Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Šablona pro CMS WordPress

Lukáš Janeček

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Lukáš Janeček

Informatika

Název práce

Šablona pro CMS Wordpress.

Název anglicky

Template for CMS Wordpress.

Cíle práce

Cílem práce je grafický návrh a naprogramování nové šablony pro prostředí CMS WordPress. Šablona bude určena pro internetový občasník a využívat prvků HTML5 a CCS3 a responzivního designu.

Metodika

Dodržování standardů softwarového inženýrství, především UML a WebML.

Doporučený rozsah práce

50 stran

Klíčová slova

wordpress, cms, šablona, vzhled, grafika, HTML5, CSS3, responzivní design

---

Doporučené zdroje informací

KUDLÁČEK, L. – COMPUTER PRESS. WordPress: Podrobný průvodce tvorbou a správou webů. 2013.
ŠESTÁKOVÁ, L. – COMPUTER PRESS. WordPress: Vlastní web bez programování. 2013.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

doc. Ing. Vojtěch Merunka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 20. 2. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 05. 03. 2016

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Šablona pro CMS WordPress" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. 3. 2016 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Vojtěchu Merunkovi, Ph.D. za pomoc s volbou tématu a vedení v průběhu celé práce. Dále bych chtěl poděkovat celé internetové komunitě, která vytváří CMS WordPress a udržuje jeho dokumentaci. Nakonec bych chtěl poděkovat všem, kteří mě během studia a psaní této práce podporovali.

Šablona pro CMS WordPress

Souhrn

Cílem práce je grafický návrh a naprogramování nové šablony pro prostředí CMS WordPress. Šablona bude určena pro internetový občasník a využívat prvků HTML5 a CCS3 a responzivního designu.

V první kapitole práce se zaměřím na obecný popis systému WordPress a jeho historie. Následně se ve druhé kapitole zaměřím na architekturu systému a technologie, které jsou využívány. Ve třetí kapitole popíši strukturu systému rozšíření, čili systémy pro vzhledy a pluginy. V poslední části se budu věnovat samotnému systému šablon.

V praktické části práce pak demonstruji, jak probíhá tvorba jedné šablony, od základního návrhu, přes wireframe až po nasazení do systému.

Klíčová slova: WordPress, CMS, šablona, vzhled, grafika, HTML5, CSS3, responzivní design

Template for CMS WordPress

Summary

Goal of my thesis is to design and create a new template for CMS WordPress for online journal or a blog with use of HTML5 and CSS3. The resulting template will be responsive.

In the first part I will focus on describing WordPress and its history in general. In the next, second, chapter, I will examine the architecture and technologies used by the system. Third chapter will deal with structure of plugin and theme systems. In the fourth and final chapter I will examine the process of creating template in a greater detail.

In the partical demonstration I will show creating a new template from scratch, going over basic conceptions, wireframes, coding and final implementation on a live website.

Keywords: WordPress, CMS, template, design, graphics, HTML5, CSS3, responsive design

Obsah

Seznam obrázků

1. Úvod

Dnes, v roce 2016, je Internet v podstatě neoddělitelnou součástí života většiny populace technologicky vyspělých zemí světa. Od svých počátků v padesátých letech minulého století se tato síť radikálně změnila a rozrostla do zcela jiné podoby, a stalo se z ní univerzální prostředí, ve kterém lze pořizovat či nabízet všemožné služby, předávat informace, komunikovat, nebo v podstatě jakékoliv spektrum lidské aktivity.

Důležitou částí tohoto vývoje jsou bezesporu webové aplikace, které uživateli poskytují přístup k informacím a prvkům internetu prostřednictvím webového prohlížeče, který tak nahrazuje klasického klienta nebo program, který by uživatel musel instalovat.

Jeden z hlavních prvků webových aplikací jsou takzvané redakční systémy, oficiálním názvem Systémy pro správu obsahu (anglicky Content Management System, zkráceně CMS, často lze narazit i na technicky řečeno chybné označení CMS System). CMS umožňují vytvářet či spravovat obsah webových stránek bez nějaké hlubší znalosti jazyka webu. Můžeme je tedy často vidět nasazené na místech, které poskytují nějakou formu obsahu, například zpravodajské servery, firemní weby, produktové stránky nebo například blogy.

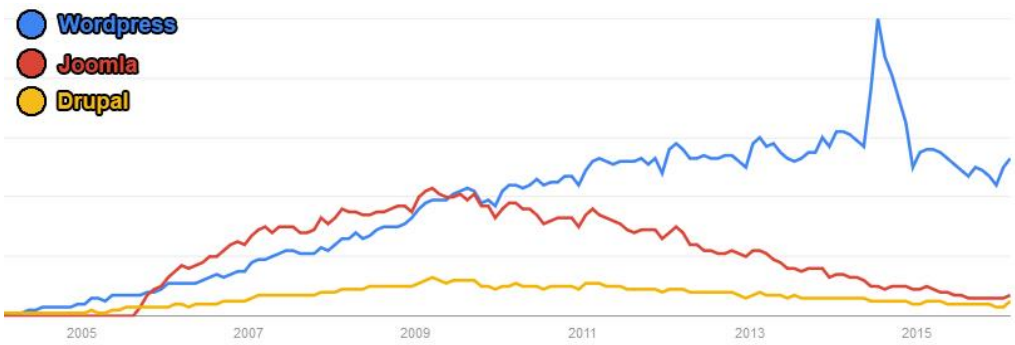
Redakční systémy rozdělují práci do několika vrstev podle takzvané vícevrstvé architektury. Tato architektura používá tři části, kdy je oddělena stránka, co se ukazuje uživateli (prezentační vrstva), logika a administrace (aplikační vrstva) a samotná data (datová vrstva) [Eckerson, 1995]. Data jsou v případě CMS většinou uložena v nějaké databázi, která obsahuje jak texty, tak i další věci, které jsou zapotřebí k funkčnosti celého systému. Díky tomuto rozdělení tak uživatele nemusí zajímat „jak to přesně funguje“, ale zároveň nepřichází o možnosti, které se od webové stránky dnes očekávají.

Content Management Systémů pochopitelně existuje celá řada, rozdílných ve funkcích a zaměřeních. Některé jsou vyvíjeny velkými společnostmi buď pro vlastní účely, nebo jako komerční produkty, jiné vyvíjí pár lidí pro svojí vlastní potřebu a tak dále. Co se rozšířenosti týká, nejpoužívanější jsou systémy na bázi open source, za kterými stojí velké komunity lidí, vytvářející systém a jeho rozšíření. Mezi tři nejpoužívanější systémy v této kategorii patří WordPress, Joomla a Drupal.

Další známé systémy z této kategorie jsou například Nucleus CMS, Radiant CMS, v České Republice a u jejích sousedů pak například phpRS či BLOG:CMS, oba z dílen českých tvůrců.

