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

Bakalářská práce

Redakční systém pro správu webových stránek

Plzeň 2020 Dominik Chlouba

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Dominik CHLOUBA**
Osobní číslo:	**A17B0230P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Redakční systém pro správu webových stránek**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte a srovnejte nejznámější open source redakční systémy.
2. Navrhněte redakční systém, který umožní správu obsahu webové prezentace, přidávání vlastních modulů a editaci šablon.
3. Realizujte navržený redakční systém s použitím technologií HTML, CSS, JavaScript, PHP, MySQL.
4. Realizovaný redakční systém otestujte a zdokumentujte.
5. Ověřte funkčnost systému použitím v praxi.

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

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Doc. Ing. Dalibor Fiala, Ph.D.**
Nové technologie pro informační společnost

Datum zadání bakalářské práce: **7. října 2019**
Termín odevzdání bakalářské práce: **7. května 2020**

Západočeská univerzita v Plzni
L.S.
Fakulta aplikovaných věd

Doc. Dr. Ing. Vlasta Radová
děkanka

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

V Plzni dne 15. října 2019

Prohlášení

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

V Plzni dne 2. května 2020

Dominik Chlouba

Abstract

The thesis deals with the creation of its own content management system for website administration. The first part of the work explains the theoretical background which is related to the issue of content management and blogging web systems. The analytical part describes technologies and architectures of web applications followed by comparison of content management systems based on PHP technology. In the practical part there is an analysis of the architecture and functionality of the basic content management system core. The practical part uses the results of the analysis, on the basis of which it builds a safe and user-friendly content management system based on PHP programming language technologies.

Abstrakt

Bakalářská práce se zabývá tvorbou vlastního redakčního systému pro správu webových stránek. V první části této práce jsou objasněna teoretická východiska, která se vztahují k problematice redakčních a blogovacích webových systémů. Analytická část popisuje technologie a architektury webových aplikací, následované porovnáním redakčních systémů založených na technologii PHP. V praktické části je provedena analýza architektury a funkčnosti základního jádra redakčního systému. Praktická část využívá výsledky provedených analýz, na jejichž základě buduje bezpečný a uživatelsky přívětivý redakční systém založený na platformě programovacího jazyka PHP.

Obsah

1 Úvod

Internet je součástí našich životů již několik desítek let a většina lidí si již bez něj málokdy dokáže smysluplně poradit. S rozšířením této sítě se neodmyslitelně začali rozvíjet také webové technologie, které mají na trhu velké zastoupení. Rozšířenost těchto aplikací je způsobena především velmi snadnou přístupností služby i samotným provozem.

Již dnes můžeme pozorovat, že pravděpodobnost nesetkání se s webovou aplikací je prakticky nulová. Používáme je každý den pro hledání informací a aktualit, komunikaci s přáteli či nakupování. Kromě sociálních sití mají vysoké zastoupení také e-shopy a blogovací nebo publicistické stránky. Informační servery, se kterými máme možnost se setkat, často obsluhují lidé bez znalostí technologií daného systému. Z důvodu usnadnění práce všem zainteresovaným jsou používány tzv. redakční systémy, které pomáhají uživatelům zpříjemnit správu webových prezentací.

Systém pro správu obsahu - CMS je software pro tvorbu, modifikaci a publikování dokumentů či článků zpravidla prostřednictvím webového rozhraní s využitím WYSIWYG editoru. CMS je velmi mocný nástroj, který kromě výše uvedeného umožňuje také řízení a správu dokumentů, souborů i obrázků či uživatelů a mnoho dalšího. S těmito systémy se především setkáme tam, kde se často mění obsah nebo ho přidává více lidí. Z pohledu vývojářů jsou CMS výhodné především díky snadnému vytváření šablon či rozšíření, aniž by museli celou webovou aplikaci psát od nuly. Tyto systémy můžeme najít na mnoha platformách. Nejrozšířenější systémy pochází z platformy jazyka PHP, následované platformou .NET Core a další.

V následujích kapitolách jsou popsány některé základní technologie, architektury, následované porovnáním open-source systémů pro správu obsahu platformy PHP a nástin některých konkurenčních systémů. V neposlední řadě je popsán Nette framework z rodiny jazyka PHP nebo samotný návrh a realizace redakčního systému.

2 Důležité pojmy

2.1 Programovací jazyk PHP

PHP je skriptovací programovací jazyk pracující na straně serveru – uživatel obdrží pouze výsledky činnosti. Vznikal mezi roky 1994 až 1996 [27] a od té doby prošel velkými změnami. Syntaxe jazyka je inspirována několika programovacími jazyky, např. Java nebo C [27]. Programovací jazyk je nezávislý na platformě, tzn. že lze skripty většinou přenášet mezi různými operačními systémy. Výhodou je podpora mnoha externích knihoven pro různé účely (práce s grafikou i soubory a přístup k databázovým systémům – např. MySQL, Oracle, MSSQL a další).

Ačkoliv je PHP velice populární, často čelí kritice z řad profesionálních vývojářů kvůli své „nekonzistentnosti“ a „nepředvídatelnosti“ [26]. Důvod velké využívanosti je prostý – provozní prostředí pro PHP bývá často poskytováno na každém webovém serveru s poměrně jednoduchou obsluhou. Příklad kostry třídy v PHP na obrázku 2.1.

```php
<?php

namespace PHPExample;

class PHPExample
{
    public function __construct(){
        $this->echoHelloWorld();
    }

    private function echoHelloWorld(){
        echo "Hello World!";
    }

}
```

Obrázek 2.1: Ukázka syntaxe programovacího jazyka PHP

2.2 Databáze MySQL

MySQL je multiplatformní systém řízení báze dat [4]. Řadí se mezi tzv. relační databáze, tedy databáze založené na tabulkách. Každá tabulka obsahuje položky jednoho typu. Tabulka se skládá z atributů (sloupce) a může obsahovat značné množství záznamů (řádky). MySQL databáze je typovaná, každý sloupec má tedy pevně stanovený datový typ [4]. Komunikace s databází probíhá pomocí jazyka SQL. Každá databáze používá proprietární dialekt tohoto jazyka s některými rozšířeními.

MySQL databáze jsou optimalizované především na rychlost [4] a nejčastěji se využívají ve spojení s platformou jazyka PHP. Spolu s touto databází se využívá nejčastěji nástroj *phpMyAdmin* [28]. Je to nástroj vytvořený v jazyce PHP, jehož účelem je umožnit jednoduchou správu MySQL databáze pomocí webového rozhraní. Odlehčenou alternativou *phpMyAdmin* je software zvaný *Adminer*.

2.3 Značkovací jazyk HTML5

HTML neboli hypertextový značkovací jazyk je jedním ze základních jazyků pro tvorbu webových aplikací. Spolu s JavaScriptem, CSS a serverovým jazykem tvoří základnu webových prezentací.

Kromě základního popisu grafického rozhraní a struktury webové stránky umožňuje HTML5 základní práci se soubory (nahrávání na server), zobrazení aplikace na celou obrazovku či poskytuje možnost zjištění aktuální pozice uživatele pomocí integrovaného rozhraní geolokace [11] bez nutnosti použití knihoven třetích stran, čímž je zajištěna i větší bezpečnost. Nová verze značkovacího jazyka ovšem nabízí i jiné funkce, které usnadňují využití plného potenciálu webových technologií. Mezi tyto funkce také patří možnost nativního přehrávání multimediálního obsahu [11] přímo na webu, práce s vektorovou grafikou, plátno (atribut Canvas) s JavaScriptovým rozhraním či podpora tzv. „offline režimu“ [11]. Offline režim umožňuje funkčnost webové aplikace i v případě, kdy nejsme připojeni k internetu (ukládání dat k uživateli, po připojení k síti se odešlou na server). Příklad zápisu struktury HTML stránky na obrázku 2.2.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HTML Example</title>
</head>
<body>
    <header>
        Header
        <nav>Navigation</nav>
    </header>
    <main>Content</main>
    <footer>Footer</footer>
</body>
</html>
```

Obrázek 2.2: Ukázka syntaxe značkovacího jazyka HTML

2.4 Skriptovací jazyk JavaScript

JavaScript je jazyk na straně klienta (součást prohlížeče webových stránek) a je nedílnou součástí všech webových aplikací. JS je multiplatformní a v současné době již objektově orientovaný skriptovací jazyk [14], který je řízen událostmi [13]. Syntaxe jazyka je inspirovaná rodinou jazyků C/C++ a Java [13], přesto je JavaScript zcela odlišný.

Přestože byl JavaScript původně určen pro klientskou stranu, využívá se také na serverech, např. NodeJS [14]. Proměnné fungují na principu dynamického přiřazování datového typu a objekty se chovají jako asociativní pole [13].

Ačkoliv klasický JavaScript nedisponuje konceptem objektů v podobě tříd, který je typický pro objektově orientované jazyky, tento mechanismus nahrazuje tzv. prototypování. Je tedy schopen simmulovat mnoho vlastností třídově založených mechanismů, např. dědičnost [13]. Funkce se nechovají pouze jako metody, ale také kontruktory objektů [13]. Příklad funkce v JavaScriptu a její volání na obrázku 2.3.

Pro zpříjemnění práce s JavaScriptem se nabízí několik možností. Jedna z nich je použití kompilátoru Babel. Babel je svobodný open source JavaScrip-

```
const alertHelloWorld = () => {
    alert("Hello World!");
};

alertHelloWorld();
```

Obrázek 2.3: Ukázka syntaxe skriptovacího jazyka JavaScript

tový kompilátor a patří mezi jazyky, které současný JS pouze rozšiřují [1], přičemž se snaží zachovat s původním JavaScriptem kompatibilitu. Babel se poměrně rychle stal populární. Hlavním důvodem této popularity je především široká podpora ES6 a ES7 [1], zároveň plně integruje JSX a React [1]. Běžně ho používají firmy jako Facebook, Mozilla nebo Netflix [2].

2.5 Kaskádové styly

Kaskádové styly, nebo-li CSS, je jazyk pro popis stylu grafického zobrazení elementů webové stránky. Nejčastěji se CSS používá ve spojení se značkovacími jazyky HTML či XML. CSS bylo postupně vydáváno ve třech verzích společně s aktuálně dostupným CSS3. Hlavním úmyslem kaskádových stylů je oddělení struktury dokumentu od jeho grafického zobrazení.

Popis pomocí kaskádových stylů se řídí několika pravidly. Syntaxe pravidla stylu se skládá ze *selektoru* následovaný *blokem deklarací* [9]. Každá deklarace je pak oddělena znakem středníku, přičemž každá deklarace obsahuje identifikátor vlastnosti a její hodnotu, obojí oddělené dvojtečkou. Jednou z vlastností kaskádových stylů je dědičnost. Dědičnost umožňuje přenos vlastností rodičů na své potomky [9]. Máme-li element s černou barvou pozadí, pak i všichni jeho potomci budou mít stejné pozadí, pokud se nevyskytne pravidlo u některého z potomků, které tuto skutečnost pozmění. Používání kaskádových stylů s sebou přináší mnoho výhod. Mezi tyto výhody patří především rozsáhlé možnosti formátování textu, tabulek či obecně elementů, zjednodušení veškerých grafických modifikací webové aplikace nebo tzv. cachování stylů.

Nevýhoda využívání CSS stylů spočívá především v nedostatečné podpoře některých vlastností či funkcí kaskádových stylů v různých prohlížečích s vysokým procentuálním zastoupením na trhu. Tyto chyby v jednotlivých implementacích CSS mohou způsobit, že se stejné styly mohou v některých

prohlížečích zobrazit zcela odlišně, případně se CSS u daných elementů vůbec neprojeví. Ačkoliv jsou CSS selektory velmi mocné, neumožňují přistupovat k rodičovským elementům z potomka [9]. Příklad použití kaskádových stylů na obrázku 2.4.

