Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Návrh a vývoj redakčního systému

Martin Toms

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Martin Toms

Systémové inženýrství a informatika
Informatika

Název práce

Návrh a vývoj redakčního systému

Název anglicky

Design and Development of a Content Management System

Cíle práce

Cílem práce je návrh a implementace webového redakčního systému, přístupného jak z mobilních, tak desktopových zařízení (mobile-first). Tento systém bude umožňovat psaní a následné zobrazení článků uživatelům. Systém bude pro ukládání článků používat SQL databázi.

Metodika

Teoretická část se bude zabývat studiem odborných zdrojů a popisu využitých technologií. Vlastní vývoj bude využívat primárně technologie PHP, JS, a další s nimi související.

Doporučený rozsah práce

35-40 stran

Klíčová slova

PHP, HTML5, CSS, webová aplikace, cms, redakční systém

Doporučené zdroje informací

ARNTZEN, Thies C., Stig BAKKEN, Shane CARAVEO, et al. PHP GROUP. PHP: Hypertext Preprocessor [online]. 1997. Dostupné z: https://www.php.net/

CASTRO, E. – HYSLOP, B. *HTML5 a CSS3 : názorný průvodce tvorbou WWW stránek.* Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

ČÁPKA, David. Jednoduchý redakční systém v PHP objektově (MVC). ITNetwork.cz [online]. [cit. 2020-06-12]. Dostupné z: https://www.itnetwork.cz/php/mvc

GUTMANS, A. – BAKKEN, S S. – RETHANS, D. *Mistrovství v PHP 5.* Brno: CP Books, 2005. ISBN 978-80-251-1519-0.

SKLAR, David. PHP 7: Praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. Zoner Press, 2018. ISBN 978-80-7413-363-3.

STACK EXCHANGE INC. Stackoverflow [online]. 2008. Dostupné z: https://stackoverflow.com/

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 2. 2021

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 15. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Návrh a vývoj redakčního systému" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15. 3. 2021 _____

Poděkování

Rád bych touto cestou poděkoval ing. Jiřímu Brožkovi, Ph.D. za vedení mé práce.

Návrh a vývoj redakčního systému

Abstrakt

Tato práce se zabývá vývojem aplikace pro publikaci článků různých typů. V první části popisuje použité technologie a související pojmy. V další části je pak představen návrh rozhraní a vývoj zajímavějších nebo komplexnějších částí této aplikace, včetně návrhů na budoucí vývoj jednotlivých funkcionalit. Návrh rozhraní je představen formou popisu snímků obrazovky z již běžící aplikace, tedy představením grafického designu.

Klíčová slova: PHP, HTML5, CSS, webová aplikace, CMS, UI design, redakční systém

Design and Development of a Content Management System

Abstract

This thesis deals with the topic of developing an application for publication of various types of articles. In the first part, it describes used technologies and related concepts. The next part contains the introduction of both the interface and development of some of the more interesting and complex parts of this application, including design and suggestions related to future development. The interface graphical design is presented in the form of described screenshots of the already running application.

Keywords: PHP, HTML5, CSS, web application, CMS, UI design, content management system

Obsah

Seznam obrázků

1 Úvod

Čtení a publikace různorodých článků, ať už v neformálních podmínkách blogů, či o něco formálnějších podmínkách webových portálů se širokým množstvím obsahu, od zpráv po bulvární články, je na denním programu většiny z nás.

Můj pohled na online publikace byl vždy jasný, chtěl jsem být součástí nejen té skupiny, která články čte, ale i té, která je připravuje a poskytuje čtenářům. Jakožto programátor mám unikátní příležitost jak docílit tohoto snu, tak si jej manuálně zrealizovat.

Tato práce se bude zabývat vývojem redakčního systému (webu), jehož primární funkce bude poskytování mých vlastních článků a textů jakéhokoliv charakteru čtenářům.

Finální verze tohoto systému bude dostupná na doméně toms.click, kde jej budu používat i v průběhu celého vývoje, jak za účelem rekreace, tak testování a rozvíjení jeho funkcionalit a vzhledu.

Práce se snaží vytvořit vlastní systém pomocí technologie PHP, MariaDB a dalších, bez použití již existujících redakčních systémů, jako je například WordPress.

V teoretické části popisuje technologie, které se následně využijí při vlastním vývoji této webové aplikace. Tento vývoj bude podrobněji popsán v praktické části.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je návrh a implementace webového redakčního systému, přístupného jak z mobilních, tak desktopových zařízení (mobile-first). Tento systém bude umožňovat psaní a následné zobrazení článků uživatelům. Systém bude pro ukládání článků používat SQL databázi.

2.2 Metodika

Teoretická část se bude zabývat studiem odborných zdrojů a popisu využitých technologií. Vlastní vývoj bude využívat primárně technologie PHP, JS, a další s nimi související.

3 Teoretická východiska

3.1 Architektura MVC

Model-View-Controller je typ architektury založený na spolupráci tří typů komponentů.[1, s. 1]

Controller je ta část aplikace, která zpracovává (kontroluje) veškerou komunikaci - ať už jde o sběr dat z URI (část adresy v adresním řádku za doménou) nebo jejich předávání ke zpracování. Každá stránka (funkce) aplikace (článek, archiv, uživatel, …) má vlastní controller.[1, s. 1]

Model se stará o data aplikace. Sám controller by data neměl zpracovávat, jen předávat mezi pohledy a modely. V modelech se tedy nachází výpočty.[1, s. 1]

Finální data jsou pak controllerem předána do view, pohledu, což je šablona obvykle obsahující HTML, popřípadě další jazyky, odesílaná uživateli. Správně by spolu view a modely měly komunikovat jen přes Controller, avšak detaily se mohou lišit autor od autora.[1, s. 1]

3.1.1 Možnost konkrétní implementace MVC struktury v PHP

Základní struktura aplikace by mohla obsahovat 3 hlavní složky:

- složka public, kam bude nasměrován vstup do aplikace, bude obsahovat index.php
- složka src, dále členěná na Model, View, Controller
- konfigurační soubor, ve kterém by bylo možné upravovat různé parametry aplikace

Controllery a modely budou moci mít podobu tříd, pohledy klasických šablon. Controllery budou mít metodu process($parameters). V indexovém souboru dojde k připojení k databázi a hlavně vytvoření RouterControlleru, který si přeparsuje (upraví do požadované formy) pomocí URL modelu URI parametry, vytvoří instanci třídy controlleru dle prvního parametru a předá mu zbylé parametry do process metody. Uvnitř controlleru se pak už jen provedou potřebné instrukce, zvolí pohled a odešle se se zpracovanými daty klientovi.[1, s. 1]

3.1.2 Příklad průchodu přes MVC strukturu

Uživatel zadá URL https://www.example.com/user/login, požadavek se dostane do vstupního souboru aplikace (index.php) v němž se vytvoří RouterController pro tuto URI (/user/login). Následně si RouterController extrahuje první parametr (user) a spustí UserController, jemuž předá zbylý parametr pro zpracování (/login).

3.1.3 .htaccess

Pro realizaci tohoto systému bude nutné patřičně nakonfigurovat webový server, pro zjednodušení procesu lze využít soubor .htaccess postavený do kořenového adresáře serveru.[1, s. 2]

Pomocí tohoto souboru též může být řešena komprese, povolení přístupu k souborům jen s patřičnou koncovkou[1, s. 2], což se hodí třeba u obrázků, nebo PDF souborů.

Zároveň tím můžeme znepřístupnit soubory aplikace a znemožnit potenciálním útočníkům jednoduše zobrazit obsah ostatních souborů (např. konfigurační soubory a .php skriptovací soubory).

3.1.4 **Autoloading**

ItNetwork ukazuje příklad načítání tříd pomocí jednoduchého konstruktu v PHP.[1, s. 2]

Jako lepší varianta může být použití Composeru a vložení autoload (a k tomu použití jmenného prostoru Model, View a Controller v rámci PSR-4 podle direkcí PHP-FIG11) argumentu do souboru composer.json a dle instrukcí dokumentace Composeru.[9]

3.1.5 **RouterController**

Jak uvádí Čápka, cílem RouterControlleru je získat z položky $_SERVER['REQUEST_URI'] pole obsahující název kontroleru a následné parametry, následně zavolat příslušný kontroler dle jeho jména.[1, s. 3]

3.2 **Front-end a back-end**

Klíčovými pojmy pro tuto práci jsou front (přední) a back (zadní) end (konec). Rozlišují umístění technologií v aplikaci. Obvykle se používají jako podstatné jméno pro označení části aplikace (množina technologií apod.) nebo jako přídavné jméno pro zařazení do ní.

Front-endové technologie jsou klientem viditelné, odesílají se prohlížeči, jejich kód i fungování může být uživatelem tím pádem (ať už chtěně či nechtěně, např. při nedostatečném zabezpečení) ovlivněno.

Back-endové technologie jsou ty, které nejsou viditelné. Jejich kód se nachází na serveru a je zpracováván též serverem. Jelikož uživatel přístup k souborům serveru za normálních zabezpečených okolností nemá, nemůže jej ovlivnit běžnými způsoby. (Je to však možné např. pomocí tzv. SQL injection, viz. kapitola o bezpečnosti.)

Obecně známými front-endovými technologiemi jsou HTML (kódování, struktura), JavaScript (skriptování funkcionalit na straně klienta) a CSS (stylování, barvy, vzhled). Na back-endu se často používá SQL (databáze, uložení dat na serveru) v konjunkci s PHP, C# nebo Javou (skriptování na straně serveru).

3.3 **Multiplatformní přístup**

Pro správné zobrazení aplikace na různých zařízeních existuje několik možností, je na ně nutné myslet v průběhu vývoje.

3.3.1 **Různé aplikace pro různá zařízení**

Castro naznačuje, že se jedná o relativně nákladnou záležitost. Uvádí též, že důvodem pro toto provedení mohou být různé požadavky v závislosti na zařízení „*To, co uživatel potřebuje, když je na cestách se svým mobilním telefonem, se většinou odlišuje od toho, co vyhledává z domova nebo kanceláře.*"[2, s. 248]

Na druhou stranu, pokud se mobilní a webová aplikace liší nejen vzhledem, ale i funkcionalitou, může se uživatel z hlediska UX ztratit a hledat funkce, které mu na druhém zařízení nejsou k dispozici.

3.3.2 **Responzivní design**

Aplikaci pro desktopové zařízení je možné napsat tak, aby se správně zobrazovala i na menších. Pomocí tzv. „media queries" se stanoví styly, které se aplikují při parametrech jako je maximální šířka obrazovky.

Castro doporučuje pro více informací o responzivním designu materiály od Ethana Marcotta.[2, s. 250]

3.3.3 **Mobile-first**

Jak už název napovídá, při vývoji mobile-first se postupuje opačně než u responzivního designu. První styly jsou pro nejmenší (mobilní) zařízení, následují tablety a počítače pomocí media queries.

Vzhledem k narůstajícímu počtu uživatelů používajících mobilní zařízení[5] je tento přístup vhodný pro většinu moderních aplikací.

3.4 **PHP**

PHP, hypertextový preprocesor, je back-endový programovací jazyk. Kód je uložen a spouštěn na serveru, není vidět uživatelem. Je primárně určen pro vývoj webových aplikací. Kód je zpracován před samotným odesláním kódu klientovi, je v něm tedy možné i skládání šablon a dalších aplikačních záležitostí zasahujících do front-endu.[8, s. 17]

Tento jazyk je dynamicky typovaný a je obecně známý svým používáním symbolu dolaru před názvy proměnných ($variable). Existuje pro něj řada Frameworků (Například Laravel, Symphony, Nette)[8, s. 281 - 286,] ale tato práce se bude zabývat vývojem přímo v „čistém" PHP, tedy bez nich.