Ve své práci se budu zabývat systémem WordPress. S tímto CMS mám dlouholeté zkušenosti jak ze strany uživatele, tak ze strany vývojáře, tak i designéra. Jeho nasazení na různé typy webů se mi vždy velice osvědčilo a doporučuji jej každému, kdo má zájem spustit web. V současné době se WordPress profiluje jako stále větší a větší „král“ CMS, který, jak uvádí [WordPress.org Features], pohání přes 24% internetových stránek, což je neskutečně velké číslo. Aktuální verze 4.4 byla od svého vydání 8. prosince podle oficiálního počítadla stažena stránek více než 38 miliony uživateli [WordPress.org Download Counter], ovšem toto číslo je reálně mnohem větší díky systému automatické aktualizace a také proto, že existují jiné zdroje. Například to, že mnoho poskytovatelů webhostingu nabízí rovnou instalaci WordPressu ze svých serverů. Jedná se jak o české poskytovatele, tak i o zahraniční hostingové giganty jako GoDaddy.com nebo Hostmonster.com

Z předchozích vět je jasné, že objektivně kvantifikovat počet všech uživatelů systému WordPress je velice obtížné, nemluvě o jeho porovnání s jinými zmíněnými systémy. Za zřejmě nejlepší zdroj informací o zájmu o jednotlivé CMS lze považovat grafy z Google Trends, které ukazují zájem o jednotlivé systémy ve vyhledávání (viz Obrázek 1). Vysvětlením výkyvu zájmu o WordPress v průběhu léta 2014 lze vysvětlit vydáním beta verze dlouho očekávaného WordPress 4.0.



Obrázek 1 - Hledanost CMS systémů; [Google Trends]

Nasazením systému jako je WordPress se zdánlivě může zdát, že nastává konec společností, které dříve poskytovaly kompletní realizace webu. Na rozdíl od dřívějších dob,

kdy bylo běžné, že se objednal celý web, a každý zásah byl vždy úkolem pro programátora, což zákazníka stálo nemalé peníze, může díky těmto systémům jednoduše zprovoznit a udržovat web v podstatě zdarma i téměř naprostý laik. Ovšem toto je pouze zdáním, protože stále máme mnoho firem, věnujícím se vytváření webových prezentací. Tyto firmy pouze musely změnit své zaměření na nové role, protože stále platí, že pokud chce mít společnost kvalitní web, musí jej nechat udělat od někoho, kdo rozumí tomu, co dělá. Vznikly tak nové služby, jako je nasazování a provozování CMS, čehož se často ujali přímo poskytovatelé webhostingu, jak bylo zmíněno v předcházejících odstavcích, nabízení profesionálního vzhledu, nebo navrhování a realizování rozšíření pro nějakou specifickou a třeba ne příliš běžnou funkci.

Tento rozvoj se stal možným díky neustále pokračujícím vývojem technologií, ale hlavně i díky různým sociálním a ekonomickým faktorům, které umožnily přístup k této technologii téměř každému člověku na planetě. Velikost internetové populace[1] v České republice je podle posledních dat 7 178 736 osob [NetMonitor, 2016]. Když tuto hodnotu porovnáme s počtem obyvatel ve stejné skupině, který je podle posledních oficiálních údajů přibližně 9 milionů [Český Statistický Úřad, 2014] (číslo je uvedeno přibližně, vzhledem k rozdílnosti definice pojmu „internetová populace“ a věkového rozdělení statistik ČSÚ), lze odvodit, že Internet v republice využívá většina dospělé nebo dospívající populace. Podobné trendy jdou nalézt i v ostatních zemích světa, kde je srovnatelná technologická vyspělost.

S rozrůstající se populací internetu došlo před více jak deseti lety k zpopularizování takzvaného Web 2.0 (na O'Reilly Media Web 2.0 Conference 2004). Tento termín použila poprvé Darcy DiNucci ve svém článku Fragmented Future z roku 1999, ve kterém napsala, že web, jak byl znám tehdy, je pouze zárodkem něčeho většího, a že se odsune od klasických „stránek s textem a obrázky“ k nějakému většímu a všudypřítomnému stavu [DiNucci, 1999]. Dnes můžeme pouze konstatovat, že měla stoprocentní pravdu. Nejčastěji se Web 2.0 obecně popisuje jako prostředí pro sdílení a společnou tvorbu obsahu všemi, kteří mají zájem se na této činnosti podílet. Redakční systémy tak představují jednu z hlavních položek tohoto trendu.

[1] Pojem označuje skupinu lidí nad 10 let, kteří se ve sledovaném období připojí k internetu

2. Cíl práce a metodika

2.1. Cíl práce

Cílem této práce je popsat postupy při vytváření nové šablony pro CMS WordPress a demonstrovat postup této tvorby na praktickém příkladu.

Cílem této práce nemá být a ani není hloubková analýza systému WordPress nebo jeho funkcí. Jsem si vědom, že taková analýza by vydala na několik samostatných prací, vzhledem k rozsahu možností systému a různých přístupů uživatelů a tvůrců. Také nemá sloužit jako kompletní příručka toho, jak psát web pomocí HTML5 a CSS3.

2.2. Metodika

V první části této práce se budu věnovat teoretickému popisu CMS WordPress jako takového. Popíši jeho prostředí a možnosti rozšíření. Ve druhé části se budu věnovat popisu použitých technologií při tvorbě šablony, tedy především jazyků PHP, HTML5 CSS3 a JavaScript. V praktické části pak demonstruji proces tvorby jedné šablony.

Tato práce je určena lidem pracujícím s CMS WordPress, nebo lidem, kteří mají o tento systém zájem. Dalšími skupinami jsou tvůrci vzhledů, kteří se chtějí seznámit se šablonovacím systémem WordPressu, či vlastníci webů hledající vhodnou šablonu pro svůj web.

3. Přehled řešené problematiky

3.1. Základní charakteristika WordPressu

WordPress je open source systém pro správu obsahu, volně šiřitelný pod General Public License Version 2 (GPL v2). Je trvale vyvíjen od roku 2003 neustále se měnícím týmem okolo jednoho pevného „hlavního“ vývojářského týmu, který také tvoří část firmy Automattic, provozující službu a web WordPress.com. WordPress.com nabízí každému uživateli Internetu možnost si jednoduše zařídit web běžící na WordPressu s určitými specializovanými funkcemi, které zjednodušují práci nezkušeným uživatelům, na jejich serverech. Nabízena je jak vlastní placená doména, tak i doména zdarma ve tvaru nazev.wordpress.com. [WordPress.com]

Ve snaze, aby se WordPress zachoval jako open source, a nebyl příliš spojován s nějakou společností (konkrétně právě Automattic), byla veškerá práva k němu převedena na neziskovou nadaci [The WordPress Foundation]. Autorem tohoto nápadu byl jeden z klíčových vývojářů systému, Matt Mullenweg.