```
body {
    padding: 0;
    margin: 0;
}

header.fixed {
    position: fixed;
}

p:nth-child(2n + 1) {
    color: #252525;
}
```

Obrázek 2.4: Ukázka zápisu kaskádových stylů

Pro zpříjemnění práce s kaskádovými styly se nabízí několik možností. Mezi tyto možnosti patří použití tzv. preprocesorů či postprocesorů. Preprocesory CSS jsou jazyky, postavené nad samotným CSS. Jelikož kaskadové styly jako takové nedovolují používání různých funkcí, proměnných, vnořených definic či matematických operací, které by ulehčovali práci s navrhováním stylů, jsou preprocesory nejlepší možností [9]. Nejznámějšími preprocesory jsou LESS, SASS a Stylus. Každý z těchto nástrojů má svoji vlastní syntaxi, přičemž jedno mají společné, dokáží z vlastního kódu vygenerovat výsledný „obyčejný“ kaskádový styl. Při generování klasického CSS se preprocesory zároveň snaží o co největší kompatibilitu se všechny majoritními prohlížeči [9].

Jelikož preprocesory umožňují psát poměrně abstraktní kód, který dovoluje vytvořit i hůře čitelné a těžko spravovatelné CSS kódy, vznikla myšlenka tzv. postprocesorů [31]. Postprocesory se snaží zachovat výhody preprocesorů a zároveň odstranit jejich nevýhody. Podobně jako preprocesory umožňují postprocesory využívaní postupů, které klasické kaskádové styly neumí. Hlavní rozdílem je standardizovaný zápis, umožňující zpřehlednění výsledného kódu. Mezi zástupce postprocesorů patří PostCSS [31].

2.6 OOP

Objektově orientované programování je způsob programování, které data a funkce navzájem svazuje do tzv. objektů. Jednotlivé objekty jsou výsledkem vyjádření tříd v reálném světě (nebo se o to alespoň pokoušejí) [23]. V OOP lze vytvářet více objektů stejné třídy, mluvíme o tzv. instancích. Definice třídy se skládá z klíčového slova *class* následovaného názvem třídy, který se dále používá jako identifikátor objektu, a dvěma složenými závorkami. V OOP je zvykem pojmenovávat dle stanovených konvencí, proto se názvy tříd řídí takzvanou velbloudí notací [24] – každé slovo v názvu třídy obsahuje velké počáteční písmeno, ostatní jsou malá.

Výslednou třídu tvoří atributy a metody. Atributy lze zjednodušeně i trochu nepřesně chápat jako klasické proměnné a metody jako funkce. Kouknemeli se blíže na objektově orientované programování v jazyce PHP, můžeme si všimnout, že některé metody třídy s určitým významem mají zvláštní název. Takové metody začínají předponou ___ (dvě podtržítka). Nejvíce využívanou metodou tohoto typu je konstruktor. Konstruktor je metoda, která se vykoná právě jednou a to při vytváření každé instance daného objektu. Cílem této specifické metody je provedení prvotního nastavení třídy. Pojmenovávání metod a atributů se obdobně jako třídy řídí velbloudí notací, ovšem první písmeno je vždy malé [24].

Třemi základními pilíři objektově orientovaného programování jsou dědičnost, zapouzdření a polymorfismus [23], které jsou podrobněji popsány v následujících odstavcích.

2.6.1 Dědičnost

Dědičnost je v objektově orientovaném programování způsob stanovení vztahu mezi objekty. U tříd lze pomocí této vlastnosti převzít atributy a metody (chování) od již existujících tříd – předků [5]. Výsledné třídy pak označujeme jako potomky.

Jednotlivé zděděné metody mohou být často překrývány (nahrazeny) nebo přetíženy [5]. Přetížení metody znamená změnu vstupních parametrů metody, přičemž název, účel a návratová hodnota metody zůstává zachována, viz Polymorfismus. Lze se setkat s třídami, které dědit nelze.

2.6.2 Zapouzdření

V OOP lze zapouzdření chápat jako ukrytí atributů a metod do jednoho objektu (třídy), které chrání data proti úmyslným i neúmyslným změnám [39]. Pro zapouzdření určitých atributů, metod či tříd se využívají zpravidla tři základní klíčová slova, která určují pravidla přístupu (*public*, *protected* a *private*) [39].

2.6.3 Polymorfismus

Polymorfismus je vlastnost objektově orientovaného jazyka umožňující objektům volání jedné metody stejně pojmenované, ale jinak implementované [30]. V polymorfismu se využívá takzvaného přetěžování, které lze aplikovat jak na metody, tak i na samotné operátory (+, -, atp.) [30].

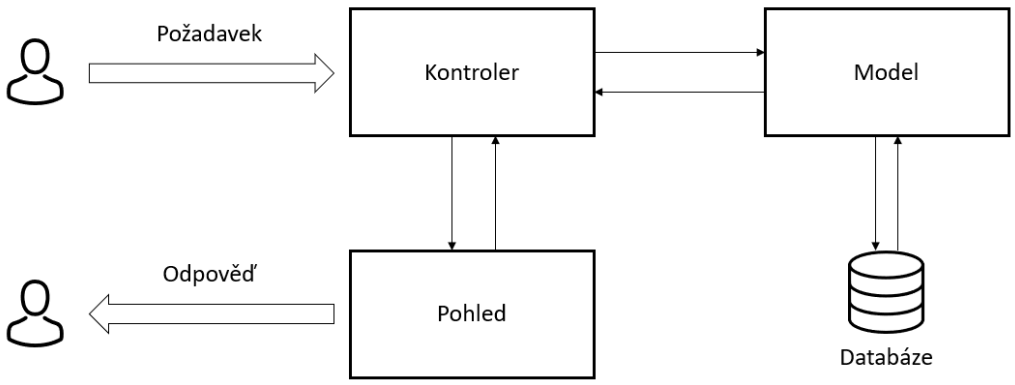
2.7 Aplikační architektury

2.7.1 MVC architektura

MVC je poměrně oblíbený architektonický vzor, který je používám zejména u webových aplikaci, ačkoliv k tomu původně určený nebyl. Model-View-Controller je využívám například u frameworků Zend pro PHP, Ruby On Rails pro Ruby či MVC framework pro ASP.NET Core [18].

Základním cílem MVC modelu je oddělení logiky od výstupu, což by mělo řešit problematiku tzv. „špagetového“ kódu. Pojmem „špagety“ se rozumí soubor, příp. třída, která obstarává zároveň logické operace a renderování výstupu.

Celá architektura je rozdělena na tři základní komponenty, viz obrázek 2.5. Model obsahuje logiku celé aplikace. Provádí se zde tedy výpočty, databázové dotazy [18] a podobně. Model vůbec neví o výstupu, pouze přijme parametry, které má zpracovat a výsledek své operace předá zpět objektu, který ho volal. Druhou komponentou je pohled (View), který se stará o zobrazení výstupu [18] uživateli v předem stanoveném formátu. Velmi často se jedná o dokument, který je součástí šablonovacího systému (např. Latte, viz dále kapitola Nette Framework). Poslední, velice důležitou součástí, je Controller. Tento prvek funguje jako prostředník mezi pohledem a modely [18]. S kontrolerem komunikuje uživatel, view i samotný model, čímž je systém kompletně propojen do jediného funkčního celku.



Obrázek 2.5: Příklad diagramu třívrstvé MVC architektury

Zdroj: Překresleno z Researchgate.net - MVC architecture

Po zadání URL adresy do webového prohlížeče zpracuje jako první požadavek tzv. router, který podle parametrů identifikuje controller, kterému předá řízení zpracování požadavku uživatele.

2.7.2 MVP architektura

MVP architektura funguje podobně jako výše zmíněný vzor MVC. Často jsou však tyto architektury chápány a implementovány různými způsoby. MVP architekturu vyzdvihuje např. český PHP framework Nette. V MVP často existuje propojení z pohledu (View) do presenteru [36], případně pohled sám přebírá kontrolu nad aplikací a vytváří si presenter [36].

2.7.3 MVVM architektura

Model-View-ViewModel je architektura, kde Model představuje a obstarává veškerou logiku a data aplikace [19]. Vrstva pohledu je pak uživatelské rozhraní, které s modelem propojuje tzv. ViewModel [19]. ViewModel, umožňující obousměrný databinding [19], připravuje data z modelu pro zobrazení v uživatelském rozhraní a naopak reaguje na změny v UI, které jsou následně propisovány do modelu.

MVVM architektura není příliš stavěna pro provoz webových služeb. Setkávame se s ní především u aplikací WPF na platformě .NET [19]. Nicméně jakožto jeden z hlavním zástupců třívrstvé architektury aplikací stojí za zmínku.

2.8 Provoz a licence

2.8.1 Open source systém

Otevřený software či svobodný software je software s tzv. otevřeným zdrojovým kódem [25]. Otevřeností se rozumí jak technickou dostupnost kódu a dokumentace, tak legální dostupnost (licence), viz kapitola níže. O Open Source vzrostl zájem především v oblasti webových technologií [25].

Z hlediska bezpečnosti software umožňuje otevřenost kódu výrazně rychlejší nalezení případných chyb v programu, díky širší skupině zainteresovaných lidí. Ačkoliv by se to mohlo zdát jako patřičná výhoda, je třeba si uvědomit také větší zranitelnost v případě vnějších útoků.

Alternativou open source je tzv. closed source [25], také nazývaný proprietární software. Především kvůli výhodám, které open source poskytuje, se s proprietárními systémy setkáváme stále méně.

2.8.2 GNU GPL licencování

GNU General Public License je licence založená na konceptu svobodného softwaru [8]. Licence zajišťuje nejenom dostupnost zdrojového kódu programů, ale také modifikovatelnost kýmkoliv, kdo o to projeví zájem. Další součástí je možnost šíření daného softwaru či jeho použití v jiných programech, které jsou taktéž šířeny pod touto licencí [8].

2.8.3 Webový server

Pojmem webový server se rozumí počítač, který přebírá zodpovědnost za odbavování HTTP požadavků jednotlivých uživatelů nebo ho lze interpretovat také jako program, který provádí výše popsanou činnost [37].

Pokud si chceme prohlédnout určitou webovou stránku, je povinností klienta (např. prohlížeče) si ji od serveru vyžádat. Jelikož klient i server jsou programy, komunikace probíhá pomocí určitých pravidel, které se nazývají *protokoly* [37]. Pro přenos webových stránek se využívá protokol HTTP, případně jeho zabezpečená verze HTTPS. Jedním z nejpoužívanějších webových serverů je software s otevřených kódem pro GNU/Linux, Apple macOS, Microsoft Windows i další platformy, tzv. Apache.

3 Redakční systémy

Pojmem redakční či publikační systém se rozumí počítačový software, který je využíván pro správu webového obsahu. Mezi základní funkce systému patří především tvorba, modifikace a publikace obsahu stránek (článků) [3], jež je prováděna zpravidla pomocí grafického webového rozhraní. Veškeré operace s obsahem jsou prováděny prostřednictvím tzv. WYSIWYG editoru [3] či jiného systému pro formátování textu, u kterého není nutná znalost HTML. Další předností systému je řízení a správa přístupových práv uživatelů nebo jednotlivých stránek, správa diskuzí, komentářů, různých souborů a další.

Výhod při využívání CMS je hned několik. Bezespornou výhodou je především možnost spravovat a vytvářet internetové stránky bez rozsáhlých znalostí webových technologií. Teoreticky lze pomocí těchto systémů vytvořit stránky od jednoduchého prezentačního webu až po menší internetový obchod [3]. V oblasti CMS softwarů má uživatel možnost výběru z mnoha platforem i redakčních systémů. Setkáme se jak s open source systémy, tak i komerčními proprietárními systémy [3] pro správu obsahu.

Následující odstavce blíže popisují a porovnávají vybrané redakční systémy založené na technologiích programovacího jazyka PHP.

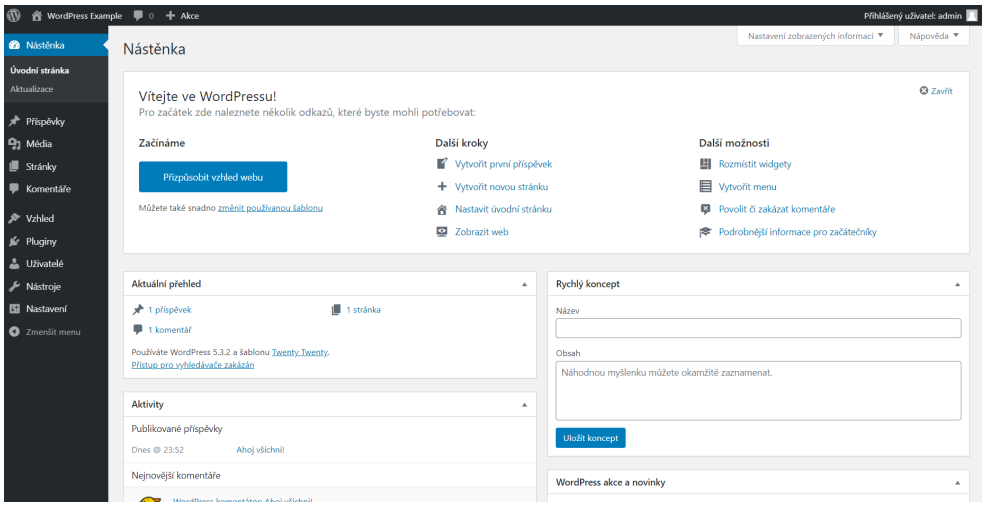
3.1 Systémy založené na PHP

3.1.1 WordPress

WordPress je svobodný open source redakční systém distribuovaný a vyvíjený pod licencí GNU GPL [38]. Software je napsaný v programovacím jazyce PHP, přičemž využívá služby databázového systému MySQL. Wordpress je oficiálním nástupcem redakčního systému *b2/cafelog* [38], který měl nahradit. Odhaduje se, že tento redakční systém pohání přibližně 33% [38] všech webových stránek na celém Internetu, od malých blogů až po největší zpravodajské online servery. Kromě této majority mezi webovými systémy se může pyšnit také širokou uživatelskou a vývojářskou komunitou. Tato skutečnost je nejspíše jeden z hlavních důvodů jeho pokračující oblíbenosti.

Výhodou systému je přizpůsobitelný vzhled, který je možné upravovat či vytvářet i s minimálními znalostmi HTML nebo PHP [10]. WordPress mimo jiné také nabízí mnoho výchozích šablon s responzivním designem, ze kterých si každý dokáže vybrat. Díky tomu je možné systém pouze nainstalovat a s jeho využíváním začít během několika málo minut. Nutno však zmínit, že po této instalaci není WordPress nakonfigurovaný zcela optimálně, což může také negativně ovlivňovat rychlost webových stránek. Nicméně pro obyčejného uživatele je to dostačující. Mezi další užitečné funkce systému patří například správa médií [10], nicméně výchozí prostředí pro správu dokumentů může být po delší době využívání a nahrávání souborů matoucí a nepřehledné. Systém WordPress je snadno ovladatelný a přístupný pro různé skupiny uživatelů. Jelikož jedním z cílů webových stranek je také zvýšení návštěvnosti a zároveň získání specifických návštěvníků, umožňuje systém poměrně intuitivní SEO optimalizaci. [10]

Důležitou úlohou, který redakční systém musí pokrýt je zabezpečení. WordPress je tvůrci aktualizován poměrně často [38], přičemž aktualizace obsahují jak opravy bezpečnostních chyb, tak zahrnují i nové funkce. Novinky či opravy se distribuují za pomoci systému automatických aktualizací. Mírnou nevýhodou a zranitelností v zabezpečení může být mimo jiné popularita systému, jelikož se zvyšuje riziko jeho napadení. Ačkoliv WordPress poskytuje velké množství rozšíření třetích stran, není vždy zajištěna stoprocentní kompatibilita. Někdy lze narazit i na rozšíření, které vyžaduje dodatečný zásah do zdrojového kódu. Grafická ukázka uživatelského rozhraní redakčního systému na obrázku 3.1.



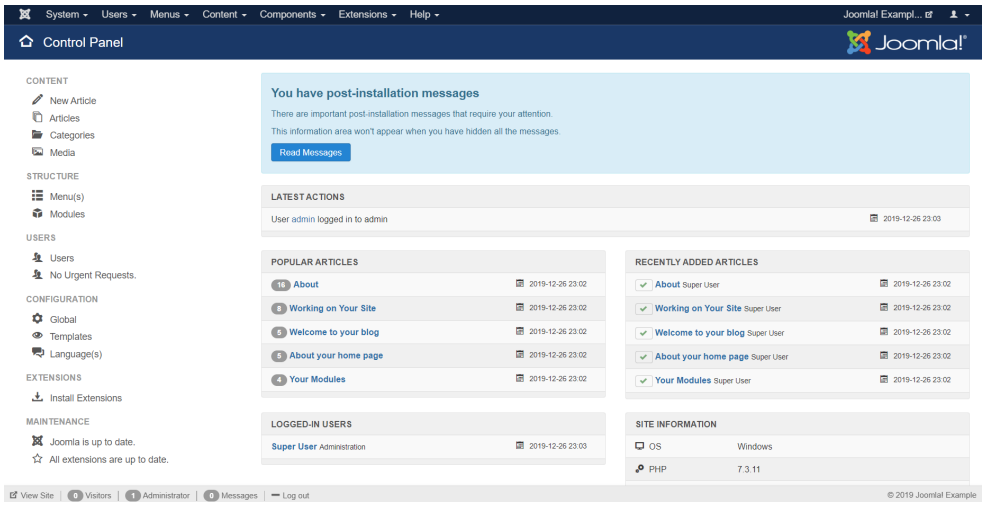
Obrázek 3.1: Snímek obrazovky redakčního systému WordPress

3.1.2 Joomla!

Joomla! je bezplatný open source systém umožňující vytváření a publikování online obsahu. Software je vyvíjený v programovacím jazyce PHP a k ukládání dat využívá databázový systém MySQL. Publikační systém je distribuován pod licencí GNU GPL [15]. Ačkoliv není Joomla! rozšířený natolik jako WordPress, jeho oblíbenost mezi uživateli i vývojáři nelze popřít.

Ačkoliv to nemusí být zřejmé, je tento redakční systém vhodný zejména pro rozsáhlejší projekty [15]. Jelikož je zde značná podpora ze strany externích vývojářů, je možné systém rozšířit pomocí různých pluginů či modulů. Na první pohled je však CMS poměrně složité až neintuitivní, zvláště pro úplného začátečníka a je tedy důležité věnovat více času na seznamení se s prostředím [15]. Podobně jako WordPress, také Joomla! nabízí základní responzivní šablony, ze kterých lze čerpat. Důležitou součástí je správa médií, přičemž soubory lze hierarchicky rozřazovat již ve výchozí konfiguraci.

Samotná instalace je snadná, pokud však chce uživatel systém zlepšovat, neobejde se bez použití rozšíření nebo modulů [15]. Základní konfigurace je poměrně důkladně očesána i o některé užitečné funkce a využívání pluginů je tak spíše nutnou záležitostí, přestože většina použitelných rozšíření je placená [15]. Grafická ukázka uživatelského rozhraní redakčního systému na obrázku 3.2.