Sklar jako přednosti jazyka PHP uvádí jeho cenu (zadarmo), open-source-nost, multiplatformnost a právě zmíněnou cílenost na vývoj webových aplikací.[8, s. 19]

3.4.1 **Zpracování dat z formulářů**

V HTML existují dvě hlavní metody pro odeslání formulářů (více o formulářích viz. kapitola Formuláře). Jsou to metody POST a GET.[8, s. 122]

Sklar uvádí, že „*URL a formulářové parametry z metody GET formulářů se dávají do $_GET. Formulářové parametry z metody POST formulářů se dávají do $_POST.*"[8, s. 122]

Přes metodu GET je navíc požadavek / obsah formuláře vidět v URL prohlížeče. Jako příklad této funkce uvádí „*Například URL http://www.example.com/catalog.php?product_id=21&category=fryingpan vloží do $_GET: $_GET['product_id'] se nastaví na 21. $_GET['category'] se nastaví na fryingpan.*"[8, s. 122]

3.4.2 **Vývojářské prostředí pro PHP**

Zatímco je kód tohoto jazyka samozřejmě možné psát v jakémkoliv textovém editoru, každý programátor ví o tzv. IDE. Integrated development interface je pokročilejší editor, obvykle s mnoha funkcemi nad rámec prostého textového editoru. Často obsahuje alespoň náznak korektury kódu, základní debuggingové funkce (breakpoints), Git integraci a další prvky známé jako „*quality of life improvements*".

Pro PHP samozřejmě existuje mnoho prostředí (stejně tak jako pro většinu velkých programovacích jazyků), avšak já preferuji PHPStorm od JetBrains nebo NetBeans. PHPStorm je placená varianta a o něco lepší v množství a provedení funkcí. Má k dispozici i studentskou licenci zdarma. NetBeans je IDE, které je zdarma.

3.4.3 **Composer**

Balíčkovací manažer jazyka PHP Composer je nástroj pro správu balíčků. Sklar říká, že umožňuje integraci cizích knihoven do vlastního kódu a to jediným příkazem.[8, s. 273]

Instalace i používání Composeru je velice snadná a spočívá v pár příkazech. Sklar uvádí že balíky můžeme hledat na webu Packagist.org.[8, s.275]

Přes Composer se také dá nainstalovat Latte (viz. kapitola Latte): „*composer require latte/latte*"[6]

3.4.3.1 Cebe's Markdown

Pro implementaci markdownu (viz. kapitola markdown, jazyk pro formátování textu) existuje spousta parserů (překladačů). Liší se v konkrétní implementaci (a tím pádem v rychlosti i stylu použití), popřípadě možností upravovat parsovací kód či v podporovaných flavorech.

Parser cebe/markdown jsem vybral, jelikož je úprava stávajících a přidávání nových elementů v příadě potřeby velice jednoduchá a dobře zdokumentovaná.

Jak říká sám Brandt (cebe): „*Cílem této implementace je, aby byla rychlá (viz měřítko) a rozšiřitelná. Překládání Markdownu na HTML je tak jednoduché, jako zavolání jediné metody*"[11, a]

Instalace je možná skrze Composer:
„composer require cebe/markdown "~1.2.0""[18]
Použití je, jak píše cebe, možné pomocí jediné metody:
„$parser = new \cebe\markdown\Markdown();
echo $parser->parse($markdown);"[18]

3.4.4 **Bezpečnost**

Klíčovým bodem při vývoji aplikace je bezpečnost. Nedostatečné zabezpečení aplikace a dat (ať už našich, či zákazníků a klientů) může vést ke ztrátě dobrého jména firmy, soudním sporům nebo až k likvidaci firmy. (Často používaná hláška k tomuto tématu je „*Never trust the user*.")

3.4.4.1 Volba front-end versus back-end řešení funkcionalit

Určitě stojí za zmínku, že ne každá funkcionalita se hodí na front-end. Pokud bude celý kód administrátorského rozhraní včetně jeho přístupových údajů řešeno pouhým JavaScriptem, nejspíše v něm budou odhaleny údaje / přístup nebude zabezpečen proti změně kódu.

Stejně tak validace formulářů tam, kde na tom záleží, nemůže být pouze na straně front-endu, neboť ten si uživatel může libovolně měnit. (Například vypnutí elementu formuláře, které uvádí Castro[2, s. 337])

Na druhou stranu, použití front-end řešení funkcionalit šetří výkon serveru, takže má své místo v aplikaci.

[a] The implementation focus is to be fast (see benchmark) and extensible. Parsing Markdown to HTML is as simple as calling a single method

3.4.4.2 SQL injection a prepared statements

Často zmiňovaným bodem bezpečnosti na stackoverflow[3] (převážně když se uživatelé ptají na otázku a nevědomky tuto bezpečnostní trhlinu ve svém kódu mají) jsou tzv. parametrizované dotazy.

Název vychází z podstaty fungování při ochraně proti SQL injection. Nejprve je třeba zmínit co je vlastně tato 'injekce SQL'.

SQL Injection je technika spočívající ve zneužití obvykle back-endového kódu přistupujícího k databázi. Útočník zneužije nedostatečnou schopnost serveru rozpoznat vlastní dotaz od neošetřeného vstupu. V praxi to může vypadat takto - původní: WHERE id='vstup', kde by vstup obsahoval apostrof a pokračování OR id=1.

Pokud by server nepoznal, že byla část po logickém součtu OR přidána útočníkem, provedl by se dotaz ačkoliv třeba není přihlášen uživatel s daným id.

Prepared statements tomuto předcházejí postupným složením (připravením) dotazu s parametry. Vzorová klauzule WHERE id='vstup' by byla WHERE id=? a vstup by byl následně předán a dosazen za otazník.[3] Parametrizace tímto způsobem umožňuje jasné odlišení vstupu od původního příkazu a kód již není pevně otevřen útoku.

V PHP existují dvě hlavní možnosti aplikace prepared statements. MySQLi a PDO. Osobně mám radši syntaxi a fungování PDO.[4]

3.4.4.3 Cross-site scripting (XSS)

Nebezpečí „*mezistránkového skriptování*" spočívá ve zneužití neošetřeného vstupu a následně výstupu na stránce. Útočník vloží například do komentářového pole skript, který (pokud není pole ošetřeno) bude zobrazen jako komentář a ovlivní frontendovou část aplikace pro ostatní uživatele.

V momentě, kdy se dostane cizí skript na stránku, může přes něj útočník sledovat vstup uživatele, tím pádem krást hesla, nebo i ukrást cookie SESSID a využít jej pro přihlášení (i bez údajů) za daného uživatele na jiném zařízení. (Jelikož přihlášení je nejspíše řešeno přes cookie typu session, to je uloženo na serveru a kontroluje se část u uživatele na počítači.[8, s. 202])

3.4.4.4 Únik dat, hashování, šifrování

Je nevyhnutelná šance, že se k datům uloženým na serveru přes veškerá snažení někdo dostane. Mělo by se s nimi tím pádem zacházet jako s takovými a snažit se jejich užitečnost co nejvíce znehodnotit z pohledu nepovolaných osob.

Jak říká o heslech Sklar, „*Když se ovšem ukládají v hashované podobě, útočník nezíská skutečná hesla, protože není způsob, jak z takového hesla získat původní čisté heslo, které je potřeba zadat při přihlašování.*"[8, s. 212]

Základním pravidlem při ukládání dat je tedy hashování hesel a dalších dat, která znát nepotřebujeme nad rámec ověření jejich správnosti.

V PHP se nachází několik nativních funkcí pro generování hashe. Hash je textový řetězec o určité délce, často bývá spojen s tzv. „*salt*", což je zjednodušeně řečeno přidání náhodného prvku k hashi. Sklar doporučuje funkce password_hash() a password_verify().[8, s. 212] Většina uživatelů v PHP dokumentaci daných funkcí nedoporučuje použití vlastní „*soli*"[13], což značí, že jej tyto funkce (které si generují salt automaticky13) tvoří lépe než člověk neznalý kryptografie.

3.4.4.5 HTTPs a certifikáty pro spojení přes něj

Pro zabezpečenou komunikaci mezi klientem a serverem se používají HTTP Secure certifikáty. Tyto certifikáty je možné získat jak placeně, tak zdarma.

Dříve bylo získat certifikát zdarma mnohem složitější a byla to jakási záruka bezpečnosti stránky, často se upozorňovalo na zelený zámeček v adresním řádku („*connection is secure*") u internetového bankovnictví a podobných stránek.

V dnešní době již je HTTPS na většině reputabilních (ale bohužel i škodlivých) stránkách a serverech a prioritně se poukazuje spíše na červený zámeček značící naopak neplatný certifikát.

Osobně pro generování certifikátů zdarma používám Let's Encrypt, který již spousta hostingových společností nabízí zdarma či za drobný poplatek k jejich službám. (Například společnost Active24 k hostingu nabízí již delší dobu automaticky se obnovující certifikát Let's Encrypt pro několik domén i subdomén.)

Zatímco certifikát zaručí, že je komunikace se serverem zabezpečená, už nezaručí, že na druhé straně nesedí osoba s nekalými záměry.

3.5 **HTML**

HTML je značkovací jazyk popisující strukturu webové stránky. Aktuálně hojně používaná verze je HTML5, která se soustředí primárně na sémantické značky a další 'quality of life' zlepšení. HTML je obecně velice známý jazyk, proto se o něm budu zmiňovat jen okrajově.

3.5.1 **Formuláře**

Velice důležitou částí webových aplikací jsou formuláře. Castro hezky říká, že formuláře jsou k tomu, aby návštěvníci stránek mohli na našich stránkách komunikovat s námi a že jejich prvky jsou element form, formulářové pole a odesílací tlačítko.[2, s. 313]

Dále říká, že, „*Když používáme formuláře, často potřebujeme skriptovací jazyk na straně serveru, v němž by napsaný skript přijímal odeslané informace.*" Podotýká, že vhodným jazykem pro začátek je PHP.[2, s.313]

V dnešní době je možné spolehlivě používat celou řadu jazyků na straně serveru - Javu (často Spring), PHP, Python, C# (ASP.NET) nebo i JavaScript (node.js). Důvodem použití jiného jazyka může být třeba i jednotnost jazyků na front/back-end částech. (Například REACT na frontendu a Express.js na back-endu[10])

3.5.2 **Latte**

Latte je šablonový engine používaný frameworkem Nette. Je možné jej použít i bez tohoto frameworku. Vhodný je pro zjednodušení nebo zpřehlednění syntaxe HTML/PHP šablon, kde je většina konstruktů PHP nahrazena složenými závorkami.

Syntaxi má téměř identickou jako PHP, ale čitelnost souborů je vylepšená. Funguje podobně jako preprocesory, kód šablon Latte je automaticky zkompilován do PHP kódu.[6]

Po instalaci pomocí composeru (viz. kapitola Composer) se velice jednoduše zprovozní celý systém přes třídu Latte\Engine, která obsahuje metodu render($template, $parameters). Render jako první parametr přijímá stringový název (cestu) šablony, například soubor.latte, a pole s parametry jako druhý.[6]

3.6 CSS3

Kaskádové styly jsou předpisem vzhledu webové stránky. Přes různé selektory tohoto jazyka se v nich předává prohlížečům informace o tom, jak mají jednotlivé prvky vypadat.

V CSS3 byly navíc přidány funkce umožňující snadný vývoj responzivních aplikací pomocí tzv. „*media queries*", které umožňují změnu stylů na bázi velikosti obrazovky.

Dalším pozitivním prvkem CSS3 jsou animace. Zatímco v dřívějších dobách byl front-end programátor odkázán na JavaScript, nyní je možné všemožné animace psát přímo jako součást stylů.

3.6.1 SASS

Volně přeloženo jako „*syntakticky úžasné stylesheety*", SASS umožňuje využití pokročilejších funkcionalit v CSS. Mezi tyto funkcionality patří nestování, mix-iny, proměnné, cykly a řada dalších. Může výrazně zpřehlednit kód, obzvláště při aplikování metodik CSS.

Jedná se o preprocesor, tedy kód napsaný v SASS se přeloží na CSS kód. To lze provést jak přes příkazovou řádku, překladače tak různé file watchery, které mohou být i doplňkem IDE. Při použití výstupu tohoto preprocesoru tím pádem není nutná instalace speciálních prvků na server ani klientské zařízení.

3.6.2 SCSS

Sassy CSS je podobné SASSu, liší se primárně v používání závorek u SCSS narozdíl od SASS.

3.6.3 BEM

Metodika block element modifier spočívá ve snaze psát udržitelný kód za pomoci předem stanovených pravidel. Prvky stránky jsou rozděleny na komponenty, komponent může mít modifikátory a podřadné prvky. K tomu se používají jednoduchá pravidla psaní názvů tříd. Dalším pravidlem je vše stylovat pouze přes třídy, což zamezí obtížím při změně HTML markupu. BEM také neklade důraz na to, jestli je podřadný prvek přímým potomkem nebo ne, ale jen jestli je potomkem.

