Ekonomická fakulta
Faculty of Economics
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vytvoření osobních webových stránek s použitím systému pro správu obsahu WordPress

Vypracoval: Quang Dat Le
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2021

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Quang Dat LE**
Osobní číslo:	**E17835**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Vytvoření osobních webových stránek s použitím systému pro správu obsahu WordPress**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je vytvořit osobní webové stránky s použitím CMS WordPress a vystavit je na internetu. Výsledná website bude navržena a vytvořena v moderním a praktickém stylu. Základem pro webové stránky bude vlastní téma pro CMS, při jeho vývoji bude využito dalších nástrojů (např. Twig, Sass). Součástí textu práce bude srozumitelné představení použití CMS na výsledné website.

Metodický postup:

1. Studium odborné literatury.
2. Zjištění funkční specifikace pro výsledné stránky.
3. Návrh a popis vývoje a implementace výsledných webových stránek.
4. Zhodnocení použitelnosti webových stránek pro nasazení v reálném prostředí.
5. Závěr a doporučení.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Robbins, J. (2018). *Learning Web Design*. 5th Edition. Sebastopol, CA (USA): O'Reilly.
2. Nixon, R. (2018). *Learning PHP, MySQL & JavaScript*. 5th Edition. Sebastopol, CA (USA): O'Reilly.
3. Krol, K. (2019). *WordPress 5 Complete*. Seventh Edition. Birmingham, UK: Packt.
4. Eduonix Learning Solutions. (2017). *Learn to Create WordPress Themes by Building 5 Projects*. Birmingham, UK: Packt.
5. Moreto, S., Lambert, M., Jakobus, B., & Marah, J. (2017). *Bootstrap 4 – Responsive Web Design*. Birmingham, UK: Packt.
6. Kyrnin, J., & Meloni, J. C. (2019). *Sams Teach Yourself: HTML, CSS, and JavaScript: All in One*. Third Edition. Old Tappan, New Jersey (USA): Pearson Education.

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **20. dubna 2020**
Termín odevzdání bakalářské práce: **16. dubna 2021**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 23. dubna 2020

Abstrakt

Tato bakalářská práce se zaměřuje na vývoj webových stránek a blogu jako forma osobní značky. Cílem práce je použití uživatelsky přívětivého systému pro správu obsahu (CMS), který je intuitivní a jednoduchý pro tvorbu a editaci obsahu na stránce. Pro tento cíl je využit Wordpress, který společně s Gutenberg block systémem splňuje požadavky pro tvorbu a editaci obsahu a poskytuje uživateli určitou svobodu a nezávislost na vývojářích a programátorech webových stránek. Teoretická část práce popisuje využití jednotlivých technologií jako je Wordpress, dále moderní nástroje jako je šablonovací system Twig, frontend nástroj SASS a další nástroje, které byly důležité pro vývoj. Praktická část práce ukazuje použití veškerých technologií během vývojové faze. Výsledkem je web se čtyřmi stránkami s přístupem do administrace pro správu obsahu.

Klíčová slova: Webová stránka, blog, CMS, Wordpress, Guterberg block system, Twig, SASS

Abstract

This bachelor thesis focuses on developing a website and blog as a form of the personal brand platform. The objective is to provide a user-friendly Content Management System (CMS) which is intuitive and simple to create and edit the content. For this objective is chosen WordPress and together with Gutenberg block system fulfil the requirements to create and edit content and gives the user certain freedom and independence when it comes to content creation and editing. The theoretical part of the thesis describes each technology such as Wordpress, modern frameworks template engine Twig and front-end toolkit SASS and other technologies, which are important for development. The practical part shows the application of the technologies during the web development phase. The result of this work is designed website with four sites with access to administration to manage the content.

Key words: Webpage, blog, CMS, Wordpress, Guterberg block system, Twig, SASS

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to - v nezkrácené podobě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou - elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum Podpis studenta

Poděkování

Rád bych touto cestou vyjádřil poděkování vedoucímu mé práce panu Mgr. Radimu Remeši za jeho cenné rady a trpělivost při vedení mé bakalářské práce. Rovněž bych chtěl poděkovat kolegům z práce za vstřícnou pomoc při získání potřebných informací a podkladů. Také bych chtěl poděkovat spolužákům za vzájemnou podporu. Velké poděkování patří mé rodině a blízkým přátelům za jejich podporu při studiu.

Obsah

1 Úvod

S každým dnem na světě přibude obrovský počet nových webových stránek. První webový dokument na internetu byl uveden v roce 1991. V roce 2014 bylo poprvé dosaženo milníku jedné miliardy vytvořených webových stránek na světe. K současnému období už číslo přesahuje přes 1.8 miliard webů zveřejněných na internetu. (*Total number of Websites - Internet Live Stats*)

Webová stránka má velké výhody, které představují přidanou hodnotu, ať už pro člověka jenž web vlastní nebo, pro obecenstvo, kterému je webová stránka prezentována:

- Výhoda při prezentování své značky či jména.
    - Pro člověka, který tvoří obsah na své stránce může přinášet výhodu při hledání nové pracovní pozice nebo při zakládání vlastních vedlejších projektů.
    - Při hledání nové pracovní pozice se jeho obecenstvem stávají rekruti firem, kteří musejí brát v potaz veškeré informace o daném potencionálním zaměstnanci. A právě webová stránka může být jedna z výhod pro zaměstnance, kde může prezentovat svůj další obsah, který už není, nebo nemusí být příliš vhodný do životopisu.
- Právě jméno značky bude obecenstvo často zadávat do vyhledávačů, pokud si chtějí vyhledat více informací o osobě samotné, jejích pracovní historii či současnosti, jejích zájmech atd.
- Ačkoliv se to na začátku zdát nemusí, do budoucna představuje webová stránka potenciál k vytváření nových konexí a příležitostí. Vše se odvíjí samozřejmě od kvality obsahu a správného rozložení na stránce. (*3 Reasons Why You Need A Personal Website,* 2016*)*

Tato bakalářská práce je zaměřená na tvorbu osobních webových stránek, tedy blog a portfolio pro blízkého přítele. Témata pro tuto stránku jsou především ekonomie, politika, pracovní kariéra, tuzemské a světové dění. Web ponese název *I Am Martin Le*. Cíle pro tuto bakalářskou práci jsou popsané v samotné kapitole 2 Cíl a motivace.

2 Cíl a motivace

V následující kapitole budou popsány cíle práce a motivace k tvorbě této bakalářské práce.

2.1 Cíl práce

Cílem práce je vytvořit webovou stránku a blog jako forma osobního portfolia. Web bude obsahovat čtyři stránky:

1. **About** – Úvodní stránka, která zároveň popisuje, kdo je autorem blogu.
2. **Graphics –** Stránka galerie obsahující obrázky.
3. **Articles –** Stránka s výpisem článků, které je možné rozřadit do vlastních kategorií.
4. **Musings –** Technicky se jedná o stejnou stránku jako **Articles**, avšak se liší obsahem. Autor zde bude psát citáty nebo své myšlenky, které budou kratšího rázu a v jednom odstavci.

Pro vývoj této webové stránky je využito hned několik technologií. Pro tvorbu, editaci a správu obsahu byl zvolen *Wordpress* a *Gutenberg blok editor*. K vývoji jsou použity šablonovací systém Twig s pomocí knihovny Timber, framework SASS, jenž rozšiřuje možnosti kaskádových stylů CSS, HTML a velmi základní znalosti v PHP a databáze MySQL, která je důležitou součástí *Wordpress*. Pro lokální vývoj je využita platforma *Docker* a pro frontendové balíčky je využito NPM.

2.2 Motivace

Jednou z motivací pro tuto bakalářskou práci je prohloubení již některých znalostí, které už mám a dalších, které jsem získal během nového zaměstnání (v období koronaviru), a které jsem si příliš stále neosvojil.

Dále pak získání know-how a určité myšlení v oblasti web developmentu, které mně může a bude stále posouvat, a to nejen z hlediska stále se vyvíjejících, nových a modernějších technologií, ale i z hlediska vlastního zájmu v tomto oboru.

Oblast web developmentu z karierního hlediska má určité možnosti a výhody. Kromě dobře placených pozic, je možné si i vybrat pracovní prostředí, a to buď jako zaměstnanec ve firmě,

nebo freelancer[1] kdekoliv si sám člověk vybere. (*What Is the Average Web Developer Salary? Here's What Data Says for 2021*, 2020)

[1] freelancer – odborník/profesionál ve svém oboru na volné noze

3 Teoretická část

V teoretické části jsou popsané technologie, které byly již zmíněné v části 2. Cíl a motivace a dále další technologie, které napomohly k vývoji webové stránky.

3.1 Systém pro správu obsahu (CMS[2])

Systém pro správu obsahu je počítačový software, který je určen uživatelům pro tvorbu, správu a editaci obsahu na webových stránkách bez potřeby technických znalostí. Zjednodušeně se tedy jedná o nástroj, který napomáhá k tvorbě webových stránek bez nutnosti psaní jakéhokoliv kódu od nuly či jakékoliv znalosti psaní kódu. Dobré CMS nabízí jednoduché a intuitivní UI. Obvykle, kromě obsahu, je zde možné upravit grafickou podobu stránek, změnou šablony. Šablony pak obvykle nabízejí přednastavené rozpoložení stránek, barvy, fonty a jiné designové aspekty stránky, které je možné upravovat podle vlastních potřeb, pokud to samotné CMS nabízí. (*What is: Content Management System (CMS)*, 2009 - 2021)

3.2 Wordpress

Wordpress je jeden z nejpopulárnějších, ne-li nejpopulárnější cesta, jak vytvořit vlastní webovou stránku či blog. Přes 40 % všech webových stránek na internetu stojí právě na tomto populárním softwaru. (*What Is WordPress? Explained for Beginners*, 2021)

Wordpress je systém pro správu obsahu, určený pro webový obsah. Uživatelé této platformy, mohou jednoduše publikovat svůj obsah na internet, a to zejména díky svému jednoduchému a intuitivnímu rozhraní. *Wordpress* ale není jen o tvorbě a správě obsahu, lidem s menšími dovednostmi v oblasti vývoje webových stránek umožňuje i výběr šablon či úprav jednotlivých designových prvků stránky na uživatelské úrovni. Z hlediska vývojářského, umožňuje tvorbu šablon a stylizačních prvků na stránce zcela od nuly, nebo úpravu již hotových šablon, které jsou tak tvořeny zákazníkům na míru. (*Sabin-Wilson*, 2015)

[2] CMS – content management systém – systém pro správu obsahu

Síla Wordpresu spočívá v:

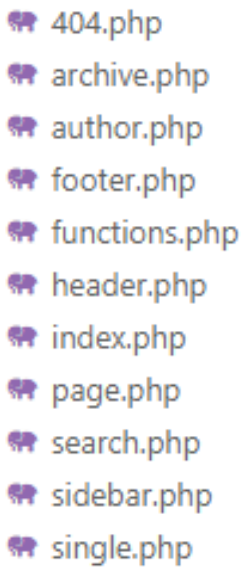
- Jedná se o open source[3] a je dostupný zdarma.
- Je spravován samotnou komunitou *Wordpress*.
- WYSIWYG[4]

Wordpress byl vytvořen v roce 2003 v jazyce *PHP* a jeho databáze je postavena v *MySQL*. (*About*, 2003)

3.2.1 Template files

Wordpress rozčleňuje svůj systém šablon do mnoha souborů. Těmito soubory rozumíme PHP soubory, které pro vykreslení určitého obsahu z databáze obsahují HTML kód, Template Tags[5] a samotný PHP kód. (*Template Files*, c2003)

Obrázek 1- Šablonovací soubory PHP



Zdroj: vlastní tvorba

Na obrázek 1 vidíme několik PHP souborů, které slouží k vykreslení některých částí či celých stránek webu. K vykreslení části stránky slouží například soubor *header.php*, jak už název napovídá. Soubor obsahuje kód pro vykreslení hlavičky na webové stránce. Výhodou rozčlenění některých částí kódu do souborů je, jako právě například *header.php*, že se zbytečně neopakuje kód. Bývá zvykem, že *header.php* se bude vykreslovat na všech stránkách. Například

[3] open source – počítačový software s otevřeným (volně dostupným) zdrojovým kódem
[4] WYSIWYG – What you see is what you get – editor pro formátování obsahu
[5] Template Tags – šablonovací tagy

pro zahrnutí na úvodní stránku, o kterou se stará soubor *frontpage.php,* ho stačí jen zavolat přes funkci. (*Template Files*, c2003)

3.2.2 Template tags

Template tags neboli také jako funkce v šablonách uvnitř kódu slouží k získání obsahu z databází a následně k vykreslení na stránce. Obsahem rozumíme nadpisy, texty jednotlivých článků a klidně i list všech článků. Template tagy se rozdělují do tří kategorií (*Template Tags*, 2003):

1. *PHP* kód tagy.
2. *Wordpress* funkce.
3. Funkce s volitelným parametrem.

Jelikož je *Wordpress* postaven na *PHP* souborech, je zde možné používat *PHP* kód a syntax. *Wordpress* funkce jsou již předdefinované a většinou slouží pro získání obsahu z databáze. Funkce s volitelným parametrem jsou pak již funkce, které navíc mohou obsahovat parametr pro získání určité informace nebo obsahu z databáze. Příkladem pro vykreslení hlavičky na úvodní stránce stačí zahrnout soubor *header.php* vložením následujícího kódu do souboru *frontpage.php.* (*Template Tags*, 2003)

Ukázka kódu 1 - Funkce pro vykreslení hlavičky

```
get_header()
```

Zdroj: (Template Tags, 2003)

3.2.3 Knihovna Timber

Knihovna *Timber* usnadňuje, zrychluje a zpřehledňuje vývoj webových stránek. Je to rozšíření do *Wordpress*, které odděluje použití *HTML* od *PHP* souborů. Touto knihovnou je zajištěná možnost využití šablonovacího systému *TWIG* uvnitř *Wordpress*. Díky knihovně Timber je zajištěno, že *PHP* soubory se soustředí na logiku a obsah z databáze, kdežto *TWIG* soubory se s pomocí užití *HTML* soustředí na vykreslení. (*Timber for WordPress | Upstatement*, 2008)

3.2.3.1 Oddělení logiky od vykreslení

Následujícím příkladem je logika, tedy získávání obsahu z *Wordpress*, lépe řečeno z databáze, oddělená od presentační vrstvy. (*Timber for WordPress | Upstatement*, 2008)

Ukázka kódu 2 – Kód logiky a vykreslení

```
$context['post'] = Timber::get_post();
Timber::render('single.twig', $context);
```

Zdroj: (Timber Docs for v1 – Timber Documentation, [2012])

Část kódu v ukázka kódu 2 se nachází v souboru *single.php*. Tento soubor je svojí logikou určen pro jednotlivé články. První řádek v kódu si do objektu *$context['post']* ukládá pomocí funkce *Timber::get_post()* veškerý dostupný obsah článku a jeho informace jako jsou nadpis, text, datum publikace, popřípadě obrázky a další položky. Na dalším řádku se pak pomocí funkce *Timber::render* posílají potřebné data do šablony. První parameter ve funkci říká, do jaké šablony se data mají odesílat, tedy do šablony *single.twig*. Druhý parameter je samotný data objekt *$context*. (*Timber Docs for v1 – Timber Documentation*, [2012])

Vykreslení dat bude dále popsáno v kapitole 3.10 Twig.