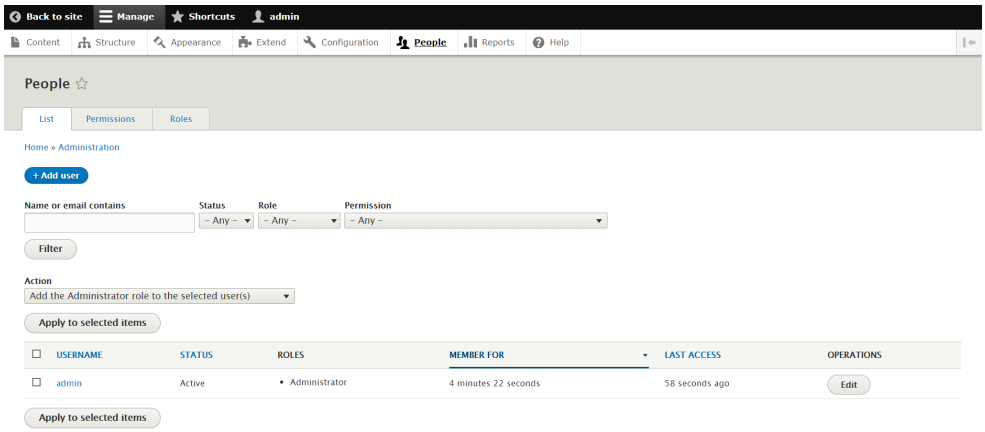
Obrázek 3.2: Snímek obrazovky redakčního systému Joomla!

3.1.3 Drupal

Drupal je redakční systém vhodný pro časopisy, blogy, internetové obchody i jiné komplexní systémy [7]. Je vyvíjen v programovacím jazyce PHP a podporuje databázové systémy MySQL, PostgreSQL a SQLite [7]. Po přidání příslušných modulů je možná podpora také databází MSSQL, Oracle či MongoDB [7]. Systém je distribuován pod licencí GNU GPL. [7]

Redakční systém se může pyšnit otevřeností rozsáhlého API [7], modulárností [7] i přehledností zdrojového kódu. Součástí jsou předpřipravené balíčky s předem definovanou funkcionalitou. CMS je postaveno na frameworku Symphony a využívá šablonovací systém Twig. Ačkoliv zde není podpora od externích vývojářů vysoká jako u Joomla! nebo WordPressu, využívá systém statisíce uživatelů. Lze rovněž vyzdvihnout jednoduché, občas méně intuitivní, ovládání systému, vysoký výkon i zabezpečení.

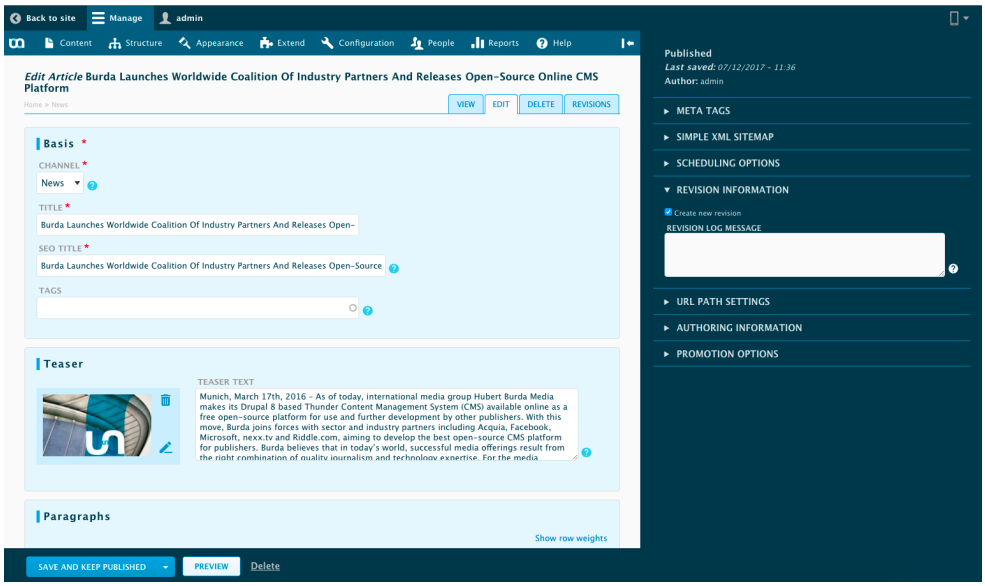
Instalace systému není složitá, ze začátku je však nutné věnovat více času výběru jednotlivých rozšiřujících modulů, jelikož je CMS v klasické konfiguraci velmi ošizen o některé základní funkce. Systém lze využívat jak pro komerční, tak i vzdělávací účely. Pro běžného uživatele bez znalostí HTML, CSS, JavaScriptu a programovacího jazyka PHP však může být příliš složitý [7]. Grafická ukázka uživatelského rozhraní redakčního systému na obrázku 3.3.



Obrázek 3.3: Snímek obrazovky redakčního systému Drupal

3.1.4 Thunder

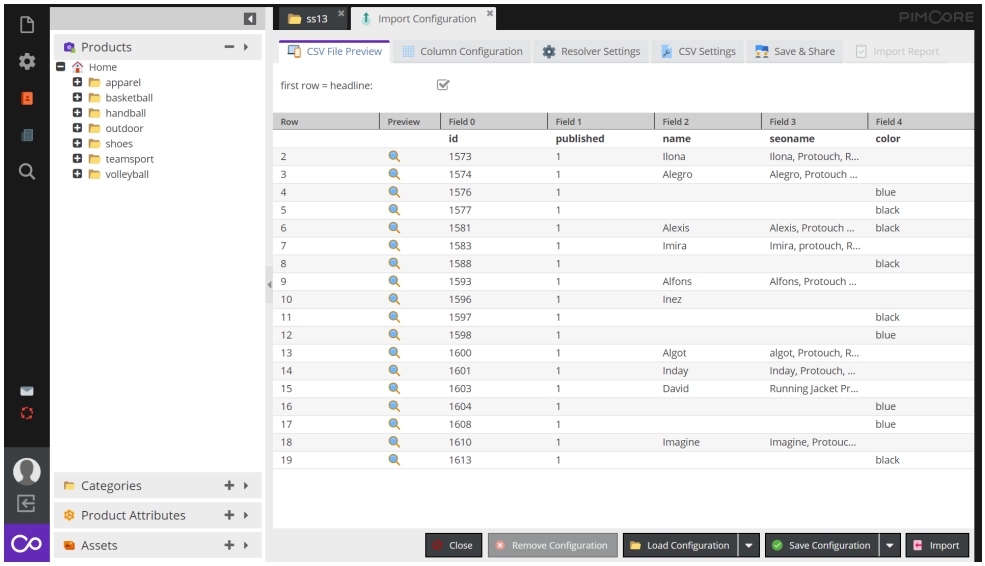
Thunder je bezplatný open source publikační systém určený zejména pro profesionální použití [34]. Je založen na systému Drupal verze 8 [34]. Jeho vývoj tedy taktéž probíhá na platformě jazyka PHP a podporuje databázový systém MySQL. Redakční systém je šířený pod licencí GNU GPL. CMS je poměrně flexibilní. Přejímá veškeré výhody i nevýhody systému Drupal, viz předchozí kapitola. Grafická ukázka uživatelského rozhraní redakčního systému na obrázku 3.4.



Obrázek 3.4: Snímek obrazovky redakčního systému Thunder

3.1.5 Pimcore

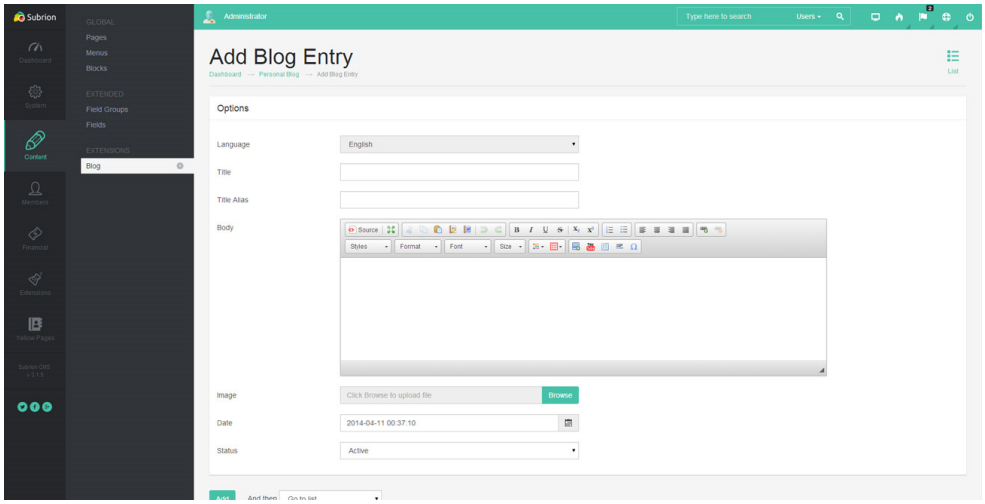
Pimcore je bezplatný open source systém pro správu obsahu doporučený pro enterprise použití, distribuovaný pod licencí GNU GPL [29]. Je vyvíjen v jazyce PHP a podporuje databáze MariaDB a MySQL [29]. Systém je založen na frameworku Symphony [29]. Pimcore lze využít jak pro data management, tak i experience management [29] (komerční využití, CMS). Komunikaci s jádrem systému obstarává RESTful API [29], což je patrně jediný způsob propojení grafického rozhraní se zbytkem systému. Grafická ukázka uživatelského rozhraní redakčního systému na obrázku 3.5.



Obrázek 3.5: Snímek obrazovky redakčního systému Pimcore

3.1.6 Subrion

Subrion je bezplatný open source systém založený na PHP5, podporující databázi MySQL. Je šířený pod licencí GNU GPL a je možné jej využít k vytváření obyčejných webových stránek až po korporátní portály [33]. Pomocí framework API lze systém rozšířit o další užitečné funkce. CMS využívá všech výhod Bootstrap 3 a šablonovacího systému Smarty. Výhodou systému je podpora vícejazyčného webu či rozsáhlý seznam různých druhů responzivních šablon nebo pluginů [32]. Administrační rozhraní nabízí přehledné rozvržení jednotlivých informací a umožňuje tak efektivnější a jednodušší práci s webovou aplikací. Grafická ukázka uživatelského rozhraní redakčního systému na obrázku 3.6.



Obrázek 3.6: Snímek obrazovky redakčního systému Subrion

3.2 Konkurenkční platformy

Systémy pro správu obsahu však nejsou záležitostí pouze platformy PHP. Vlastní systémy má celá řada webových platforem jako ASP.NET Core, NodeJS a další. Za velice úspěšné systémy na .NET platformě lze označit například Kentico CMS [16] určený především pro komerční či korporátní využití, nopCommerce [22] umožňující tvorbu jednoduchých i komplexních eshopů a další.

4 Nette Framework

Nette Framework je open source framework platformy PHP zaměřený na PHP verze 5.x a dnes již také 7.x [21]. Hlavní motivací vzniku frameworku byla eliminace bezpečnostních rizik, znovupoužitelnost kódu a MVC, resp. MVP architektura [21]. Framework je založen na komponentách a je řízen především událostmi systému [21], přičemž využívá principu OOP – objektově orientovaného programování. Nette bylo vytvořeno českým vývojářářem Davidem Grudlem [21] a je nabízen pod licencí GNU GPL. Systém se stal oblíbený především v České republice a je využívám i významnými společnostmi jako GE Money, Slevomat či ESET [21].

Začít s Nette Frameworkem je velice snadné, vývojáři předpřipravili tzv. sandbox. Sandbox je kostra webové aplikace Nette, která má zpřehlednit kód výsledné aplikace pomocí doporučené adresářové struktury. Struktura není samozřejmě závazná, je tedy možné si ji upravit dle svých potřeb. Jednou z předností Nette je podpora modulů, které umožňují členění kódu do více rozvětvených logických celků.

Součástí Nette Frameworku je tzv. DI kontejner [6]. Základní podstatou vkládání závislostí (Dependency Injection – DI) je poskytování objektů, které třídy potřebují ke své funkčnosti, aniž by se o to samy musely starat. Tím se třídám odebere veškerá zodpovědnost za získávání objektů [6].

4.1 MVP a router

Jak již bylo zmíněno na začátku kapitoly, je PHP framework založený na architektuře MVP. Aplikace se tedy skládá z tří základních komponent – presentery, modely a pohledy, viz kapitola Aplikační architektury.

Než samotný požadavek dorazí k presenteru, musí nejdříve projít tzv. směrovačem [12]. Směrovač přijme URL adresu podle které pozná, kterému presenteru má předat obsluhu pro zpracování požadavku klienta.

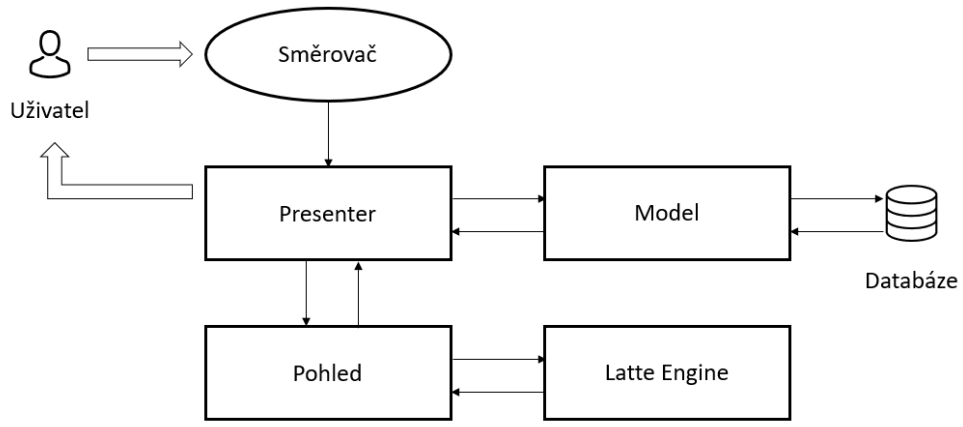
4.2 Životní cyklus

Aby mohl Nette Framework vykonat nějaké operace, je nutné na daný server odeslat požadavek z klientské aplikace. Nejdříve se požadavek dostane ke směrovači, který daný požadavek zpracuje, zavolá příslušný presenter a předá mu zbylé parametry [12].

Presenter dle parametrů zjistí akci, jež má vykonat a kontaktuje konkrétní model, kterému sdělí, co se od něj očekává. Jednotlivé akce, např. zobrazení výstupu či modifikace dat, jsou v presenteru reprezentované metodami.

Model je vrstva, která přímo či nepřímo manipuluje se samotnými informacemi [12]. Většina požadavků na model očekává přístup k databázi s daty, které se mohou i nemusí dále v metodě upravovat. Posledním krokem metody modelu je předání získaných dat presenteru, kterým byl vyvolán.

Poté, co presenter obdrží informace od modelu, jsou data předána pohledu (šabloně). Šablona je popsána pomocí HTML a obvykle obsahuje dodatečné značky šablonovací systému Latte, které data zobrazují. O vykreslení těchto dat se stará automaticky tzv. Latte Engine [12]. Po dokončení sestavování stránky je výsledný soubor odeslán zpět tázající se aplikaci, která webovou aplikaci zobrazí uživateli. Celý životní cyklus Nette Frameworku je znázorněn na obrázku 4.1.



Obrázek 4.1: Životní cyklus Nette Frameworku

Zdroj: Překresleno z ITnetwork.cz - Životní cyklus Nette

4.3 Šablonovací systém Latte

Latte je výchozí šablonovací systém Nette Frameworku, který se snaží zjednodušit a zpříjemnit práci s výsledným grafickým rozhraním webové aplikace a zároveň chrání výstup před základními útoky jako je XSS [17].

Předností systému je možnost rozdělení výsledné stránky na sadu menších komponent, ze kterých se pak stránka poskládá [17]. Důvodem rozdělení je především možnost využití dané komponenty na více místech aplikace, což také umožňuje držení se principu DRY (Don't repeat yourself) – tedy neopakování zdrojového kódu na více místech aplikace. Způsobů, jak šablony vytvářet, je několik a je pouze na programátorovi, kterou se bude řídit. Při volbě je důležité myslet především na budoucí udržitelnost kódu a rozšiřitelnost webových stránek.

4.4 Ladicí nástroj Tracy

Knihovna Tracy, nebo-li Laděnka, je nástroj pro ladění PHP aplikací. Pomáhá vizualizovat a logovat případné chyby či překlepy a snaží se navrhnout opravu [35]. Ladicí knihovnu je možné rozšířit pomocí dalších kompatibilních knihoven nebo si vytvořit své vlastní rozšíření [35].

4.5 NEON

NEON označuje formát textového souboru. Syntaxe NEONu je odvozena od formátu JSON [20], avšak jsou zde odstraněny veškeré uvozovky, závorky a čárky na konci řádků oddělující jednotlivé atributy a další, viz obrázek 4.2. Soubory formátu NEON se v Nette Frameworku používají jako konfigurační soubory, např. pro konfiguraci DI kontejneru či databázového přístupu.

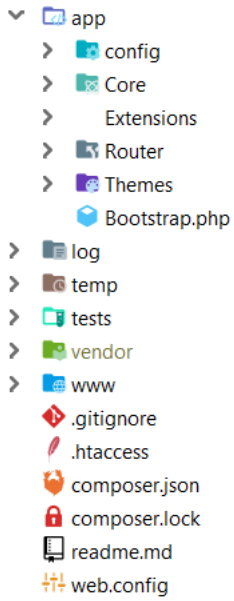
```
services:
    createRouter: Anker\RouterFactory
    - Anker\CoreClass
```

Obrázek 4.2: Ukázka syntaxe textového souboru NEON

5 Návrh redakčního systému

Následující kapitoly se zaměřují již na samotný návrh a konstrukci redakčního systému, který během vývoje obdržel symbolický název *Anker CMS*. S tímto označením se pojí také konstrukce názvů některých důležitých konfiguračních souborů či struktur. Návrh redakčního systému založeného na platformě PHP počítá s plným využitím všech funkcí a možností Nette Frameworku. Framework bude přizpůsoben pro účely systémy, zajistí veškeré zabezpečí, business logiku a manipulaci s datovou vrstvou. Datová vrstva bude pracovat na principech ORM, tedy objektově relačního mapování, uživatelé i vývojáři rozšíření pro redakční systém tak budou úplně odstíněni od databáze i samotného SQL jazyka. Vlastní projekt redakčního systému bude volně dostupný s otevřených kódem dle licence GNU GPL v3 na online GIT repozitáři.

5.1 Adresářová struktura



Obrázek 5.1: Adresářová struktura navrhovaného redakčního systému

Adresářová struktura redakčního systému vychází z tzv. Sandbox projektu Nette Frameworku, viz obrázek 5.1. Kořenový adresář projektu obsahuje 6 základních složek, jejichž účel bude rozebrán níže. Soubory *.htaccess* či

web.config obsahují veškeré konfigurační parametry určené pro správný a optimální chod webového serveru. *composer.json* uchovává informace o závislostech projektu na komponentách třetích stran. Komponenty většinou zahrnují balíčky potřebné pro samotný Nette Framework či ORM, případně pro jiné součásti, které dodatečně využívá redakční systém.

5.1.1 Adresář „app“

Adresář *app* obsahuje všechny komponenty redakčního systému. Skládá se především z konfiguračních souborů pro systém obsahující informace pro připojení k databázi, konfigurace potřebných nástrojů systému či závislosti tříd na službách.

Jádro celého redakčního systému je ukryto ve složce *Core*. Adresář obsahuje základní objekty pro logické řízení systému, bázové objekty stránek, databázových entit či aplikačních rozšíření.

Aplikační rozšíření budou uložena ve složce *Extensions*. Rošíření bude možno spravovat ručně, případně použít nástroj systému pro automatickou správu. Více o rozšířeních v kapitole *Integrace rozšíření*.

Posledním hlavním adresářem je *Themes*, který se dále dělí na podadresáře *Backend* a *Frontend*. Složka *Backend* obsahuje šablony pro administrační část systému. Toto rozhraní pro správce bude možné napojit na API redakčního systému a lze ho tak přizpůsobit pro své potřeby. Zpočátku bude obsahovat výchozí šablonu systémy, kterou lze kdykoliv deaktivovat a přidat tak šablonu novou. Obdobně adresář *Frontend* obsahuje šablony pro veřejnou část webu. Stejně jako u administrace je možné vytváření vlastních šablon, které lze napojit pomocí API. Aktivace či deaktivace šablon bude možné řídit v administračním rozhraní systému.

5.1.2 Adresář „log“

V adresáři *log* jsou spravovány veškeré logovací soubory jak Nette Frameworku, tak i samotného redakčního systému.

5.1.3 Adresář „temp“

Temp je využívám samotným Nette Frameworkem pro uchovávání *cache* souborů, který na produkční instanci redakční systémů dokáží urychlit vyřizování požadavků uživatelů.

5.1.4 Adresář „tests“

Tests uchovává možné PHP Unit testy, které je možné v případě potřeby spouštět či vytvářet. Testy slouží především pro ověřování správné funkčnosti systému.

5.1.5 Adresář „vendor“

Vendor je adresář, který využívá PHP balíčkovací nástroj zvaný *Composer*. Composer obstarává a udržuje závislosti projektu na externích komponentách. Při využívání GIT nástroje se nedoporučuje tento adresář nahrávat do vzdáleného repozitáře. Veškeré závislosti jsou uloženy v dříve zmíněném souboru *composer.json* či *composer.lock*. Potřebné balíčky je tedy možné získat i při pozdějším použití.

5.1.6 Adresář „www“

Adresář „www“ je tzv. „startovací“ bod celé aplikace. Jsou zde uchovávány veškeré veřejné soubory (obrázky, dokumenty, ...) a další. Důležitou součástí složky je soubor *index.php*, který přijímá požadavky uživatelů a předává je komponentám obstarávajícím zpracování a vyhodnocení požadavků.

5.2 Design

Designový návrh všech výchozích šablon redakčního systému se bude řídit alespoň minimálními požadavky na UI a UX. Šablony budou responzivního charakteru, přičemž minimální podporovaná šířka zařízení je stanovena na *350 pixelů*.

Pro vytváření šablon je možné využívat zdroje z doplňkových nástrojů, například NPM či Bower. Ačkoliv výchozí šablony budou z větší části sestaveny z vlastních komponent, je možné využívat také Bootstrap knihovnu.

5.3 Integrace rozšíření

Rozšíření redakčního systému budou uchovávána v adresáři *Extensions* a pro jejich správnou integraci je nutné dodržet určitou základní kostru souborů. Každé rozšíření musí být zabaleno v tzv. kořenovém adresáři, který je uložen ve složce pro rozšíření. V kořenovém adresáři musí být umístěny soubory *config.neon* a *info.neon*. Nejsou-li soubory k dispozici, je rozšíření ignorováno.