Samotný systém je postavený na jazyce PHP a používá databázový systém MySQL. Díky tomu je možné systém používat na většině dostupných serverů bez jakýchkoliv problémů, i když často některé funkce vyžadují určitá nastavení serveru jako například modul mod_rewrite. Některá rozšíření také mohou vyžadovat nějaká další specifická nastavení, jako například safe_mode. Ovšem ani jedno z těchto nastavení není problém upravit, pokud k tomu svolí vlastník serveru.

WordPress patří mezi celosvětově nejoblíbenější redakční systémy (viz Obrázek 1). Podle statistik z webu [w3techs.com] WordPress využívá 25,8% sledovaných webů, čili ještě vyšší číslo, než kterým se chlubí oficiální stránky. Tento podíl dává, podle stejných statistik, WordPressu téměř 60% podíl na „trhu“ Content Management Systémů. Systémy Joomla a Drupal přitom dohromady dosahují pouze 5% všech sledovaných webů s podílem 11,2%.

Tato popularita WordPressu pramení primárně z jeho jednoduché instalace a provozu, ale také z toho, že ač je v základu jednoduchý, je otevřený rozšiřování o v podstatě libovolné funkce, což umožňuje pomocí WordPressu postavit i mnohem více specializované služby, než pouze novinkový portál nebo osobní blog.

3.1.1. Klíčové vlastnosti

WordPress nabízí v základu mnoho zajímavých a užitečných funkcí, z nichž některé jsem již zmínil v předcházejících odstavcích. Oficiální stránky [WordPress.org Features] uvádějí 23 vlastností, které WordPress nejlépe vystihují. Osobně považuji za nejdůležitější tyto vlastnosti:

- Jednoduchost zprovoznění
- Jednoduchost vytváření obsahu pomocí integrovaného editoru TinyMCE
- Možnost komplexně organizovat obsah pomocí kategorií a štítků
- Flexibilita, umožňující postavit jakýkoliv web na jednotném základu
- Rozšiřitelnost pomocí stále se rozrůstajícího množství pluginů s nejrůznějšími funkcemi
- V základu integrované komentáře, vyhledávání, automatické vytvoření RSS kanálu

3.1.2. Historie

Jak bylo zmíněno výše, WordPress je konstantně vyvíjen od roku 2003. První oficiální verzí byla verze 0.70, vydaná 27. května 2003 [WordPress.org Roadmap]. Verze 1.0 vyšla o půl roku později, 3. ledna 2004. Od této verze také každá hlavní verze nese jméno nějakého známého jazzového hudebníka. První verze tak nesla jméno „Miles Davis“.

Verze z počátku vycházely nepravidelně, ale od verze 2.1 „Ella Fitzgerald“ byl přijat systém, že každé 3-4 měsíce bude jedna hlavní verze, která přinese novinky na základě podnětů značné komunity uživatelů. Mezi těmito klíčovými pochopitelně nepravidelně vycházejí nepojmenované menší verze (číslované třemi čísly, například 4.4.2), obsahující záplaty bezpečnostních chyb a jiných kritických problémů.

Krátce po uvedení WordPressu byl také zaveden systém [WordPress Trac], umožňující jednoduchou komunikaci mezi vývojáři a sledování nebo nahlašování chyb.

V době psaní této práce je poslední verzí 4.4, nesoucí jméno „Clifford Brown“, která přinesla především automatické přidání responzivní třídy všem obrázkům, které tak bez dalšího zásahu autora textu budou vypadat dobře na všech zařízeních, jejichž velikost je v dnešní době stále rozmanitější, a možnost jednoduše vkládat úryvky z textů na jiných WordPressem poháněných webech. „Clifford“ dále přinesl řadu změn „pod kapotou“ systému, například integrování REST API, zlepšení dotazování na komentáře

a zlepšení systémů štítků a kategorií. Pro tento rok jsou v naplánované tři verze. Nejbližší nová verze, 4.5 (zatím nepojmenovaná), je naplánována zatím nespecifikované období tohoto roku, stejně jako verze 4.6.

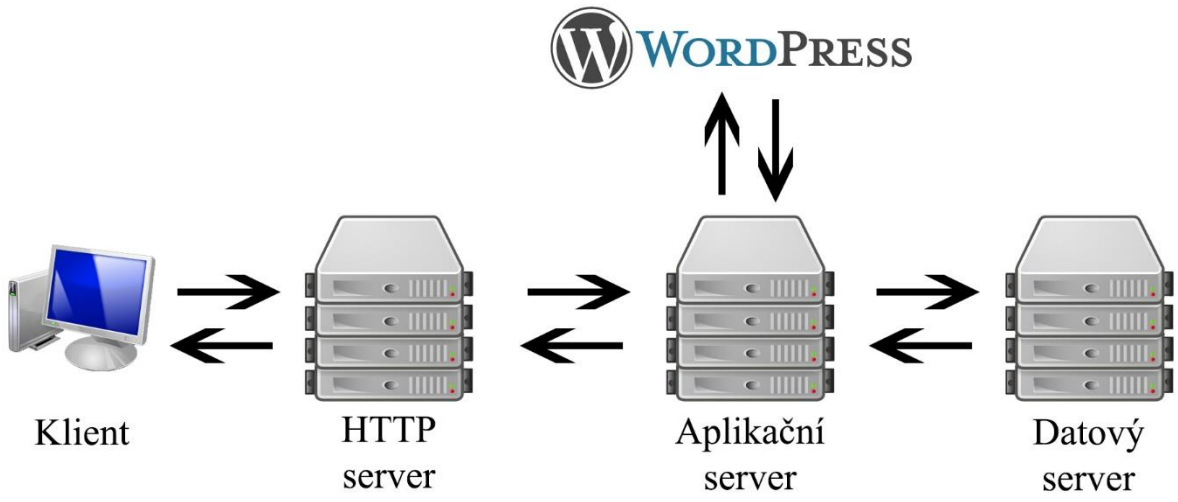
3.2. Architektura WordPressu

Jak bylo zmíněno v úvodu, redakční systémy jsou webové aplikace, a WordPress není výjimkou. To znamená, že je uživateli zobrazován skrze libovolný webový prohlížeč přes Internet. Zde je využívána klasická architektura „klient-server“, složená z klienta, serveru a propojení mezi nimi (viz Obrázek 2).

Klienta v případě WordPressu, stejně jako u dalších webových aplikací, zastupuje zmíněný webový prohlížeč, který je dnes součástí víceméně každé instalace všech majoritních operačních systémů a uživatelům je nabízeno značné množství volně stažitelných alternativ.

Server je zde zastoupen několika různými částmi, HTTP serverem, Aplikačním serverem a Datovým serverem. HTTP server přijímá požadavky od klienta a vrací mu žádaný výsledek. Tento výsledek je založen na logice provedené Aplikačním serverem pomocí údajů poskytnutých Datovým serverem.