3.3 Gutenberg blok editor

S příchodem verze *WordPress* verze 5.0 v listopadu 2018 vyšel nový způsob, jak vkládat obsah na své webové stránky. Celá myšlenka tohoto editoru spočívá v tom, že veškerý obsah vkládaný na stránky probíhá přes jednotlivé bloky, proto je také nazýván blok editorem. Bloky, které se mohou přidávat do editoru je spousta, například obyčejný text, nadpis, obrázky, galerie, tlačítka, youtube videa a opravdu mnoho dalších bloků. *Wordpress* také umožňuje vytvořit vlastní blok podle potřeby. (*How to Use the New WordPress Gutenberg Block Editor*, 2020)

Gutenberg blok editor je velice příjemným a rychlým způsobem, jak vkládat veškerý obsah do stránek a lidé, blogeři nebo administrátoři stránek nemusí umět kódovat ani programovat díky tomuto editoru. Mají dokonce takovou volnost, že mohou měnit pořadí obsahu, který se vykresluje na stránce a také ho mohou formátovat podle svých představ. (*How to Use the New WordPress Gutenberg Block Editor*, 2020)

3.4 HTML

HTML (*Hyper Text Markup Language*) je standardním značkovacím jazykem pro dokumenty na internetu. Spolu s *CSS* a *Javascriptem* tvoří základní trojici pro tvorbu webových stránek. *HTML* je kostra celé stránky a tvoří její strukturu. Pro vypsání obsahu musí webové prohlížeče získat *HTML* dokument z webového serveru, nebo jiného uložiště a následně vykreslit obsah z dokumentu do webových stránek. (*Duckett,* [2011].)

Celá struktura *HTML* je tvořena pomocí tzv. elementů, které jsou stavebními bloky. *HTML* elementy často bývají párové s otevírajícím a zavírajícím tagem, které se píšou do ostrých závorek. V otevírajícím tagu je možné použít navíc atributy, které doplňují informaci o daném tagu nebo rozšiřují jeho funkce. Uzavírající tag je ještě doplněn o lomítko. Obsah či informace pak leží mezi párovými tagy. Kromě párových tagů jsou v *HTML* i samostatné tagy, u kterých v drtivé většině potřebují mít atributy. (*Duckett,* [2011].)

3.5 CSS

CSS (*Cascading Style Sheets*) umožňuje vytvářet a aplikovat pravidla, jak bude obsah vyobrazen na stránce. Slovíčkem „jak“ je myšleno zejména po stránce stylizační. Upravuje způsob zobrazení elementů na webových stránkách, které jsou napsané v *HTML*. Ten sám o sobě už s některými tagy dokáže stylizovat svůj obsah, avšak se zcela nedokáže vyrovnat samostatnému *CSS*, který by měl být oddělený od *HTML* souboru, a taky tomu často bývá, pro přehlednost a čitelnost. V *CSS* můžeme upravovat formátování textu, barvy pozadí, velikosti objektů, animace objektů, pozicování objektů a mnoho dalších… (*Duckett,* [2011].)

Pomocí *CSS* můžeme upravovat veškeré selektory na stránce, které jsou pomocí *HTML* definovány. Mezi selektory patří veškeré elementy, třídy a id, které svým způsobem označují daný element pro přehlednost. Pro přidání jakýchkoliv stylizačních úprav pro daný selektor se deklarují uvnitř složených závorek, kterým předchází název selektoru. (*Duckett,* [2011].)

3.6 Javascript

Javascript je skriptovací nebo také programovací jazyk, pomocí něhož se webová stránka ze statické stránky stává více interaktivní a pro návštěvníka více atraktivní. *Javascript* patří do rodiny *HTML* a *CSS*, který spolu s nimi tvoří jádro veškerých internetových stránek. Drtivá většina webových stránek používá *Javascript* na straně klienta. Stejně tak jako *CSS* by *Javascript* měl být ve svém vlastním **.js* souboru, pro přehlednost a čitelnost. (*Duckett,* c2014)

Javascript umožňuje přistupovat k jakýmkoliv elementům, atributům nebo obsahu (text, obrázek, …) a následně s nimi dále pracovat. K elementům dokáže přidávat, upravovat nebo odebírat atributy, či zcela pozměnit celý element a také i obsah elementu. Dokáže i přesouvat element definovaný v *HTML* struktuře. Kromě těchto možností dokáže i změnit stylizaci již deklarovaných stylů pro dané elementy a tímto je velmi silným nástrojem v oblasti web developmentu. (*Duckett,* c2014)

3.7 PHP

PHP je skriptovací jazyk pracující na straně serveru vytvořen speciálně pro webové stránky. *PHP* se taky přidává k rodině *HTML, CSS* a *Javascript* a pomocí něj je stránka také o něco více interaktivnější. Na rozdíl od *Javascriptu*, který pracuje na straně klienta, uživatel vidí už jen výsledek scriptu, který tak často slouží ke generování *HTML* a následně ho odesílá klientovi k zobrazení. Efektivně pracuje s databází a stává se tak ideálním komunikačním prostředníkem mezi databází a stránkou. (*Thomson & Weilling,* 2016)

3.8 MySQL

MySQL využívá relační databázový model, který efektivně ukládá, hledá, třídí a získává data. Server pro *MySQL* zajišťuje přístup k datům pro uživatele, aby s nimi mohli pracovat naráz. Používá standardní *SQL* jazyk pro komunikaci s databází a spolu s *PHP* pracuje na straně serveru. (*Thomson & Weilling,* 2016)

3.9 React

React je open-source, javascriptová knihovna určená k tvorbě uživatelského rozhraní nebo UI[6] komponent. *React* byl vyvinut a spravován společností *Facebook* a komunitou jednotlivých vývojářů a společností. *React* s rozšířením *JSX*[7] svým zápisem napodobuje šablonovací jazyky, avšak využívá plnou sílu *Javascript*. Díky tomu, že je jeho práce s dynamicky měnícími daty optimální, využívá se jako základ pro vývoj jednostránkových webu či mobilních aplikací. (*React – A JavaScript library for building user interfaces*, 2021)

[6] UI – user interface – uživatelské rozhraní
[7] JSX – Javascript XML

3.10 Twig

Šablonovací systém *Twig* svojí syntaxí velice urychluje a usnadňuje vývoj stránek. *PHP* jazyk, jako takový, také slouží jako šablonovací systém, ale psaní v něm je pro vývojáře celkem zdlouhavé a nepříjemné. Syntaxe pro *Twig* je celkem jednoduché na pochopení. *Twig* šablony kromě běžného *HTML* obsahují i proměnné a výrazy, které jsou nahrazeny hodnotami při vykreslování. V *Twig* existují dva způsoby, jak vypsat či zpracovat proměnné nebo výrazy. (*Twig - The flexible, fast, and secure PHP template engine*, c2010-2021)

3.10.1 Syntax

Ukázka kódu 3 – Způsob výpisu výrazů a proměnných

```
1  {%  %} // pro výrazy
2  {{  }} // pro proměnné
```

Zdroj: (Twig - The flexible, fast, and secure PHP template engine, c2010-2021)

V ukázka kódu 3 na prvním řádku je možné vidět zápis, kam se do prostoru mezi složené závorky s procenty vkládají výrazy, které mají vykonat své úkony, například *for loop* smyčka, podmínka *if else* nebo deklarace proměnných. Na druhém řádku je zápis, kam mezi dvojité složené závorky výrazy, které při vykreslování šablony vypisují svoji hodnotu, typicky jsou to proměnné. (*Twig - The flexible, fast, and secure PHP template engine*, c2010-2021)

3.10.2 Proměnné

V ukázka kódu 3 bylo popsáno, jak vypsat proměnné do stránky. Pokud by v proměnné byl uložen objekt a my bychom potřebovali vypsat jeho konkrétní atribut či element, pak by kód vypadal: (*Twig - The flexible, fast, and secure PHP template engine*, c2010-2021)

Ukázka kódu 4 - Přístup k atributu "bar" z objektu " foo"

```
{{ foo.bar }}
```

Zdroj: (Twig - The flexible, fast, and secure PHP template engine, c2010-2021)

Tímto zápisem je řečeno, že potřeba vypsat hodnotu uloženou v atributu *bar*, která náleží objektu *foo* a jak je vidět, k atributu se přistupuje přes tečku. Pro deklaraci proměnných uvnitř šablony, jak bylo v ukázka kódu 3 naznačeno, se zapisuje: (*Twig - The flexible, fast, and secure PHP template engine*, c2010-2021)

Ukázka kódu 5 - Deklarace proměnných uvnitř šablony

```
1  {% set foo = 'foo' %}
2  {% set foo = [1, 2] %}
3  {% set foo = {'foo': 'bar'} %}
```

Zdroj: (Twig - The flexible, fast, and secure PHP template engine, c2010-2021)

V ukázka kódu 5 je vidět, že se deklaruje proměnná pro textové řetězce na 1. řádku, pole na 2. řádku nebo objekty na 3. řádku. (*Twig - The flexible, fast, and secure PHP template engine*, c2010-2021)

3.10.3 Filtry

Filtry se obecně využívají pro proměnné, které svým účelem upravují hodnotu uložené uvnitř proměnné. Filtry jsou oddělené od proměnné pomocí svislé čáry. Pro proměnnou je také možné použít více filtrů najednou, které jsou zpracovávány popořadě a výsledek jednoho filtru je použit pro ten následující. (*Twig - The flexible, fast, and secure PHP template engine*, c2010-2021)

Ukázka kódu 6 - Použití více filtru najednou

```
{{ name|striptags|title }}
```

Zdroj: (Twig - The flexible, fast, and secure PHP template engine, c2010-2021)

V ukázka kódu 6 se pro proměnnou *name* používají 2 filtry najednou. Prvním je *striptags*, který odstraňuje veškeré *HTML* tagy uvnitř proměnné *name*. Druhým filtrem je *title*, který v tomto příkladu získává hodnotu z proměnné *name*, na kterou byl již zprvu aplikován filtr *striptags* a který konkrétně nastaví počáteční písmeno jako velké písmeno. (*Twig - The flexible, fast, and secure PHP template engine*, c2010-2021)

3.11 SASS

SASS[8] je preprocesorový jazyk, který se kompiluje do běžných kaskádových stylů *CSS*. Díky *SASS* je možné využít *proměnných, nesting, mixins, partials a import rule* a dalších, které jsou plně kompatibilní s CSS syntaxí a rozšiřují tak možnosti oproti *CSS*. *SASS* podporuje dvě syntaxe, *SCSS* a *SASS*. Záleží jen na vývojáři, který z nich si vybere, a který mu bude více vyhovovat. Pro projekt byl využit syntax *SASS*, ale pro představu ukážu kousek kódu i ze syntaxe *SCSS*. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

[8] SASS – Syntactically awesome style sheets – syntakticky úžasné kaskádové styly

3.11.1 SCSS syntax

SCSS syntax se používá v souborech s koncovkou *.scss*. *SCSS* syntax je velmi podobný, pomalu až stejný jako *CSS*. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Ukázka kódu 7 - SCSS syntax

```
1  .content {
2      width: 300px;
3      color: red;
4
5      a {
6          text-decoration: none;
7      }
8
9      p {
10         text-align: justify;
11     }
12
13 }
```

Zdroj: vlastní tvorba, (Visual Studio Code - Code Editing. Redefined, 2021)

Na ukázka kódu 7 je vidět zápis kaskádových stylů v *SCSS*. Na začátku se definuje vlastní třída *.content* na 1. řádku. Prvek nesoucí danou třídu má obarvený text na červenou se šířkou 300 px. *SCSS* a zároveň i *SASS* jsou speciální právě v tom, že je možné využít *nesting*. Pokud chceme upravit stylizaci atributů *a*, *p* na 5. a 9. řádku, které se mají týkat zejména třídy *.content*, pak je stačí zapsat uvnitř složených závorek tak, jak je naznačeno v ukázce. Pokud se dané atributy či třída použije mimo třídu *.content*, potom na ně nebudou aplikovány dané styly. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Ukázka kódu 8 - Zkompilovaný CSS syntax

```
1  .content {
2      width: 300px;
3      color: red;
4  }
5  .content a {
6      text-decoration: none;
7  }
8  .content p {
9      text-align: justify;
10 }
```

Zdroj: vlastní tvorba, (Visual Studio Code - Code Editing. Redefined, 2021)

Na ukázka kódu 8 je vidět, jak vypadá zkompilovaný kód do *CSS*.

Pro demonstraci je v ukázka kódu 9 aplikace předešlých stylů v *HTML* kódu. Její vizuální podoba je vidět v obrázek 2.