```
anker:
    name: My Extension 1
    identifier: my.unique.identifier
    class: MyExtension1\MyExtension1
    description: Nové rozšíření pro redakční systém Anker
```

Obrázek 5.2: Požadovaná struktura konfiguračního souboru rozšíření

Info.neon obsahuje informace o rozšíření, přičemž struktura zápisu je předem stanovena, viz obr. 5.2. *Config.neon* je přímo napojen na hlavní konfigurační soubor systému. Soubor je určen především pro registraci závislostí tříd na jiných objektech, což umožní vývojářům využívat možností DI Injection.

Každé rozšíření musí být umístěno v jedinečném jmenném prostoru, aby je bylo možné od sebe jednoduše rozeznat. Hlavní třída musí dědit od předka s názvem *ExtensionBase*.

6 Programátorská dokumentace

6.1 Princip

Princip celého redakčního systému je založen na jednoduchosti, rozšířitelnosti a přizpůsobitelnosti. Jednoduchost spočívá především v použití Nette Frameworku verze 2.4, které nabízí mnoho funkcí, jenž vývojáři usnadňují práci a může se tak věnovat řešení vlastních funkcionalit, aniž by se musel starat o ty úplně nejzákladnější problémy. Těmito problémy se především myslí funkcionalita rozpoznávání a směrování požadavků uživatele. Dále pak dostupnost užitečných programovacích struktur, které nejsou dostupné a jádře programovacího jazyka PHP a je tak nutné si je vytvořit svépomocí či využít již hotové knihovny třetích stran, například pomocí balíčkovacího systému Composer.

Vyvinutý redakční systém se pak zaměřuje především na rozšířitelnost. Ačkoliv jádro systému již obsahuje velké množství logiky, které je určeno k administrativě či tvorbě vlastních webových stránek či příspěvků, každý uživatel očekává od svého webového portálu mírně odlišné vlastnosti. Součastí systému je možnost tvorby vlastních rozšíření, pomocí kterých se tato očekávání mohou naplnit a eliminovat tak případné subjektivní nedostatky systému. Jednoduchá integrace vlastních rozšíření umožňuje například vytváření a registraci vlastních presenterů a šablon v rámci rozšíření či tvorbu vlastních „widgetů“, které se na jednotlivé stránky či do jednotlivých příspěvků vkládají pomocí speciálních značek, jenž principiálně fungují stejným způsobem jako *ShortCodes* v redakčním systému WordPress.

Posledním důležitých faktorem je přizpůsobitelnost. Tato vlastnost redakčního systému je především definována pomocí témat, nebo-li šablon webových stránek. Jelikož jádro systému samo o sobě obsahuje pouze rozhraní pro manipulaci s daty, základní datové struktury či užitečné logické a grafické komponenty, jsou vlastnosti a chování stránky, jak veřejné, tak i administrační části, zcela definovány šablonami. Přestože v rámci tématu je možné vytvářet struktury, které se chovají jako rozšíření, není tento způsob, především kvůli přehlednosti, doporučován. Chování jednotlivých stránek či příspěvků jsou pak určena implementací jednotlivých presenterů či šablon.

6.2 Struktura

Projekt se skladá z několika základních adresářových struktur, které obsahují veřejně dostupné soubory, dočasné soubory (cache), skripty pro účely vývojářů, jádro redakčního systému, katalogy překladů, konfigurační soubory, soubory s informacemi pro migraci databázového systému a mnoho dalších důležitých struktur či souborů. Jednotlivé součásti projektu jsou podrobněji rozebrány v následujících kapitolách.

6.2.1 Konfigurační soubory

Součástí frameworku i samotného redakčního systému jsou různé druhy konfiguračních souborů, jenž plní důležitou úlohu v samotné funkčnosti aplikace. Tyto konfigurační soubory lze rozdělit do tří kategorií: jádro, témata a rozšíření.

Jádro redakčního systému obsahuje celkem pět různých souborů typu *NEON*. *config.neon* a *config.local.neon* jsou určeny pro konfiguraci Nette Framework rozšíření, Dependency Injection kontejnerů a k uchovávání informací o přihlašovacích údajích k databázovému serveru aplikace. Posledními konfigurační soubory jsou *anker.themes.neon* (viz obr. 6.1), *anker.extensions.neon* (viz obr. 6.2) a *anker.config.neon*. První ze souborů obsahuje informace o aktuálně aktivním tématech ve veřejné a privátní části webu. Druhý uchovává identifikátory jednotlivých aktivních rozšíření pro redakční systém. Poslední ze souborů pak umožňuje registraci presenterů jádra systému, především pak těch, které poskytují RESTful API vstup do aplikace.

