozici velice účinný backend pro správu obsahu kuchařky. V prostředí backendu můžeme vytvářet a pracovat se všemi objekty projektu, jak lze vidět na obrázku 4.1.

Abysme mohli v backendu přistupovat k vlastním modelům, je však třeba jedné operace. Je zapotřebí vlastní model registrovat v souboru *admin.py* (ukázka 4.7) nacházející se v příslušné aplikaci.

```
1 # admin.py
2 from .models import Recipe
3 admin.site.register(Recipe)
```

Ukázka kódu 4.7: Registrace vlastního modelu



Obrázek 4.1: Snímek backendu již hotového projektu kuchařka Bchlrr

4.4 Identifikace objektů

Pro přístup k jednotlivým objektům pomocí URL požadavku potřebujeme dané objekty nějakým způsobem rozlišit. Django v základním nastavení nabízí dva různé identifikátory. První možností je primární klíč objektu - atribut id, druhou možností je tzv. slug. Atribut id automaticky obsahuje každý objekt aplikace Django, ať už je tento atribut definovaný nebo ne. Proto se tento identifikátor jeví jako ideální. Bohužel URL adresa *http://www.bchlrr.cz/recipe/23/* uživateli mnohé neřekne a zároveň není vhodná pro optimalizaci vyhledávacích enginů. U adresy *www.bchlrr.cz/recipe/segedin/* takové problémy nehrozí. Tento způsob identifikace objektů se nazývá slug a v projektu kuchařky byl použit. Slug je ve většině případů dynamicky generován z názvu článku, receptu, zprávy. U takto generovaného slugu si však musíme dát pozor na jeho jedinečnost. V případě, že by se dva recepty jmenovali stejně, nastává problém. A z tohoto důvodu byl v projektu použit vlastní generátor slugů, který si na jejich jedinečnost dává pozor. V případě již existující slugu generátor přidá ke slugu objektu unikátní příponu. Tento generátor se nachází v souboru *utils.py*, vždy v příslušné aplikaci.

4.4.1 Regulární výrazy

Podobně jako Git, tak i regulární výrazy jsou mimo téma této práce. Avšak Django je využívá při validaci URL požadavků (ukázka 4.8) a autor je v projektu použil při parsování tagů receptu.

Regulární výraz (také regular expression nebo regex) slouží k vyhledání části řetězce, kterou předem (úplně) neznáme, nebo která může mít více podob. Používá se v programovacích a skriptovacích jazycích. [19]

```
1 r'^(?P<slug>[\w-]+)
2 # r'^ uvozuje regulární výraz
3 # ?P značí Pattern, česky vzor, v tomto případě slug
4 # [\w-] přijme veškeré znaky, obdoba [a-zA-Z0-9_]
```

Ukázka kódu 4.8: Regex slugu

Čtenářům, které dané téma zajímá, doporučuje autor navštívit stránky `http://www.regularnivyrazy.info/shrnuti-syntaxe.html`, kde naleznou syntax většiny regexů a mnohé další ohledně problematiky regulárních výrazů.

4.5 Aplikace recipes

4.5.1 Model receptu

Model aplikace byl již popsán v návrhu aplikace. Jeho kompletní implementaci lze nálezt v souboru *recipes/models.py*. Z důvodu rozsáhlosti nebyl kód metod třídy Recipe a pomocných tříd zahrnut v ukázce kódu 4.9.

```
1  class Recipe(models.Model):
2      title          = models.CharField(max_length=100,
           null=False, blank=False)
3      slug           = models.SlugField(editable=False)
4      author         = models.ForeignKey(Profile,
           related_name='profile_recipes')
5      ingredients    = models.TextField()
6      cook_time      = models.CharField(max_length=50,
           choices=COOK_TIME, default='MINUTE', null=False,
           blank=False)
7      servings       = models.CharField(max_length=10,
           null=False, blank=False)
8      content        = models.TextField()
9      date_added     = models.DateTimeField(auto_now_add=True)
10     date_edited    = models.DateTimeField(auto_now=True)
11     edited         = models.BooleanField(default=False)
12     likes          = models.ManyToManyField(Profile,
           related_name='profile_likes', blank=True)
13     tags           = TaggableManager(related_name='recipes')
14     # tags are not directly editable because their manytomany
           relations -> store duplicate tags as string in temp_tags
15     temp_tags      = models.CharField(max_length=200,
           null=True, blank=True)
16     # strip title all accent symbols i.e. ěščř... for correct
           ordering
17     unicode_title  = models.CharField(editable=False,
           max_length=100, default=title)
18     banner_photo   = models.ImageField()
```