Poslední zmíněná část, propojení, je představována síťovou komunikací mezi klientem a serverem prostřednictvím Internetu.



Obrázek 2 - Schéma komunikace WordPressu podle modelu klient-server

3.2.1. Technologie

WordPress je napsán v jazyce PHP, a tak ke své funkci vyžaduje server, který tento jazyk podporuje, a MySQL databázi pro ukládání dat. Aktuální verze WordPressu, 4.4, doporučuje verzi PHP 5.6 nebo vyšší a MySQL verze 5.6 nebo vyšší [WordPress.org

Requirements], čili požadavky, které jsou dnes běžné u většiny poskytovatelů webových hostingů. Vývojáři pochopitelně doporučují poslední verze obou systémů pro zajištění maximální bezpečnosti. Některé funkce systému, především „uživatelský přívětivé URL“, v oficiální dokumentaci nazývané „Pretty Permalinks“ [WordPress.org Introduction to Blogging], vyžadují, aby byl na webovém serveru povolený modul mod_rewrite, který běžně v základu povolen není, a je tak třeba jeho povolení vykomunikovat s poskytovatelem. Některá rozšíření (pluginy) dále vyžadují vypnutou direktivu safe_mode.

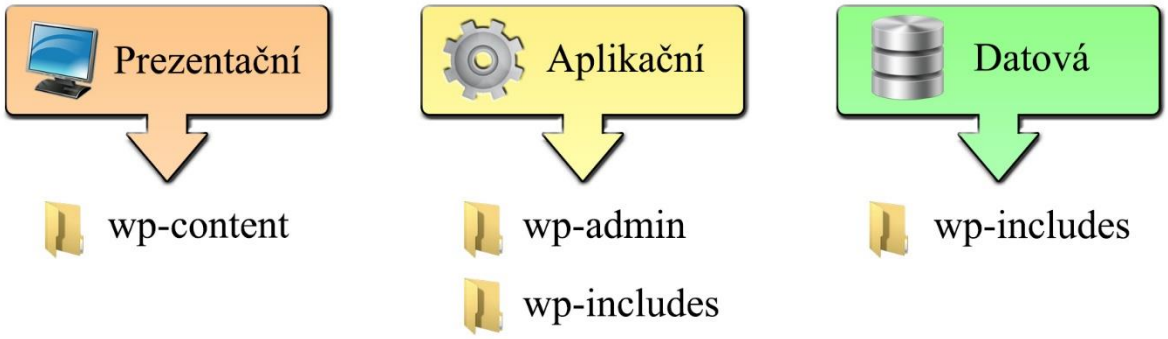
Pokud webový server splňuje všechny výše zmíněné podmínky, může na něm WordPress běžet. Mezi oficiálně doporučované však patří Apache nebo Ngnix, v současnosti asi nejpoužívanější řešení pro webový server. Dále také existují způsoby, jak WordPress spustit na jiném databázovém systému, než je MySQL, jako například SQLite nebo Microsoft SQL Server. Ovšem tyto metody vyžadují další nastavení, které není pro běžného uživatele jednoduché a tato práce se instalací WordPressu nezabývá.

3.2.2. Vrstvy systému

Jak bylo zmíněno, WordPress využívá vícevrstvou architekturu o třech vrstvách – prezentační, aplikační a datovou. Toto rozdělení znamená, že aplikace je rozdělena do několika částí, z nichž každá běží samostatně a často na jiném hardwaru [Eckerson, 1995], a společně tvoří jeden ucelený výsledek, který je následně předáván uživateli.

Při obecném pohledu na toto rozdělní lze říct, že prezentační vrstvu představuje uživatelův internetový prohlížeč, aplikační vrstvu pak server s podporou PHP a datovou vrstvu MySQL databáze na datovém serveru.

Čistá instalace WordPressu je sice rozdělena od tří složek, ale nelze tvrdit, že každá složka je odpovědná za jednu vrstvu systému. Spíše se jedná o rozdělení podle toho, za jakou část zpracování obsahu je která složka a její skripty zodpovědná, a všechny tak v podstatě tvoří aplikační vrstvu. Ovšem pro zjednodušení si lze představit jednoduché rozdělení podle tohoto modelu (viz. Obrázek 3).



Obrázek 3 - Rozdělení složek do třívrstvého modelu

3.2.2.1. Prezentační vrstva

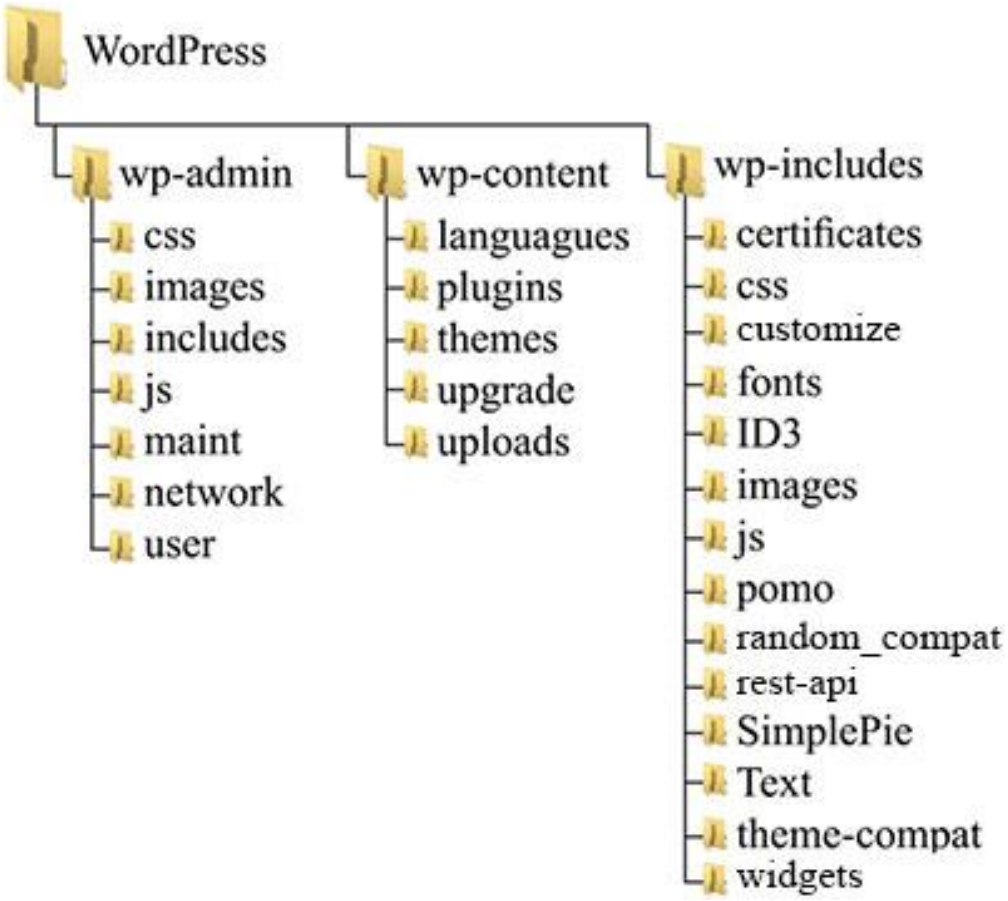
Vrstva prezentační odpovídá, jak bylo zmíněno, za to, co se uživateli zobrazuje. V rámci WordPressu tuto funkci řeší především takzvané šablony (anglicky Themes). Šablony jsou v podstatě kombinací několika PHP souborů a CSS stylu, případně i nějakých jiných skriptů, například velice často používané javascriptové knihovny jQuery.