```
anker:
    themes:
        frontend:
            - Sjakkspill

        backend:
            - Kapteinbrua
```

Obrázek 6.1: Struktura konfiguračního souboru aktivních témat

Velmi důležité konfigurační soubory jsou součástí jednotlivých témat. Každé téma má předem definovanou strukturu, viz dále v následujících kapitolách, jenž obsahuje tři základní konfigurační soubory: *services.config.neon*, *anker.config.neon* a *info.neon*. První soubor, podobně jako *config.neon*, je určen

především k registraci dodatečných služeb do DI kontejneru aplikace, čímž je vývojář zcela odstíněn od obstarávání jednotlivých služeb, které mohou témata využívat. Dalším důležitým souborem je *anker.config.neon*, který je určen k registraci presenterů tématu, inicializační třídy, případně je možné ho využít k vlastním účelům, nicméně je nutné nezasahovat do struktur definovaných jádrem systému. Posledním souborem je *info.neon*, který není povinnou součástí tématu, přesto je doporučené jeho využívání. Tento soubor obsahuje detailní informace o účelu a možnostech tématu, které je možné následně zobrazit uživateli, například v rámci administračního rozhraní.

```
active:
    - Extension1
    - Extension2
```

Obrázek 6.2: Struktura konfiguračního souboru aktivních rozšíření

Posledním typem jsou konfigurační soubory, které jsou součástí jednotlivých rozšíření systému. Podobně jako konfigurační soubory tématu, jsou určeny především k registraci služeb do DI kontejneru či k inicializaci a připojení rozšíření do jádra systému. Těmito soubory jsou *config.neon* a *info.neon*.

6.2.2 Jádro systému

Srdcem celého systému je jádro, které obstarává veškerou logiku systému, jenž umožňuje chod a provoz služeb, které plní funkce reálné aplikace pro správu webového obsahu. Samotné jádro se skládá ze tří vrstev, které se snaží od sebe co nejvíce oddělit business logiku a funkce, databázový systém a grafické uživatelské rozhraní.

Prezentační vrstva aplikace se stará o zobrazování informací pro uživatele, většinou pomocí grafického uživatelského rozhraní, může kontrolovat zadávané vstupy, nicméně neobsahuje zpracování dat. Do této vrstvy spadá část jádra zajišťující inicializaci a operace s presentery systému. Jádro rozlišuje dva druhy presenterů. Prvním druhem jsou tzv. „handler“ presentery, které přijímají požadavky přímo z Nette frameworku, a dle kterých se rozpoznává onen druhý druh presenterů. Mezi handler presentery patří *PagePresenter*, který zpracovává požadavky pro veřejnou část webové stránky, *AdminPresenter*, jehož účelem je rozpoznávání a operace nad požadavky administrační části, a *ApiPresenter*, který nepatří mezi presentery s grafickým výstupem, nýbrž poskytuje rozhraní pro vytváření a zpracovávání RESTful API uživatelských požadavků. Handler presentery sami o sobě nejsou

přímo využívány uživatelskými tématy, ale poskytují základní logiku pro směrování příslušných požadavku již do presenterů druhé skupiny. Touto druhou a zároveň poslední skupinou presenterů jsou tzv. „base“ presentery. Base presentery poskytují přístup jednotlivým presenterům témat k obalové struktuře redakčního systému, díky které mohou témata využívat či upravovat různá data či nastavení v rámci vytvořené instance. Tyto presentery jsou celkem tři: *PublicPresenter* – poskytuje služby presenteru veřejné části webové aplikace, *PrivatePresenter* – obdobně poskytuje služby pro administrační část a *ApiPresenter* – služby pro RESTful API požadavky. Mezi posledními prezentačními částmi je dobré zmínit službu pro renderování uživatelských menu či například Vindu filtr v rámci *Latte engine*, který umožňuje vyrenderování šablony widgetu a její umístění do šablony stránky.

Další vrstvou je business logika, ve které, jak již název napovídá, leží veškerá logika a funkce, výpočty a zpracování dat. Do této vrstvy řadíme především části, které se starají o registraci, inicializaci a validaci rozšíření v rámci služeb *ExtensionManager* a *ExtensionChecker*, logiku spravující registrovaná témata a jejich presentery či fasády a managery obstarávající zpracování požadavků na výpočetní, datové či souborové zdroje.

Poslední vrstvou je vrstva datová, která není svým rozsahem rozlehlá, nicméně hraje klíčovou roli v celé aplikaci. Datová vrstva je tvořena databází, která uchovává, zpřístupňuje a zaručuje konzistenci dat. Součástí vrstvy jsou všechny entity služeb, které jádro poskytuje. Důležitou součástí jsou repozitáře, jenž poskytují rozhraní fasádám a managerům pro přístup k těmto datům. Nejdůležitější částí infrastruktury je však třída *AppDbContext*, která obaluje služby *Entity manageru* databázového rozšíření *Doctrine*.

Celé jádro redakčního systému je obaleno v jediné datové struktuře, pomocí které je umožněn přístup k datům jádra v rámci jednotlivých presenterů témat. Celé jádro je také dostupné skrze Dependency Injection kontejner. Zmíněnou obalovací vrstvou je třída *AnkerWrapper*.

6.2.3 Rozšíření

Součástí aplikační struktury jsou také rozšíření redakčního systému. Každé rozšíření má předem jasně definovanou strukturu, kterou je nutné dodržet, jinak by daná funkce nebyla úspěšně integrována do jádra systému. To by mělo za následek nemožnost využívat možnosti daného rozšíření.

Jádro systému rozlišuje každé rozšíření pomocí jedinečného identifikátoru, který je přidělen samotnými vývojáři. Je tedy nutné se přidávání či vytváření nového rozšíření do systému ujistit, že je daný identifikátor volný. Tento jedinečný řetězec je součástí konfiguračního souboru *info.neon*, viz obr. 6.3. Mimo jiné obsahuje také název, popis či jiné informace o rozšíření a umožňuje registraci dodatečných presenterů či možnost vytvářet strukturu administračního menu, které bude jádrem přidáno mezi ostatní položky seznamu. Poslední důležitou součástí tohoto souboru je definice inicializační třídy, která je volána během startovacího procesu redakčního systému.

```
anker:
    name: My Extension 1
    identifier: my.unique.identifier
    class: MyExtension1\MyExtension1
    description: Nové rozšíření pro redakční systém Anker
    author: Dominik Chlouba
    version: 1.0.0
    image: https://lh3.googleusercontent.com/proxy/q9-yy_jAhjQI4JhZ2tB64
    presenters:
        - MyExtension1\Presenters\Presenter1
    admin:
        menu:
            tohomepage:
                link: /
```

Obrázek 6.3: Struktura informačního souboru rozšíření (*info.neon*)

Nutno dodat, že každé rozšíření je v samostatném adresáři, který je pojmenován stejně, jako inicializační třída. Tato třída může být také součástí podadresáře v hlavním adresáři rozšíření, nicméně není tento způsob umísťování doporučen. Hlavní adresář by měl také obsahovat všechny konfigurační soubory s definovanou strukturou, která vede k úspěšnému propojení s jádrem systému. Přes všechna tato základní omezení struktury rozšíření, není vývojař nijak omezován ve vytváření podadresářů či jiných programovacích struktur.

6.2.4 Témata

Důležitou součástí aplikace jsou uživatelská témata. Témata, podobně jako rozšíření, mají předem danou strukturu, kterou je nutné dodržet. Témata jsou rozdělena do dvou kategorií: veřejná a soukromá (administrační). Jak bude vysvětleno v dalších kapitolách, jsou témata přímo odpovědná za chování webové stránky. Je tedy možné mít zcela odlišná chování, tedy témata, pro dvě různé uživatelské zkušenosti s aplikací.

```
anker:
    initializer: Themes\Backend\Kapteinbrua\Initializer
    presenters:
        - Themes\Backend\Kapteinbrua\Presenters\LoginPresenter
        - Themes\Backend\Kapteinbrua\Presenters\API\ImagesPresenter
    admin:
        menu:
            dashboard:
                link: /admin/dashboard
            media:
                link: ""
                permissions:
                    resource: anker_media
                    privilege: all
                items:
                    overview:
                        link: '/admin/media?page=1&limit=20'
                        permissions:
                            resource: anker_media
                            privilege: overview
                    upload:
                        link: /admin/media/upload
                        permissions:
                            resource: anker_media
                            privilege: upload
```

Obrázek 6.4: Struktura konfiguračního souboru tématu (*anker.config.neon*)

Jednotlivá témata se skládají především z konfiguračních souborů, presenterů a Latte šablon. V rámci tématu existují tři druhy konfiguračních souborů, jejichž úkolem je integrace nových funkcí tématu do jádra systému či konfigurace DI kontejneru. Prvním z těchto souborů je *anker.config.neon* (viz obr. 6.4), jehož účelem je registrace presenterů do jádra redakčního systému a v případě administračního tématu je umožněna konfigurace obsahu hlavního menu poskytovaného samotným jádrem. Druhý souborem je

services.config.neon, jehož jediným účelem je integrace nových služeb do Nette frameworku, převážně pak registrace nových služeb pro Dependency Injection kontejner. Posledním souborem je již známý *info.neon* obsahující data informativního charakteru, především pak název, autora či popis.

Všechny soubory každého tématu jsou součástí samostatných adresářů. Hlavní adresář každého tématu by měl obsahovat již zmiňované konfigurační soubory. Soubor *info.neon* (viz obr. 6.5) není povinný, přesto je doporučovaný zejména z důvodu přehlednosti v administračním rozhraní. Dalším pravidlem pro správný chod tématu je dodržení struktury pro presentery a šablony. Jednotlivé presentery musí být součástí nějakého podadresáře v tématu, například Presenters, adresář se šablonami pak musí být umístěn ve stejném adresáři jako složka s presentery. Tato podmínka musí být bezpodmínečně splněna, jinak nedojde k integraci tématu s jádrem systému. Součástí tématu může být inicializační třída, kterou je nutné registrovat v *anker.config.neon*. Tato třída je určena především k možnosti vlastního nastavení či úpravě dat v rámci redakčního systému před samotným zahájením akcí spojených s vyřizováním uživatelských požadavků. Přes všechna zmíněná strukturální omezení je výsledná hierarchie jednotlivých témat zcela v rukou vývojářů.