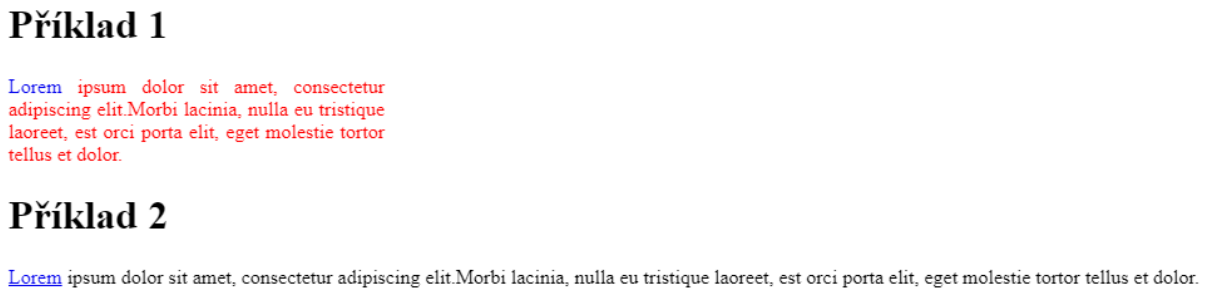
Ukázka kódu 9 - Aplikace kaskádových stylů do HTML

```
1  <!-- Příklad 1 -->
2  <div class="content">
3      <p><a href="#">Lorem</a> ipsum dolor sit amet, consectetur
4      adipiscing elit.Morbi lacinia, nulla eu tristique laoreet,
5      est orci porta elit, eget molestie tortor tellus et dolor.</p>
6  </div>
7
8  <!-- Příklad 2 -->
9  <div>
10     <p><a href="#">Lorem</a> ipsum dolor sit amet, consectetur
11     adipiscing elit.Morbi lacinia, nulla eu tristique laoreet,
12     est orci porta elit, eget molestie tortor tellus et dolor.</p>
13 </div>
```

Zdroj: vlastní tvorba, (Visual Studio Code - Code Editing. Redefined, 2021)

V ukázka kódu 9 je *Příklad 1* s *div* tagem na řádcích 2–6, na který se aplikuje třída *content* a uvnitř tohoto divu jsou elementy pro paragraf *p* a pro odkaz *a* na řádku 3. V *příkladu 2* na řádcích 9–13 je div prvek bez aplikace třídy *content*. Na obrázek 2 je znázorněn vizuální rozdíl mezi *příkladem 1* a *příkladem 2*. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Obrázek 2 - Vizuální podoba aplikace kaskádových stylů do HTML



Zdroj: vlastní tvorba, (Sass: Syntactically Awesome Style Sheets, c2006–2021)

3.11.2 SASS syntax

Syntax *SASS* se používá v souborech s koncovkou *.sass* a jeho zápis je obdobný jako *SCSS*, a dokonce i pohodlnější. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Ukázka kódu 10- SASS syntax

```
1  .content
2    width: 300px
3    color: red
4
5    a
6      text-decoration: none
7
8    p
9      text-align: justify
```

Zdroj: vlastní tvorba, (Visual Studio Code - Code Editing. Redefined, 2021), (Sass: Syntactically Awesome Style Sheets, c2006–2021)

Syntax SASS v ukázka kódu 10 je téměř identický k příkladu v ukázka kódu 7, ale celkem na první pohled rozeznatelný. Oproti *SCSS* pro *nesting* se nepíšou složené závorky a místo nich se používá odsazení. Kompilace do *CSS* je stejná jako na ukázka kódu 8. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

3.11.3 Proměnné

Proměnné ve skriptovacím jazyce *SASS*, fungují obdobně jako v kterýchkoliv jiných programovacích jazycích. Do proměnných je možné uložit hodnoty jako barvy, font-family[9] a jiné, které chceme a potřebujeme používat stále dokola. Deklarace proměnných vypadá v podstatě stejně jako při deklaraci vlastností v *CSS*. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Ukázka kódu 11 - Deklarace proměnných

```
<variable>: <expression>
```

Zdroj: (Sass: Syntactically Awesome Style Sheets, c2006–2021)

V ukázka kódu 11 se na levé straně od dvojtečky píše název proměnné a na pravé straně pak hodnota, nebo hodnoty, které do ní chceme uložit. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

[9] font-family – rodina písma, skupina fontů

Ukázka kódu 12 – Proměnné v SASS

```
1  $font-family: Helvetica, sans-serif
2  $black-color: #000 // černá
3
4  body
5    font-family: $font-family
6    color: $black-color
```

Zdroj: vlastní tvorba, (Sass: Syntactically Awesome Style Sheets, c2006–2021)

V ukázka kódu 12 je vidět konkrétně příklad zápisu proměnných pro *SASS*. Deklarace a následný výpis je velmi jednoduchý proces. Na řádcích 1 a 2 vidíme deklaraci dvou proměnných *$font-family* a *$black-color* a jak je již z názvů jasné, tak první proměnná nese hodnoty skupinu fontů a druhá proměnná nese hodnotu barvy černé. Následně jejich použití je možné vidět v elementu *body***,** kde proměnné jednoduše vložíme ke správným vlastnostem, tedy *font-family* a *color* na řádcích 5 a 6. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

3.11.4 Nesting

Nesting ze *SASS*, tvoří krásnou vizuální hierarchii a dělá tak kód velmi přehledným, příjemným na čtení a hlavně i jednoduchým na zápis, který šetří čas. Syntax pro *nesting* byl již představeny v ukázka kódu 10 a ukázka kódu 7. A velice užitečným se také stává při vytvoření vlastních názvů tříd. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

3.11.5 Mixins

Mixins jsou takové vlastní funkce pro *SASS*. Umožňují vytvářet skupinu vlastních skupin *CSS* deklarací, které jsou znovupoužitelné v *SASS*. Do *mixins* je možné předávat parametry. Na následujících příkladech bude nejlépe vidět, jak dokážou ušetřit vývojářům čas a námahu. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Ukázka kódu 13 - Deklarace a použití mixins

```
1  =transform($property) // deklarace mixinu transform
2    -webkit-transform: $property // vendor prefix pro prohlížeče
3    safari a iOS
4    -ms-transform: $property // vendor prefix pro Edge a zastaralý IE
5    transform: $property // obecný zápis
6
7  .box
8    +transform(rotate(30deg))
```

Zdroj: (Sass: Syntactically Awesome Style Sheets, c2006–2021)

V ukázka kódu 13 je příklad vytvořeného *mixin* pro vlastnost *transform* s přidáním vendor prefixů[10]. Ne všechny prohlížeče podporují všechny vlastnosti *CSS* a musejí se tedy psát speciální vendor prefixy tak aby aplikovaný styl byl podporován ve všech prohlížečích. Na prvním řádku vidíme příklad vytvořeného *mixin*. Znaménkem *„rovná se“*, dále přidáním názvu pro daný *mixin* a do závorek parametr, se kterým se dále pracuje, deklarujeme *mixin transform*. Na dalších třech řádcích pak parametr vypisujeme k vendor prefixům. Na posledním řádku je zápis použití *mixin*. Pro jeho použití se na začátku píše znaménko *„plus“,* následně název a do závorek daný parametr. Po kompilaci do *CSS* by kód vypadal následovně v ukázka kódu 14. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Ukázka kódu 14 - Zkompilované CSS při použití mixinu transform

```
1  .box {
2    -webkit-transform: rotate(30deg);
3    -ms-transform: rotate(30deg);
4    transform: rotate(30deg);
5  }
```

Zdroj: (Sass: Syntactically Awesome Style Sheets, c2006–2021)

Díky *Mixins* stačí napsat jeden řádek, který po zkompilování do *CSS* vypíše vše potřebné a tím šetří čas. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

3.11.6 Partials a import rule

Díky *partials* a import rule je možné rozdělit kusy kaskádových stylů do svých vlastních souborů. *Partials* je konkrétně soubor, který na začátku názvu obsahuje podtržítko, například *_variables.sass*. *SASS* tak ví, že s takto nazvaným souborem může jen pracovat, a není kompilován do *CSS* souboru. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

Import rule umožňuje sloučit několik vlastních, rozdělených souborů do jednoho. *Import* je tak výborný pro sloučení svých proměnných, *mixins* a dalších jiných *SASS* souborů, které mohou upravovat jednotlivé části stránky, například hlavička, patička, navigace a další… Zápis takové *import* je představen v ukázka kódu 15. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

[10] vendor prefix – způsob přidání podpory pro CSS do prohlížečů, než se stanou standardně podporované

Ukázka kódu 15 - Import vlastních souborů

```
1  // soubor index.sass
2  @import 'variables'
3  @import 'mixins/mixins'
4  @import 'frontend/frontend'
```

Zdroj: vlastní tvorba, (Visual Studio Code - Code Editing. Redefined, 2021)

Zápis vypadá tak, že na začátku zapíšeme znaménko „zavináč“, dále slovíčko *import* a následně do uvozovek píšeme název souboru. Pokud importovaný soubor není ve stejném adresáři, pak píšeme i relativní cestu k souboru. V ukázce je konkrétní příklad pro *import* tří souborů do souboru *index.sass*. (*Sass: Syntactically Awesome Style Sheets*, c2006–2021)

3.12 Docker

Docker je open source, a jedná se o sadu *PaaS*[11], které pomocí virtualizace na úrovni operačního systému poskytuje software v balíčcích, tedy tzv. kontejnerech. A díky *Docker* je možné zařídit běh *Wordpress* a její databáze *MySQL* pro lokální vývoj. (*Docker*, 2013)

3.13 NPM

Když se řekne *NPM* (*Node Package Manager*), jsou na mysli dvě věci. Za prvé online repositář s publikovanými open source *Node.js* projekty a za druhé utilita pro příkazovou řádku, která pracuje s online repositářem. Zjednodušeně se jedná o správce balíčků pro Javascript. (*What is npm?*, 2011)

Instalace knihoven je velice jednoduchá. Například pro instalaci knihovny s názvem *„async“* stačí napsat do příkazové řádky zápis z ukázka kódu 16 a tím se následně nainstaluje knihovna do adresáře *./node_modules*. (*What is npm?*, 2011)

Ukázka kódu 16 - Příkaz pro instalaci knihovny "async" pomocí NPM

```
npm install async
```

Zdroj: (What is npm?, 2011)

[11] PaaS – Platform as a service – platforma jako služba

Další důležitá náležitost k *NPM* je správa závislostí. Veškeré knihovny, které jsou součástí projektu jsou zapsané jako závislosti v souboru *package.json*. Po každé instalaci nového balíčku je zároveň zapsán jako závislost do tohoto souboru. Pokud je tedy třeba nainstalovat stejné knihovny z jednoho projektu do druhého, pak stačí použít soubor *package.json* z prvního projektu do druhého, tím že se v příkazové řádce použije příkaz v ukázka kódu 17 a ve složce *./node_modules* se následně objeví všechny závislosti, což jsou stažené knihovny. (*What is npm?,* 2011)

Ukázka kódu 17 – Příkaz pro instalaci závislostí pro NPM

```
npm install
```

Zdroj: (What is npm?, 2011)

3.14 Composer

Podobně jako *NPM, Composer* je nástroj pro správu knihoven (závislostí) pro PHP. Stejně tak jako *NPM* se *Composer* ovládá z příkazové řádky. Knihovny se přes *Composer* instalují do složky *./vendor* a stejně jako NPM, obsahuje soubor *composer.json*, do kterého jsou zapsány veškeré závislosti pomocí příkazu na ukázka kódu 18: (*Introduction - Composer*, [2012])

Ukázka kódu 18 - Příkaz pro instalaci závislostí pro Composer

```
composer install
```

Zdroj: (Introduction - Composer, [2012])

Pro instalaci konkrétní knihovny stačí zapsat příkaz na ukázka kódu 19. Zde místo „*php*“ můžeme zapsat jakýkoliv jiný název knihovny, kterou potřebujeme nainstalovat. (*Introduction - Composer*, [2012])

Ukázka kódu 19 - Příkaz pro instalaci knihovny „php“ pomocí Composer

```
composer require php
```

Zdroj: (Introduction - Composer, [2012])

4 Praktická část

Praktická část je rozdělená do několika podkapitol. 4.1 Instalace technologií popisuje způsob instalace a použití jednotlivých technologií, 4.2 Tvorba stránek popisuje samotnou tvorbu webových stránek. Tvorba bude popsána způsobem vizuální ukázky stránky (obrázkem) a následně přiložením veškerých souvislostí ke stránce, buďto dalšími obrázky nebo zdrojovým kódem.

4.1 Instalace technologií

V následujících podkapitolách jsou popsány způsoby instalace jednotlivých technologií, které zajišťují běh nebo umožňují vývoj webových stránek.

4.1.1 Docker, Wordpress a MySQL

Jak už bylo v teoretické části naznačeno, pro vývoj bylo využito *Docker*, který ve své dokumentaci již přichystal návod na nakonfigurování *Wordpress* a MySQL databáze. Konfigurace vypadá následovně v ukázka kódu 20.

Ukázka kódu 20 - Konfigurace Wordpress a MySQL databáze pomocí Docker

```
1  version: '3.3'
2  services:
3    db:
4      image: mysql:5.7
5      volumes:
6        - db_data:/var/lib/mysql
7      restart: always
8      environment:
9        MYSQL_ROOT_PASSWORD: 123
10       MYSQL_DATABASE: iammartinle
11       MYSQL_USER: root
12       MYSQL_PASSWORD: 123
13
14   wordpress:
15     depends_on:
16       - db
17     image: wordpress:latest
18     ports:
19       - "8000:80"
20     restart: always
21     volumes:
22       - ./wordpress:/var/www/html
23     environment:
24       WORDPRESS_DB_HOST: db:3306
25       WORDPRESS_DB_USER: root
26       WORDPRESS_DB_PASSWORD: 123
27       WORDPRESS_DB_NAME: iammartinle

28   adminer:
29     image: adminer
30     restart: always
31     ports:
32       - 8080:8080
33 volumes:
34   db_data: {}
```

Zdroj: (Bachelor-thesis, 2021), (Quickstart: Compose and WordPress, c2013-2021)

Celá konfigurace se zapisuje do souboru docker-compose.yaml a obsahuje celkem tři služby. První je databázová konfigurace na s klíčovým názvem *db* na řádcích 3–12. Druhá konfigurace se týká *Wordpress*, který je spolu s databází propojen, kde služba nese klíčový stejnojmenný název *wordpress* na řádcích 14–27. A poslední je *adminer*, který umožňuje přístup do databáze pomocí grafického rozhraní, není však pro tento projekt tolik zásadní, na řádcích 28–32.