Ukázka kódu 4.9: Model receptu

4.5.2 Signály

Signály jsou unikátní funkcí Djanga, spojenou právě s aplikačním modelem. Jedná se o signály (pokyny pro vykonání akce), které se zavolají v určitý okamžik. Existuje několik typů signálů: *pre_save, post_save, pre_delete, post_delete, pre_init, post_init, pre_migrate a post_migrate*. U modelu *Recipe* je signál *pre_save* použit k vygenerování a přiřazení unikátního slugu receptu (ukázka 4.10). Kompletní signály lze najít v souboru *models.py* v příslušné aplikaci.

```
1 # generování unikátního slugu před uložením objektu receptu
2 def add_unique_slug(sender, instance, *args, **kwargs):
3     if not instance.slug:
4         instance.slug = unique_slug_generator(instance)
5
6 pre_save.connect(add_unique_slug, sender=Recipe)
```

Ukázka kódu 4.10: Signál pro vytvoření slugu

4.5.3 Tagy

Značky, tagy, hashtagy receptu jsou řešeny pomocí externí aplikace pro správu tagů Django-Taggit. K jejich implementaci stačí objektu definovat atribut *tags = TaggableManager()*. Aby všechny značky receptu měly stejnou podobu (tj. začínaly mřížkou) je v projektu použit parser tagů, který přijímá značky v jakémkoliv formátu (oddělené mezerou, čárkou, malá a velká písmena), z těch poté vypustí všechny speciální znaky a interpunkci. Ke každé značce přidá mřížku a uloží ji k receptu. Tento nástroj se nachází v souboru *recipes/utils.py*

4.6 Aplikace profiles a Django uživatelský model

Podobně jako model receptu, byl již uživatelský model popsán v návrhu aplikace. Avšak na rozdíl od receptu je implementace uživatelského modelu složitější. Django již v sobě má zakomponovaný uživatelský model, který zajišťuje autorizaci uživatelů. Bohužel tento model není pro potřebu projektu dostačující. V návrhu aplikace autor stanovil, že uživatel bude mít možnost na svůj profil nahrát profilovou a úvodní fotku. Django uživateli žádné *ImageField* atributy nenabízí. Je tedy třeba tento model nějakým způsobem upravit.

Existují tři způsoby, jak vestavěný uživatelský model upravit nebo rozšířit. První způsob je tzv. proxy model. Jedná se o model dědičnosti, kdy třída např. *Profil* dědí právě z jedné třídy *User*. Tento způsob nijak nezmění podobu databáze, jelikož třída *Profil* nedědí ze třídy *Model* z Django balíčku *models*[1], a tedy není možné použít vlastní atributy profilu, které by se promítly v tabulce databáze. Tento způsob se volí pouze v případě, kdy nám stačí k profilu přidat nějakou funkci, metodu, změnit výchozí řazení nebo členit objekty podle námi požadovaných kritérií.

Dalším způsobem je použití One-to-One vazby mezi profilem a uživatelem. Vytvořením třídy *Profil* s vlastními atributy a propojení One-to-One vazbou s vestavěným Django uživatelem, dáme frameworku vědět, že třída *Profil* vestavěného uživatele rozšiřuje. Tato varianta je nejvhodnější pro projekt kuchařky, a také byla zvolena.

Třetí způsob úpravy uživatele je implementace vlastního uživatelského modelu na základě Django *AbstractUser*. Tento způsob by měl být brán v potaz na začátku projektu. Vyžaduje totiž velké zásahy do nastavení, databáze a celkového fungování projektu. Varianta vlastního uživatelského modelu by měla být zvážena pouze v krajních případech, např. u požadovaného vlastního procesu ověření uživatele. [20]

Atributy, kterými budeme Django uživatelský model rozšiřovat jsou vlastní jméno uživatele, slug pro identifikaci, *ImageField* profilová a úvodní fotka a uživatelovi sledovaní uživatelé. Vztah sledovaní uživatelé je řešen pomocí vazby Many-to-Many, tzn. jeden uživatel může několik dalších sledovat a několik dalších může sledovat uživatele. V tomto případě je u Many-to-Many vazby nutno specifikovat parametr *symetrical=False*, z důvodu zabránění vzájemného propojení, tj. pokud sleduji dalšího uživatele, neznamená to, že on sleduje mě (na rozdíl od Facebookových přátel nebo oblíbených receptů). Atributy, které využijeme z Django uživatele jsou přihlašovací jméno a heslo.

Zabezpeční hesla je řešeno pomocí hashování, a není tedy v databázi uloženo jako prostý text. Hashování probíhá pomocí 50ti místného klíče, který se nachází v souboru *settings.py* a je pro každý projekt unikátní (generovaný při vzniku). Klíč produkční verze aplikace by se nikdy neměl dostat na veřejnost.