Alternativou k BEM může být třeba atomické CSS nebo řada dalších.

3.7 JavaScript

Jazyk často zaměňovaný s Javou, ačkoliv je radikálně odlišný, JavaScript, je front-sidový skriptovací jazyk. Umožňuje programovat různé funkce typu validace formulářů, animace (i když v dnešní době je již možné animovat přímo pomocí CSS3) nebo třeba provádění výpočtů na straně klienta pro odlehčení zátěže serveru.

Ne všude je však použití JavaScriptu na místě, neboť nejen že uživatel může do kódu zasáhnout, ale i může jeho user experience být ochuzena pomalejším načítáním a celkovým zpomalením stránek na slabších zařízeních.

3.7.1 jQuery

Webové stránky této knihovny uvádí, že „*jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling,*

animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers."[16]

Tato knihovna umožňuje minimalizaci (nejen ve smyslu minimalizace odstraněním mezer apod.) JavaScriptového kódu, který je v jeho nativní formě dlouhý [například pro výběr elementu s id="foo" a změnou jeho obsahu musíme použít dlouhý konstrukt document.getElementById("foo").innerHtml = "bar", kdežto v jQuery stačí napsat $("#foo").html("bar").]

Spousta vývojářů používá jQuery automaticky, což je vidět i na StackOverflow, kde již pár let téměř všechny otázky zabývající se JavaScriptem mají odpovědi primárně v jQuery nebo se jQuery objeví i v otázkách specificky se odkazujících na nepoužití této knihovny.

Jeden z důvodů nepoužití jQuery může být malé množství JavaScriptu. Načtení jQuery o něco zpomalí prvotní načtení stránky a tím pádem nemá v těchto případech stoprocentní opodstatnění.

Alternativa k jQuery mohou být i jiné JavaScriptové knihovny. V dnešní době je pro vývoj front-endové části aplikace populární REACT, který ale není předmětem této práce. Hojně využívaný pro vývoj webových aplikací je i Angular.

3.8 **Markdown**

„*Markdown is a lightweight markup language that you can use to add formatting elements to plaintext text documents.*"[17]

Tento jazyk je jeden z mých oblíbených. Umožňuje formátování textu bez rozptylování spisovatele. Je velice jednoduchý, neboť jeho syntaxe obsahuje jen pár konstruktů.[17, b]

Jelikož je veškeré formátování zprostředkováno pomocí prostých znaků a textových řetězců, je možné jej jednoduše využít i implementovat na většině míst. Taktéž není problém (ať už bezpečnostní či velikostní) s uložením tohoto formátování do databáze.

Markdown je ikonickou součástí formátování ve verzovacích serverech GitHub i GitLab (různé aplikace mají často vlastní tzv. flavor[17], to je typ syntaxe Markdownu), StackOverflow a FB Messenger jej taktéž používají.

3.9 **SEO**

Search Engine Optimization, přeloženo jako optimalizace pro vyhledávač, je snaha o zpřístupnění obsahu aplikace internetovým vyhledávačům (např. Google.) Zatímco vyhledávač se k veřejné stránce dostane téměř vždy, ne vždy zcela jistě pochopí význam jednotlivých položek na ní. Čím lépe vyhledávače rozpoznají obsah naší aplikace, tím spíše jej také doporučí cílovému uživateli.

Základy SEO spočívají ve validitě HTML a dalšího kódu, využití mikrodat a spoustě dalších prvků.

3.9.1 **Schema.org**

Mikrodata typu Schema.org obsahují bližší informace o struktuře a obsahu daného prvku na www stránce. Jsou čtena při procházení stránky roboty (web crawler) třeba při procesu

[b] Těmito „konstrukty" jsou myšleny například odstavce a nadpisy

indexování. Mohou být obsažena v markupu stránky (značkách HTML) nebo ve skriptových značkách, mají pak JSON formát.

Schema.org je slovník prvků, od nejobecnějších („*Thing*"), přes obecnější („*CreativeWork*"), po specializované pro nějaký účel („*Article*"). Každý prvek má určité parametry, třeba titulek nebo datum vytvoření. (Parametrem může být i celý další prvek.)

3.9.2 **OpenGraph**

Stránky OpenGraph uvádí, že „*Open Graph protokol umožňuje jakékoliv webové stránce stát se bohatým objektem v sociálním grafu. Tohoto je použito například na Facebooku, aby bylo umožněno jakékoliv stránce mít stejné funkce jako jakýkoliv jiný objekt na Facebooku.*"[14, c]

V praxi to znamená, že protokol OpenGraph je technologie nebo možná lépe řečeno styl zápisu metadat webové stránky, který jí dovolí zobrazit se například ve formě náhledu na sociálních sítích, při sdílení přes messengery (aplikace pro psaní zpráv přes internet typu Signal nebo WhatsApp) a další. (Z vlastní zkušenosti vím, že OpenGraph funguje ve Facebook Messengeru i v příspěvcích na Facebooku.)

Jak uvádí dokumentace Twitteru, funguje na něm podobný systém, který se nazývá Twitter Cards. Twitter uvádí, že „*Když procesor Twitterových karet hledá tagy na stránce, nejprve hledá vlastnosti specifické pro Twitter, a pokud nejsou přítomny, použije podporované Open Graph vlastnosti.*"[15, d] Tím pádem jsou kompatibilní.

Příklad zápisu titulku, který se zobrazí v náhledu:

„<meta property="og:title" content="The Rock" />"[14]

3.9.3 **Robots.txt, meta značka robots**

Roboti a crawleři běžně prochází webové stránky za různými účely. Někteří indexují obsah pro vyhledávače, jiní zase hledají kontaktní formuláře pro odesílání spamu.

Roboti se ale také často řídí instrukcemi vývojáře těchto stránek. Soubor robots.txt umístěný obvykle v kořenovém adresáři říká těmto artificiálním návštěvníkům kam smí a kam ne, popřípadě co mají indexovat a co ne. (Není žádoucí indexovat errorovou hlášku, ale domovskou stránku rozhodně ano.) Další důležitou instrukcí je, jestli mají roboti navštívit linky nacházející se na dané stránce. (Taktéž nemusí být žádoucí, aby robot prošel na stránku, například v obsahu článku, nacházející se na jiné, cizí doméně.)

Alternativou k tomuto souboru jsou metadata obsažená v meta značce s názvem robots.

Příklad:

<meta name="robots" content="index,follow">

3.9.4 **Sitemap**

Mapa stránek je obvykle XML soubor (může mít i jiný formát) obsahující seznam stránek obsažených ve webové aplikaci. Tento seznam využívají již zmínění roboti a crawleři, kteří podle nich mohou jednodušeji procházet webové stránky.

Tato mapa může obsahovat i prioritu dané stránky, například může být žádoucí, aby naše domovská stránka měla vyšší prioritu než náhodný článek.

---

[c] The Open Graph protocol enables any web page to become a rich object in a social graph. For instance, this is used on Facebook to allow any web page to have the same functionality as any other object on Facebook.

[d] When the Twitter card processor looks for tags on a page, it first checks for the Twitter-specific property, and if not present, falls back to the supported Open Graph property

Dále obvykle obsahuje informaci o tom jak často je stránka aktualizovaná. To může být měsíčně, týdně, nikdy, ale klidně i pokaždé. (Například u stránek, které jsou generovány s novým obsahem, třeba domovská stránka se články, při každém zobrazení.)

3.10 **SQL**

Dotazovací jazyk SQL slouží k práci s daty v databázi typu SQL. Je možné v něm psát jak jednoduché (SELECT, UPDATE, INSERT, DELETE), tak složitější příkazy. (Například pomocí klíčového slova JOIN, nebo rekurzivní CTE.)

3.10.1 **MariaDB**

Tento open-source SŘBD je hojně využívaný pro spoustu projektů. Kupříkladu je to i defaultní databáze u Windowsovského řešení lokálního vývoje XAMPP.

MariaDB Foundation uvádí, že je tento SŘBD garantován zůstat open-source „*MariaDB server je jeden z nejpopulárnějších databázových serverů na světě. Je vytvořen původními vývojáři MySQL a je garantováno, že zůstane open source. Mezi významné uživatele patří Wikipedia, WordPress.com a Google.*"[7, e] Zároveň mu přidává na kredibilitě i použití uvedenými technologickými giganty.

3.10.2 **Normalizace databáze**

Microsoft uvádí, že „*Normalizace je proces organizování dat v databázi. To zahrnuje tvoření tabulek a zakládání vazeb mezi těmito tabulkami podle pravidel určených zároveň k ochraně dat a ujištění o větší flexibilitě databáze odstraňováním redundance a nekonzistentních závislostí.*"[23, f]

Pokud by databáze nebyla normalizovaná, bude práce s ní jednodušší z hlediska psaní dotazů, protože mít stejná data na více místech znamená, že je získáme na více místech. Ale, na druhou stranu, máme-li je na více místech, může dojít k nekonzistenci dat.

Pokud by data, která mají být na více místech stejná, byla rozdílná, není snadné určit, která jsou ta správná. Normalizace předchází těmto problémům.

3.10.3 **Normální formy**

Pro normalizaci existují návody / poučky, které radí jak k ní přistupovat.

Microsoft uvádí, že první normální forma spočívá v identifikaci každé tabulky primárním klíčem.[23]

Dále říká, že druhá normální forma stojí na vytváření tabulek pro opakující se data a následném propojení těchto tabulek s původními pomocí cizích klíčů.[23]

Třetí normální forma podle Microsoftu vyžaduje eliminaci nesouvisejících sloupců z tabulek. Opět by měly být tato data ve vlastních tabulkách.[23]

[e] MariaDB Server is one of the most popular database servers in the world. It's made by the original developers of MySQL and guaranteed to stay open source. Notable users include Wikipedia, WordPress.com and Google.

[f] Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

3.10.4 **Databázový vrapper**

Pro práci s databází je možné mít třídu se sadou metod, ať už statickou nebo ne, která bude schopna s databází pracovat.[1, s. 7]

Čápka vychází z návodu na ItNetworku:

„*class MyDB{*
private $spojeni;
function __construct($host,$user,$pass,$name){
$options=array(
PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"
);
$this->spojeni= @new PDO("mysql:host=$host;dbname=$name", $user, $pass, $options); }
function query($query,$param=Array()){
$navrat=$this->spojeni->prepare($query);
$navrat->execute($param);
return $navrat->rowCount();
}
function queryOne($query,$param=Array()){
$navrat=$this->spojeni->prepare($query);
$navrat->execute($param);
return $navrat->fetch(PDO::FETCH_ASSOC);
}
function queryAll($query,$param=Array()){
$navrat=$this->spojeni->prepare($query);
$navrat->execute($param);
return $navrat->fetchAll(PDO::FETCH_ASSOC);
}}"[19]

Na první pohled bych možná sloučil funkce queryOne a queryAll do jedné, například pomocí přidaného booleánského parametru, neboť jsou obě operace výběr z databáze.

3.10.5 **MySQL Workbench**

Oracle uvádí, že „*MySQL Workbench umožňuje DBA, vývojáři, nebo datovému architektovi vizuálně designovat, modelovat, generovat a spravovat databáze. Toto obsahuje všechno co datový modelér potřebuje pro tvrobu komplexních ER modelů, forwardnímu a reverznímu vývoji...*"[28, g]

Toto grafické rozhraní pro vývoj databáze může být použito pro kompletní vývoj databáze. Na větší aplikaci by ale nejspíše bylo vhodnější najmout databázového inženýra, který by ji navrhl efektivněji.

Rozhraní je možné dobře upravovat a umožňuje zobrazení entitně-relačního diagramu v různých podobách, export jeho obrázku v různých formátech (.png, .svg) a hlavně již zmíněný forward engineering, což je vygenerování kompletního kódu databáze po pár kliknutích z naklikaného diagramu.

[g] MySQL Workbench enables a DBA, developer, or data architect to visually design, model, generate, and manage databases. It includes everything a data modeler needs for creating complex ER models, forward and reverse engineering...