Ukázka kódu 21 - Příkaz pro spuštění Docker příkazu

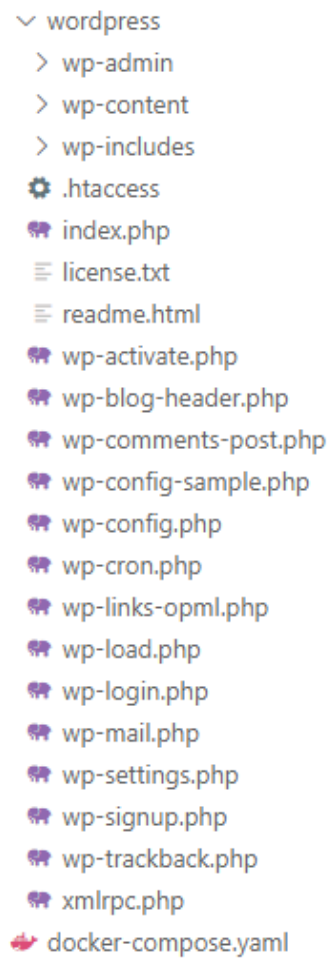
```
docker-compose up
```

Zdroj: (Quickstart: Compose and Wordpress, c2013-2021)

Následně příkazem z ukázka kódu 21 je zahájeni stahování tzv. *obrazů* daných konfigurací, tedy *Wordpress*, *MySQL* databáze a *adminer* a spustí se kontejnery jednotlivých konfigurací. (*Quickstart: Compose and WordPress*, c2013-2021)

Je třeba specifikovat, kam se *Wordpress* bude ukládat, aby se objevily veškeré potřebné adresáře a soubory lokálně. V ukázka kódu 20 je to definováno řádkem 22 - *./wordpress:/var/www/html*, kde se na levé straně od dvojtečky specifikuje, do jakého adresáře se *Wordpress* uloží, a na pravé straně od dvojtečky se specifikuje, které adresáře, podadresáře a soubory se chtějí ukládat. Pro získání všech adresářů a souborů se uvede adresář *wordpress*. Po spuštění se získá celá stromová struktura *Wordpress*:

Obrázek 3 - Stromová struktura Wordpress po spuštění Docker



Zdroj: (Bachelor-thesis, 2021), (Quickstart: Compose and WordPress, c2013-2021)

4.1.2 Twig a Timber

Fungování šablonovacího systému *Twig* uvnitř *Wordpress* je zajištěno pomocí knihovny Timber. Pro instalaci této knihovny je možné využít dva způsoby, pomocí:

1. Plugin
2. Composer

Plugin je ve spojitosti s *Wordpress* celkem běžná záležitost. Dokážou ho totiž rozšířit o další funkcionality a dělají z něho celkem silný nástroj pro tvorbu obsahu na webových stránkách.

Pro tento projekt byl však využit způsob instalace pomocí Composer, který z mého osobního hlediska umožňuje snadnější kontrolu. Instalace se spustí v kořenovém adresáři příkazem:

Ukázka kódu 22 - Příkaz pro instalaci Timber

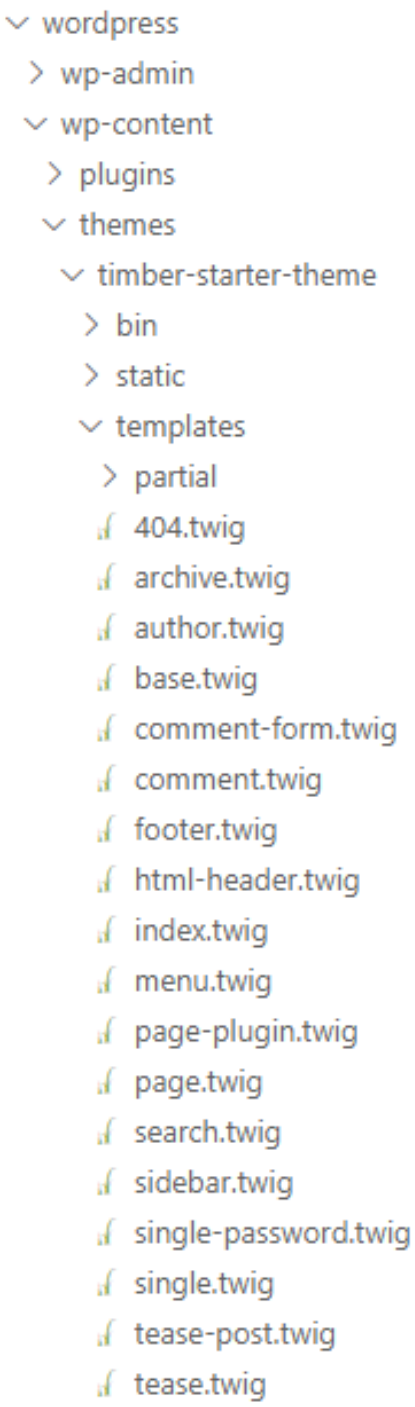
```
composer require timber/timber
```

Zdroj: (Timber for WordPress | Upstatement, 2008)

Dále byla pro projekt využita základní šablona *timber-starter-theme,* která byla také nainstalována pomocí Composer ve složce w*ordpress/wp-content/themes/* mezi ostatní šablony, které *Wordpress* nabízí.

Složka *timber-starter-theme* v sobě ukrývá jednu z nejdůležitějších adresářů. Jedná se o adresář *templates* s šablonami, které slouží pro vykreslení některých částí či celých stránek na webu.

Obrázek 4 - Stromová struktura timber-starter-theme s důležitým adresářem templates



Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Timber for WordPress | Upstatement, 2008)

4.1.3 SASS

Pro použití SASS v projektu je potřeba psát kód do souborů s koncovkou *.sass* a pro kompilaci do souborů s koncovkou *.css* je pak potřeba nainstalovat knihovnu z *NPM*, která zajišťuje tento proces. Do projektu je nainstalována stejnojmenná knihovna jako technologie, tedy *SASS*, a pro zahájení kompilace se spouští příkazem (*Sass*, [2009]):

Ukázka kódu 23 - Příkaz pro jednorázovou kompilaci souborů SASS do CSS

```
npm run build:style
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Sass, [2009])

Příkaz v ukázka kódu 23, kompiluje jednorázově.

Ukázka kódu 24 - Příkaz pro neustálou kompilaci souborů SASS do CSS

```
npm run watch:style
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Sass, [2009])

V ukázka kódu 24 je příkaz, pomocí kterého se kompilují soubory neustále a které je vhodné při vývoji stránek. Tedy ve chvíli, kdy se přidá, odebere nebo jednoduše upraví kód uvnitř, některého SASS souboru, pak se automaticky překompiluje do CSS souboru.

Veškeré SASS soubory uvnitř projektu v adresáři */static/styles/* se kompilují do adresáře */build/styles/* s koncovkou *.css*, kde jak už bylo řečeno výše v této kapitole.

4.1.4 Gutenberg blok

Gutenberg blok není úplně potřeba instalovat, jelikož je již součástí *Wordpress*. Avšak pro vytvoření vlastního bloku bylo využito knihovny z *NPM @wordpress/scripts* od *Wordpress*, který zajišťuje kompilaci javascriptového souboru tak, aby byl optimální pro použití na webovou stránku (*@wordpress/scripts*, [2009]).

Ukázka kódu 25 - Příkaz pro jednorázovou kompilaci Javascriptových souborů

```
npm run build:scripts
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (@wordpress/scripts, [2009]).

Ukázka kódu 26 - Příkaz pro neustálou kompilaci Javascriptových souborů

```
npm run watch:scripts
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (@wordpress/scripts, [2009]).

Stejně tak jako u *SASS*, kde jsou příkazy pro kompilaci, tak i v tomto případě, s tím rozdílem, že kompilace je určená pro javascriptové soubory. Příkaz v ukázka kódu 25 je pro jednorázovou kompilaci. Ukázka kódu 26 se používá pro neustálou kompilaci, který je určen pro fázi vývoje, tak jako je tomu u *SASS*.

4.2 Tvorba stránek

Tvorba stránek je rozdělená do několika podkapitol, které prezentují rozložení/části, které se na stránce vyskytují opakovaně, například hlavička, patička a dále celé stránky webu.

Wordpress definuje v souboru *functions.php* veškeré funkcionality, rozšíření atd., které jsou buďto vlastnoručně vytvořené anebo jsou vytvořené jinými technologiemi či pluginy. V tomto souboru si například *Timber* deklaruje svoji třídu pro práci se šablonou *timber-starter-theme* nebo je zde deklarována funkce pro vytvoření vlastního *Gutenberg bloku*. Při popisu v podkapitolách této kapitoly se často bude odkazovat na tento soubor.

Dále každá šablona pro určitou stránku obsahuje v stromové struktuře vlastní PHP soubor s koncovkou **.php*, ze kterého se následně posílají potřebná data a obsah do obvykle stejnojmenných souborů, avšak s koncovkou *.twig*. Každá podkapitola obsahuje zdrojový kód.

Zdrojový kód v *SASS* v praktické části je jedna ukázka v podkapitole 4.2.2.2 Zdrojový kód v SASS v ukázka kódu 30, jelikož se jedná o analogický postup vytváření stylů pro všechny ostatní styly na stránce.

4.2.1 Layout

Layout neboli také rozložení stránky, je jedna ze zásadních věcí při vývoji stránek. V layoutu definujeme části webu, které se na stránce opakují a bylo by zbytečné a nesprávné je vypisovat v každé vlastní šabloně znovu. Typicky se jedná o hlavičku a patičku. Ty jsou na každé stránce stejné a v ukázka kódu 27 je jejich vykreslení zajištěno pomocí kódů na řádcích 12 pro hlavičku a 23 pro patičku.

Ukázka kódu 27 - Zdrojový kód v Twig pro layout

```
1  {% block html_head_container %}
2
3   {% include 'html-header.twig' %}
4   {% block head %}
5   {% endblock %}
6     </head>
7  {% endblock %}
8
9    <body class="{{body_class}}" data-template="base.twig">
10
11     {# Header #}
12     {% include 'header.twig' %}
13
14     {# Content #}
15     {% block fullPage %}
16       <main role="main" class="site">
17         {% block content %}
18         {% endblock %}
19       </main>
20     {% endblock %}
21
22     {# Footer #}
23     {% include 'footer.twig' %}
24   </body>
25 </html>
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Na řádcích 15–20 je definován *{% block fullPage %}* a uvnitř tohoto blocku je dále *{% block content %}* na řádcích 17–18, které slouží pro vložení veškerého dynamického obsahu, textů, článků atd. Dynamický obsah se konkrétně nevkládá do tomto souboru *base.twig*, zde se tyto *blocky* totiž definují a jednotlivé šablony je používají pro výpis svého obsahu. Ukázky použití jsou naznačeny v dalších podkapitolách.

4.2.2 Header

Header neboli hlavička je důležitou součástí této stránky. Jedná se celkem o jednoduché rozložení. Jako první největší část na obrázek 5 vidíme logo nesoucí název webové stránky. V případě kliknutí, logo odkáže na úvodní stránku. Pod logem pak vidíme horizontální navigaci na další stránky. Uživatel si může sám upravovat, které odkazy se mu budou zobrazovat v navigaci, díky připravenému rozhraní od *Wordpress*.

Obrázek 5 - Hlavička stránky

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

4.2.2.1 Zdrojový kód v Twig

Ukázka kódu 28 - Zdrojový kód v Twig pro hlavičku

```
1  <header class="header">
2    <div class="site">
3      <div class="header-logo">
4        <a href="{{ site.link }}" class="header-logo-link">
5          <img src="{{ static }}/images/logo-black-transparent-
   500x76.png" alt="I Am Martin Le" class="header-image">
6        </a>
7      </div>
8      {% include "menu.twig" with {'items': navigation.get_items} %}
9    </div>
10 </header>
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

V ukázka kódu 28 je celá hlavička zabalená do tagu *header* v šabloně *header.twig*. Samotné logo jako obrázek je uvnitř elementu *img* na řádku 5. Do atributu *src* se zapisuje cesta k obrázku a jak je možné si povšimnout, je zde využito proměnné *{{ static }}*, která v sobě ukrývá cestu do adresáře *static* a je definována v souboru *functions.php* jako globální proměnná. Dále zápis pokračuje určením cesty k obrázku *logo-black-transparent-500x76.png*. Následně je element *img* zabalen do tagu *a*, který pomocí hodnoty uvnitř proměnné *{{ site.link }}* odkazuje na hlavní stránku.

Navigace, v předešlé ukázce, je definována uvnitř vlastní šablony *menu.twig* a pro jeho vykreslení uvnitř šablony *header.twig* je na něj odkazováno pomocí zápisu na řádku 8.

Ukázka kódu 29 - Zdrojový kód v Twig pro navigaci

```
1  {% if navigation %}
2    <nav id="menu" class="menu" role="navigation">
3      <ul class="menu-list">
4        {% for item in items %}
5          <li class="{{ item.classes | join(' ') }} {{ item.current ? 'v
   iew-active' }}">
6            <a target="{{ item.target }}" href="{{ item.link }}">{{ item
   .title }}</a>
7            {% include "menu.twig" with {'items': item.children} %}
8          </li>
9        {% endfor %}
10     </ul>
11   </nav>
12 {% endif %}
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Na ukázka kódu 29 je zdrojový kód pro navigaci. Celá navigace je obalená podmínkou *{% if navigation %}*. To značí, že k vykreslení navigace dochází jen v případě, že existuje. Následně pomocí smyčky *{% for item in items %}* na řádku 4 se vypisuje jeden odkaz po druhém.

4.2.2.2 Zdrojový kód v SASS

Ukázka kódu 30 – Zdrojový kód v SASS pro Hlavičku

```
1  .header
2      margin-bottom: $spacing * 2 // $spacing == 24 px
3
4      &-logo
5          max-width: 500px
6          margin: 20px auto
7
8          &-link:hover
9
10             .header
11                 &-image
12                     transform: scale(1.01)
13
14     &-image
15         transition: transform $transition ease // $transition == 0.2 s
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Pro docílení grafického vzhledu hlavičky z obrázek 5, je přidán soubor *header.sass*, který obsahuje zdrojový kód v ukázka kódu 30. Díky tomuto zdrojovému kódu stylizuje hlavičku na stránce do potřebné podoby.

Pod prvkem obalený ve třídě *.header* se pomocí zápisu na 2. řádku vytváří mezera 48 px, jelikož proměnná *$spacing* obsahuje hodnotu *24 px* a následným vynásobením číslem *2* se získá potřebná hodnota.