[1] pokud by dědila, už by se nejednalo o proxy model

4.6.1 Model profilu

Následující ukázka kódu 4.11 byla z důvodu rozsáhlosti kódu zjednodušena. Metody, funkce a pomocné třídy byly vypuštěny úplně. Kompletní kód modelu Profil lze nalézt v souboru *profiles/models.py*.

```
class Profile(models.Model):
    user                = models.OneToOneField(User)
    name                = models.CharField(max_length=30)
    slug                = models.SlugField()
    profile_photo       = models.ImageField()
    profile_banner      = models.ImageField()
    followers           = models.ManyToManyField('self',
        related_name='following', blank=True, symmetrical=False)
```

Ukázka kódu 4.11: Model třídy Profil, která rozšiřuje Django User

4.6.2 Signály

U profilu také použijeme již dříve zmíněné signály. Jelikož rozšiřujeme Django uživatelský model, potřebujeme po vytvoření nového uživatele také vytvořit jeho profil. Toto zajistíme pomocí signálu *post_save* (ukázka 4.12).

```
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        # vytvoření nového profilu, s atributem user, který
           představuje nově vytvořeného uživatele
        profile = Profile.objects.create(user=instance)
        # nastavení jména profilu, defaultně totožné s
           přihlašovacím jménem
        profile.name = instance.username

post_save.connect(create_user_profile, sender=User)
```

Ukázka kódu 4.12: Signál, který novému uživateli vytvoří nový profil

U profilu, je stejně jako u receptu nastavený signál *pre_save*, který zajišťuje generování slugu profilu. Poslední věc, kterou u uživatelova účtu zajistíme signálem, je odstranění uživatele při odstranění profilu, a naopak. K této funkcionalitě využijeme signál *post_delete*.

4.7 Práce s obsahem

Modelovou vrstvu kuchařky máme připravenou. Je tedy na čase se pustit do zobrazování obsahu a práce s ním.

Práci s objekty zajišťují pohledy. Jak již bylo řečeno, každá URL adresa ze souboru *urls.py* je spojená s právě jedním pohledem. Při požadavku porovná Django požadovanou URL adresu s definovanými adresami a zavolá příslušný pohled. Ten poté požadavek vyhodnotí, provede požadované akce a vrátí odpověď v podobě kontextových dat a šablony. Tyto kontextové data obsahují QuerySet objektů a další nepovinná rozšiřující data, které jsou v projektu použity například k rozlišení právě aktivní nabídky. Veškeré pohledy se nachází v souboru *views.py* v příslušné aplikaci.

4.7.1 Class-based pohledy

V předchozí kapitole popisu použitých technologií byl znázorněn pohled založený na metodě. V projektu jsou však použity pohledy založené na generických třídách (class-based). Použití class-based pohledů má několik výhod. Tyto třídy již obsahují metody, které daný požadavek zpracují a vrátí příslušnou odpověď. Pokud nám tato metoda nevyhovuje, můžeme jí přepsat[2]. Class-based pohledy, které se nachází v projektu, jsou popsány v tabulce 4.1. Kompletní přehled lze nalézt na webové stránce `https://ccbv.co.uk/`

V následující ukázce 4.13 lze vidět syntaxi pohledu detailu receptu. Jediné, co stačí uvést je model receptu a šablonu. Nepovinná metoda *get_context_data* slouží k přidání kontextových údajů, ke kterým se dá přistoupit v šabloně stránky. V této ukázce přidáváme data *active_menu = 'recipe'*, která slouží k rozpoznání aktivní nabídky.

```
1 class RecipeDetailView(DetailView):
2     model = Recipe
3     template_name = 'recipes/recipe_detail.html'
4 
5     def get_context_data(self, **kwargs):
6         context['active_menu'] = 'recipe'
7         return context
```

Ukázka kódu 4.13: Pohled detailu receptu

[2]method override

Pohled	Funkce pohledu
View	Záměrně jednoduchá rodičovská třída pro všechny pohledy. Implementuje pouze metodu vrácení odpovědi a jednoduchou kontrolu požadavku.
CreateView	Pohled pro vytváření nového objektu instance, který u GET požadavku vrací odpověď a šablonu. Při POST požadavku vytváří nový objekt.
DetailView	Vrací jediný objekt a šablonu detailu tohoto objektu.
UpdateView	Pohled určený k editaci objektu. GET požadavek vrací objekt k úpravě a šablonu úpravy. POST požadavek objekt upraví.
DeleteView	Pohled určený ke smazání objektu.
ListView	Vrací šablonu listu a QuerySet objektů definovaný v metodě *get_queryset*.
TemplateView	Jednoduchý pohled, který vrací samotnou šablonu
LoginView	Pohled zajišťující přihlášení uživatele.
LogoutView	Pohled zajišťující odhlášení uživatele.