```
anker:
    theme:
        name: Kapteinbrua
        description: Hlavní administrační téma
        version: 1.0.0
        author: Dominik Chlouba
        image: "https://images.squarespace-cdn.com/content/57e0c8c07
```

Obrázek 6.5: Struktura informačního souboru tématu (*info.neon*)

6.2.5 Katalogy překladů

Součástí jádra systému je překladová vrstva, která zajišťuje možnosti pro konfiguraci vícejazyčné webové aplikace. Každý jazyk je jednoznačně identifikovatelný kódem jazyka složený z kombinace zkratek pro jazyk a stát dle norem ISO 639-1 a ISO 3166-1 (například pro český jazyk cs_CZ).

Každý jazyk se skládá z tzv. katalogů. Katalog je soubor s identifikátorem překladu, který se skládá z klíčových slov oddělenými tečkami (například *an-*

ker.text.zdroj), a samotným textem (viz obr. 6.6). Veškerá data katalogu jsou uložena ve formátu NEON. Všechny katalogy jsou součástí podadresáře *lang*, uloženém v hlavní složce aplikace. Kromě dodržení formátu jednotlivých katalogů, je nutné dodržen i jeho název. Každý katalog obsahuje klíčové slovo, následované tečkou s identifikátorem jazyka (například *anker.cs_CZ*). Uvedené klíčové slovo je pak vždy použito pro identifikaci katalogu, ze kterého bude získán překlad pro danou část šablony.