3.10.6 **phpMyAdmin**

Standardně dostupný nástroj dodávaný s XAMPPem phpMyAdmin, je, jak říká dokumentace *„...zdarma softwarový nástroj napsaný v PHP, který je určen ke zvládnutí administrace MySQL nebo MariaDB databázového serveru. Můžete použít phpMyAdmin k vykonání většiny administračních úkonů, včetně tvorby databáze, spouštění dotazů, a přidávání uživatelských účtů.*"[22, h]

Jeho rozhraní pro vývoj databáze je poněkud chudší než MySQL Workbench, avšak z vlastní zkušenosti po vyzkoušení jsem přesvědčen, že pro administraci menšího webu je více než dostačující.

3.10.7 **Rekurzivní CTE**

Jako vhodnou funkci SQL pro mou práci (využiji ji u výpočtu článků v kategoriích) jsem vyhodnotil rekurzivní CTE, které umožňují komplexní rekurzivní dotaz v SQL.[12]

Princip jejího fungování stojí na výběru určitých dat dle parametrů a následném rekurzivním dotazování pomocí vybraných hodnot a jejich spojení pomocí klíčového slova UNION.

Existuje i nerekurzivní CTE, což je ve své podstatě jen na místě vytvořený dočasný pohled (VIEW) bez vložené rekurze.[12]

3.10.8 **Full-textový index pro vyhledávání**

Snaidero uvádí, že *„Full-textový index je speciální typ indexu, který poskytuje indexový přístup pro full-textové dotazy."*[20, i] Podporuje jej i MariaDb.

Tento typ indexu je tím pádem vhodný tam, kde uživatel zadává výraz, který nemusí být stoprocentně shodný s obsahem databáze, ale i přesto chceme najít shodu, alespoň částečnou či nejbližší. (Z podstaty fungování full-text vyhledávání.)

3.11 **Nutné legální záležitosti**

3.11.1 **Cookie Banner**

Každá webová stránka používající cookies musí mít cookie banner, kde informuje uživatele o tom, že je využívá a dává mu jakousi možnost kontroly nad nimi. (Na většině stránek je vidět možnost akceptování všech cookies, nebo jen nezbytných - to jsou právě třeba dříve zmíněné SESSID.)

Cookies jsou ukládána na zařízení uživatele, tudíž dává smysl, aby o nich uživatel věděl. Na druhou stranu, téměř každá stránka je používá, takže jsou tyto bannery uživatel často přehlédne (možná i kvůli tzv. banner blindness fenoménu), nebo je odklikne jen aby od nich měl pokoj.

[h] ...a free software tool written in PHP that is intended to handle the administration of a MySQL or MariaDB database server. You can use phpMyAdmin to perform most administration tasks, including creating a database, running queries, and adding user accounts.

[i] A full-text index is a special type of index that provides index access for full-text queries.

3.11.2 **GDPR**

Evropské nařízení o ochraně osobních údajů 2016/679 lehce komplikuje vývoj všech webových aplikací. Spočívá v tom, že by tvůrce / vlastník stránek měl chránit data v maximální rozumné možné míře.

Toho může být dosaženo například zabezpečením databáze, hashováním hesel, ukládáním jen dat, které jsou opravdu potřebná a podobně.

3.11.3 **Legální dokumenty**

Každá aplikace by taktéž měla mít informace o ochraně osobních dat (pokud je shromažďuje), obchodní podmínky či podmínky užití, ideálně i vyloučení odpovědnosti (pro předcházení případným soudním sporům).

Díky již zmíněnému cookie banneru by též měla obsahovat cookie policy (na který se v cookie bannerech obvykle uvádí odkaz), dokument hovořící o přesném využití cookies. (Obzvláště při užití pro trackování, sledování, uživatele.)

3.12 **Zprovoznění aplikace v reálném prostředí**

„*Jakmile dokončíte své umělecké dílo a cítíte, že už ho můžete vypustit do světa, musíte přenést své stránky na server, který bude poskytovat hosting vašim stránkám, a tím je zpřístupnit lidem.*"[2, s. 387]

3.12.1 **Doména**

Pro přístup k webové aplikaci uživatel použije doménu. Tuto doménu musí tvůrce (nebo provozovatel) webových stránek zaplatit.[2, s. 387] Obvykle se jedná o částky v řádech stovek korun, ale odvíjí se to primárně od koncovky, někdy i od délky názvu domény. (Příliš krátké nebo dlouhé názvy mohou stát více než relativně normální délky)

Národní domény typu .cz, stejně tak obyčejné domény jako .com, .eu nebývají dražší než pár set za rok. Barvitější domény typu .tech, .us nebo .design už mohou vyjít i na tisíce, ne-li více.

3.12.2 **Hostitel**

Při hledání hostitele je dobré koukat na všechny jeho parametry. Z osobní zkušenosti vím, že ne každý hosting je si roven.[2, s. 388]

Existují i hostingové služby zdarma (dříve jsem míval osobní web na Orgfree, které umožňovalo přístup přes FTPs, databázi a všechny ostatní potřebné parametry zdarma, po nějaké době však nastolili režim, ve kterém byly stránky vyplé - několik hodin denně.)

V závislosti na tom, co od serveru (nebo jestli třeba nepotřebujeme celý server, což je podstatně dražší) požadujeme se i odvíjí cena. Většina mých hostingů stála od 20 do 50 korun za měsíc, což jsou částky zanedbatelné, ale při větším množství vedených webů se mohou nastřádat a je možná lepší volit balíčky pro více webů. V součtu s cenou domény může jeden web vyjít kupříkladu i na 150 korun měsíčně a více.

3.12.3 **Přesun souborů aplikace na server**

K přesunu mívají hostingové společnosti (po sjednání služeb) buď vlastní rozhraní, kam se soubory nahrají nebo zpřístupní přes protokol FTP či FTPs adresář na serveru.[2, s. 390]

4 Vlastní práce

4.1 Návrh uživatelského rozhraní

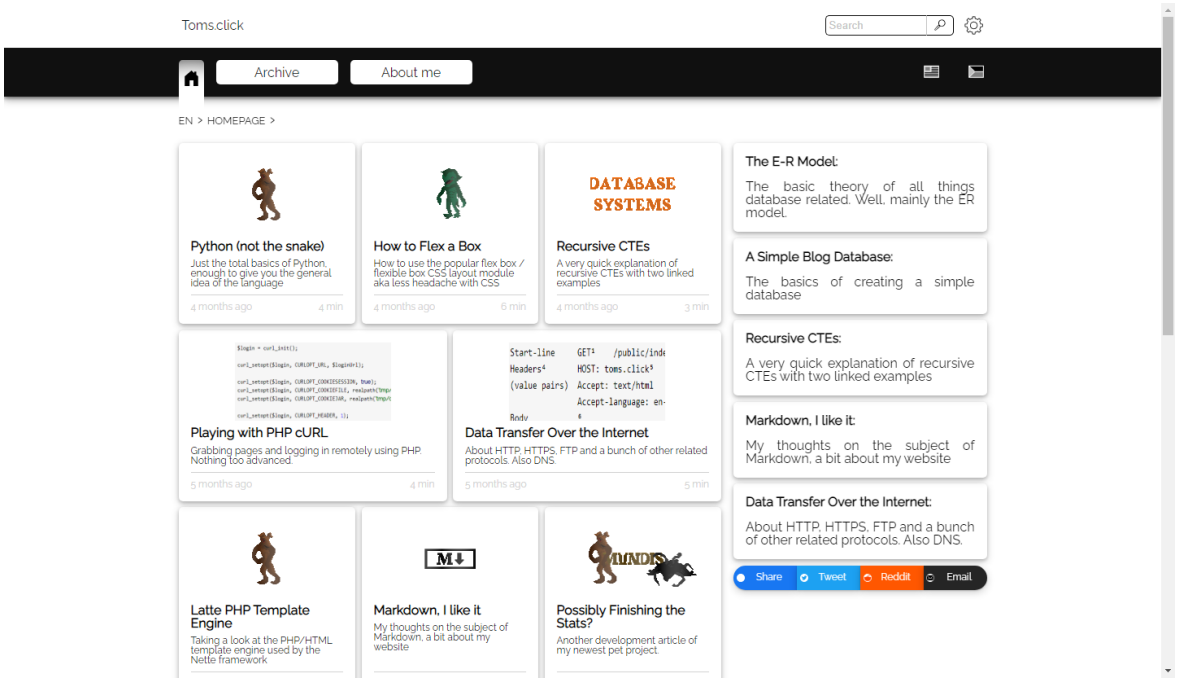
Jako celkové rozložení stránky jsem zvolil maximální šířku stránky 1024 pixelů. Důvodem je článkový obsah systému. Příliš široká stránka by se špatně četla a uživatel by mohl být zahrnut příliš informacemi. V CSS byla použita primárně technologii flexbox.

4.1.1 Hlavička stránky

Mimo obsah loga a navigačních prvků je zde zajímavá funkce menu, která se odemkne po odstranění cookie banneru. Je zde možné odebrat souhlas s cookies a přejít do kontaktního formuláře. Zobrazí se zde též aktivní stránka (její titulek / ikona v případě hlavní stránky) se zobrazí.

4.1.2 Hlavní stránka

Jedna z nejdůležitějších částí každé webové aplikace je úvodní stránka. Na hlavní stránce je obsažen výpis nejnovějších článků, dále vybraných článků (featured) a call to action pro sdílení.

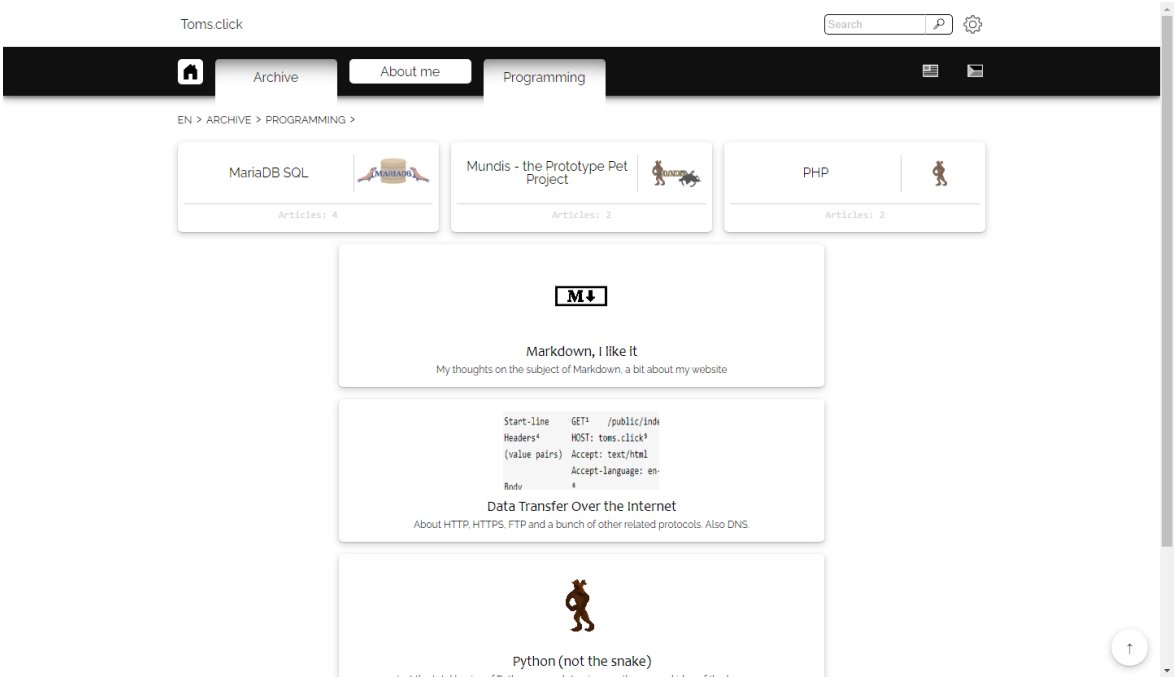


Obrázek č. 1: Domovská stránka

4.1.3 Archiv

Pro existenci článků je nezbytná existence archivu, kde budou dohledatelné. U každé sekce/rubriky jsem se rozhodl vypisovat jak počet rubrik pod ní obsažených a článků pod ní obsažených (viz. kapitola Rekurzivní CTE pro součet článků v rubrice). Tato funkce dá uživateli představu o obsáhlosti sekcí.