Samotný obsah webu je následně získáván prostřednictvím databáze prostřednictvím aplikační a datové vrstvy. Samotná databáze pochopitelně ukládá pouze textové řetězce, ale díky tomu, že může obsahovat v textové podobě i odkazy na jiné soubory, není pro prezentační vrstvu problém zobrazovat obrázky a jiné části obsahu.

3.2.2.2. Aplikační vrstva

Aplikační vrstva je místem, kde se odehrává hlavní část práce každé webové aplikace. Ve WordPressu tuto práci zastává několik stovek PHP skriptů, rozdělených do tří hlavních složek (viz. Obrázek 4) a patnácti v kořenovém adresáři, které se starají o správné přiřazení a spuštění ostatních skriptů.

Složka wp-admin se stará především o správné zobrazení a funkčnost administračního prostředí. Druhá složka, wp-content, obsahuje obsah vytvořený nebo nahraný uživatelem, který s ním může relativně bezpečně manipulovat. Jedná se tedy především o šablony vzhledu, rozšíření (pluginy) a nahrané mediální soubory. Poslední složka, wp-includes, obsahuje zbylé skripty logiky systému a komunikace s databází.



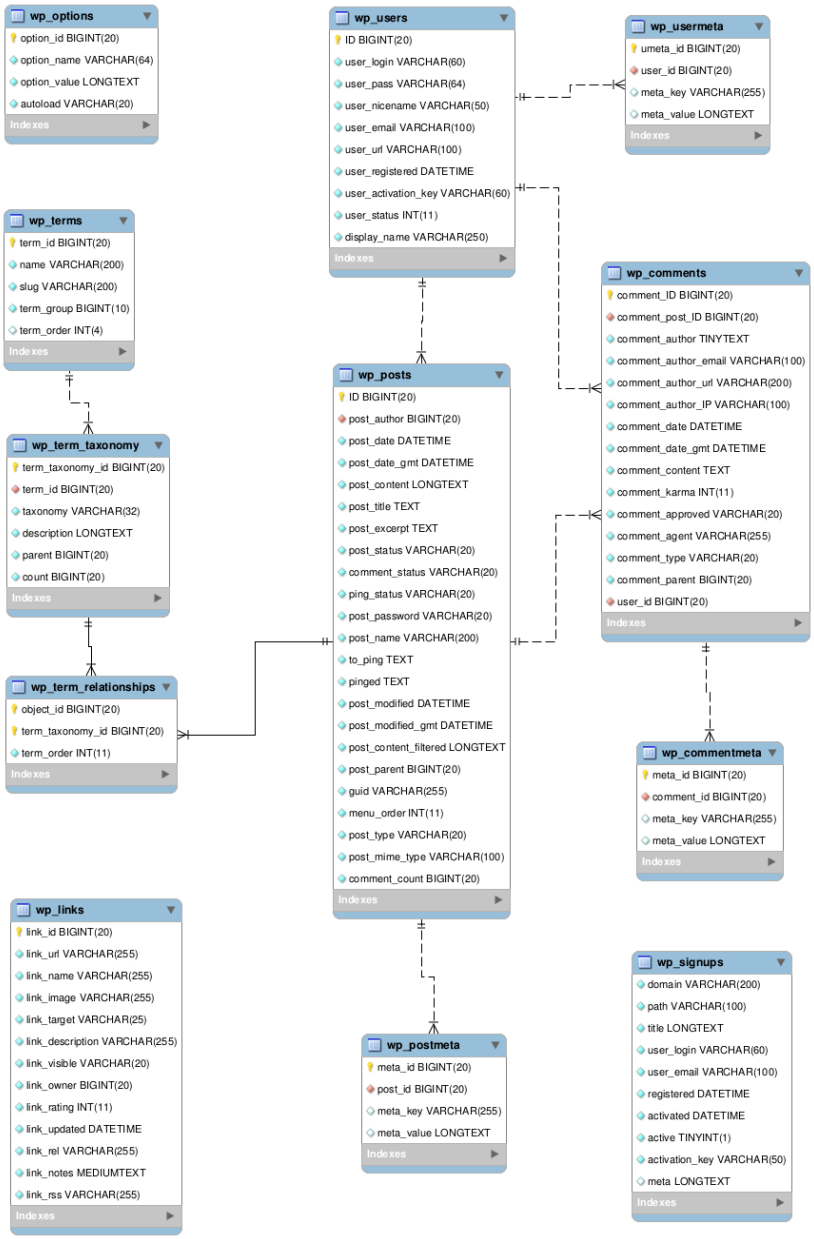
Obrázek 4 – Základní adresářová struktura WordPress 4.4

3.2.2.3. Datová vrstva

Jak jsem zmínil, WordPress využívá databázový systém MySQL, vyvinutý švédskou firmou MySQL AB, později koupenou společností Sun Microsystems, kterou později koupila, a do dnešní doby zůstává vlastníkem, Oracle Corporation. Samotný systém ale přes všechny změny vlastníků zůstal volně dostupný pod licenci GPL, což je důvodem jeho značné obliby.

Databáze WordPressu se v základu skládá z 12 tabulek (celé schéma viz Obrázek 5). Hlavní tabulkou je wp_posts, obsahující údaje a obsah jednotlivých příspěvků a článků. Na tuto tabulku jsou pak napojeny další, jako wp_comments, starající se o komentáře, nebo wp_users, ukládající informace o registrovaných uživatelích. Všechny tyto tři tabulky pak k sobě mají ještě takzvané tabulky meta (wp_postmeta, wp_commentmeta, wp_usermeta), ukládající metadata, většinou generována z individuálních uživatelských polí. Tabulky wp_terms, wp_term_taxonomy a wp_term_relationships se pak starají o organizaci obsahu na webu prostřednictvím kategorií a štítků. Tabulka wp_links ukládá odkazy spravované

přes dnes již rozšiřující plugin Link Manager (vydávaný vývojáři). Tato funkce byla původně součástí systému, ale od verze 3.5 byla pro nižší využívanost odebrána ze základní instalace. Poslední tabulkou ve standardní instalaci je wp_options, která je jako jediná nepropojená, a která obsahuje základní informace a nastavení blogu, jako je jméno, domovská adresa, kontakt na administrátora, povolení emotikonů a atd. [WordPress.org Database Description]. Instalace WordPressu podporující provozování více webů z jedné společné administrace pak přidává několik dalších tabulek, například na Obrázku 5 viditelnou wp_signups.