```
media.library.title: Knihovna médií
media.library.actions.upload: Nahrát
media.library.empty: Knihovna neobsahuje žádná média
media.library.upload.title: Nahrávání médií
media.library.upload.actions.library: Přejít do knihovny
media.library.upload.files: Vyberte soubory
media.library.upload.submit: Nahrát
media.library.detail.actions.open: Otevřít
media.library.detail.actions.download: Stáhnout
```

Obrázek 6.6: Struktura katalogu jazykové vrstvy

6.2.6 Ostatní

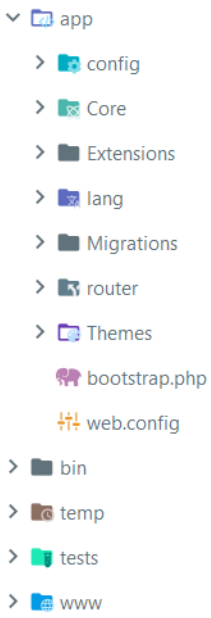
Hlavní části redakčního systému a jejich struktura byla vylíčena v předchozích kapitolách, přesto se aplikace skládá z dalších součástí, které jsou specifické pro Nette framework. Framework i samotný redakční systém využívá možnosti migrací databázových entit. Tyto migrace jsou umístěny ve složce *Migrations*.

Poslední část hlavního adresáře aplikace je tzv. router. Router obsahuje informace o formátu a směrování jednotlivých HTTP požadavků. Tato funkce je přímo součástí Nette frameworku a není doporučeno s touto částí jakkoliv manipulovat.

V poslední řadě je třeba zmínit adresář *www*, který obsahuje veškeré soubory spojené s veřejně dostupnými daty. Složka je využívaná například pro uchovávání JavaScriptových, CSS i jiných souborů. Samotné jádro systému využívá tuto složku, přesněji její podadresář *uploads*, pro uchovávání souborů nahraných uživateli.

Všechny ostatní adresáře v rámci projektu jsou důležité pro optimalizaci či samotný chod aplikace. Nette framework nabízí možnost cache paměti pro

optimalizaci a zrychlení vyřizování požadavků uživatele. Veškeré generované dočasné soubory jsou součástí adresáře *temp*. Poslední adresář, který stojí za zmínku, je *vendor*. Tento adresář slouží k uchovávání balíčků z Composeru. Složka je automaticky generovaná při každém spuštění jakékoliv akce spojené s Composerem, tedy například aktualizace či instalace nového balíčku. Z tohoto důvodu se nedoporučuje jakkoliv manipulovat se soubory obsaženými v tomto adresáři. Důležitá část výsledné struktury projektu je znázorněna na obrázku 6.7



Obrázek 6.7: Výsledná struktura projektu redakčního systému Anker

6.3 Základní funkce

Jádro redakční systému s sebou přináší mnoho užiteční funkcionalit, které pomáhají uživatelům zpříjemnit správu webového i multimediálního obsahu a využít tak plného potenciálu aplikace. V následujících kapitolách budou popsány jednotlivé funkce jádra, včetně možnosti vytváření rozšíření či témat a jejich integrace se systémem.

6.3.1 Správa mediálních souborů

Správa mediálních souborů je základem každého redakčního systému. Jádro aplikace nabízí základní možnosti této správy. Veškeré soubory nahrané

uživateli jsou uloženy v adresáři *uploads*. Tato služba umožňuje základní operace, mezi které patří nahrávání jednoho či více souborů, zobrazení detailu souboru nebo odstranení média z aplikačního uložiště. Přestože jsou veškeré soubory fyzicky ukládány do veřejné složky aplikace, systém si vytváří vlastní záznamy o existenci jednotlivých nahraných souborů do databázové tabulky *anker_media*. Tabulka je v rámci jádra reprezentována entitou *Media* a k ní přidružené entitě *MediaMeta*, obsahující případné dodatečné informace o daném souboru. Pro získávání dat z databáze se využívá tzv. repozitář. V tomto případě se jedná o třídu *MediaRepository*. Prostředkem, pro zpracovávání dat týkající se medií, je *MediaFacade*. Hlavní službu zajišťující sběr veškerých dat však poskytuje třída *MediaManager*, která obstarává veškeré operace spojené například s nahráváním či odstraňováním souborů z uložiště.

Z důvodu zajištění konzistence dat, nejsou soubory shromažďovány přímo v adresáři *uploads*, ale jsou dále rozřazovány do podadresářů, kde první ze složek nese část číslovky daného roku nahrávání (pro rok 2020 je to číslovka 20), podsložky obsahující číslo měsíce (pro duben číslo 04) a v poslední řadě také podadresáře identifikující den v měsíci. Během nahrávání souboru není nijak zásadně měněn název souboru, pouze se na konec řetězce přidá aktuální datum a čas ve formátu *<celý rok>-<číslo měsíce>-<den>-<hodina>-<minuta>-<sekunda>*, aby nedošlo k případné kolizi v pojmenování souborů.

6.3.2 Vlastní typy příspěvků

Důležitou součástí jádra systému, které zároveň umožňuje jednoduché rozšíření bez nutnosti výrazného programování, je možnost vytváření vlastních typů příspěvků. Příspěvky jsou základním kamenem při tvorbě a správě obsahu webové aplikace. Ve výchozích nastavení systému řadíme mezi příspěvky i typ *stránka*. Je nutné zdůraznit, že nehledě na strukturu daného typu, lze vytvořit jakékoliv příspěvky s jakýmkoliv obsahem. Veškeré typy příspěvků jsou reprezentovány entitou *PostType*, která uchovává informace důležité pro identifikaci typu i jeho samotnou strukturu. Identifikátor každého typu musí být zcela jedinečný, aby nedocházelo k nekonzistenci dat. Struktura příspěvku je uchovávána v tabulce objektem ve formátu JSON, který je reprezentován polem s jednotlivými položkami obsahující identifikátor továrny políčka, registrované v jádře redakčního systému, a zkratku či identifikátor dané položky v rámci typu.

Jak již bylo zmíněno, veškerá pole jednotlivých typů příspěvků jsou vázána na továrnu reprezentující vlastnosti a schopnosti políčka. Některé tyto továrny jsou již součástí jádra systému, přesto mnohdy uživatel požaduje specifické možnosti od pole. Z tohoto důvodu jádro poskytuje možnost vytváření a registrace vlastních továren. Uživatel tak má prostředek k tvorbě polí různých typů, například pro číselná data, odkazy či interaktivní mediální widget a další. Tato vlastnost jádra zlehčuje rozšíření systému nejen obyčejným uživatelům, ale také samotným vývojářům, neboť není pro vytváření vlastních typů příspěvků nutný velký zásah do kódu aplikace. Jediný takový zásah spočívá ve vytvoření vlastní továrny či definování šablony pro daný typ příspěvku v rámci jednotlivých témat.

Všechny továrny, poskytované jádrem či registrované do systému, zastřešuje třída *PostFactory*, která se zároveň stará o generování formuláře pro jednotlivé typy příspěvků dle předem definované struktury. Generátor typu je zcela kompatibilní s překladovou vrstvou, což zároveň poskytuje volnost uživatelům ve vytváření textů vhodných k jednotlivých polím. Důležitou funkcí třídy je také možnost samotné registrace továrny, která se provádí pomocí metody *addInput*. Samotná metoda provede registraci pouze v rámci generátoru typů. Důležitý krok při registraci továrny je zaregistrovat rozšíření jako metodu do kontejneru generátoru formuláře pomocí statické metody *extensionMethod* třídy *Container*. Formulář jádra systému rozšiřuje výchozí balíček poskytovaný Nette frameworkem, proto je způsob práce s ním zcela totožný v obou systémech. Přes tuto podobnost se však doporučuje využívat třídu *Form* v rámci balíčku *Anker*.

Možnost vytváření typů příspěvků by ovšem byla zbytečná, pokud by nebylo možné vytvářet samotné příspěvky. Jednotlivé příspěvky jsou reprezentovány entitou *Post* a přidruženou entitou *PostMeta* obsahující dodatečné informace k příspěvkům. V databázovém systému je entita příspěvku definována tabulkou *anker_posts*, pro metadata pak *anker_postmeta*. Každý příspěvek uchovává mimo jiné informace o uživateli, který ho vytvořil, typu příspěvku, jedinečném identifikátoru a především samotný obsah příspěvku. Obsah je v tabulce uložen ve formátu JSON jako objekt s položkami identifikovatelnými pomocí zkratky pole typu daného příspěvku a jeho příslušnou hodnotou. Pro manipulaci s daty typů příspěvků i jednotlivými příspěvky se využívá třída *PostRepository*, služby pro správu těchto dat pak poskytuje fasáda *PostFacade*. Mezi poskytované operace patří například přidávání příspěvků a typů, jejich případná editace a mazání.

6.3.3 Vytváření menu

Jelikož se obsah webových stránek často mění a upravuje, dochází ke změnám struktury aplikace jako takové. Součástí každé webové prezentace je možnost navigace mezi stránkami, nebo-li menu. Se zmíněnými změna dochází také ke změnám odkazů menu a tak je nutné zajistit co nejpohodlnější správu i této části webu. Jádro systému tyto služby poskytuje a umožňuje tak tvorbu různých navigací s hierarchickou strukturou až do hloubky s číslem pět. Jednotlivá menu jsou v rámci jádra reprezentována entitou *Menu*, v databázovém systému jsou pak data uložena v tabulce *anker_menu*. Entita obsahuje zcela základní informace poskytující jednoznačný a jedinečný identifikátor menu a popis samotné struktury menu. Struktura je v tabulce uložena ve formátu JSON.

Pro přístup k datům uloženým v databázi se využívá repozitář *MenuRepository*. Třída *MenuFacade* pak slouží ke zpracovánání dat pro repozitář. Služby pro generování struktury jednotlivých menu poskytuje třída *MenuManager*. Součástí jádra je také *Vindu* widget, který umožňuje jednoduché vygenerování menu v rámci šablony či příspěvku.

6.3.4 Správa uživatelů

Dalším aspektem důležitým pro redakční systémy je správa uživatelů. Přestože z větší části jsou stránky spravované administrátory nebo redaktory, můžeme se setkat i se systémy, které nabízí omezené možnosti správy či konfigurace systému i externím registrovaným uživatelům.

V jádře systému je uživatel reprezentován entitou *User*, která společně s entitou *UserMeta* uchovává veškerá data uživatele v aplikaci. V databázovém systému jsou veškeré instance entit uživatele uloženy v tabulce *anker_users*, jejich metadata pak v *anker_usermeta*. Pro manipulaci s daty o uživatelích a jejich metadatech využívá jádro systému repozitář *UserRepository*. Služby s operacemi nad uživateli pak nabízí fasáda *UserFacade* umístěná v business vrstvě aplikace. Mezi tyto základní operace patří například vytváření uživatele, autentizace, editace a mazání.

V základu systém uchovává základní osobní údaje o uživateli, mezi které patří přezdívka či identifikátor, heslo, celé jméno, e-mailová adresa či uživatelskou roli. Uživatelská role není součástí tabulky s uživateli, ale s metadaty. Uživatelské role budou podrobněji rozepsány v následující kapitole.

6.3.5 Uživatelské role

Jak již bylo předestřeno v předchozích kapitolách, důležitou součástí systému jsou nejen příspěvky, ale také uživatelé. Velmi často nechceme, aby každý registrovaný uživatel měl stejné možnosti manipulace v administrační části webové aplikace. Uživatele tak můžeme zcela libovolně rozřazovat do kategorií dle jejich zaměření. Často se setkáváme například s rolemi *administrátor*, *redaktor* nebo *návštěvník*. Každá z těchto rolí pak nabízí různé možnosti oprávnění, které dovolují uživateli spravovat jednotlivé příspěvky, multimediální soubory či nastavení webové aplikace.

Jednotlivé role jsou v systému reprezentovány entitami třídy *Role*, uloženými v databázové tabulce *anker_roles*. Každá role uchovává dvě základní informace, tj. identifikátor role a jednotlivá oprávnění, která byla roli přiřazena. Oprávnění jsou v tabulce uložena ve formátu JSON ve formě pole obsahující identifikátory oprávnění. Pro manipulaci s daty jednotlivých rolí využívá jádro systému repozitář *RoleRepository*. Služby s operacemi nad rolemi pak nabízí fasáda *RoleFacade*. Mezi tyto základní operace patří například vytváření rolí, jejich editace a mazání.

6.3.6 Opravnění rolí

Poslední funkcionalitou spojenou s rolemi jsou oprávnění. Jednotlivá oprávnění mohou být sice vytvářena v rámci administračního rozhraní, přesto kontrola samotných povolených akcí se provádí v rámci presenteru tématu či jádra systému. Je tedy zcela v kompetenci těchto tříd, zda nově vytvořená oprávnění budou respektovat či nikoliv.

Jednotlivé role jsou v systému reprezentovány entitami třídy *Permission*, uloženými v databázové tabulce *anker_permissions*. Každé oprávnění se skládá ze tří základních informace: identifikátor oprávnění, zdroje a privilegia. Zdroj je klíčové slovo, které slouží k identifikaci oblasti oprávnění (například operace nad multimediálními soubory). Privilegium pak určuje konkrétní povolenou operaci nad touto oblastí (například vytváření). Pro manipulaci s daty jednotlivých rolí využívá jádro systému repozitář *PermissionRepository*. Služby s operacemi nad oprávněními pak nabízí fasáda *PermissionFacade*. Mezi tyto základní operace patří například vytváření oprávnění, jejich editace a mazání.

6.3.7 Audit uživatelů

Uživatel v rámci své role může provádět jakkékoliv operace, které mu byly přiděleny během registrace či samotným administrátorem. Velmi často potřebuje mít administrátor webové aplikace přehled o jednotlivých provedených akcích uživatelů s nižšími oprávněními. Přestože je tato funkce přímo součástí jádra systém, je zcela na uvážení jednotlivých témat, zda tuto funkcionalita využijí.

Jednotlivé záznamy auditu jsou reprezentovány entitami *Audit* a *AuditMeta*. Každé instance těchto entit jsou uloženy v databázovém systému v rámci tabulek *anker_audit* a *anker_auditmeta*.

6.3.8 Tvorba rozšíření

Ačkoliv jádro redakčního systému poskytuje mnoho užitečných funkcí týkající se tvorby a správy webových stránek a příspěvků, oprávnění a uživatelů či správy multimediálních souborů, je nutné umožnit uživatelům či vývojářům vytvářet vlastní rozšíření. Možnosti jednotlivých rozšíření nejsou přímo limitovány, nabízí se celá škála způsobů, které mohou zcela ovlivnit chování i možnosti daného systému. Jak již bylo zmíněno v předchozích kapitolách, všechna rozšíření jsou umístěna ve složce *Extensions* v rámci hlavního adresáře aplikace. Veškeré soubory rozšíření jsou součástí adresáře, jehož název je zcela libovolný, nicméně se doporučuje stejné pojmenování jako má hlavní inicializační třída. S vytvářením nových objektů, v rámci těchto projektů, souvisí také správné umístění v rámci jmenného prostoru. Doporučeným jmenným prostorem pro každé rozšíření je *<název-rozšíření>\<...>\<...>*, čímž se zajistí konzistence programovacích struktur a nedojde tak k možným chybám během registrace rozšíření do jádra systému.

Hlavní inicializační třída každého rozšíření musí dědit od abstraktní třídy ExtensionBase, která implementuje rozhraní IExtension. Rozhraní poskytuje definici konstruktoru a tři základní metody, které v určitý čas hrají důležitou roli v propojení navazujících funkcionalit s jádrem redakčního systému. První z těchto metod je *initialize()*, která je volána při každé registraci rozšíření do jádra. Tuto metodu je tady možné například používat pro registraci *Vindu* widgetů či podobné služby a mnoho dalšího. Následují metody *install()* a *uninstall()*, které jsou volány vždy při aktivaci či deaktivaci rozšíření. Časté využití těchto metod souvisí především s inicializační částí rozšíření pro databázový systém.

Rozšíření, kromě základní struktury popsané v předchozích kapitolách, nejsou nijak omezená. Je tak zcela na vývojáři, k jaké potřebě tuto funkcionalitu využije. Veškerá aktivní rozšíření jsou ukládána v rámci konfiguračního souboru *anker_extensions.neon*.

6.3.9 Tvorba témat

Jádro samo o sobě poskytuje služby, které jsou jednotlivými tématy využívány, přesto tyto služby nedefinují samotné vlastnosti a chování webové aplikace. Tuto skutečnost umožňují tzv. témata, která jsou součástí redakčního systému, případně mohou být vytvořena zcela dle představ zákazníka. Jádro rozeznává dva druhy témat: veřejné a soukromé. Veřejná témata definují vlastnosti veřejné části webové stránky, tedy té části, která je obvykle volně dostupná neregistrovaným uživatelům. Soukromá témata pak popisují chování administrační části aplikace. Umožněním vlastní tvorby administračního tématu je možné zcela upravit a přizpůsobit webovou stránku dle požadavků, které jsou mnohdy velmi specifické.