Pravděpodobně přehlédnutou funkcí je zobrazení prvního článku v sekci již v zobrazení sekce (vpravo nahoře), jedná se o prokliknutelný link umožňující rychlejší navigaci pomocí méně kliknutí, pokud o ní uživatel ví.



Obrázek č. 2: Archiv

4.1.4 Článek

Každý článek musí obsahovat titulek, datum vytvoření (obzvláště pokud budu chtít psát o technologiích či programování, pro ujištění uživatele o aktualitě / neaktualitě článku) a obsah článku.

Datum se standardně nachází nerušivě v pravé horní části článku. Uvažoval jsem o jeho umístění pod titulkem, ale rozhodl jsem se, že takto to vypadá lépe i z hlediska formátování textu.

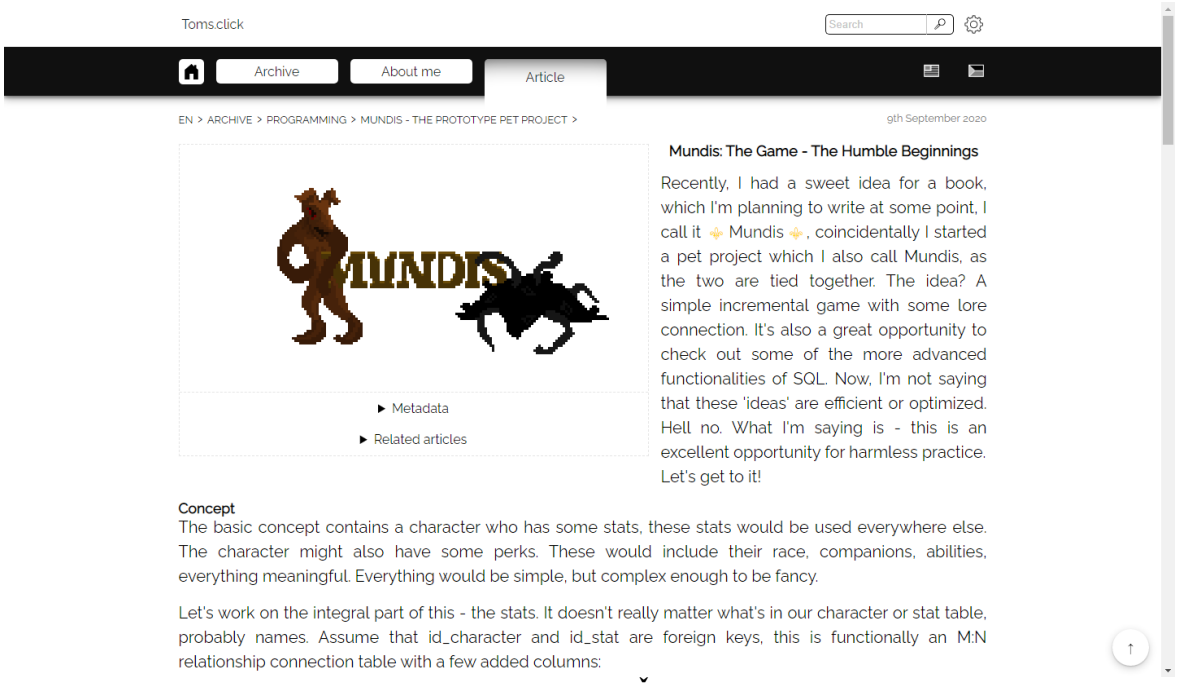
Na bázi neformálního UI testu jsem skryl metadata článku do levé části stránky pod obrázek k rozkliknutí. Nepovažuji je za dostatečně důležité a mohly by rozptylovat uživatele od obsahu.

Obsah článku je v databázi uložen jako text s formátováním typu markdown a při zobrazení přeložen do HTML pomocí cebeho markdownnového parseru.

Implementoval jsem si několik značek, které mi přišly jako zajímavé. Konkrétně rozkliknutelný text s definicemi, text nad textem, barevný text a několik dalších.

Přidal jsem i funkci připnutí audio nahrávky ke článku (na domovské stránce jsou označeny notou, což při menším počtu namluvených článků může být hůře viditelné).

Pod článkem se nachází call to action ve formě několika tlačítek pro sdílení. Pakliže existují předchozí či další články v sérii (v databázi pole next tabulky článek), zobrazí se též zde. Následoval jsem běžný princip procházení stránky zleva nahoře dolů doprava.



Obrázek č. 3: Článek

4.1.5 O mně

Tato aplikace měla v původní myšlence zároveň sloužit i jako můj životopis pro snadnější vstup na pracovní trh. Sekce o mně by tudíž měla obsahovat nejen nějakou informaci o mně, ale i dovednosti a časem i pracovní zkušenosti.

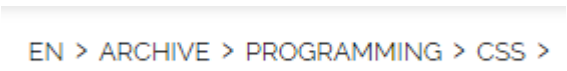
Po pár měsících jsem přidal i namluvené krátké texty ve všech jazycích, které ovládám na uznané úrovni A1 a výše. Nikde jsem to neviděl, čili se jedná o experimentální funkci a z mé strany snahu o prokázání dovednosti jazyka již na mých stránkách.

Následně jsem do této sekce připnul i odkaz (se zobrazením reputace) na můj účet StackOverflow a oblíbenou hlášku.

4.1.6 Drobečková navigace (breadcrumbs)

Celá aplikace obsahuje drobečkovou navigaci pro snadný průchod uživatele. Uživatel si bude na každé stránce vědom kde se právě nachází.

Jako první komponent navigace jsem zvolil lokalizaci (jazyk CS, nebo EN). Opět se jedná o krok se snahou usnadnění orientace uživateli.

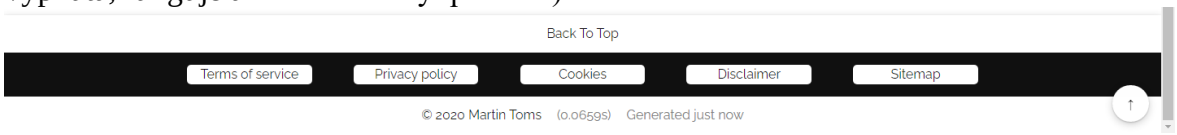


Obrázek č. 4: Drobečková navigace

4.1.7 Patička stránky a zobrazení legálních dokumentů

Žádná větší webová aplikace se v dnešní době neobejde bez legálních dokumentů typu zásady ochrany osobních dat a zásady cookies. Jelikož nepředpokládám časté změny, implementoval jsem tyto stránky jako natvrdo zapsané do aplikace. Výhodou této implementace je potenciálně vyšší rychlost načítání, i když u tohoto množství textu je to zanedbatelná výhoda.

V patičce se také nachází informace o rychlosti načtení stránky, popřípadě zda byla stránka vygenerována poprvé, či načtena z cache (ačkoliv je funkce cachování stránek vypnutá, funguje a tudíž může být použita.)

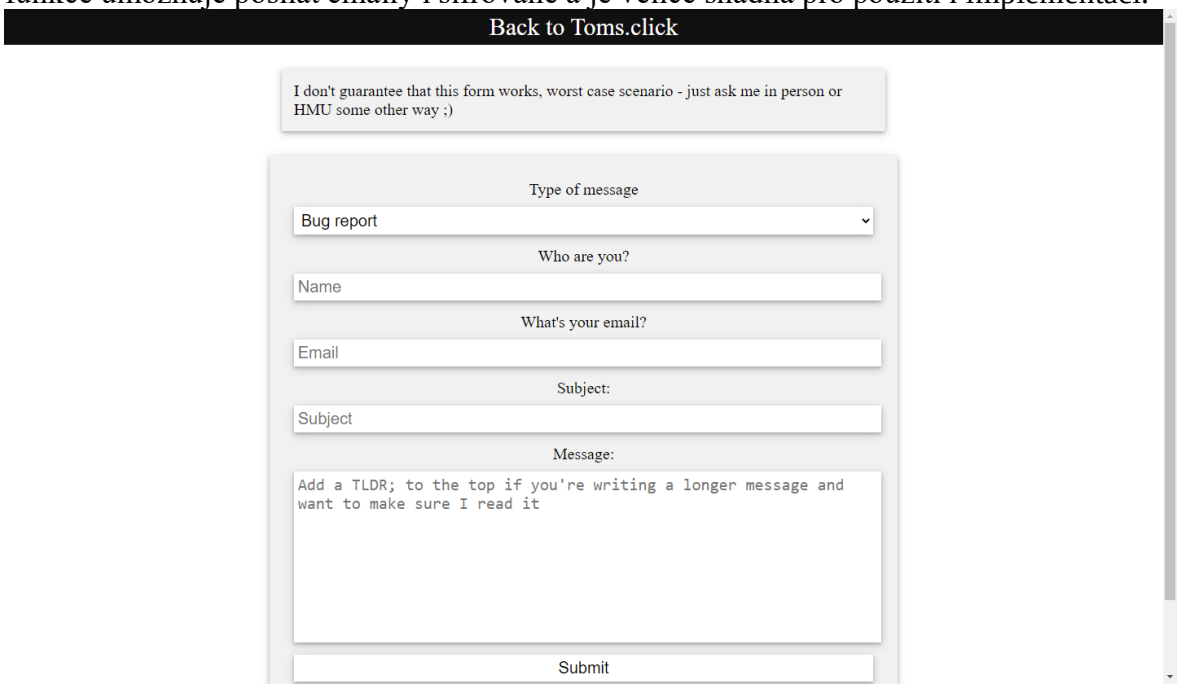


Obrázek č. 5: Patička stránky

4.1.8 **Kontaktní formulář**

Pomocí funkcí PHP je tento formulář schopen odeslat email s obsahem formuláře. Jako drobnou nevýhodu mé implementace vidím nedostatečnou ochranu proti spamu, což by se dalo opravit například přidáním captcha. Nejedná se o nutnou součást aplikace, tudíž by bylo možné ho kdykoliv vypnout bez ovlivnění fungování celku. Funguje na subdoméně contact.toms.click.

Jako výhodu vidím simplicitu fungování a možnost jednoduché úpravy v případě nutnosti. Pro samotné odesílání emailu jsem použil funkci PHP mb_send_mail(). Tato funkce umožňuje posílat emaily i šifrovaně a je velice snadná pro použití i implementaci.



Obrázek č. 6: Kontaktní formulář

4.1.9 **Obrázky**

Většinu náhledových obrázků jsem si nakreslil sám v grafickém programu Aseprite pixel-artovým stylem. Pakliže se někde objevují čáry přes obrázek při zobrazení jednoho z nich, pak se nejedná o vadu aplikace, nýbrž o vedlejší efekt exportu z Aseprite do svg formátu. Vychází z podstaty, že jednotlivé pixely jsou čtverci v svg, ale při změně velikosti se při špatném poměru ukážou okraje těchto čtverců.

4.1.10 **Zpět nahoru (Back to top)**

Tlačítko back to top se hodí pro umožnění uživateli rychle se vrátit k navigačním prvkům. Alternativou by mohlo být použití tzv. sticky hlavičky, což by ale zabralo určité místo na obrazovce uživatele a mohlo by mírně zhoršit celkový dojem na mobilním zařízení.

Toto tlačítko se obvykle nachází ve spodní pravé části obrazovky, typicky v kulaté formě, proto jsem jej sem umístil také.

4.1.11 **Ikony (iconfinder a licence)**

Pro ikony rozhraní jsem se rozhodl použít službu iconfinder, které obsahují i kvalitní ikony zdarma pro komerční využití bez nutnosti linkbacku (odkazu na zdroj).

U každé ikony je pak na iconfinderu k dispozici informace o licenci (některé podléhají např. MIT) a link na stažení .png nebo .svg souborů. Lze zde najít i celé sady podobně vypadajících ikon pro konzistentně vypadající rozhraní.