Obrázek 5 - Databázová struktura WordPressu;[WordPress.org Database Description]

3.3. Přizpůsobení WordPressu

WordPress nabízí značný stupeň přizpůsobitelnosti tomu, co od něj vlastník webu očekává, ať už se jedná o vzhled nebo o funkce. WordPress tak může sloužit nejen jako blogovací systém, jak byl původně navržen, ale jako téměř univerzální systém, pomocí kterého lze provozovat například i elektronický obchod či interní sociální síť. Vždy záleží pouze na tom, jak si uživatel rozhodne systém přizpůsobit pomocí rozšíření a vzhledů, které si do systému nahraje, nebo někým nechá nahrát.

3.3.1. Rozšíření ve WordPressu

Domnívám se, že jedním z hlavních důvodů, proč je WordPress tak populární platformou, je jeho neuvěřitelná rozšiřitelnost o téměř libovolné funkce (jak bylo zmíněno, například elektronický obchod). Ta je prováděna přes takzvané pluginy (v české verzi je někdy používán výraz rozšíření), kterých je k dispozici celá řada s různými funkcemi a v různé kvalitě.

Jak jsem se zmínil v přechozích kapitolách o požadavcích na technologii, jsou to právě pluginy, které pro svojí plnou funkčnost často potřebují nějaké speciální nastavení. Většinou se jedná o vypnutí direktivy safe_mode na webovém serveru, nebo o nastavení určitých přístupových práv k nějaké složce na serveru.

Hlavním zdrojem pro stahování pluginů je oficiální repositář, který v době psaní této práce obsahuje přes 43 tisíc pluginů, které byly staženy více než 1,2 miliardkrát [WordPress.org Plugins]. Uživatel může plugin stáhnout buďto ručně z repositáře a nahrát jej na svůj server přes FTP, nebo jej může stáhnout a nainstalovat přímo z administrace WordPressu. Plugin pak v obou případech bude čekat v seznamu, až jej uživatel aktivuje.

3.3.1.1. Struktura rozšíření

Každý plugin, ať už nahraný ručně, nebo stažený z administrace, je uložený jako PHP soubor ve složce plugins ve wp-content (viz Obrázek 4). Pokud je jedná o nějaké komplexnější rozšíření, kterých je většina, bývá uložen ve vlastní složce v tomto adresáři, aby se usnadnila orientace a předešlo konfliktům v pojmenování.

Samotný plugin se obvykle skládá z několika samostatně nefungujících skriptů, stejně jako celý CMS systém napsaných v jazyce PHP a případně s nějakou funkcí

doplněnou přes JavaScript. Po integraci do systému nějakým způsobem rozšiřují jeho funkce. Pokud má rozšíření nějaké uživatelské rozhraní nebo zobrazuje uživatelům nějaký výstup, využívá kaskádové styly CSS.

Při vytváření rozšíření musí brát autor v potaz několik základních věcí nutných ke správnému fungování. První věcí je, aby soubory pluginu obsahovaly jasné identifikující hlavičky, druhou věcí je využívání [WordPress Plugin API], které se stará o takzvané hooky. Hook propojuje samotnou funkci pluginu a systému, aby při načítání stránky bylo načteno i dané rozšíření, a aby při případném odstranění pluginu systém fungoval bez problémů.

Hooky se dělí do dvou skupin, Actions a Filters. Actions se spouštějí, pokud nastane nějaká určitá akce a přidávají nějaký kód. Příkladem je třeba odeslání emailu na určité adresy pokaždé, když je na webu přidán nový příspěvek. Filters se spouští, pokud dochází k nějaké komunikaci mezi prohlížečem a databází (čili například vždy, když se vytváří nebo zobrazuje nový příspěvek na stránce), a mohou upravovat procházející data. Příkladem může být automatická cenzura slov podle nějakého nastaveného seznamu.

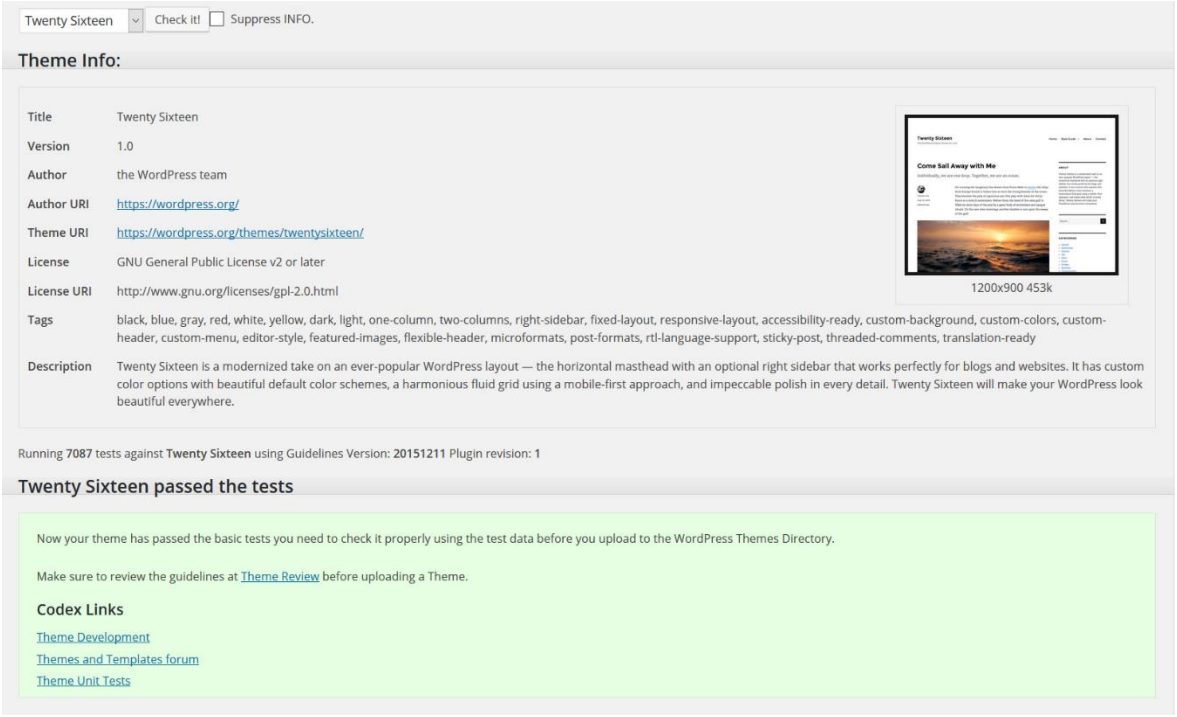
Pochopitelně existují i další pravidla pro vytváření pluginů. Například pokud autor chce svůj plugin zveřejnit ve zmíněném oficiálním repositáři, musí k němu přidat i textový soubor readme s informacemi o funkčnosti a licenci.