Tabulka 4.1: Základní Djagno class based pohledy [21]

Uživatelské práva a přístupy

Jedním z důvodu implementace uživatelské modelu je omezení přístupu a autorizace návštěvníka stránek. Anonymní návštěvník smí kuchařku pouze prohlížet. Jakmile se neautorizovaný uživatel pokusí obsah přidat, nebo jakkoliv upravit, je vyzván k přihlášení, případně registraci. Této funkcionality dosáhneme pomocí tzv. mixinů, přesněji pomocí *LoginRequiredMixin*.

Použití mixinů je speciální způsob, jak třídě přidat extra chování. Nejedná se o nic jiného, než jednoduché třídy, které poskytují modularitu a v Pythonu nejsou ničím výjimečným. Mixiny by měly mít jeden specifický účel a neměly by být instancovány nebo dál rozšiřovány. Typický příklad mixinu je Django *LoginRequiredMixin*, jehož jediná funkce je zjištění, zda-li je uživatel požadavku přihlášený (ukázka kódu 4.14).

```
class LoginRequiredMixin(object):

    def dispatch(self, request, *args, **kwargs):
        if not request.user.is_authenticated():
            raise PermissionDenied
        return super(LoginRequiredMixin, self).dispatch(
            request, *args, **kwargs)
```

Ukázka kódu 4.14: Django LoginRequiredMixin

Použití mixinu je velice jednoduché. Na rozdíl od Javy podporuje Python multiple inheritance[3], a díky tomu může Python třída dědit z více předků. Tento způsob použití je znázorněn na úkazce kódu 4.15. Tato ukázka obsahuje pohled *MyRecipesView*, který dědí z generického pohledu *ListView* a zároveň z mixinu *LoginRequiredMixin*. Tento pohled vrací list všech receptů přidaných uživatelem, který specifikujeme pomocí override metody *get_ queryset*. V této metodě se dotážeme na recepty, ve kterých je autorem uživatel požadavku. Pokud by si o daný pohled zažádal nepřihlášený uživatel, nastala by vyjímka, jelikož *AnonymousUser* žádné pole receptů nemá. A právě pomocí *LoginRequiredMixin* zajistíme, aby byl tento pohled dostupný pouze pro přihlášené uživatele. V nastavení projektu je nutno specifikovat redirect adresu *login_ url*, na kterou budou nepřihlášení uživatelé přesměrováni.

```
class MyRecipesView(LoginRequiredMixin, ListView):
    model = Recipe
    template_name = 'recipes/my_recipes.html'

    def get_queryset(self):
        qs = Recipe.objects.filter(
                author=self.request.user.profile
            )
        return qs
```

Ukázka kódu 4.15: Pohled, který vrací pouze recepty přidané uživatelem

Další problém, který vzniká ve spojitosti s uživatelskými účty, je editace a mazání obsahu cizího uživatele. Řešení této záležitosti lze provést přímo v pohledech spojených s danou operací. V projektu u pohledu *UpdateView* a *DeleteView* jsou

[3] vícenásobné dědění

přepsány metody *get_ queryset*, které vrací QuerySet uživatelů, kteří mají právo daný objekt upravovat, případně mazat.

```
def get_queryset(self):
    if self.request.user.is_superuser:
        return Profile.objects.all()
    return Profile.objects.filter(user=self.request.user)
```

Ukázka kódu 4.16: Zajištění editace pouze uživatelova vlastního obsahu

Kód 4.16 je úryvek metody z pohledu *ProfileEditView*. Metoda *get_ queryset* vrací QuerySet uživatele, který tento profil smí upravovat. V případě, že požadavek pochází od administrátora, je vrácen QuerySet všech profilů. Pokud se uživatel, od kterého požadavek pochází, v tomto QuerySetu nenachází, je přesměrován na chybovou stránku 404 - nenalezeno.

4.7.2 Šablony

Pohledy a uživatelské přístupy máme za sebou, což znamená, že můžeme začít tvorbu finální vrstvy MVC architektury, View Layer. Tato vrstva zajišťuje zobrazování jednotlivých stránek a v systému Django se nazývá Template Layer. Základní šablony se nachází v kořenovém adresáři projektu ve složce *templates*, šablony týkající se objektů aplikací se nachází ve složce *templates* v dané aplikaci.

Django Template Language

Poté, co nám pohled vrátí šablonu, je potřeba ji Djangem zpracovat a převést do běžného HTML (ukázka 4.17). Tato šablona totiž obsahuje kombinaci HTML a Django Template Language (dále jen DTL) značek, které běžný prohlížeč přečíst neumí. DTL nabízí nepřeberné množství značek a filtrů, pomocí kterých lze obsah šablony formátovat. Značky a filtry použité v praktickém projektu jsou uvedeny a vysvětleny v tabulce 4.2. Značky se uvádějí pomocí složených závorek a procenta, proměnné pomocí dvou složených závorek.