4.2 Grafický design

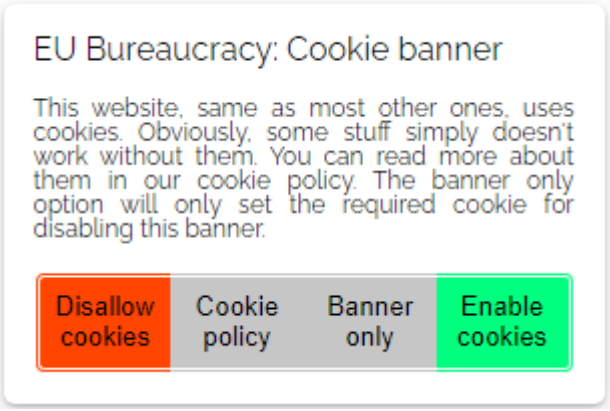
V rámci celkového vzhledu jsem se snažil cílit na relativní simplicitu. Rozhraní obsahuje převážně černou a bílou barvu, popřípadě výraznější barvy u tlačítek dle funkcí (zelená = provedení funkce, červená = zrušit / storno).

4.2.1 **Cookie banner**

Nedílnou součástí jakékoliv aplikace používající cookies je již zmíněný cookie banner. Předpokládám, že se zobrazí úplně každému uživateli alespoň jednou, proto jsem se jej snažil designovat s touto úvahou v mysli.

Jako hlavní jsem považoval umístění banneru. Souhlas s cookies pro mě není dostatečně důležitý pro obtěžování uživatele, proto jsem se rozhodl jej umístit dolů do pravého rohu. Výhoda tohoto umístění je to, že nepřekáží (alespoň na desktopových zařízeních) v ovládání celé aplikace.

Tlačítka banneru jsem vzhledově spojil do jednoho, pouze s odlišením barev dle funkce. Tlačítko nepovolení cookies pouze banner dočasně skryje (do dalšího obnovení stránky, nebo do přechodu na jinou stránku), neboť nemůže nastavit cookie pro jeho semi-permanentní zrušení.



Obrázek č. 7: Cookie banner

4.2.2 **Animace**

S animacemi jsem byl opatrnější, jelikož přílišné hýbání prvků na stránce může uživatele rozptylovat (osobně moc animace poblíž textu, který se snažím číst, nemám rád).

4.2.3 **Zvýrazňování kódu v code blocích**

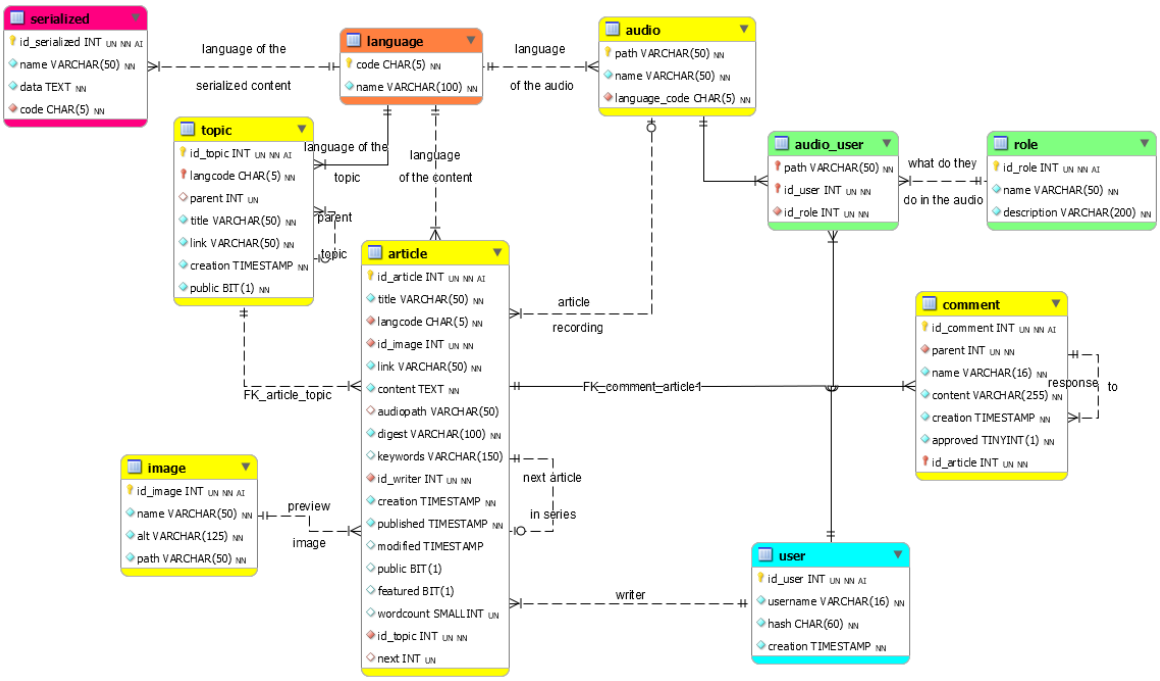
Jelikož předpokládám, že v mnoha z mých článků bude obsažen i nějaký kód, přišlo mi vhodné implementovat i zvýrazňování kódu.

Pro tento účel jsem použil knihovnu highlight.js psanou v JavaScriptu, která je schopna podle třídy elementu na stránce aplikovat zvýraznění syntaxe daného jazyka.

Jako hlavní výhodu vidím možnost výběru konkrétních jazyků, které bude highlight.js podporovat a také široké množství šablon (stylů zvýraznění, barvy, písmo apod.), které jsou k dispozici.

Obrázek č. 8: Příklad stránky se zvýrazněným kódem

4.3 Návrh databáze



Obrázek č. 9: Schéma databáze

Databázi jsem navrhoval v nástroji MySQL Workbench. Její prvotní podoba se podstatně lišila od nynější, například v systému lokalizace, neboť jsem původně předpokládal možnost překládání všech článků přes jedno id. Tato funkce se ukázala jako zbytečná a kompletně jsem databázi předělal.

Centrálním prvkem databáze je article, článek, který obsahuje veškerý text článku. Články mohou náležet do rubriky (topic) a mají obrázek, který se zobrazí jako náhled na domovské stránce, v archivu a v horní části zobrazeného článku.

Jelikož se předpokládá dvojjazyčnost, je zde i language, jazyk, který stanovuje jazyk obsahu.

Audio je komplexnější než je momentálně nutné, ale na druhou stranu to nepřekáží dočasnému simiplifikovanějšímu používání bez všech prvků. Jedná se spíše o předpoklad rozšíření audio funkce v aplikaci do budoucna.

Tabulku serialized jsem přidal pro zpětnou kompatibilitu v průběhu vývoje, neboť původní aplikace neobsahovala překlady v kódu aplikace, ale v serializované formě v databázi. To se s průběhem času ukázalo jako obtížné na obsluhu přímo přes phpMyAdmin a nakonec jsem od tohoto způsobu upustil v prospěch překladu přímo v aplikaci.

Tuto tabulku bych v příští iteraci větších úprav databáze / úprav aplikace odstranil, nebo bych ji zužitkoval pro implementaci nových funkcionalit, které by potřebovaly data ukládat do databáze. (Na místě mě napadá třeba editor textu, který by si sem mohl ukládat neuložená data, ačkoliv by nejspíše bylo lepší užití databáze na straně klienta.)

Uživatel je uchováván s informací o hashi pro přihlašování. Nepředpokládal jsem v průběhu vývoje více než jednoho uživatele (redaktora).

4.4 Pohledy pro výběr dat

Většina SELECT dotazů směřovaných na databázi prochází skrze jeden ze tří hlavních pohledů. Pohledy jsem aplikoval proto, abych v průběhu vývoje mohl popřípadě upravovat databázi. V tom případě by stačilo jen upravit pole v pohledech.

Těmito pohledy jsou: „*publicArticles*", „*articles*" a „*hiddenArticles*". Po přidání pole published (datum publikace) jsem taktéž přidal pohled „*unpublishedArticles*", avšak ten je spíše pro mou vlastní orientaci v databázi.

Jak již jejich názvy napovídají, jedná se o členění všech článků v databázi na bázi toho, jestli mají být viditelné uživatelům. Rozdíl mezi public articles a articles je prostý - public articles jsou ve veřejné rubrice, zatímco articles jsou veřejné, avšak jejich rubrika již nemusí být. Toto umožňuje sdílení určitého obsahu jen přes link a nezobrazení části obsahu v archivu.

Samotné pohledy jsou relativně komplexní, avšak myslím, že jsou dostatečně pochopitelné.[j]

CREATE OR REPLACE VIEW publicArticles AS
SELECT
a.id_article, l.code AS 'id_language', t.id_topic, a.title, a.link, i.path AS 'src', i.alt AS 'name', a.content, a.digest, a.keywords, a.creation, a.modified, u.id_user, u.username, next, a.wordcount, a.audiopath AS audio
FROM article a
JOIN image i ON a.id_image = i.id_image
JOIN topic t ON a.id_topic = t.id_topic
JOIN language l ON a.langcode = l.code
JOIN user u ON a.id_writer = u.id_user
WHERE a.public = 1 AND t.public = 1 AND published < CURRENT_TIMESTAMP;

Například u tohoto pohledu je klíčová část "*WHERE a.public = 1 AND t.public = 1 AND published < CURRENT_TIMESTAMP*". Vyberou se tedy jen články, které jsou publikovány, označené jako veřejné a navíc jsou ve veřejné rubrice.

4.5 Implementace MVC struktury

Architekturu aplikace jsem implementoval přesně podle návrhu v teoretické části. Každá stránka[k] má vlastní Controller. Výpočty jsou prováděny v Modelových funkcích a veškeré šablony jsou pohledy.

Jelikož jsem Latte engine přidal v pozdní fázi vývoje s primárním účelem simplifikace velkého množství šablon, neboť se v nich již obtížně orientovalo, část aplikace stále používá pohledy bez Latte. (Konkrétně je používá hlavní výstup stránky kolem pohledů.)

Na jednu stranu to není ideální situace, jelikož se pohledy načítají dvěma způsoby. Na druhou stranu to dokazuje možnost rychlé a jednoduché implementace knihovny v PHP pomocí Composeru i do pouhé části aplikace.

4.5.1 Konkrétní implementace Latte

Pro samotný výstup je použita metoda render, tato metoda si vezme jako parametry cestu k latte šabloně a pole (popřípadě jiný typ, dle dokumentace) s daty:

protected function latteView() {

[j] Uvádím jen jeden, jelikož se liší pouze parametry v koncové části.
[k] Například domovská stránka, článek, archiv

```
$latte = new Engine();
try {
$latte->render(__DIR__ . '/../View/' . $this->view . '.latte', $this->latteParameters);
} catch (\RuntimeException $exception) {
  //File not found
  echo '[Error] Work in progress.';
}}
```

V metodě výpisu pohledu se pak použije tato metoda pro všechny pohledy, které nejsou hlavním:

```
public function view()
{
  $this->processParametersForView();
  if ($this->view === 'main/base') {
        extract($this->data);
        include(__DIR__ . '/../View/' . $this->view . '.php');
        return;
  }
  $this->latteView();
}
```

4.6 Konfigurace aplikace (config.php)

Soubor pro úpravu určitých hodnot jsem provedl pomocí globálních proměnných. Globální proměnné s sebou přináší výhodu dostupnosti na všech místech aplikace.

Řeším zde verzi souborů [define('FILE_VERSION', 1.512);] a přesměrování českých verzí stránek (např. /domovska-stranka) na příslušný controller (HomepageController). Toto je nutné z důvodu fungování překladu URI na controller v této aplikaci.

```
define('ROUTING_TABLE', [
  'Domovska-strankaController' => 'HomepageController',
  'ClanekController' => 'ArticleController',
  'O-nasController' => 'AboutController',
  'ArchivController' => 'ArchiveController',
  'RecepceController' => 'HomepageController',
  'UzivatelController' => 'UserController'
]);
```

Dále zde řeším momentálně vypnutou funkci cachování, dobu, po jakou se má cache držet na serveru a také které stránky se mají cachovat. Zapnout ji lze prostým přehozením hodnoty false na hodnotu true:

```
define('CACHE_ENABLED', false);
```

Jako poslední se zde nachází, možná nejdůležitější část, připojení k databázi a údaje pro připojení v proměnné $dbParameters.

Tento soubor je připnut k souboru index.php pomocí instrukce require_once "*../config.php*"; a je jedním ze dvou prvků (druhým jest autoload.php), které nejsou načteny automatickým načítáním tříd. Veškeré ostatní třídy již autoloading a namespaces využívají.

4.7 Implementace .htaccess

Nad souborem .htaccess jsem strávil delší dobu než jsem čekal, ačkoliv jeho syntaxe není až tak složitá. Zjistil jsem, že je v něm možné implementovat nejen přesměrování a lépe vypadající URL (například odstranění koncovky .php), ale i kompresi a cachování souborů.