3.3.2. Šablony ve WordPressu

V kapitole o prezentační vrstvě jsem ve stručnosti popsal, že šablony slouží k převedení kódu poskytovaného z databáze do podoby, která se zobrazuje uživateli, čili že odpovídají za to, jak daná stránka vypadá.

Šablony ve WordPressu jsou asi nejblíže tomu, co si u tohoto CMS lze spojit s klasickým webdesignem, a mnoho autorů a i celých společností se jejich tvorbou zabývá na profesionální úrovni. Ovšem díky tomu, že šablony může vytvářet každý, kdo má znalosti psaní stránek v PHP a HTML, jich existuje celá paleta kompletně zdarma. Pokud šablona splňuje určité požadavky, nic nebrání autorovi jí nahrát do oficiálního repositáře, odkud si ji mohou ostatní zdarma stáhnout. Tento repositář nabízí široký výběr z více než 2000 různých vzhledů [WordPress.org Themes]. Mezi požadavky pro to, aby byl nahraný vzhled uznaný, patří kromě očekávaných věcí, jako validita kódu a absence chyb,

například obrázek sloužící jako náhled, soubor readme a informace o licenci. Pro kontrolu všech těchto požadavků je autory WordPressu udržován plugin ThemeCheck, který projde soubory zvoleného vzhledu, a vypíše případné chyby (viz obrázek 6, ukazující aplikaci pluginu na výchozí styl WordPress 4.4).

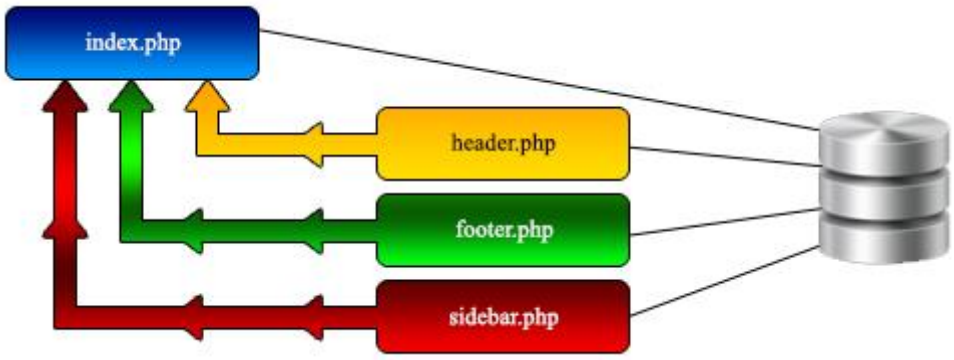


Obrázek 6 - Výstup ThemeCheck pluginu

3.3.2.1. Struktura šablon

Šablony ve WordPressu mají několik částí, které jsou nezbytně nutné, a bez kterých šablona nemůže fungovat, a takových, které by sice nutné být měly, ale šablona bez nich fungovat bude, i když značně omezeně. Poslední skupinou jsou zcela nepovinné rozšiřující části, kterých je asi nejvíce.

Protože je WordPress napsaný v PHP, využívá variace na funkci include(), která umožňuje vložit do stránky kus jiného kódu a následně jej provést. To umožňuje změnit část stránky (například hlavičku) v jednom souboru, a tato změna se projeví všude, bez nutnosti upravovat každý soubor individuálně. Například index.php si ve většině šablon bere obsah ze souborů pro hlavičku (header.php), patičku (footer.php) a postranní panel (sidebar.php) (viz obrázek 7).



Obrázek 7 – Příklad složení index.php

Mezi ty absolutně povinné patří pouze dva soubory, index.php a style.css. První z nich, index.php se stará o vypsání textu získaného z databáze. Ve většině stylů se využívá jako šablona pro zobrazení hlavní stránky, ovšem pokud neexistují šablony pro individuální typy obsahu, využívá se k jejich zobrazení právě index.php. Druhý soubor, style.css, se jednak využívá pro nastavení kaskádových stylů CSS, ale v rámci WordPressu také pro načtení základních informací o daném vzhledu, jako je název a autor, zapsaných jako komentář na prvních řádcích souboru.

Druhou skupinu tvoří zmíněné soubory, které nejsou zcela nutné, ale pro plnou funkčnost systému a webu je ideální, aby se v šabloně nacházely. Jedná se především o šablony pro základní kusy obsahu, jako jednotlivé novinky, stránky, komentáře a tak dále. Dále se také jedná o součásti stránek, jejichž část lze vidět na Obrázku 7.

Poslední kategorii tvoří především šablony pro speciální typy obsahu, nebo pro variace regulérního obsahu, které je zapotřebí zobrazit jiným způsobem. Například se může jednat o obsah spojený s nějakou speciální akcí s vlastní barevnou paletou, přehrávač videí, a podobně.

3.3.2.2. Child šablony

Pojmem „child theme“ se ve WordPressu označuje takový typ šablon, které využívají dědění z nějaké nadřazené šablony (oficiálně pojmenované „parent theme“). Z této šablony pak přebírají její soubory a styly. Jedná se o doporučovanou metodu pro upravování existujících šablon. Oproti zásahu přímo do existující šablony se jejich použitím lze vyhnout ztrátě uprav v případě, že by šablona byla aktualizována. [Wordpress.org Child Themes]

WordPress organizuje soubory následné šablony do tří kategorií:

- Výpis – soubory šablony, zodpovědné za výpis kódu
- Styl – soubory šablony, zodpovědné za kaskádové styly
- Funkce – soubor šablony, aktivující funkce WordPressu

Soubory z první kategorie jsou kompletně přepsány, pokud následná šablona obsahuje stejně pojmenovaný soubor. Styly jsou rozšiřovány a přepisovány podle pravidel kaskádových stylů. Soubor s funkcemi je načten před stejnojmenným souborem nadřazené šablony. Z toho důvodu by měly být funkce deklarovány v podmínkách, aby je bylo možné snadno přepsat předcházející funkcí stejného jména. Ukázku takovéto podmínky lze vidět na následujícím obrázku.

```php
<?php
if (!function_exists('funkce_sablony')) {
    function funkce_sablony() {
        //  Kod funkce
    }
}
?>
```

Obrázek 8 - Podmíněná funkce

3.3.3. Frameworky

Jak jsem zmínil v úvodní kapitole, s příchodem Web 2.0 a rozšířením webových aplikacích jako takových muselo dojít k přeorientování firem zabývajících se vývojem webů, aby mohly přežít měnící se prostředí a požadavky zákazníků.