Značka/filtr	Funkce
{% url 'xyz' xyz.slug %}	Tato značka vrací absolutní cestu schodného pohledu
{% if xyz %} {% else %} {% endif %}	Značky podmínky IF
{% for xyz in xyz_list %} {% empty %} {% endfor %}	Značky cyklu FOR
{% extends xyz.html %}	Značka dědění
{% include xyz.html %}	Značka vložení snippetu
{{ xyz|truncatechars:10 }}	Filtr oříznutí proměnné po 10ti znacích
{{ xyz|safe }}	Filtr renderování textu proměnné jako HTML, namísto prostého textu
{{ xyz|capfirst }}	Filtr pro kapitalizaci prvního symbolu proměnné

Tabulka 4.2: Legenda použitých značek a filtrů DTL

```
<!--DTL-->
{% for recipe in recipes %}
    <a href='{% url 'recipe_detail' recipe.slug %}'>
        {{recipe.title}}
    </a>
{% endfor %}
<!--Po převedení do obyčejného HTML-->
<a href='/recipe/svickova-se-sesti'>Svíčková se šesti</a>
<a href='/recipe/segedin'>Segedín</a>
<a href='/recipe/nepeceny-dort'>Nepečený dort</a>
<a href='/recipe/segedin-x8ey'>Segedín</a>
<a href='/recipe/italske-rizoto'>Italské rizoto</a>
```

Ukázka kódu 4.17: Převedení šablony do čistého HTML

Jednou z funkcí tohoto značkovacího jazyku je dědičnost. Můžeme nakonfigurovat základní šablonu, která bude obsahovat CSS, faviconu, JavaScript, metadata a další znovupoužitelná data. Z této šablony mohou ostatní šablony dědit a využít tak kódu šablony základu.

Tzv. snippety jsou úryvky kódu, které se vyskytují ve více šablonách. Typic-

kým příkladem je navigační menu. DTL umožňuje tento kód definovat v jednom souboru, a poté ho do jiných souborů šablony 'vložit'. Ukázka snippetu 4.18 znázorňuje vložení navigačního menu podle uživatelovy autorizace.

```
{% if user.is_authenticated %}
    {% include 'snippets/navbar_logged_in.html' %}
{% else %}
    {% include 'snippets/navbar_logged_out.html' %}
{% endif %}
```

Ukázka kódu 4.18: Snippet navigačního menu závislého na přihlášení uživatele

4.7.3 Formuláře

Pro vytváření a úpravu dat objektů využívá Django HTML formuláře. Tyto formuláře definujeme v souboru *forms.py*, kde nastavíme požadované inputové pole a metody pro kontrolu polí. U pohledu poté stanovíme $form\_class = XYZForm$ a v šabloně tento formulář uvedeme pomocí *{{form}}*.

Clean data

Validování obsahu formuláře probíhá pomocí metody $clean\_data\_FIELD$. Při této kontrole lze zjistit, jestli zadaný login již existuje, jestli se zadaná hesla schodují (ukázka 4.19), kontrola povolených znaků a další. V případě selhání kontroly vracíme uživateli chybnou hlášku.

```
def clean_password_repeat(self):
    password = self.cleaned_data.get('password')
    password_repeat = self.cleaned_data.get('password_repeat')
    if password and password_repeat and password !=
       password_repeat:
        raise forms.ValidationError('Hesla se neschodují')
    return password_repeat
```

Ukázka kódu 4.19: Zjištění, jestli se zadaná hesla schodují

Token CSFR

Pro správnou funkčnost formulářů vyžaduje Django na začátku formuláře uvést značku *{% csrf_token %}*. Jedná se o ochranu proti Cross Site Request Forgery

útoku. CSRF útok je tvořen speciálním odkazem, na který se útočník snaží nalákat své oběti, aby tak pod jejich identitou provedl skrytou akci, kterou by za běžných okolností samotní návštěvníci nikdy neudělali. [22]

4.8 Ostatní

V této sekci jsou popsané funkce projektu, které se zařazením nehodí do žádné jiné sekce či podsekce. Tyto funkce nemají přímý vliv na chod systému, pouze jej vylepšují a zpřehledňují.