Zápis *&-logo* na 4. řádku se po překompilování stává *.header-logo*, protože je vnořen do třídy *.header* pomocí odsazení. Logo díky zápisu na 5. řádku je široký maximálně 500 px. Díky vlastnosti *margin* a jeho hodnotám na 6. řádku je shora a zdola mezera 20 px a je zarovnán na střed díky zápisu *auto*. Při přejetí myší na logo, se jeho původní velikost proporcionálně zvětší díky zápisu na řádce 12, kde je vlastnost *transform* s hodnotou *scale(1.01)*. Tento zápis je po kompilaci ve třídě *.header-logo-link:hover .header-image* díky zápisu na řádcích 8–11.

Pro plynulý přechod při zvětšování loga slouží zápis na 15. řádce, kde vlastnost *transition* umožňuje přidat, nebo upravit přechod pro vlastnost *transform*. Dále proměnná *$transition* nese časovou hodnotu *0.2 s* a říká, jak dlouho přechod bude trvat. Poslední hodnota *ease* představuje, jakým způsobem se bude chovat přechod při daném čase. Tedy v tomto konkrétním příkladě dojde na začátku přechodu k pomalé transformaci, následně se zrychlí a ke konci přechodu se zase zpomalí.

4.2.3 Footer

Footer neboli také patička je vyobrazena na obrázek 6. Na levé straně najdeme ikonky, ne které po kliknutí vedou na danou sociální síť autora blogu. Na pravé stráně je copyright a popis kým byla stránka vytvořena. Sociální sítě si může uživatel upravovat v rozhraní *Wordpress*, kam uvádí url a název své sociální sítě.

Obrázek 6 – Patička stránky

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

4.2.3.1 Zdrojový kód v Twig

Ukázka kódu 31 - Zdrojový kód v Twig pro patičku

```
1  <footer class="footer">
2    <div class="site">
3      <div class="footer-in">
4        <div class="footer-socials">
5          {% for item in social_channels.items %}
6            {% if item.post_title == 'email' %}
7              {% set email = item.url|replace({'https://':''}) %}
8            {% endif %}
9            <div class="footer-social">
10             <a class="footer-social-link"
11                href='{{ item.post_title == "email" ? "mailto:" ~ email
    : item.url }}'
12                target='{{ item.post_title != "email" ? "_blank" }}'
13             >
14               {{ svgArray[item.post_title] ?: item.post_title }}
15             </a>
16           </div>
17         {% endfor %}
18       </div>
19       <div class="footer-copyright">
20         <p class="footer-copyright-text">
21           Copyright {{"now"|date('Y')}}, made by
22           <a class="footer-copyright-link"
23              target="_blank"
24              href=https://github.com/Le-David
25            >
26              Quang Dat Le
27            </a>
28         </p>
29       </div>
30     </div>
31   </div>
32 </footer>
```

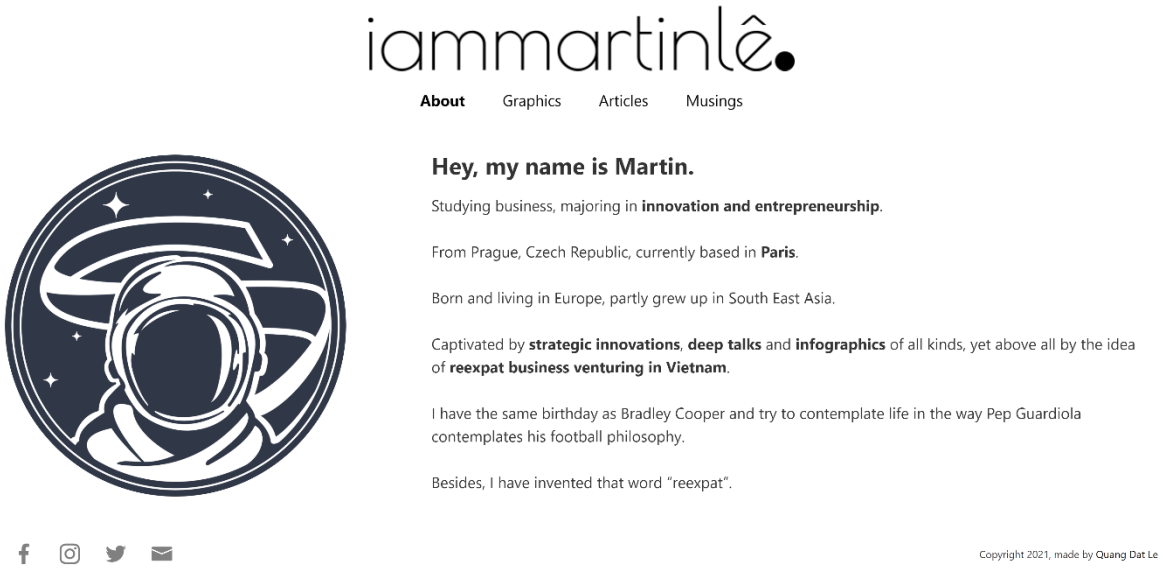
Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

V souboru *footer.twig* je pole obsahující hodnoty sociálních sítí uvnitř *social_channels.items*, které si uživatel nastavuje v rozhraní Wordpress. Pomocí smyčky na řádku 5–17 jsou vypisovány na stránku. Ikonky pro sociální sítě jsou uložené v objektu *svgArray*, kde klíčem je název sociální sítě a hodnota je SVG ikonka. Napsáním správného názvu sociální sítě v rozhraní *Wordpress* se automaticky přiřadí správná ikonka na řádku 14 a naopak pokud neexistuje ikonka s danou sociální sítí, pak se díky ternárnímu operátoru vypíše název. Na řádcích 19–29 se vypisuje copyright s aktuálním rokem (řádek 21) a jméno autora, který vyvíjel stránku s prokliknutelným odkazem.

4.2.4 About

Stránka *About* jak naznačuje obrázek 7, vypadá celkem jednoduše na vývoj. Dokonce by se dalo i říct, že již existuje *Gutenberg blok*, který by vykresloval obrázek na levé straně a text na pravé straně. Samozřejmě, že existuje, ale neobsahuje nastavitelné parametry podle, kterých si uživatel přeje vykreslit obrázek a obsah tímto způsobem. Proto je vytvořen vlastní *Gutenberg blok*, který bude více popsán v podkapitole 4.2.4.2 Vytvoření vlastního Gutenberg bloku.

Obrázek 7 - Úvodní stránka: „About“



Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

4.2.4.1 Zdrojový kód v Twig

Šablona *frontpage.twig* obsahuje jen pár řádku kódu. Na 1. řádku je *{% extends "base.twig" %}*, díky nímž se automaticky na stránce již vykreslí hlavička a patička, aniž bychom museli explicitně volat další funkce k tomu určené. A pro vložení obsahu na stránku se využívá *{% block content %}* na 3. řádku, který je již definován v *base.twig*. Uvnitř tohoto *blocku* se vypíšou veškeré *Gutenberg bloky*, které uživatel přidá v rozhraní *Wordpress*, pomocí zápisu na 4. řádku.

Ukázka kódu 32 - Zdrojový kód v Twig pro About

```
1  {% extends "base.twig" %}
2
3  {% block content %}
4    {{ post.get_content }}
5  {% endblock %}
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

4.2.4.2 Vytvoření vlastního Gutenberg bloku

K vytvoření vlastního *Gutenberg bloku* je potřeba ho nejdříve registrovat ve *functions.php*. Pro lepší přehlednost je registrace tohoto bloku ve vlastním souboru *src/gutenberg_blocks/asideMediaContent.php* a ve *functions.php* v ukázka kódu 33 je kód pomocí jímž se odkazuje na daný soubor.

Ukázka kódu 33 - Reference na soubor asideMediaContent.php v souboru functions.php

```
require get_template_directory() . '/src/gutenberg_blocks/asideMediaContent.php';
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Soubor *asideMediaContent.php* obsahuje zdrojový kód z ukázka kódu 34. Funkce *aside_media_content()* z 1. řádku v sobě obsahuje tři části.

1. Registrace javascriptového souboru na řádcích 3–7.
2. Registrace souboru pro kaskádové styly určené pro editor na řádcích 9–13.
3. Registrace *Gutenberg bloku* na řádcích 15–22.

Javascriptový soubor z bodu 1 bude sloužit k vytvoření samotného *Gutenberg bloku* v editoru *Wordpress.* Jedná se o *index.js* soubor a zde se s pomocí *React* a javascriptových knihoven připravené od *Wordpress,* vytvoří prvky, kam bude moct uživatel vkládat svůj obsah, obrázek, nadpis a text.

Soubor pro kaskádové styly z bodu 2 slouží k stylizaci daných prvků pro daný *blok.* Zde soubor nese název *styleEditor.css.* Výhodou je, že je možné nastylovat blok v editoru tak, aby vypadal podobněji, či stejně jako na stránce díky tomuto připravenému souboru

V neposlední řadě pro registraci vlastního *Gutenberg bloku* je třeba skloubit již zmíněné soubory, tedy javascriptový a stylizační soubor. Nakonec je třeba říct *Wordpress,* aby volal funkci pro vytvoření vlastního *Gutenberg bloku* na řádce 25.

Ukázka kódu 34 - Zdrojový kód v PHP pro vytvoření Gutenberg bloku

```
1  function aside_media_content() {
2
3    wp_register_script(
4      'aside-media-content-script',
5      get_template_directory_uri() . '/build/scripts/index.js',
6      array( 'wp-i18n', 'wp-blocks', 'wp-block-editor', 'wp-
   components', 'wp-editor' )
7    );
8
9    wp_register_style(
10     'aside-media-content-styleEditor',
11     get_template_directory_uri() . '/build/styles/styleEditor.css',
12     array( 'wp-edit-blocks' )
13   );
14
15   register_block_type(
16     'gutenberg-block/aside-media-content',
17      array(
18        'apiVersion' => 2,
19        'editor_script' => 'aside-media-content-script',
20        'editor_style' => 'aside-media-content-styleEditor',
21      )
22   );
23 }
24
25 add_action( 'init', 'aside_media_content' );
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Zdrojový kód uvnitř *index.js* je rozdělen do čtyř ukázek kódu (ukázka kódu 35, ukázka kódu 36, ukázka kódu 37 a ukázka kódu 38) pro lepší přehlednost.

Ukázka kódu 35 - Zdrojový kód v index.js. Deklarace knihoven a proměnné.

```
1  const { __ } = wp.i18n
2  const { registerBlockType } = wp.blocks
3  const { IconButton } = wp.components
4  const {
5    MediaUpload,
6    MediaUploadCheck,
7    RichText,
8    useBlockProps
9  } = wp.blockEditor
10
11 const ALLOWED_MEDIA_TYPES = [ 'image' ];
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

Na začátku kódu v ukázka kódu 35 na řádcích 1–9 jsou deklarovány knihovny a na řádce 11 je deklarována proměnná.

Ukázka kódu 36 - Zdrojový kód v index.js. Registrace Gutenberg bloku

```
12 registerBlockType('gutenberg-block/aside-media-content', {
13   apiVersion: 2,
14   title: 'Aside media content',
15   description: 'Block to generate a custom aside media content',
16   icon: 'align-left',
17   category: 'layout',
18   keywords: [ __( 'media' ), __( 'image' ), __( 'photo' ), __( 'conten
   t' ), __( 'text' ) ],
19
20   attributes: {
21     title: {
22       type: 'string',
23       source: 'html',
24       selector: 'h2'
25     },
26     content: {
27       type: 'string',
28       source: 'html',
29       selector: 'p'
30     },
31     media: {
32       type: 'string',
33       default: 'https://via.placeholder.com/710',
34     }
35   },
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Na řádce 12 v ukázka kódu 36 registrujeme vlastní Gutenberg blok a uvnitř něho deklarujeme několik základních hodnot jako je název, popis, ikonka, kategorie, klíčová slova, které jsou důležitými informacemi v rozhraní editoru a atributy pro jednotlivé prvky jako je *title, content, media*.

Ukázka kódu 37 - Zdrojový kód v index.js. Edit funkce.

```
36 edit: (props) => {
37
38   const {
39     attributes: {
40       title,
41       content,
42       media,
43     },
44     setAttributes,
45   } = props
46
47   const blockProps = useBlockProps();
48
```

```
49   function onSelectImage(newMedia){
50     setAttributes({ media: newMedia.sizes.full.url })
51   }
52   return ([
53     <div { ...blockProps} className={ `${blockProps.className} asideMe
   diaContent` }>
54       <MediaUploadCheck>
55         <MediaUpload
56           onSelect={ onSelectImage }
57           allowedTypes={ ALLOWED_MEDIA_TYPES }
58           value={ media }
59           render={ ( { open } ) => {
60             return <div className="asideMediaContentEditor-
   mediaUpload asideMediaContent-media">
61               <img className="asideMediaContentEditor-mediaUpload-
   image" src={ media }/>
62               <IconButton
63                 onClick={ open }
64                 icon="upload"
65                 className="asideMediaContentEditor-mediaUpload-
   button editor-media-placeholder__button is-button is-default is-large"
66               >
67                 Upload/edit image
68               </IconButton>
69             </div>
70           } }
71         />
72       </MediaUploadCheck>
73       <div className="asideMediaContentEditor-
   content asideMediaContent-content">
74         <RichText
75           tagName="h2"
76           value={ title }
77           placeholder="Add title..."
78           onChange={ (title) => setAttributes({ title }) }
79           className="asideMediaContent-title"
80         />
81         <RichText
82           tagName="p"
83           value={ content }
84           placeholder="Add content..."
85           onChange={ (content) => setAttributes({ content }) }
86           className="asideMediaContent-text"
87         />
88       </div>
89     </div>
90   ]) },
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Edit funkce v ukázka kódu 37 reprezentuje fázi, během které uživatel upravuje, přidává či odstraňuje svůj obsah pro stránku. Zde se již využívají již hotové *React* knihovny s přidáním vlastností a vlastních hodnot.

Pomocí *MediaUploadCheck* začínající na řádce 54 a končící na řádce 72 se kontroluje, jestli je uživatel oprávněn k nahrávání obrázků. *MediaUpload* z řádků 55–71 umožňuje uživateli nahrát obrázek, a to pomocí tlačítka *IconButton* definovaný na řádcích 62–68. Obrázek, který nahraje uživatel do editoru je následně uložen do proměnné *media* z řádky 58.