Všechna témata jsou umístěna ve složce *Themes* v hlavním adresáři aplikace. Tato složka je dále rozdělena na dvě podsložky: *Frontend* – veřejná témata a *Backend* – soukromá témata. Povinností každého tématu je nutnost dodržen stanovenou strukturu, která byla popsána v předchozích kapitolách. Šablona, kromě presenterů a šablon, může teoreticky obsahovat i dodatečné funkce, které by za normálních okolností byly součástí nějakého rozšíření.

Veškeré soubory tématu jsou součástí adresáře, jehož název je zcela libovolný. S vytvářením nových objektů, v rámci těchto témat, souvisí také správné umístění v rámci jmenného prostoru. Doporučeným jmenným prostorem pro každé téma je *Themes\<[Frontend] nebo [Backend]>\<název-tématu>\<...>\<...>*, čímž se zajistí konzistence programovacích struktur a nedojde tak k možným chybám během registrace tématu do jádra systému. Redakční systém poskytuje jednoduchý přehled všech dostupných témat a umožňuje jejich snadné přepínání. Aktivní témata jsou ukládáná v rámci konfiguračního souboru *anker_themes.neon*.

6.3.10 Jazykové mutace

Jako jedna z výhod byla vyzdvihnuta možnost multijazyčnosti aplikace. Ačkoliv tuto možnost systém poskytuje, je zcela na jednotlivých tématech, zda tuto funkcionalitu budou využívat. Překlady jsou uloženy v rámci kata-

logů, kterých může být hned několik. Redakční systém si uchovává informace o aktivním jazyce a tyto data využivá při získávání překladů z konkrétních katalogů. Samotný název katalogu je pak využívám při identifikaci zdroje. Každá šablona tématu může obsahovat několik těchto identifikátorů. Pro využití této funkcionality v rámci šablony Latte se využívá tzv. filtr. Tvar tohoto filtru je předem stanovený překladovou vrstvou: _*<identifikátor-katalogu>.<identifikátor-zdroje>.<...>.<...>*. Kromě překladů v šablonách rozlišuje redakční systém také své interní identifikátory, které často začínají klíčových slovem katalogu *anker*. Tyto překlady se generují například v rámci vlastních typů příspěvků a jiných interních položek.

Přestože samotné překlady nejsou součástí žádného databázového systému, operace nad jednotlivými jazyky a překlady poskytuje *LanguageManager*. Mezi tyto operaci patří získávání katalogů a překladů, jejich vytváření, editace a mazání.

6.3.11 Obecná nastavení

Jádro redakčního systému také obsahuje některé základní možnosti nastavení, mezi které patří globální definice názvu stránky, její popis, nastavení údajů pro *OpenGraph* či změnu ikonky webové stránky. Veškeré položky nastavení a jejich hodnoty jsou uchovávány pomocí entity *Settings*, která je v databázovém systému reprezentována tabulkou *anker_settings*. Struktura této tabulky obsahuje sloupce, které uchovávají identifikátor nastavení a jeho hodnotu.

Přestože jádro systému obsahuje omezené množství nastavení, jakékoliv rozšiřující možnosti jsou jednoduše integrovatelné s již zavedenými funkcemi.

6.3.12 Vindu widgety

Důležitou součástí systému, která pomáhá ke zjednodušení integrace rozšiřujících funkcí jsou tzv. *Vindu widgety*. Vindu principiálně fungují na způsobu *ShortCodes* redakčního systému WordPress. Widget je možné registrovat do jádra systému pomocí metody *registerVindu*, která je součástí třídy *AnkerWrapper*. Metoda požaduje předání dvou parametrů. Prvním z nich je jedinečný identifikátor Vindu, druhý pak samotná instance Vindu. Každý widget je reprezentován třídním objektem, který dědí od abstraktní třídy *AnkerVinduBase*, jenž implementuje rozhraní IAnkerVindu s metodou *renderVindu*. Každá Vindu třída musí tuto metodu implementovat, neboť jejím

výsledkem je vyrenderovaná šablona widgetu, která bude přidána na příslušné místo v rámci šablony či příspěvku. Vývojář widgetu má přístup k jakýmkoliv datům pomocí *AnkerWrapperu* nebo DI kontejneru.

6.4 Životní cyklus

Životní cyklus redakčního systému vychází z velké části z Nette frameworku. Veškeré požadavky uživatelů jsou posílány přes jediný PHP skript (*index.php*), jenž je umístěn ve veřejném adresáři *www*. Tento soubor přenechá řízení zpracování aplikačnímu souboru *bootstrap.php*, kde proběhne základní konfigurace prostředí, DI kontejnerů a registrace služeb aktivních rozšíření a témat. Samotný kontejner aplikace nedokáže posoudit přijaté požadavky, proto jsou předány routeru, který ho přeloží do srozumitelných dat. Následuje zavolání příslušného typu presenteru, který provede inicializaci jádra redakčního systému.

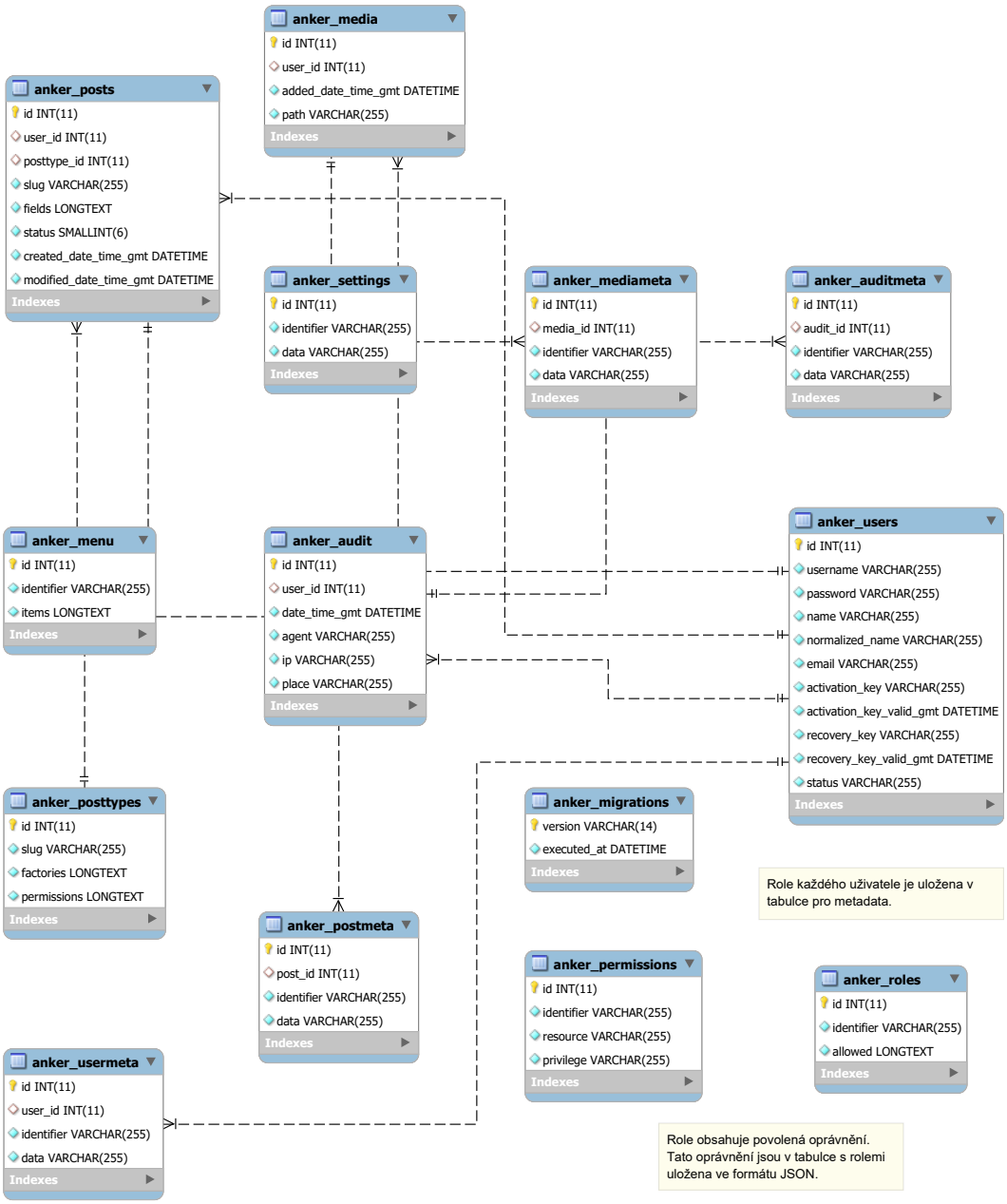
Během inicializace jádra se provádí několik důležitých úkonů, mezi které patří integrace aktivních témat či rozšíření, shromáždění registrovaných presenterů redakčního systému, registrace veškerých Vindu widgetů a inicializace připojených rozšíření. Poté se dle životního cyklu presenteru zpracovávají operace *action<Akce>*. Následně se inicializuje požadovaný presenter redakčního systému a zpracují se operace *handle<Signál>*. Jednotlivé operace systémových presenterů jsou pak záležitostí každého tématu. Nakonec se provede vyrenderování šablony s poskytnutými daty, které jsou odeslány klientovi a následně zobrazeny.

6.5 Databázový model

Databázový systém základního jádra redakčního systému se sestává z celkem čtrnácti tabulek, které jsou rozepsány v tabulce 6.1. Některé tabulky jsou na sebe přímo závislé, některé spolu vůbec nemusí souviset. Kardinalita jednotlivých vazeb tabulek je znázorněna na obrázku 6.8.

Název tabulky	Popis
anker_audit	Záznamy auditů uživatelů
anker_auditmeta	Metadata auditů
anker_media	Záznamy dostupných multimediálních souborů
anker_mediameta	Metadata multimediálních souborů
anker_menu	Záznamy vytvořených struktur menu
anker_migrations	Seznam aktivních migrací
anker_permissions	Seznam registrovaných oprávnění
anker_postmeta	Metadata příspěvků
anker_posts	Záznamy vytvořených příspěvků
anker_posttypes	Seznam dostupných typů příspěvků
anker_roles	Seznam registrovaných rolí
anker_settings	Záznamy s dostupnými nastaveními
anker_usermeta	Metadata uživatelů
anker_users	Seznam uživatelů systému

Tabulka 6.1: Seznam tabulek v databázovém systému MySQL



Obrázek 6.8: ERA model databáze redakčního systému Anker

7 Závěr

V první teoretické části práce byly zmapovány standardně používané webové technologie pro tvorbu PHP aplikací, definovány některé základní pojmy pro vývoj i samotný provoz systému a v poslední řadě prostudovány některé redakční systémy, se kterými máme možnost se setkat nejvíce. Přínosem této části je zmapování specifických rysů a vlastností jednotlivých redakčních systémů založených na technologii PHP, dále šířené jako Open Source software. Tyto poznatky se dají prakticky využít při tvorbě vlastní aplikace tohoto druhu. Prostudováním jednotlivých redakčních systémů můžeme snadněji vystihnout jejich kladné vlastnosti či funkce, které následně zužitkujeme, a poskytneme tak uživateli příjemný nástroj pro správu webových stránek, příspěvků nebo multimediálních souborů. V závěru této části byl popsán *Nette framework* platformy PHP, který značně usnadnil tvorbu vlastního redakčního systému i jednotlivé operace pro vývojáře rozšíření.

Druhá praktická část práce se zaměřuje na vytvoření návrhu redakčního systému založeném na frameworku Nette z platformy PHP. V úvodní části samotné realizace je představen princip funkčnosti a životního cyklu celé aplikace, včetně možností vytváření a integrace vlastních rozšíření společně s tvorbou tématů pro webovou aplikaci. Následuje popis struktury projektu, přičemž je zde vyzdvižena nejen hierarchická struktura jádra aplikace, ale především pak obsah jednotlivých důležitých konfiguračních souborů. Další část práce více rozvádí možnosti a funkcionalitu redakčního systému. Jednotlivé funkce jsou zdokumentovány a okomentovány, v případě tvorby rozšíření, témat či jazykových mutací webové aplikace jsou blíže popsány základní požadavky, které vedou k úspěšné integraci těchto rozšíření do jádra redakčního systému. Na závěr je představena struktura databáze, na které celá aplikace stojí. Tento koncept může být využit k inspiraci pro řešení podobných, menších projektů a nalézt uplatnění u začínajících tvůrců podobných systémů. Není příliš složitý na pochopení, tudíž není problém s uvedením do běžného provozu. Technologické prostředky pro fungování toho sytému jsou dostupné vesměs velmi levně. Řešení postačuje pro správu jednoduché dynamické prezentační webové stránky s možností publikování a úpravy stránek či příspěvků, jednoduchou správou multimediálních souborů a tvorbě vlastních menu, témat nebo rozšíření.

Téma redakčních systémů je dnes natolik rozsáhlé, že není možné vystihnout všechny aspekty, které s uvedeným tématem souvisí. V dnešní době existuje mnoho systémů, které nabízejí správu webových stránek na různých platformách, přičemž každý je specifický nabízenými funkcionalitami, které mnohdy bývají odlišně. Z tohoto důvodu je výběr redakčního systému pro zákazníky zcela subjektivní záležitostí, často zkoumáme účel a budoucí zaměření portálu a snažíme se najít nejlepší možné řešení naší problematiky.

Literatura

[1] *What is Babel?* [online]. Babel, . [cit. 2019/11/16]. Babel. Dostupné z: `https://babeljs.io/docs/en/`.

[2] *Babel: Sponsors* [online]. Babel, . [cit. 2019/11/16]. Babel. Dostupné z: `https://babeljs.io/support`.

[3] *Systém pro správu obsahu* [online]. ZONER software, 2011. [cit. 2019/11/19]. CMS. Dostupné z: `https://www.interval.cz/clanky/10-nejlepsich-redakcnich-systemu-cms/`.

[4] David M. Kroenke, D. J. A. *Databáze*. Albatros Media a.s., 2015. ISBN 978-80-251-4352-0.

[5] *OOP Concept for Beginners: What is Inheritance?* [online]. Stackify, 2017. [cit. 2019/11/20]. Dědičnost. Dostupné z: `https://stackify.com/oop-concept-inheritance/`.

[6] *Dependency Injection* [online]. Nette Foundation. [cit. 2019/12/01]. Dependency Injection. Dostupné z: `https://doc.nette.org/cs/3.0/dependency-injection`.

[7] *Drupal - Open source CMS* [online]. Drupal, 2018. [cit. 2019/11/20]. Drupal. Dostupné z: `https://www.drupal.org`.

[8] *GNU General Public License* [online]. Tibor Peták. [cit. 2019/11/25]. GNU General Public License. Dostupné z: `http://www.gnugpl.cz`.

[9] Greig, S. *CSS3 - Publishing the Limits*. John Wiley & Sons Ltd, 2013. ISBN 978-1-118-65263-3.

[10] Horňáková, M. *333 tipů a triků pro WordPress*. Computer Press, a.s., 2011. ISBN 978-80-251-3443-6.

[11] *HTML: Living Standard* [online]. Whatwg. [cit. 2019/11/16]. HTML. Dostupné z: `https://html.spec.whatwg.org/multipage/`.

[12] *Úvod do Nette frameworku pro PHP* [online]. David Čápka. [cit. 2019/12/01]. ITnetwork. Dostupné z: `https://www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette`.

[13] *JavaScript* [online]. W3Schools, 2006. [cit. 2019/11/16]. JavaScript. Dostupné z: `https://www.w3schools.com/js/`.

[14] *JavaScript (JS)* [online]. MDN contributors, . [cit. 2019/11/16]. JavaScript. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[15] *Co je Joomla!?* [online]. JoomlaPorta.cz, 2018. [cit. 2019/11/20]. Joomla! Dostupné z: https://www.joomlaportal.cz/o-joomla.

[16] *Kentico CMS* [online]. Kentico. [cit. 2019/11/22]. Kentico. Dostupné z: https://www.kentico.com.

[17] *Latte Templating Engine* [online]. Nette Foundation. [cit. 2019/12/01]. Latte. Dostupné z: https://latte.nette.org/cs/guide.

[18] *MVC architektura* [online]. David Čápka. [cit. 2019/11/23]. MVC architektura. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

[19] *MVVM: model-view-viewmodel* [online]. Václav Dajbych, 2009. [cit. 2019/11/23]. MVVM. Dostupné z: https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel.

[20] *Práce s formátem NEON* [online]. Nette Foundation. [cit. 2019/12/01]. NEON. Dostupné z: https://doc.nette.org/cs/3.0/neon.

[21] *Proč používat Nette?* [online]. Nette Foundation. [cit. 2019/12/01]. Nette Framework. Dostupné z: https://doc.nette.org/cs/3.0/why-use-nette.

[22] *nopCommerce - Free and open-source eCommerce platform* [online]. nopCommerce. [cit. 2019/11/22]. nopCommerce. Dostupné z: https://www.nopcommerce.com.

[23] *Objektově orientované programování* [online]. ZONER software, 2006. [cit. 2019/11/20]. Objektově orientované programování. Dostupné z: https://www.interval.cz/clanky/oop-v-php-vyuziti-oop-v-praxi/.

[24] *OOP v PHP: Základy OOP* [online]. Jakub Mrozek, 2006. [cit. 2019/11/20]. OOP v PHP. Dostupné z: https://www.interval.cz/clanky/oop-v-php-zaklady-oop/.

[25] *Otevřený software* [online]. Open Source Initiative, 2004. [cit. 2019/11/19]. Open-source software. Dostupné z: https://opensource.org.

[26] *PHP: Hypertext Preprocessor* [online]. The PHP Group, 2005. [cit. 2019/11/16]. PHP. Dostupné z: https://www.php.net/manual/en/.

[27] *History of PHP* [online]. The PHP Group, 2001. [cit. 2019/11/15]. History of PHP. Dostupné z: https://www.php.net/manual/en/history.php.php.

[28] *Čím je Adminer lepší než phpMyAdmin?* [online]. Jakub Vrána. [cit. 2019/12/02]. phpMyAdmin. Dostupné z: https://www.adminer.org/cs/phpmyadmin/.

[29] *Pimcore* [online]. Pimcore. [cit. 2019/11/21]. Pimcore. Dostupné z: https://pimcore.com/en/products/experience-manager/web-content-management/introduction.

[30] *OOP Concepts for Beginners: What is Polymorphism* [online]. Stackify, 2017. [cit. 2019/11/20]. Polymorfismus. Dostupné z: https://stackify.com/oop-concept-polymorphism/.

[31] *Some things you may think about PostCSS... and you might be wrong* [online]. Julian Ćwirko, 2016. [cit. 2019/11/16]. PostCSS. Dostupné z: http://julian.io/some-things-you-may-think-about-postcss-and-you-might-be-wrong/.

[32] *Subrion - Open Source CMS* [online]. Subrion, . [cit. 2019/11/21]. Subrion. Dostupné z: https://subrion.org.

[33] *Subrion* [online]. CMSWire, . [cit. 2019/11/21]. Subrion. Dostupné z: https://www.cmswire.com/d/subrion-cms-p002537.

[34] *Thunder* [online]. Drupal, 2016. [cit. 2019/11/21]. Thunder. Dostupné z: https://www.drupal.org/project/thunder.

[35] *Tracy debugging tool* [online]. Nette Foundation. [cit. 2019/12/01]. Tracy. Dostupné z: https://tracy.nette.org/cs/guide.

[36] *Třívrstvá architektura a další vícevrstvé architektury* [online]. David Čápka. [cit. 2019/11/23]. Třívrstvá architektura. Dostupné z: https://www.itnetwork.cz/navrh/architektury-a-dependency-injection/trivrstva-architektura-a-dalsi-vicevrstve-architektury.

[37] *Webový server* [online]. Mozilla, 2005. [cit. 2019/11/25]. Webový server. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server.

[38] *WordPress česky* [online]. WordPress. [cit. 2019/11/19]. WordPress. Dostupné z: https://cs.wordpress.org.

[39] *OOP Concept for Beginners: What is Encapsulation* [online]. Stackify, 2017. [cit. 2019/11/20]. Zapouzdření. Dostupné z: https://stackify.com/oop-concept-for-beginners-what-is-encapsulation/.

Seznam obrázků

Seznam tabulek

A Instalace aplikace

Ačkoliv aplikace přímo nenabízí grafické rozhraní, které se stará o automatickou inicializaci datových struktur, je instalace a rozchození redakčního systému celkem jednoduché. Důležitým krokem ke zprovoznění systému je umístění zdrojového projektu do veřejné složky webového hostingu (často označované jako *public* či *htdocs*). Minimálním požadavkem na provoz této aplikace je PHP verze 7.2, doporučená je však novější verze 7.3. Součástí projektu je ve složce *www* inicializační SQL soubor pro MySQL databázi. V následujícím postupu vytvoříme, například pomocí programu *phpMyAdmin*, novou databázi pro náš systém, do které importujeme již zmíněný *anker_init.sql*. Poslední krokem je změna přihlašovacích údajů k databázi, kterou bude redakční systém využívat. Soubor s těmito informacemi se nachází v hlavní složce aplikace *app* v podadresáři *config* – *config.local.neon*. Struktura tohoto souboru je znázorněna na obrázku A.1. Položku *metadata* pokud možno neměníme.

```
doctrine:
    user: root
    password:
    dbname: anker
    metadata:
        Anker: %appDir%
```

Obrázek A.1: Struktura lokálního konfiguračního souboru pro databázi