4.8.1 Hlášení nevhodného obsahu

Problému nevhodného obsahu se u systémů s uživatelsky generovaných obsahem nevyhneme. Ať už se jedná o obsah nevhodný pro děti a mladistvé, nebo spam různých botů. Funkce hlášení nevhodného obsahu je u takového typu systému povinností. V kuchařce Bchlrr lze nahlásit recept nebo uživatele. Všechny tyto hlášení se poté administrátorovi zobrazí v backendové části aplikace. S příslušným nevhodným obsahem může dále zacházet dle svého uvážení. Ukázka 4.20 představuje třídu jednotlivého nahlášení.

```
1 class RecipeReport(models.Model):
2     author          = models.ForeignKey(Profile)
3     reported_recipe = models.ForeignKey(Recipe)
4     reason          = models.TextField()
5     date_reported   = models.DateTimeField(auto_now_add=True)
```

Ukázka kódu 4.20: Model hlášení receptu

4.8.2 Vyhledávání a řazení

Vyhledávat a řadit lze všechny seznamy receptů na úvodní stránce. Možnosti řazení jsou následující, od nejnovějšího, od nejoblíbenějšího a abecedně. Pohled příslušné stránky tyto filtry přijme v GET paramatrech a následně vrací vyfiltrovaný QuerySet. Ukázka kódu 4.21 byla z důvodu rozsáhlosti zjednodušena. Kompletní metoda *filter_recipe_view* se nachází v *recipes/utils.py*.

```
def filter_recipe_view(view, qs):
    if view.request.GET.get('q') and
       view.request.GET.get('q').split():
        query_list = view.request.GET.get('q').split()
        qs = qs.filter(title__icontains=q) for q in
           query_list)) | (unicode_title__icontains=q) for q
           in query_list)) | (ingredients__icontains=q) for q
           in query_list)) | (content__icontains=q for q in
           query_list))
        )

    if view.request.GET.get('sort'):
        if view.request.GET.get('sort') == 'new':
            return qs.order_by('-date_added')
        elif view.request.GET.get('sort') == 'top':
            return qs.order_by('-likes_count')
        elif view.request.GET.get('sort') == 'abc':
            return qs.order_by('title')

    return qs
```

Ukázka kódu 4.21: Vyhledávání a řazení QuerySetu

4.8.3 Systém zpráv

V projektu kuchařky je implementovaný systém zpráv. Jedná se o reakce na uživatelovi akce. Tzn. zobrazení úspěšné zprávy po úspěšném vytvoření objektu (obrázek 4.2), editaci objektu, mazání. Varovné zprávy při požadavku na nepřístupné pohledy. Tento systém je implementován pomocí mixinů. Použití je stejné jako *LoginRequiredMixin*. Stačí, aby objekt, který má zprávy přidávat, dědil z mixinu *SuccessMessageMixin* a definoval atribut $success\_message = 'uspesna\_zprava'$

4.8.4 Pagination

V případě, že by se kuchařka stala populární, mohl by nastat problém se zobrazováním úvodní stránky. Požadovat tisíce receptů pokaždé, když uživatel navštíví domovskou stránku není vhodné z důvodu doby nahrávání stránek a serverové zátěže. Tento problém je řešen pomocí externí aplikace Django-Pagination, která

Obrázek 4.2: Úspěšná zpráva editace receptu

seznam receptů rozdělí na jednotlivé stránky s určitým počtem receptů. Implementace tohoto balíčku spočívá v obalení listu receptů ve značkách *{% paginate pocet_receptu_na_strance recipe_list %}* a *{% show_pages %}*

4.8.5 Thumbnaily obrázků

Stejně jako nahrávat tisíce receptů je zbytečné stahovat celý obrázek pouze pro zobrazení jeho thumbnailu. Thumbnaily jsou dynamicky generované pomocí externí aplikace Sorl Thumbnails. Pro správnou funkčnost dynamických thumbnailů stačí *<img>* obalit ve značce *{% thumbnail obrazek "rozmery"crop="center"%}* a *{% endthumbnail %}*. Galerie je poté řešena javascriptovým lightboxem Magnific Popup.

4.9 Plány do budoucna

Sám překvapený autor kuchařky má v plánu na projektu pokračovat. Mezi funkce, které by autor rád přidal, ale bohužel už z časových důvodů nestihl, patří systém notifikací, AJAX prvky typu infinity load, lepší práce s nahráváním fotek, ochranu proti botům, dynamičtější práci s tagy a mnohé další. Je tedy možné, že GitHubová verze projektu bude odlišná od té přiložené a live verze kuchařky.

Závěr

Práce se věnuje porovnání nejznámějších open source redakčních systémů a tvorbě vlastního redakčního systému, který se stará o obsah kuchařky se sociálními prvky Bchlrr.

Výstupem teoretické části práce je zasvěcení do problematiky CMS, analýza a porovnání nejznámějších variant. Tento výstup by měl čtenáři posloužit jako pomůcka, která mu usnadní volbu redakčního systému přesně podle jeho technických zkušeností a požadavků na projekt.