K přidání nadpisu slouží *RichText* na řádcích 74–80, který je v editoru obalen do HTML elementu *H2*. Pro samotný text slouží také *RichText*, který je na rozdíl od nadpisu obalen do HTML elementu *p*. Text pro nadpis je po vložení uložen v proměnné *title* na řádce 76 a samotný text do proměnné *content* na řádce 83.

Ukázka kódu 38 - Zdrojový kód v index.js. Save funkce.

```
91 save: (props) => {
92
93   const {
94     attributes: {
95       title,
96       content,
97       media,
98     },
99   } = props
100
101  const blockProps = useBlockProps.save();
102
103  return (
104    <div { ...blockProps} className={ `${blockProps.className} asideMe
   diaContent` }>
105      <div className="asideMediaContent-media">
106        <img className="asideMediaContent-
   image" src={ media } alt={ title }/>
107      </div>
108      <div className="asideMediaContent-content">
109        <RichText.Content
110          tagName="h2"
111          value={ title }
112          className="asideMediaContent-title"
113        />
114        <RichText.Content
115          tagName="p"
116          value={ content }
117          className="asideMediaContent-text"
118        />
119      </div>
120    </div>
121  )
122},
123})
```

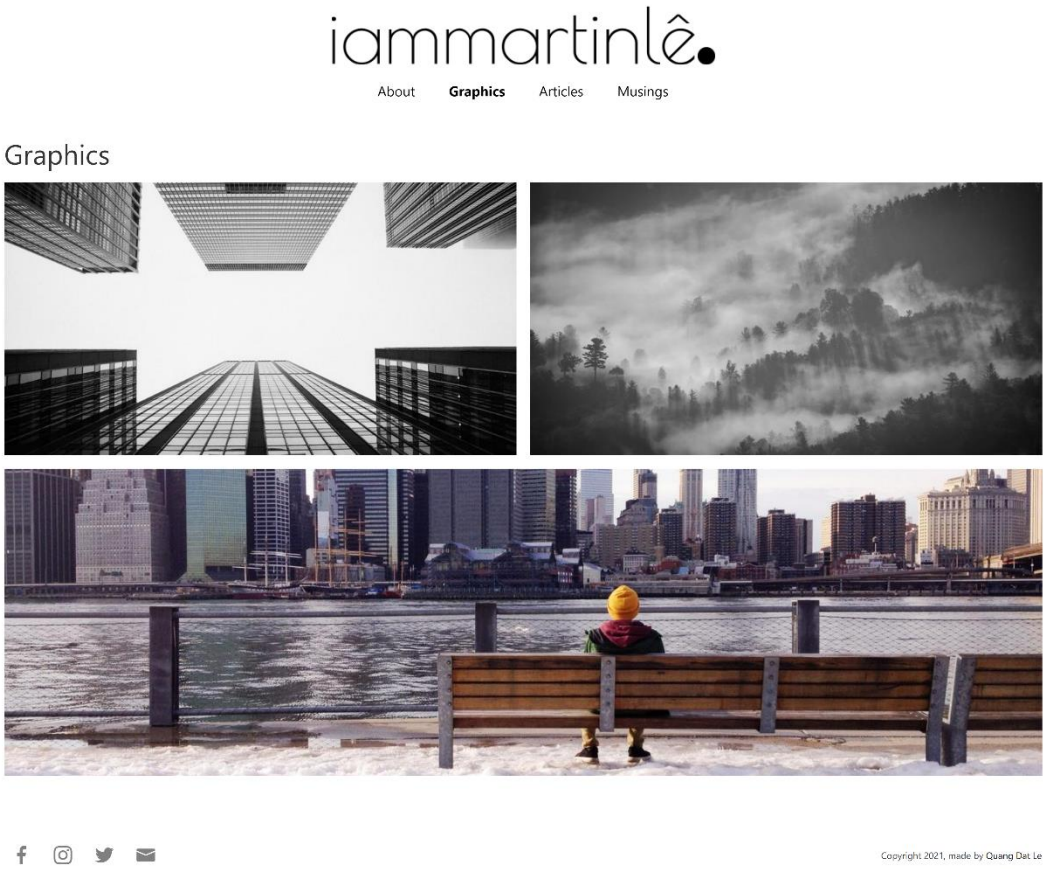
Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Funkce *save* je fáze, ve které uživatel uloží obsah pro jeho následné vykreslení na stránce. V ukázka kódu 38 je část kódu, která se použije pro vykreslení obsahu na stránce. Tak jako v *Edit* funkci, *Save* funkce obsahuje část pro vykreslení obrázku na řádcích 107–109, kde zdrojem obrázku je hodnota uvnitř proměnné *media*. Pro nadpis je určen kód na řádcích 110–115, kde text pro nadpis je v proměnné *title* a pro obyčejný text je vyhrazen kód na řádcích 116–120, kde obsah tohoto textu je v proměnné *content*.

4.2.5 Graphics

Stránka *Graphics*, nebyla třeba vůbec kódovat, ani vytvářet žádnou šablonu. Gutenberg blok ve svém rozhraní již nabízí komponentu, s názvem *Gallery* do kterého je možné vkládat obrázky podle toho, jak uživatel potřebuje.

Obrázek 8 – Stránka galerie: "Graphics"



Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

4.2.6 Articles

Articles neboli stránka s výpisem článků byla spolu se stránkou *About* pomalu jedny z nejdelších na vývoj. Stránka má trochu jiné rozložení oproti stránce *About*. Jak ukazuje obrázek 9, na levé stráně je postranní panel, kde nalezneme výpis veškerých dostupných kategorií článků a po kliknutí na některou z nich se přesměrovává na stránku, kde se vypíšou články jen z dané kategorie. Uprostřed nalezneme výpis veškerých článků patřící ke stránce *Articles* seřazené od nejstaršího po nejnovější s tím, že každý článek obsahuje nadpis, datum publikace a přiřazené kategorie. Kliknutím na některých z článků pak vede na detail článků, a kde je ve stejném rozpoložení vykreslen i obsah článku. Více ohledně článku je popsáno v podkapitole 4.2.6.3 Detail article.

Obrázek 9 - Stránka s články: "Articles"

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

4.2.6.1 Vytvoření Articles v rozhraní Wordpress

Běžnou praxí pro vytvoření místa, kde může uživatel přidávat svoje články by bylo vytvořit nový tzv. *Posts*. *Wordpress* však má již jeden takový nastavený, ale je mu třeba ještě nastavit přesnější název. Na ukázka kódu 39 je kód, díky kterému dochází k přejmenování původního *„Posts“* na *„Articles*“. Aby to však bylo možné, je třeba deklarovat globální proměnné pro *menu* na řádce 2, *submenu* na řádce 3 a *wp_post_types* na řádce 11. Následně na řádcích 4–7 a 12–22 dochází k přepsání původních názvů na nové. Na závěr je třeba ještě volat dané funkce pro přepsání názvů pomocí funkce *add_action* na řádcích 25 a 26.

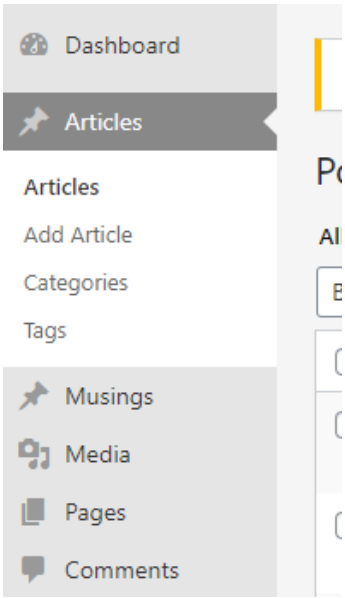
Ukázka kódu 39 - Přejmenování "Posts" na "Articles"

```php
function change_post_menu_label() {
    global $menu;
    global $submenu;
    $menu[5][0] = 'Articles';
    $submenu['edit.php'][5][0] = 'Articles';
    $submenu['edit.php'][10][0] = 'Add Article';
    $submenu['edit.php'][16][0] = 'Tags';
}

function change_post_object_label() {
    global $wp_post_types;
    $labels = &$wp_post_types['post']->labels;
    $labels->name = 'Articles';
    $labels->singular_name = 'Article';
    $labels->add_new = 'Add Article';
    $labels->add_new_item = 'Add Article';
    $labels->edit_item = 'Edit Article';
    $labels->new_item = 'Article';
    $labels->view_item = 'View Article';
    $labels->search_items = 'Search Articles';
    $labels->not_found = 'No Articles found';
    $labels->not_found_in_trash = 'No Articles found in Trash';
}

add_action( 'admin_menu', 'change_post_menu_label' );
add_action( 'init', 'change_post_object_label' );
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Rozhraní pro *Articles* nyní vypadá tak, jak ukazuje obrázek 10.

Obrázek 10 - Rozhraní pro Articles



Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

4.2.6.2 Zdrojový kód v Twig

Zdrojový kód pro šablonu *template-articles.twig* je rozdělen na ukázka kódu 40 a ukázka kódu 41 pro lepší přehlednost.

Ukázka kódu 40 - Zdrojový kód v Twig pro Articles #1

```
1  {% extends "base.twig" %}
2  {% set posts = related_posts|default([]) %}
3
4  {% block fullPage %}
5    <main class="post post-type-articles">
6
7    {% include 'sidebar.twig' with {
8      categories: categories,
9    } %}
10   {% block article_content %}
11     <article class="articles-list" id="articles-list">
12     {% for post in posts %}
13       {% set post_categories = post.terms('category') %}
14       {% set uncategorized %}
15         {% for post_category in post_categories %}
16           {% if post_category.name == 'Uncategorized' %}
17             true
18           {% endif %}
19         {% endfor %}
20       {% endset %}
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Ukázka kódu 40 začíná kódem *{% extends "base.twig" %}*, díky němuž se přepisuje, či upravuje obsah prvků, které jsou v souboru *base.twig* definovány. Na 2. řádku se do proměnné *posts* ukládají veškeré dostupné články, které budou sloužit k výpisu na stránku. Uvnitř *{% block fullPage %}* začínající na 4. řádku je veškerý kód sloužící k vykreslení obsahu na stránku.

Pro vykreslení postranního panelu s kategoriemi slouží kód na 7–9 řádku, kde se využívá šablony *sidebar.twig* a které se zároveň předává parametr *categories*, která obsahuje pole všech kategorií článků.

Na 10. řádku se deklaruje nový *block {% block article_content %}*, který není v původním *base.twig*. Tento *block* slouží jak pro výpis článků na stránce *Articles*, tak zároveň i pro jednotlivé detaily článku.

Na 12 řádku je připravená *for* smyčka pro výpis jednotlivých článku. Na řádku 13 jsou nejdříve získány kategorie každého článku. Článkům, ke kterým nebyla přiřazena žádná kategorie jsou automaticky umístěny v kategorii *Uncategorized*, tuto kategorii není však třeba vypisovat. Na

řádcích 15–20 je kód, který slouží pro použití v podmínce, tak aby se daná kategorie nevypisovala.

Ukázka kódu 41 - Zdrojový kód v Twig pro Articles #2

```
21 <article class="articles-in">
22   <a href="/articles/{{ post.slug }}" class="articles-in-link">
23     <h1 class="articles-in-title">{{ post.post_title }}</h1>
24     <time class="articles-in-
   date" datetime="{{ (post.post_date)|date('Y-m-
   d') }}">{{ (post.post_date)|date('F j, Y') }}</time>
25   </a>
26   {% if post_categories and ('true' not in uncategorized) %}
27     <ul class="articles-in-categories">
28       <li class="articles-in-category-icon">
29         <svg class="shape shape-tag articles-in-category-icon-
   in" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 436.4 436.4">
30           <path d="M340.4 23.2h-134a32 32 0 0 0-
   22.6 9.4L9.4 207a32 32 0 0 0 0 45.2l134 134a32 32 0 0 0 45.2 0L363 211
   .8a32 32 0 0 0 9.4-22.6v-134a32 32 0 0 0-32-32zm-56 112a24 24 0 1 1 0-
   48 24 24 0 0 1 0 48z"/>
31           <path d="M404.4 55.2v149a27.8 27.8 0 0 1-
   8.2 19.6L219 401.1l2.7 2.7a32 32 0 0 0 45.3 0l160-160a32 32 0 0 0 9.4-
   22.6v-134a32 32 0 0 0-32-32z"/>
32         </svg>
33       </li>
34       {% for post_category in post_categories %}
35         <li class="articles-in-category">
36           <a href="/articles/category/{{ post_category.slug }}" class=
   "articles-in-category-link">
37             <h5 class="articles-in-category-
   title">{{ post_category.name }}</h5>
38           </a>
39         </li>
40       {% endfor %}
41     </ul>
42   {% endif %}
43 </article>
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Ukázka kódu 41 už slouží k vypsání veškerých důležitých prvků článku. Na řádce 23 se vypisuje titulek a na řádce 24 datum publikace. Na 26. řádce začíná podmínka, která vykreslí kategorie jen v případě, že kategorie vůbec existují a zároveň pokud neobsahuje název kategorie *Uncategorized*.

4.2.6.3 Detail article

Pro detail článku je využitá šablona s názvem *single.twig*. Detail článku, jak ukazuje obrázek 11, vykresluje obsah ve stejném rozložení jako na původní stránce *Articles*, tedy postranní panel s kategoriemi přiřazené k článku nalevo a samotný článek s veškerým obsahem na středu.

Obrázek 11 - Detail article

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

4.2.6.3.1 Zdrojový kód v Twig pro detail článku

Ukázka kódu 42 - Zdrojový kód v Twig pro detail článku