Pro rychlejší načítání stránky je vhodné některé prvky, jako obrázky, cachovat a odesílání komprimovaných souborů trvá kratší dobu. Řeším zde i přesměrování z http na https variantu stránky.

4.7.1 Chybové stránky

Pakliže stránka vyhodí chybu, je přesměrována na chybové stránky podle následujícího předpisu:

ErrorDocument kód URL

Jako zajímavou chybu, na kterou jsem v průběhu vývoje narazil uvedu situaci, kdy sama chybová stránka odeslala chybový kód a program se zacyklil, neboť se neustále přesměrovával na sebe a ve výsledku přešel na jinou chybovou hlášku.

4.7.2 Komprese

Pro kompresi jsem využil modul gzip.c, bohužel se mi ho nepodařilo aplikovat na obrázky typu SVG a ICO. Jako řešení jsem vybral modul deflate.c, který si již s těmito formáty poradil bez problémů.

Po implementaci komprese bylo načítání celé aplikace znatelně rychlejší, nejen v místech se spoustou obrázků, ale všude, neboť se komprimují i soubory CSS a JavaScriptu.

```
<IfModule mod_gzip.c>
mod_gzip_on Yes
mod_gzip_dechunk Yes
mod_gzip_item_include file .(html?|txt|css|js|php|pl|svg)$
mod_gzip_item_include handler ^cgi-script$
mod_gzip_item_include mime ^text/.*
mod_gzip_item_include mime ^application/x-javascript.*
mod_gzip_item_exclude mime ^image/.*
mod_gzip_item_exclude rsphead
^Content-Encoding:.*gzip.*
</IfModule>
<IfModule mod_deflate.c>
AddType image/svg+xml .svg
AddOutputFilterByType DEFLATE image/svg+xml
AddType image/x-icon .ico
AddOutputFilterByType DEFLATE image/x-icon
</IfModule>
```

4.8 Implementace lokalizace

Původní plán tohoto systému obsahoval jen jeden jazyk, avšak po zvážení jsem se rozhodl implementovat i druhý jazyk. Protože jediným předpokládaným redaktorem jsem já, obsáhl jsem jen jazyky, kterými jsem schopen hovořit plynně.

Informaci o jazyku dané stránky jsem umístil do drobečkové navigace a také do URL.

Při implementaci jsem postupoval s myšlenkou pouze dvou jazyků po celou dobu existence aplikace (a to Čeština a Angličtina). Z hlediska rozšíření a dalšího vývoje na straně lokalizace by to mohlo způsobit drobné problémy (nutnost refactoringu a přepsání některých funkcionalit) a dále nutnost reevaluace použití tlačítek vlajek pro změnu jazyka. (Možná by byl na místě drop down list při obsažení více než 3 jazyků, popřípadě umístění dole v patičce stránky.)

Na druhou stranu, můj způsob lokalizace mi umožnil jednoduchou správu jazyků i pomocí prostých návratových výrazů obsahující ternární výrazy.

4.8.1 **Locale Access Method (LocaleAM)**

Jelikož k lokalizaci přistupuji přes session cookie $_SESSION[‘localization’], vytvořil jsem si statickou třídu, která mi tento zápis umožní zkrátit na LocaleAm.

Metodami této přístupové třídy jsou zobrazení kódu jazyka (např. en-us), nastavení lokalizace a získání formátu dat daného jazyka.

4.8.2 **Localization Dictionary**

Slovníková třída obsahuje veškeré překlady uživatelského rozhraní. Limitována je momentálně na dva jazyky, je implementována formou ternárních výrazů, kde pokud je nastavena anglická lokalizace, vypíšou se Anglické popisky, jinak České.

Tento systém je možné jednoduše upravit tak, aby podporoval i více jazyků.

4.8.3 **Localization Container**

Tato třída je samotnou lokalizací obsaženou v $_SESSION[‘localization’]. Jejími primárními prvky je kód jazyka a formát dat. (Jelikož český formát je 1. ledna, zatímco anglický je 1st January apod.)

Její metody též řeší vypis data publikace článku[1] na hlavní stránce.

4.8.4 **Implementace v prvotním controlleru**

Pro snadný překlad celých stránek jsem zvolil metodu nejmenšího odporu. Fungování této metody spočívá v nalezení překladů dle zadaného indexu v LocalizationDictionary() a následném extrahování těchto dat do proměnných při zobrazování pohledu dané stránky z $this->data.

```
protected function localize($item) {
  $ld = new LocalizationDictionary();
  if (method_exists($ld, $item)) {
        //entire page
        $this->data['localizationData'] = $ld->$item();
        $this->latteParameters['localizationData'] = $ld->$item();
        return true;
  } else {
        //a piece of page
        return $ld->searchAndTranslate($item);
  }
}
```

[1] Před x měsíci apod.

Použití je následující: $this->localize('about');

Metoda $this->localize se dá použít i na překlad jednotlivých slov/částí, které jsou zvlášť uchovány v LocalizationDictionary. Tohoto je využito například v metodě breadcrumbs(), která se taktéž nachází v prvotním controlleru.

4.8.5 **Volba jazyka dle url, nebo prohlížeče**

Při příchodu na stránku se RouterController rozhoduje o použití jazyka. Pokud je v URL obsažen jazyk (například en-us), použije tento jazyk, za předpokladu, že je v souladu s povolenými / předpokládanými. (To znamená, že se vytvoří v $_SESSION[‘localization’] instance LocalizationContainer:

```
private function useUrlLanguage($check) {
  $accepting = ['en-us', 'cs-cz'];
  $code = in_array($check, $accepting) ? $check : null;
  if ($code === null)
        $this->useBrowserLanguage();
        else {
  $_SESSION['localization'] = new LocalizationContainer($code);
}}
```

Pakliže však není jazyk k dispozici v url, což může být třeba při příchodu přímo k doméně[m], pak si RouterController zvolí jazyk podle jazyku prohlížeče.

To tento controller udělá podle obsahu serverové globální proměnné $_SERVER[‘HTTP_ACCEPT_LANGUAGE’], která bude obsahovat kód jazyka, ve kterém by se stránka měla zobrazit. Pokud neobsahuje Češtinu, pak se použije defaultně angličtina.

```
private function useBrowserLanguage() {
  $lang = strtolower(substr( $_SERVER['HTTP_ACCEPT_LANGUAGE'], 0, 2));
  if ($lang === 'cs')
        $code = 'cs-cz';
  else { $code = 'en-us'; }
        $_SESSION['localization'] = new LocalizationContainer($code);
}
```

4.9 **Implementace cachování stránek**

Tuto funkci jsem se rozhodl implementovat pro zrychlení načítání stránek, ale kvůli chybám a potenciálním bezpečnostním trhlinám jsem se jej následně rozhodl deaktivovat.

Primární nedostatek mé implementace spočívá v zacachování cookie banneru a následném zobrazení i uživatelům, kteří již s cookies souhlasili. Jelikož výstup cookie banneru zpracovávám přes serverový jazyk, možností by bylo cachovat i PHP kód pro jeho generování, avšak to se mi nezdálo jako dobrý nápad.

Po zvážení jsem se rozhodl tuto funkci ponechat v programu pro případ budoucího vývoje, ale nechat jej ve vyplém stavu na serveru, ačkoliv se při zaplém stavu podstatně zrychlilo načítání stránek (před refactoringem až o desetinásobek, což mě vedlo právě k tomuto refactoringu a odhalení problému v jedné z mých tříd).

[m] Míněno jako název domény bez parametrů: https://www.toms.click

4.10 **Rekurzivní CTE pro součet článků v rubrice**

Jsem toho názoru, že rekurze v jakékoliv podobě je zajímavá. Tento příkaz je jeden ze složitějších SQL dotazů nacházejících se ve výsledné aplikaci.

Jeho fungování spočívá ve vybrání počtu článků v dané sekci, včetně těch článků, které jsou v sekcích pod sekcí, kterou referencuje otazník v příkazu.

Nejprve se vytvoří rekurzivní CTE (princip viz. teoretická část) a následně se v něm prostým SELECT příkazem vybere počet článků.

```
WITH RECURSIVE articlesUnderTopic AS (
  SELECT * FROM topic WHERE id_topic = ?
  UNION
  SELECT children.* FROM topic AS children,
  articlesUnderTopic AS thisWholeThing
  WHERE children.parent = thisWholeThing.id_topic
) SELECT COUNT(*) AS articleCount
FROM article
WHERE id_topic IN (SELECT id_topic FROM articlesUnderTopic);
```

4.11 **OpenGraph Protocol**

Pro zápis mikrodata a hlavičky celkově používám metodu uvnitř hlavního controlleru setHeaders().

Tato metoda jako parametry bere titulek stránky, klíčová slova, popisek stránky, instrukce pro roboty a meta značky. Právě do těchto meta značek se může vepsat pole s metadaty OG.

Příklad použití pro stárnku /about:

```
$this->setHeaders('Martin Toms', 'Martin Toms, toms.click', $this->data['localizationData']['about-me'], 'all', [
  ['og:description', $this->data['localizationData']['about-me']],
  ['og:url', 'https://www.toms.click' . $_SERVER['REQUEST_URI']],
  ['og:type', 'website'],
  ['og:site_name', 'Toms.click'],
  ['og:title', 'Martin Toms | Toms.click'],
  ['og:image', 'https://www.toms.click/src/View/image/anubite-preview.png'],
  ['og:locale', LocaleAM::code()]
]);
```

4.12 **Implementace vyhledávání**

Jejím cílem bylo vyzkoušet si napsat vlastní menší aplikační rozhraní na bázi dotazu na stránku a následné odpovědi ve formátu JSON.

Zvažoval jsem podobnou implementaci i v ostatních částech aplikace, například pro zobrazení článků. Od této myšlenky jsem odstoupil, neboť jsem již byl příliš daleko ve fázi vývoje a takováto změna by byla relativně monumentální, obzvláště s přihlédnutím k faktu, že tyto funkce již byly kompletní a funkční.

Primární výhoda tohoto "*API*" řešení je možnost získání dat odkudkoliv, i bez přímého přístupu k databázi v jiné aplikaci. Mohlo by tudíž být využito pokud bych zvolil styl vývoje co zařízení.

Po určitou dobu jsem toto API na serveru používat k vyhledávání z localhostu při vývoji, jelikož jsem neměl nastavenou místní subdoménu.

Samotný vyhledávací formulář (pole vyhledávání) funguje na AJAX principu okamžitého načtení bez obnovení stránky.

4.12.1 **SQL příkaz pro vyhledávací funkci**

K samotnému vyhledávání jsem použil rekurzivní CTE (viz. teoretická východiska), konkrétní příkaz je toto:

```
WITH RECURSIVE articlesUnderTopic AS (
  SELECT * FROM topic WHERE id_topic = ?
  UNION
  SELECT children.* FROM topic AS children,
  articlesUnderTopic AS thisWholeThing
  WHERE children.parent = thisWholeThing.id_topic
) SELECT COUNT(*) AS articleCount
FROM article
WHERE id_topic IN (SELECT id_topic FROM articlesUnderTopic);
```

Použití tohoto dotazu je podmíněno FULLTEXT indexem a dosazením hodnoty id dané rubriky na místo otazníku pomocí prepared statements.

Přidání FULLTEXTového indexu do této konrétní databáze je možné následujícím příkazem přes SQL řádku:

```
ALTER TABLE `article` ADD FULLTEXT( `title`, `link`, `content`, `digest`);
```

Tato vyhledávací funkce by fungovala i bez předpřipravených výrazů, ale nebylo by to bezpečné, neboť dotaz na tuto API může přijít i ze strany útočníka a dosazený výraz do vyhledávacího pole by mohl vést k SQL injection.

4.13 **Rozhraní pro vkládání dat**

Zde jsem delší dobu váhal, zda implementovat vlastní rozhraní, nebo použít knihovnu s již existujícím rozhraním (např. stažení nějaké přes Composer ze stránek Packagist).

Aplikaci jsem v průběhu vývoje aktivně používal, shodou okolností jsem došel k tomu, že mi bohatě stačí jako administrátorské rozhraní phpMyAdmin a rozhodl jsem se neimplementovat pokročilejší funkce přímo v aplikaci.