Z analýzy a porovnání je jasné, že na trhu redakčních systémů vede WordPress. Tato varianta disponuje ohromnou komunitou, velikým počtem pluginů a nekonečným množstvím dostupných, krásně zpracovaných šablon. Systém je určený provozovateli stránek, který od nich požaduje funkčnost bez další zbytečné konfigurace. Jedná se o kompletní systém, který již v základu poskytuje účinné nástroje. Ovšem ne vždy je takový komplexní systém žádaný.

Systémy Joomla a Drupal nemají tolik základních funkcí, ale není problém je pomocí pluginů doinstalovat. Joomla oproti Drupalu nabízí početnější komunitu, lépe zpracované rozšíření, kvalitnější šablony a menší technické požadavky. Drupal na druhou stranu odmění zkušenějšího správce větší kontrolou nad systémem.

Svoji vlastní cestou si jde Django CMS. Hlavní zaměření systému je uživatelova absolutní kontrola nad vzhledem, chováním a obsahem stránek. Systém nemá příliš velikou komunitu. Co však tato komunita ztrácí na velikosti, to vynahrazuje ochotou a užitečností řešených problémů. Při zprovozňování systému musí správce obětovat svoje úsilí a čas. Odměnou mu ale je jednoduchý, přehledný a funkční systém, který přesně odpovídá představám a požadavkům provozovatele stránek.

Praktická část práce se věnuje představení, popisu základní struktury a vývoji projektu ve frameworku pro tvorbu webových aplikací Django. Na této ukázkové aplikaci jsou znázorněny potřebné vlastnosti systému, který má sloužit ke správě obsahu. Mezi tyto vlastnosti patří práce s obsahem, databází, uživateli a multimédii. Live verze projektu je dostupná na `http://bchlrr.pythonanywhere.com`.

Literatura

[1] Michael Kutý *Bachelor / Master Thesis Latex Seed (FIM UHK Fork)* [online]. [cit. 2015-01-04]. Dostupný z URL: `https://github.com/uhk-fim/thesis-latex-seed`

[2] Deane Barker *Web Content Management* O'Reilly Media, 2015, ISBN: 978-1-49190-812-9, 352 stran

[3] Ivey Brent Laminack *A Brief History of Content Management Systems* [online]. [cit. 2016-11-29]. Dostupný z URL: `http://laminack.com/index.php/technology/11-a-brief-history-of-content-management-systems`

[4] Matt Mullenweg *WordPress Now Available* [online]. [cit. 2003-05-27]. Dostupný z URL: `https://wordpress.org/news/2003/05/wordpress-now-available/`

[5] George Nigel *Beginning Django CMS* Apress, 2016, ISBN: 978-1-484216-70-5, 171 stran

[6] Steven Johnson *History of Joomla* [online]. [cit. 2014-01-13]. Dostupný z URL: `https://www.intownwebdesign.com/joomla/history-of-joomla.html`

[7] Dries Buytaert.*About Drupal History* [online]. Dostupný z URL: `https://www.drupal.org/about/history`

[8] W3 Techs *Usage of content management systems for websites* [online]. [cit. květen 2017]. Dostupný z URL: `https://w3techs.com/technologies/overview/content_management/all`

[9] Plain Black Corporation *Compare Content Management Systems* [online]. Dostupný z URL: `http://www.cmsmatrix.org/`

[10] Aristeides Stathopoulos *Why you should use twig for WordPress* [online]. [cit. 2015-04-12]. Dostupný z URL: https://aristath.github.io/blog/using-twig-in-wordpress

[11] James Bruce *How Do Search Engines Work?* [online]. [cit. 2013-04-17]. Dostupný z URL: http://www.makeuseof.com/tag/how-do-search-engines-work-makeuseof-explains/

[12] gcx11 *Úvod do Pythonu* [online]. [cit. 2016-12-08]. Dostupný z URL: https://goo.gl/TQ3AT5

[13] Elvis Pranskevichus, Yury Selivanov *What's New In Python 3.6* [online]. [cit. 2016-12-23]. Dostupný z URL: https://docs.python.org/3/whatsnew/3.6.html

[14] Adrian Holovaty, Jacob Kaplan-Moss *Introducing Django* [online]. [cit. prosinec 2009]. Dostupný z URL: https://djangobook.com/introducing-django/

[15] Django Software Foundation *Django Documentation* [online]. Dostupný z URL: https://docs.djangoproject.com/en/1.11/

[16] Dharanee Dharan *Django Introduction* [online]. [cit. 2017-04-04]. Dostupný z URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[17] Blackrock Digital LLC *Start Bootstrap - Clean Blog* [online]. [2017-03-06]. Dostupný z URL: https://github.com/BlackrockDigital/startbootstrap-clean-blog