```
1  {% extends "template-articles.twig" %}
2
3  {% block article_content %}
4    <article class="article" id="article">
5      <h1 class="article-title">{{ post.post_title }}</h1>
6        <time class="article-
   date" datetime="{{ (post.post_date)|date('Y-m-
   d') }}">{{ (post.post_date)|date('F j, Y') }}</time>
7      {% if post.thumbnail %}
8        <img class="article-thumbnail"
9             src="{{ post.thumbnail.src }}"
              alt="{{ post.thumbnail.alt ?: 'featured post thumbnail' }}"
        />
10     {% endif %}
11     {{ post.get_content }}
12   </article>
13 {% endblock %}
```

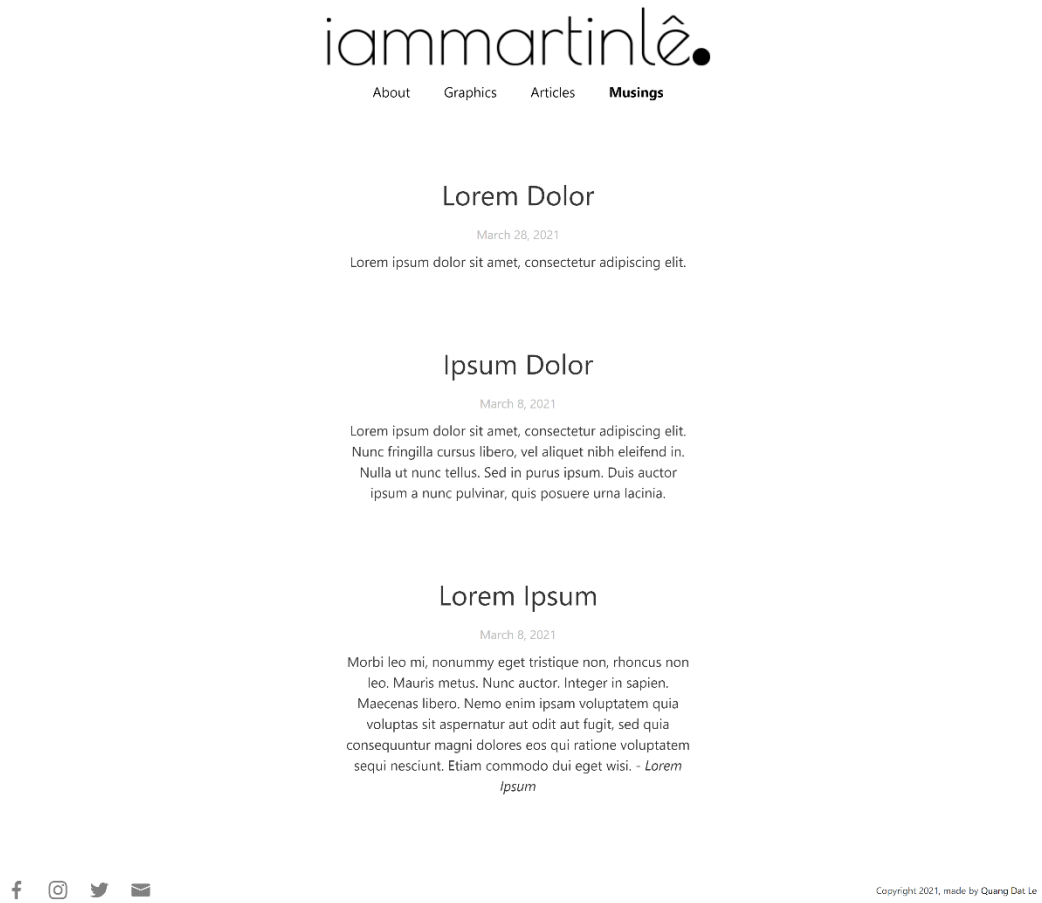
Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Díky 1. řádku v ukázka kódu 42 je zajištěno, že se využije stejného rozložení stránky jako na stránce *Articles.* Na 3. řádce se využívá *{% block article_content %},* který je právě definován v šabloně *template-articles.twig* a dovnitř něho se vkládá nadpis, datum publikace, nepovinný obrázek a text článku.

4.2.7 Musings

Stránka *Musings* je velice podobná stránce *Articles*, avšak se liší formátem vykreslení na stránce. Na obrázek 12 je vidět výpis jednotlivých *Musings* s nadpisem, datumem publikace a na rozdíl od *Articles* i textem. Záměrem *Musings* je rovnou vypsat veškerý dostupný obsah a není tedy třeba žádného odkazu na detail *Musing*.

Obrázek 12 - Stránka "Musings"



Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

4.2.7.1 Vytvoření Musings v rozhraní Wordpress

Pro *Musings* byla vytvořeno tzv. CPT (*Custom post type*[12]) v souboru *functions.php*, aby uživatel v rozhraní *Wordpress* mohl přidávat vlastní posty stejným způsobem jako v *Articles*.

Ukázka kódu 43 - Vytvoření "Custom post type" Musings

```
1  $musing_labels = array(
2    'name'               => _x( 'Musings', 'post type general name' ),
3    'singular_name'      => _x( 'Musing', 'post type singular name' ),
4    'add_new'            => _x( 'Add musing', 'musing' ),
5    'add_new_item'       => __( 'Add New Musing' ),
6    'edit_item'          => __( 'Edit Musing' ),
7    'new_item'           => __( 'New Musing' ),
8    'all_items'          => __( 'All Musings' ),
9    'view_item'          => __( 'View Musing' ),
10   'search_items'       => __( 'Search Musings' ),
11   'not_found'          => __( 'No musings found' ),
12   'not_found_in_trash' => __( 'No musings found in the Trash' ),
13   'menu_name'          => 'Musings'
14 );
15
16 $musing_args = array(
17   'labels'        => $musing_labels,
18   'description'   => 'Posts for musings',
19   'public'        => true,
20   'menu_position' => 5,
21   'supports'      => array( 'title', 'editor', 'thumbnail', 'excerpt'
   ),
22   'show_in_rest'  => true, // allow gutenberg
23   'rewrite'       => array( 'slug' => 'musings', 'with_front' => false
    )
24 );
25
26 register_post_type( 'musings', $musing_args );
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

Na ukázka kódu 43 se registruje CPT *Musings* na řádce 26. Funkce získává dva parametry, *musings* což je název, pod kterým je nalezen v databázi a *$musing_args*, což je pole s informacemi na řádce 16–24 nutné pro založení CPT. Aby CPT v rozhraní nesl správné názvy a štítky, je mu předána na řádce 17 *$musing_labels*, což je pole s nastavenými názvy a štítky na řádcích 1–14.

[12] Custom post type – vlastní post typ, také jako CPT.

4.2.7.2 Zdrojový kód v Twig

Kód pro vypsání obsahu je velice podobný jako v *Articles* a je v šabloně *template-musings.twig*. V ukázka kódu 44 jako obvykle, je díky 1. řádku možné použít *{% block fullPage %}* k vložení obsahu pro Musings. Na 2. řádku se získávají veškeré dostupné posty patřící k *Musings* a ukládají se do proměnné *posts*. Následně se používají ve *for* smyčce na řádku 6, uvnitř kterého se vypisuje jeden post po druhém.

Ukázka kódu 44 - Zdrojový kód v Twig pro Musings

```
1  {% extends "base.twig" %}
2  {% set posts = related_posts|default([]) %}
3
4  {% block fullPage %}
5    <article class="post post-type-musings">
6      {% for post in posts %}
7        <article class="musing">
8          <h1 class="musing-title">{{ post.post_title }}</h1>
9          <time class="musing-
   date" datetime="{{(post.post_date)|date('Y-m-
   d')}}">{{(post.post_date)|date('F j, Y')}}</time>
10         {{ post.post_content }}
11       </article>
12     {% endfor %}
13   </article>
14 {% endblock %}
```

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021), (Visual Studio Code - Code Editing. Redefined, 2021)

5 Závěr

Závěrem bych chtěl shrnout svoji bakalářskou práci, ve které se zabývám vývojem webové stránky. Předpokladem projektu bylo vytvořit čtyři stránky s ohledem na autorovy preference. Pro tento projekt jsou využity technologie, které jsou podrobněji popsány v teoretické části za použití odborné literatury a zejména online dokumentací. Praktická část je rozdělena na dvě části. První část popisuje způsob implementace a instalace potřebných technologií pro vývoj a tvorbu webových stránek. Druhá část se věnuje popisu procesu vývoje webových stránek. Tato sekce je rozdělena na čtyři části, ve kterých každá jedna část odpovídá jedné stránce. Jednotlivé části obsahují obrázky s vizuální podobou stránky a dále jsou doplněny zdrojovým kódem a popisem. V příloze jsou dostupné obrázky jednotlivých stránek a částí.

Tvorbou těchto webových stránek, jsem si potvrdil, že ačkoliv jsem se naučil postupy, jak vytvořit web složením několika technologií dohromady, nejsem si v nich zcela jistý a některé mám ještě neúplně probádané. Největší problémy při vývoji nastávaly při tvorbě vlastního *Gutenberg bloku.* Popis pro jeho tvorbu je dostupná v online dokumentaci, ze které jsem měl však často problém chápat dané postupy. V těchto případech jsem se snažil hledat další články, dokumentace a někdy i videa, která mi pomohla k lepšímu porozumění a řešení problému. Využil jsem i pomoci od svých kolegů z práce, kteří jsou v oboru zkušenější a mají o mnohem více znalostí, které jsem od nich mohl čerpat a použít při vývoji stránek.

Výsledkem této práce jsou osobní webové stránky a blog. Zdrojový kód této práce je veřejně dostupný na adrese https://github.com/Le-David/bachelor-thesis. Repositář neobsahuje potřebné knihovny pro lokální spuštění projektu, avšak případný zájemce může pomocí několika příkazů popsaných v této bakalářské práci spustit projekt na svém lokálním zařízení. (Bachelor-thesis, 2021)

Vývoj webových stránek popsané v této práci byl pro mě velkým přínosem z hlediska získání nových znalostí, dovedností jako je efektivnější čtení z dokumentací nebo hledání a řešení problémů, při kterých se často objevují. Tento projekt počítá s tím, že se bude v budoucnu dále vyvíjet, a je zde prostor pro další vývoj stránek či jejich částí.

6 Summary

At the end I would like to summarize my bachelor thesis in which I develop a website. The premise of the project was to create four pages with respect to the author's preferences. Technologies for this project are described more detailed in the theoretical part using the literature and especially online documentation. The practical part is divided into two parts. The first part describes how to implement and install the necessary technologies for the development and creation of website. The second part describes the process of website development. This section is divided into four sections, in which each section corresponds to one page. The individual sections contain images with a visual appearance of the page and they are further supplemented by source code and descriptions. In attachment are pictures of individual pages and parts.

By creating this website, I have confirmed that although I have learned how to create a website by putting several technologies together, I am not entirely confident to work with them and I should continue to learn them. The biggest development problems occurred when it came to creating own Gutenberg block. A description for its creation is available in the online documentation. I often had trouble to understand some methods in the documentation. In these cases, I tried to look for other articles, documentation and sometimes videos that helped me to better understand and solve the problem. I also used the help of my colleagues from work who are more experienced in the field and have much more knowledge, so they practically helped me to solve some issues.

The result of this work is a personal website and blog. The source code of this work is publicly available at https://github.com/Le-David/bachelor-thesis. The repository does not contain the necessary libraries for local run of the project, but a potential applicant can use several commands described in this bachelor's thesis to run the project on their local device. (Bachelor thesis, 2021)

The development of the website described in this work was a great benefit for me in terms of gaining new knowledge, skills such as more effective reading from documentation or finding and solving problems that often appears. This project counts on further development in the future and there is space for further development of the site or its parts.

I Seznam použité literatury

Total number of Websites - Internet Live Stats. Retrieved March 14, 2021, from https://www.internetlivestats.com/total-number-of-websites/

3 Reasons Why You Need A Personal Website. (2016). Forbes. Retrieved March 14, 2021, from https://www.forbes.com/sites/laurencebradford/2016/09/27/3-reasons-why-you-need-a-website/?sh=726725f62460

What Is the Average Web Developer Salary? Here's What Data Says for 2021. (2020). Retrieved March 14, 2021, from https://kinsta.com/blog/web-developer-salary/

What is: Content Management System (CMS). (2009 - 2021). Wpbeginner. Retrieved March 14, 2021, from https://www.wpbeginner.com/glossary/content-management-system-cms/

What Is WordPress? Explained for Beginners. (2021). Kinsta. Retrieved March 15, 2021, from https://kinsta.com/knowledgebase/what-is-wordpress/

Sabin-Wilson, L. (2015). *WordPress For Dummies* (7th). Wiley.

McHarry, S. (2013). *WordPress To Go - How To Build A WordPress Website On Your Own Domain, From Scratch, Even If You Are A Complete Beginner*. CreateSpace Independent Publishing Platform

About. (2003). WordPress. Retrieved March 15, 2021, from https://wordpress.org/about/

Template Files. (c2003). WordPress. Retrieved March 15, 2021, from https://developer.wordpress.org/themes/basics/template-files/

Timber for WordPress | Upstatement. (2008). Upstatement. Retrieved March 15, 2021, from https://upstatement.com/timber/

Timber Docs for v1 – Timber Documentation. ([2012]). Retrieved March 16, 2021, from https://timber.github.io/docs/

How to Use the New WordPress Gutenberg Block Editor. (2020). Bluehost Resource Center. Retrieved March 21, 2021, from https://www.bluehost.com/resources/how-to-use-the-new-wordpress-gutenberg-block-editor/?utm_source=google&utm_medium=genericsearch&gclid=Cj0KCQjwutaCBhDfARIsAJHWnHsrIpdIoywD9Rs80_kpCbwcTJhuYtc1LtPF-XTyxA9xUYxPrQzkj40aArGNEALw_wcB&gclsrc=aw.ds

Duckett, J. ([2011].). *HTML and CSS: Design and Build Websites: design and build websites*. Wiley.

Duckett, J. (c2014). *JavaScript and JQuery: Interactive Front–End Web Development: interactive front-end web development*. Wiley.

Thomson, L., & Weilling, L. (2016). *PHP and MySQL® Web Development* (5th ed.). Sams.