Toto rozhodnutí má několik výhod i nevýhod. Nevýhody jsou primárně nutnost přihlašovat se mimo aplikaci do rozhraní databáze, přístup ke všem datům bez omezení (a náchylnost na lidský faktor), o něco složitější přidávání obrázků, článků i rubrik.

Na druhou stranu, má to i své výhody. Hodiny, které bych strávil implementací pro mě nepotřebného rozhraní, jsem mohl strávit v jiných částech vývoje.

Další výhodou je jednoduchost a zároveň síla tohoto rozhraní. Přístup k datům přímo přes příkazovou řádku SQL umožňuje editaci i na úrovni, která by nemusela být implementována v GUI aplikace. Již jsem se za svůj život setkal s aplikacemi, které phpMyAdmin nebo jiné nástroje používali i při existenci více redaktorů.

4.14 **Volba hostingu, doména**

Jak jsem již zmínil, aplikaci jsem aktivně používal v průběhu vývoje. Nejen na lokálním serveru, ale i na serveru veřejně přístupném. Zaregistroval jsem si tedy doménu toms.click a hosting zřídil u Active24 za pro mě příznivě přijatelnou částku.

Součástí tohoto hostingu jsou všechny potřebné součásti k fungování mé aplikace, přístup k databázi, dokonce i samo-obnovující https certifikát od Let's Encrypt.

Jelikož se za celý rok nevyskytl žádný problém ze strany poskytovatele služeb, řekl bych, že jsem si vybral správně. Alternativní možností by byl pronájem vlastního serveru, což by pro aplikaci této velikosti bylo podle mě zbytečné.

5 Výsledky a diskuse

5.1 Vzhled aplikace

Vzhledově je aplikace, alespoň dle mého názoru, pěkná a pokud se bude vzhled měnit, budou to převážně změny malého charakteru. Celý design je plně responzivní (respektive mobile-first) a správně se tedy zobrazuje jak na mobilních, tak stolních zařízeních.

5.2 Testování aplikace

Tuto aplikaci jsem aktivně používal po celou dobu vývoje a valnou většinu prvků jsem upravoval na bázi vlastního testování a používání.

Rovněž jsem vykonal několik neformálních UI testů (vlastní typ kognitivního průchodu s uživateli a podobných), kterých bych za ideální situace vykonal podstatně více, formálně a osobně, ale vzhledem k situaci s momentální globální pandemií koronaviru jsem se rozhodl nenásledovat tuto cestu nad rámec vlastního testování pro bezpečnost sebe i ostatních.

5.3 Latte vs čisté PHP šablony

S nadhledem mohu říci, že kdybych začal zakomponovávat Latte dříve, rozhodně bych jej použil v celé aplikaci, nejen v pohledech stránek. Duální provedení šablon není ideální.

5.4 Komentáře k aplikaci z hlediska budoucího vývoje

5.4.1 Design

Prvním bodem je určitě administrátorské rozhraní nad rámec phpMyAdmin. Do budoucna by určitě stálo za to tuto funkci implementovat, pakliže by se přidalo více redaktorů, což ale v momentální fázi vývoje nehrozí.

Pro budoucí vývoj by mohlo být vhodné užití již existující knihovny pro cachování, nebo přehodnocení fungování momentální implementace.

Ve finální verzi projektu jsem se rozhodl prosadit jen animace v navigačním panelu, popřípadě efekty při najetí na prvky UI myší. Příkladem může být zvětšení efektu stínu kolem náhledů článků nebo i točení kolečka nastavení po odsouhlasení cookies v cookie banneru.

Linky na jednotlivé dokumenty jsem umístil do patičky stránky. V případě přidání více jazyků, nebo většího rozvoje aplikace bych volil uložení těchto dokumentů v databázi.

Pro další vývoj archivu bych volil více viditelné zobrazení prvních článků a písmo s vyšším kontrastem u počtu sekcí/článků v rámci přístupnosti (accessibility) webu.

Dříve byla ve hlavičce dalších funkcí, jako černobílý režim či režim pro barvoslepé, ale při refaktoringu kódu celé aplikace jsem se rozhodl tyto funkce vynechat, neboť obsahovaly pár chyb (špatné zobrazování obrázků apod.), které by nebylo snadné v rozumném čase opravit

Z hlediska UI designu jsem možná zvolil nešťastně zkušenostní načítání u jednotlivých dovedností, ale chtěl jsem přesně dát najevo co umím nejlépe a jsem si vědom toho, že se tento styl ne vždy doporučuje. Jako související zajímavost uvádím, že mi u mého

prvního pohovoru tato hotová aplikace velice pomohla, byla označena za ‘zajímavou prezentaci’ a alespoň momentálně jsem zaměstnaný, zčásti i díky této bakalářské práci.

5.4.2 **Databáze**

Komentáře jsou v databázi pouze jako vhled do budoucnosti. Pro budoucí vývoj by také bylo vhodné překlady přesunout do databáze, pakliže by se v aplikaci nachály více než dva jazyky. Mohlo by též být k užitku přidání nějakého systému oprávnění pro uživatele, pakliže by bylo přidáno administrátorské rozhraní přímo do aplikace a nacházelo se zde více uživatelů. Dále by mohla být detailnost audio části databáze více využita.

Z hlediska budoucího rozvoje obsahu této aplikace bych mohl implementovat i čistě audio články, popřípadě podcast (neboť ty jsou zrovna v módě). Možná bych nahradil notu na domovské stránce viditelnějším zeleným textem audio, popřípadě ji přesunul do horního pravého rohu a do formy zeleného kruhu.

Závěr

V průběhu tvorby této práce jsem vytvořil funkční systém pro publikaci vlastní tvorby podobnou blogům a jiným serverům s volně dostupnými články. Nepoužil jsem již existující redakční systémy (například Wordpress), ale vytvořil jsem si vlastní přímo na míru mým požadavkům. Systém jsem plně používal a došel jsem k závěru, že jej i nadále budu používat na doméně toms.click.

V teoretické části byla popsána výchozí stanoviska a popis technologií, které jsem následně použil při tvorbě této aplikace. Jednotlivé části práce - od databáze, přes lokalizaci, po vzhled webu jsou pak detailněji popsány v praktické části práce, společně s popisem možných úprav a komentářů pro budoucí vývoj aplikace.

Momentální nedostatky aplikace pro uživatele reálně nejsou vidět, tudíž je aplikace plně funkční a jedná se převážně o nedostatky, které by mohly být odstraněny pro jisté quality of life improvements na straně mě, jakožto tvůrce obsahu stránek.

6 Seznam použitých zdrojů

1. SKLAR, David, 2018. PHP 7: Praktický průvodce nejrozšířenějším skriptovacím jazykem pro web. Zoner Press. ISBN 978-80-7413-363-3.
2. CASTRO, Elizabeth a Bruce HYSLOP, 2012. HTML5 a CSS3: Názorný průvodce tvorbou WWW stránek. Brno: Computer Press. ISBN 978-80-251-3733-8.
3. How can I prevent SQL injection in PHP?, 2008. Stackoverflow [online]. -: STACK EXCHANGE, Sep 12 '08 at 23:55 [cit. 2021-01-14]. Dostupné z: https://stackoverflow.com/questions/60174/how-can-i-prevent-sql-injection-in-php
4. Mysqli or PDO - what are the pros and cons? [closed], 2008. Stackoverflow [online]. -: STACK EXCHANGE, Sep 12 '08 at 23:55 [cit. 2021-01-15]. Dostupné z: https://stackoverflow.com/questions/13569/mysqli-or-pdo-what-are-the-pros-and-cons
5. Desktop vs Mobile vs Tablet Market Share Worldwide, 1999. In: Statcounter: GlobalStats [online]. StatCounter [cit. 2021-02-05]. Dostupné z: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-200901-202101
6. Latte: nejbezpečnější & opravdu intuitivní šablony pro PHP [online], 2008. Nette Foundation [cit. 2021-02-06]. Dostupné z: https://latte.nette.org/
7. MariaDB [online], 2009. MariaDB Foundation [cit. 2021-02-11]. Dostupné z: https://mariadb.org/
8. ČÁPKA, David, 2012. MVC - Jednoduchý redakční systém v PHP objektově - Online kurz [online]. ITnetwork.cz [cit. 2020-05-17]. Dostupné z: https://www.itnetwork.cz/php/mvc
9. Composer [online], 2012. Nils Adermann, Jordi Boggiano and many community contributions [cit. 2020-05-19]. Dostupné z: https://getcomposer.org/doc
10. NAIK, Sapnesh a Shanika WICKRAMASHINGHE, 2016. Most Compatible Frontend and Backend Framework Pairings. Crowdbotics: Your knowledge base for building business-ready apps, features, integrations and more. [online]. Berkeley, California: Crowdbotics, 13 OCTOBER 2020 [cit. 2021-02-11]. Dostupné z: https://blog.crowdbotics.com/most-compatible-frontend-backend-framework-pairings/#:~:text=With%20React%2C%20we%20recommend%20Express,many%20other%20Node%20web%20frameworks

11. PHP-FIG [online], 2009. The PHP Framework Interop Group [cit. 2020-05-13]. Dostupné z: https://www.php-fig.org/
12. SQL Server Recursive CTE. SQL Server Tutorial [online]. SQL Server Tutorial [cit. 2021-02-13]. Dostupné z: https://www.sqlservertutorial.net/sql-server-basics/sql-server-recursive-cte/
13. ARNTZEN, Thies C., Stig BAKKEN, Shane CARAVEO, et al. PHP GROUP. PHP: Hypertext Preprocessor [online]. 1997. Dostupné z: https://www.php.net/
14. The Open Graph protocol [online]. [cit. 2021-02-13]. Dostupné z: https://ogp.me/
15. Twitter Developer Documentation [online]. Twitter [cit. 2021-02-14]. Dostupné z: https://developer.twitter.com/en/docs
16. JQuery: write less, do more [online], 1997. OpenJS Foundation and jQuery contributors [cit. 2021-02-14]. Dostupné z: https://jquery.com/
17. Markdown Guide [online]. New Mexico: Matt Cone [cit. 2021-02-14]. Dostupné z: https://www.markdownguide.org/
18. BRANDT, Carsten. Cebe/markdown: A super fast, highly extensible markdown parser for PHP. Https://github.com/ [online]. GitHub, Feb 6, 2014 [cit. 2021-02-15]. Dostupné z: https://github.com/cebe/markdown
19. KIT. Ekce 5 - Práca s MySQL v PHP - PDO objektovo a modulárne [online]. 2012 [cit. 2021-02-15]. Dostupné z: https://www.itnetwork.cz/sk/php/ostatni/php-prace-s-mysql-pdo-objektove-a-modularne
20. SNAIDERO, Ben, 2006. SQL Server Full Text Indexes. MSSQLTips: SQL Server Tips, Articles and Training [online]. Edgewood Solutions, 7/25/2018 [cit. 2021-02-15]. Dostupné z: https://www.mssqltips.com/sqlservertutorial/9136/sql-server-full-text-indexes/#:~:text=What%20is%20a%20Full%20Text,make%20up%20the%20index%20data.
21. MySQL Workbench. MySQL [online]. Oracle Corporation [cit. 2021-02-15]. Dostupné z: https://www.mysql.com/products/workbench/
22. PhpMyAdmin documentation [online], 2012. The phpMyAdmin devel team [cit. 2021-02-19]. Dostupné z: https://docs.phpmyadmin.net/en/latest/intro.html
23. Description of the database normalization basics. Docs.microsoft.com [online]. Microsoft, 05/22/2020 [cit. 2021-02-26]. Dostupné z: https://docs.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description

7 Přílohy

Příloha A: Zdrojový kód aplikace

Příloha A: Zdrojový kód aplikace

Kód aplikace se nachází v archivu *thesis-toms.zip*

Databáze se nachází v souboru */src/Database/0.database.sql*

Instalace:

Pro snadné spuštění je možné přesunout obsah celého adresáře po rozbalení do složky htdocs při defaultní instalaci XAMPPu (se zapnutými moduly Apache a MySQL, testováno na XAMPP Version 7.4.5). Aplikace je nyní přístupná z adresy *https://localhost*

Druhý krok je nahrání databáze na server libovolnou metodou (například pomocí SQL konzole na adrese *localhost/phpMyAdmin*).