[18] Kayla Ngan *What is Git?* [online]. [cit. 2017-04-04]. Dostupný z URL: https://www.visualstudio.com/learn/what-is-git/

[19] David Watzke *Regulární výrazy* [online]. [cit. 2008-01-23]. Dostupný z URL: http://www.abclinuxu.cz/clanky/programovani/regularni-vyrazy

[20] Vitor Freitas *How to Extend Django User Model* [online]. [cit. 2016-07-22]. Dostupný z URL: https://simpleisbetterthancomplex.com/tutorial/2016/07/22/how-to-extend-django-user-model.html

[21] Refresh Oxford *Classy Class-Based Views* [online]. [cit. 2017-04-29]. Dostupný z URL: `https://ccbv.co.uk/`

[22] Roman Kümmel *Cross Site Request Forgery* [online]. [cit. 2008-05-19]. Dostupný z URL: `https://www.soom.cz/clanky/484--Cross-Site-Request-Forgery`

Přílohy

Datové médium

Struktura praktického projektu

- složka **bchlrr_cookbook**
  - **settings.py** obsahuje nastavení projektu
  - **urls.py** obsahuje URL seznam dostupných požadavků
- složka **media**
  - složka **cache** obsahuje cache thumbnailů
  - složka **profiles** obsahuje uživateli nahrané fotky profilů
  - složka **recipes** obsahuje uživateli nahrané fotky receptů
- složky **profiles** a **recipes** obsahují příslušné soubory aplikace
  - složka **templates** obsahuje šablony a snippety příslušné aplikace
  - **admin.py** obsahuje registrované modely
  - **forms.py** obsahuje formuláře
  - **models.py** obsahuje definované třídy modelu
  - **urls.py** obsahuje rozšiřující URL požadavky příslušné aplikace
  - **utils.py** obsahuje pomocné metody
  - **views.py** obsahuje pohledy
- složka **templates** obsahuje základní šablony, snippety, CSS a JS
- **manage.py** spravuje a spouští projekt

Instalace a spuštění praktického projektu

1. Z adresy `https://www.python.org/` nainstalujeme Python verze 3 nebo novější (balíčkový manažer pip je již v instalaci obsažený).
2. (Volitelný krok) Zprovozníme virtuální prostředí. Viz kapitola 4.1.1.
3. Pomocí příkazu *pip install Django* nainstalujeme Django.
4. Nainstalujeme externí aplikace a knihovny.
   a) *pip install Pillow* - knihovna umožňující Pythonu práci s obrázky
   b) *pip install django-taggit* - aplikace pro správu tagů
   c) *pip install django-el-pagination* - aplikace zajišťující stránkování
   d) *pip install Unidecode* - knihovna pro práci s diakritikou
5. Složku *bchlrr_cookbook* nakopírujeme do požadovaného adresáře. V případě použití virtuálního prostředí nakopírujeme projekt do složky, ve které se nachází složka virtuální prostředí *myvenv*.
6. Ve složce projektu otevřeme terminálové okno.
7. Pomocí příkazu *python manage.py makemigrations* zjistíme změny modelu aplikace.
8. Příkazem *python manage.py migrate* vytvoříme databázi podle modelu aplikace.
9. Nyní je potřeba vytvořit administrátorský účet. To provedeme pomocí příkazu *python manage.py createsuperuser*.
10. Pomocí *python manage.py runserver* lokálně spustíme aplikaci (výchozí adresa je 127.0.0.1:8000).

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Horčičko Michal	Jičínská 101, Valdice	I14077

TÉMA ČESKY:

Porovnání volně dostupných Open Source redakčních systémů

TÉMA ANGLICKY:

Comparison of freely available Open Source Content Management Systems

VEDOUCÍ PRÁCE:

Ing Andrea Vokálová - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce je popis a porovnání volně dostupných redakčních systémů a s nimi spojenými technologiemi, uživatelská přívětivost těchto systémů, dostupnost a kvalita rozšíření (pluginů) a šablon. Výsledkem bude celkový přehled o světě redakčních systémů, klady a zápory jednotlivých variant. Praktická část práce bude zaměřena na tvorbu vlastního (jednoduchého) redakčního systému.

OSNOVA:
Úvod,
1. Definice CMS
2. Hypotézy a použité metody
3. Návrh, vývoj vlastního CMS v Django Python
4. Výsledky
Závěr, zdroje

SEZNAM DOPORUČENÉ LITERATURY:

Nigel George - Beginning Django CMS, Apress, 2016, ISBN 978-1-484216-70-5
Deane Barker - Web Content Management, O'Reilly Media, 2015, ISBN 978-1-49190-812-9

Podpis studenta: Horčičko ..... **Datum:** 12.10.2016

Podpis vedoucího práce: Andrea Vokálová ..... **Datum:** 12.10.2016