React – A JavaScript library for building user interfaces. (2021). Retrieved March 26, 2021, from https://reactjs.org/

Twig - The flexible, fast, and secure PHP template engine. (c2010-2021). Retrieved March 17, 2021, from https://twig.symfony.com/

Sass: Syntactically Awesome Style Sheets. (c2006–2021). Retrieved March 20, 2021, from https://sass-lang.com/

Docker. (2013). Retrieved March 21, 2021, from https://www.docker.com/

What is npm?. (2011). Node.js. Retrieved March 21, 2021, from https://nodejs.org/en/knowledge/getting-started/npm/what-is-npm/

Introduction - Composer. ([2012]). Composer. Retrieved March 22, 2021, from https://getcomposer.org/doc/00-intro.md

Quickstart: Compose and WordPress. (c2013-2021). Docker Documentation | Docker Documentation. Retrieved March 22, 2021, from https://docs.docker.com/compose/wordpress/

Sass. ([2009]). Npm. Retrieved March 27, 2021, from https://www.npmjs.com/package/sass

@wordpress/scripts. ([2009]). Npm. Retrieved March 27, 2021, from https://www.npmjs.com/package/@wordpress/scripts

Visual Studio Code - Code Editing. Redefined. (2021). Vistual Studio Code. Retrieved April 12, 2021, from https://code.visualstudio.com/

Bachelor-thesis. (2021). GitHub. Retrieved April 11, 2021, from https://github.com/Le-David/bachelor-thesis

II Seznam obrázků

III Seznam zdrojových kódů

IV Seznam Příloh

A Zdrojové kódy

B Obrázky v příloze

A Zdrojové kódy

Celý zdrojový kód k aplikaci je veřejně přístupný na stránce:

https://github.com/Le-David/bachelor-thesis

Příloha A 1 – Zdrojový kód webové stránky „I Am Martin Le“

B Obrázky v příloze

Příloha B 1 - Hlavička stránky

Zdroj: vlastní tvorba

Příloha B 2 - Patička stránky

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

Příloha B 3 - Stránka "About"

iammartinlê.

About Graphics Articles Musings

Hey, my name is Martin.

Studying business, majoring in **innovation and entrepreneurship**.

From Prague, Czech Republic, currently based in **Paris**.

Born and living in Europe, partly grew up in South East Asia.

Captivated by **strategic innovations**, **deep talks** and **infographics** of all kinds, yet above all by the idea of **reexpat business venturing in Vietnam**.

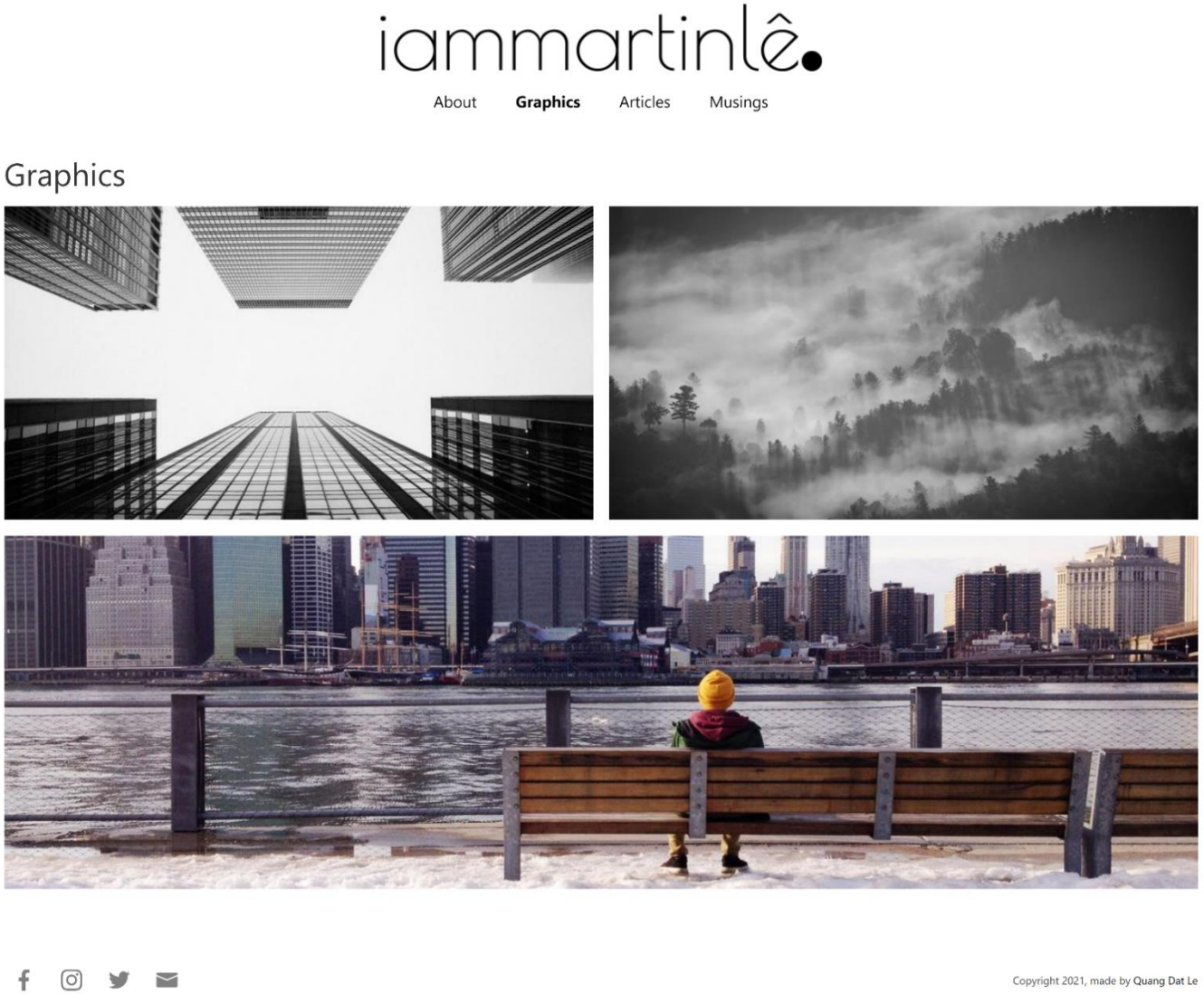
I have the same birthday as Bradley Cooper and try to contemplate life in the way Pep Guardiola contemplates his football philosophy.

Besides, I have invented that word "reexpat".

Copyright 2021, made by Quang Dat Le

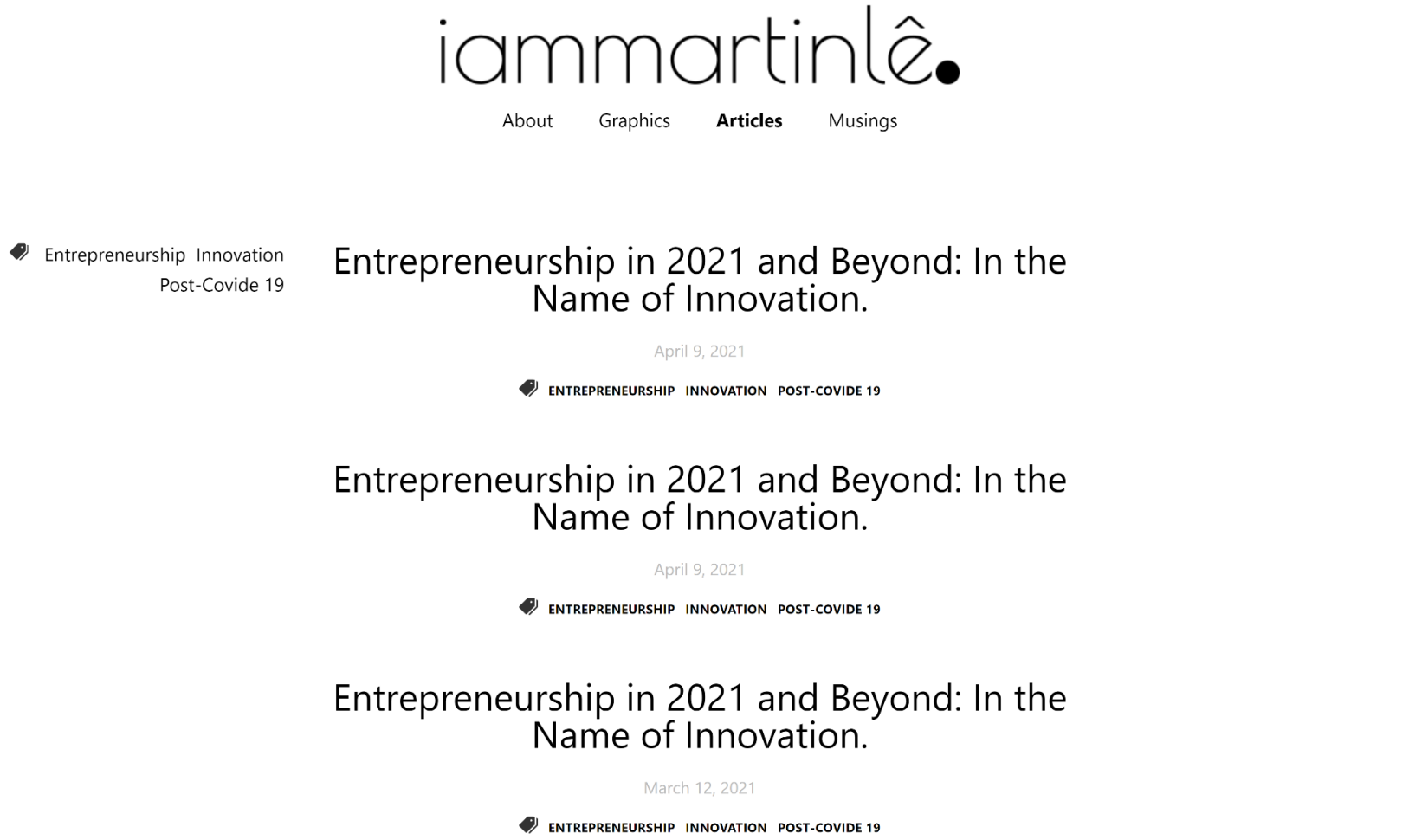
Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

Příloha B 4 - Stránka "Graphics"



Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

Příloha B 5 - Stránka "Articles"



Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

Příloha B 6 - Detail article

iammartinlê.

About Graphics Articles Musings

Entrepreneurship Innovation
Post-Covide 19

Entrepreneurship in 2021 and Beyond: In the Name of Innovation.

April 9, 2021

A new year begins.

Although this text is still written in lockdown and, just like yesterday, Covid-19 has not gone, this moment is ideal to **switch the entrepreneurial mindset to a new beginning** and, at the same time, keep in mind a valuable fact: **2020 has drifted entrepreneurship**.

There is probably no need to go through *"What happened?"* – everybody knows, but the up-to-date question to ask is ***"What now?" – What is the inertia from 2020 going to be like?*** Coming up is **"the new (ab)normal**" – the new era after "the normal" (prior to 2020) and "the abnormal" (2020) that is going to be different from both previous. *Will consumer attitudes, expectations, and standards return to the normal prior to 2020?* Probably not. *Will the cost-cutting be enough to respond to the recession?* Certainly not. **Businesses have been helping society overcome the crisis. Next (thus now), they should be creating post-crisis innovations** as here come many more new, unknown questions. *What is going to be a nonessential purchase? How is online going to be integrated with offline? Who is going to create the workforce? How is the work life going to be organized? ...**What is the business speed going to be? After drifts and inertia, is acceleration going to follow?***

According to McKinsey & Company, the performance gap has widened dramatically between firms and industries, so **acceleration is even a must.** Acceleration is a must because **the whole system is accelerated.** Acceleration is a must because sources of finance for short-term operations dry up.

So *how can businesses accelerate?*

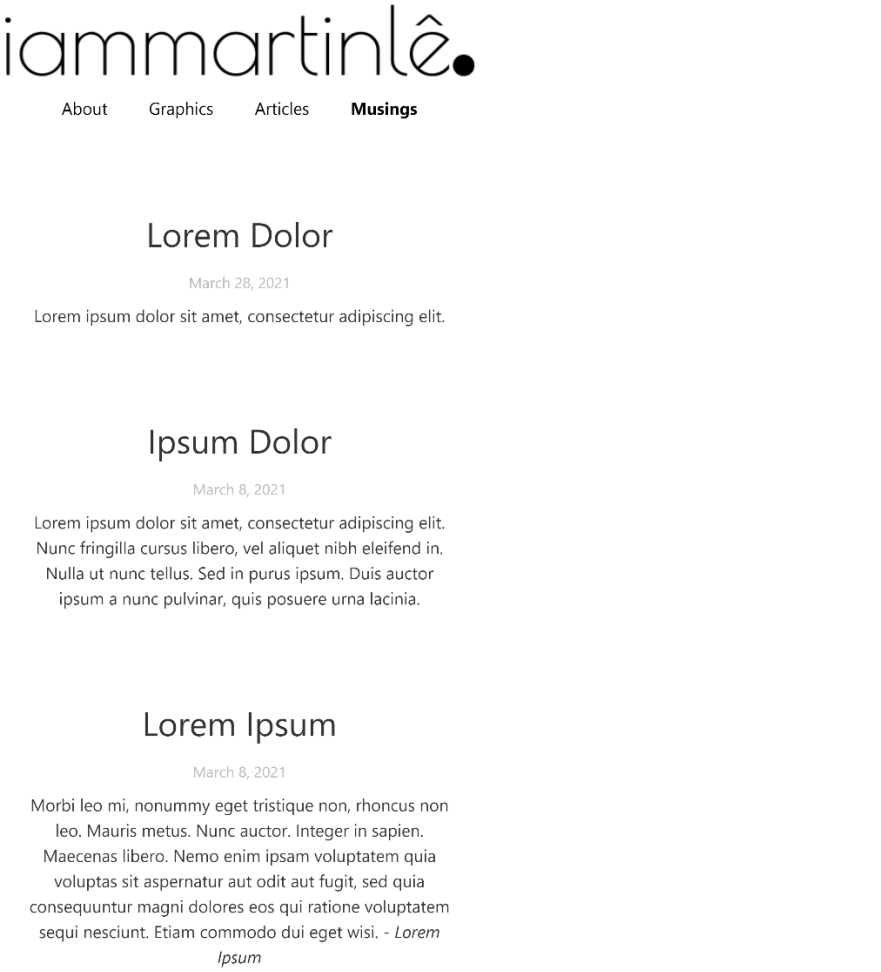
The answer is motivating: **By further proactive (instead of reactive) innovation that is targeted at a large, long-term market beyond crisis, meaning meeting crisis-born, higher standards for innovation-driven businesses that must keep engaging in new entrepreneurial activities – in the new era when innovation and entrepreneurship are democratized.**

Sources: WEF, The Conversation, HBR, McKinsey & Company.

Copyright 2021, made by Quang Dat Le

Zdroj: vlastní tvorba, (Bachelor-thesis, 2021)

Příloha B 7 - Stránka "Musings"



Zdroj: Vlastní tvorba, (Bachelor-thesis, 2021)