V případě WordPressu mnohé komerční subjekty za pomoci extenzivního využití systémů šablon a rozšíření vytvořily celé frameworky, které umožňují běžnému uživateli jednoduché nasazení na jakékoliv stránce a případně i snadné vybudování vlastního vzhledu bez jakýchkoliv znalostí kódování, vše z prostředí administrace webu a prostřednictvím drag-and-drop. Příkladem takovýchto frameworků je například Genesis od StudioPress, Headway od HeadwayThemes, nebo Gantry Framework. Uživatel jednoduše stáhne (většinou po zakoupení) jeden velký balík s celým frameworkem, a ten pak v prostředí WordPressu jednoduše aktivuje stejně jako běžná rozšíření či vzhledy.

3.4. Lokalizace WordPressu

Vzhledem k tomu, že výsledná šablona bude podporovat překlady do jiných jazyků, což je při rozšířenosti WordPressu velice vyhledávaná vlastnost, popíši v této kapitole základní funkce pro překládání, které WordPress využívá.

Celý redakční systém byl od začátků navržen tak, aby umožňoval snadné překládání do různých jazyků, důkazem čehož je 160 různých jazykových mutací, ovšem některé z nich jsou v podstatě duplikáty, rozlišené pouze v určitých specifických jazyka v dané oblasti (například kanadská a australská angličtina) [Translation Teams].

Pro překládání WordPressu se používají dva anglické termíny, localization a internationalization, často zkracované jako l10n a i18n (podle počátečního a posledního písmene a podle počtu písmen ve slově [Glossary of W3C Jargon]).

3.4.1. Technologie lokalizace

Autoři WordPressu se při vývoji infrastruktury pro překlad jejich díla rozhodli pro využití prověřeného frameworku GNU gettext, který je v podstatě standardem pro lokalizace veškerého open source softwaru.

Framework gettext pracuje na principu překládání individuálních textových řetězců, ať už je řetězcem jedno slovo, nebo celý odstavec. Uvnitř WordPressu jsou tyto řetězce vytvářeny a zpracovávány pomocí několika lokalizačních PHP funkcí, přičemž se nejčastěji používají tři z nich: __(), _e() a _n().

3.4.1.1. Funkce __()

Tato funkce využívá formátu __($text), kdy prohledá celý lokalizační soubor pro překlad řetězce z proměnné $text a předá jej PHP příkazu return. Pokud tento překlad nenajde, předá se původní hodnota proměnné $text.

Tato funkce tak má využití, pokud potřebujeme, aby se text ukládal do proměnné, například pokud jí potřebujeme používat jako parametr, například pro funkci sprintf, nebo pokud jí vypisujeme až později.

3.4.1.2. Funkce _e()

Tato funkce využívá formátu _e($text), kdy prohledá celý lokalizační soubor pro překlad řetězce obsaženého v proměnné $text a předá jej PHP příkazu echo. Pokud tento překlad nenajde, předá se pouze původní hodnota proměnné $text.

Využití najde tato funkce především, pokud se jedná o text, který je součástí uživatelského rozhraní, například součást nadpisu nebo popisku, a není zapotřebí jej využívat později či jako parametr funkce.

3.4.1.3. Funkce _n()

Tato funkce je variantou funkce __(), čili se její výsledek vrací jako proměnná. Hlavní rozdíl je v tom, že se používá pro označení textu, kde jsou množná čísla. Využívá se zde formát _n($single, $plural, $number), čili máme zadané jednotné číslo, množné číslo a počet. Na základě počtu je rozhodnuto, jestli funkce předá hodnotu jednotného ($single) nebo množného ($plural) čísla.

3.4.1.4. Funkce _x(), _ex() a _nx()

Tyto tři funkce jsou rozšířením funkcí předcházející o takzvaný „kontext“, čili že přidávají význam. Ten je důležitý především v případech, že není jasné, co dané slovo má znamenat, a v případech, že se například sloveso a podstatné jméno píší shodně. Překladatelé tak mohou poznat, jak mají slovo přeložit, aby dávalo ve výsledku smysl.

3.5. Použité technologie

Jak bylo práci již zmíněno, WordPress je naprogramovaný v jazyce PHP, zatímco samotné šablony využívají kombinaci známou z běžného vývoje webových stránek, čili jazyky HTML a CSS, zodpovědné za vzhled stránky, a jazyk JavaScript, zodpovědný za interaktivnost pro koncového uživatele, pochopitelně v kombinaci s PHP pro generování obsahu.

V této kapitole se zaměřím na popis této čtveřice jazyků, kterou budu následně využívat v rámci praktické části své práce pro realizaci tvorby šablony.

3.5.1. HTML

HyperText Markup Language, mnohem častěji známý pouze pod svojí zkratkou HTML, je základním stavebním kamenem webu. Jedná se o takzvaně značkovací jazyk, čili jazyk, který přidává k textu nějaké dodatečné informace, nejčastěji sloužící k rozložení dokumentu.

I v dnešní době lze narazit na jednotlivce, kteří HTML považují za programovací jazyk, což je ale zcela špatná definice. Nejjednodušší definicí pro poznání rozdílu je, že programovací jazyk obsahuje nějaký algoritmus, čili instrukce, jak něco udělat, zatímco značkovací jazyk pouze říká, jak se má nějaký výsledný text vykreslit.

3.5.1.1. Struktura HTML

HTML historicky vzešel z SGML, *Standard Generalized Markup Language*, univerzálního značkovacího metajazyka, a až do současné verze, HTML5, jej šlo brát jako podmnožinu právě tohoto jazyka.

To znamená, že celý jazyk je postavený na sérii takzvaných "**tagů**" (lze také narazit na výraz "element", nebo v češtině také někdy "prvek" či "značka" se stejným významem), jejich atributů (vlastnostech) a speciálních entitách. První veřejně dostupná zmínka o značkách byla v dokumentu od sira Tima Bernse-Leeho, který obsahoval 18 tagů využívaných v jeho původním návrhu systému pro CERN. Jedenáct z oněch 18 značek je využíváno do dnešní doby. Tagy se v základě dělí na dvě skupiny:

- **Párové**
    - Mají začátek a konec, většinou obalují nějaký obsah

- Příklad - *<h1>Nadpis</h1>* - nadpis první úrovně

- **Nepárové**
    - Pouze jeden tag, jejich obsah je určený pomocí atributů
    - Příklad - *<img src=”zdroj.jpg”>* - vložení obrázku zdroj.jpg

Atributy nějakým způsobem rozšiřují tagy o určité vlastnosti. Jsou důležitou součástí především u nepárových značek, kterým přidávají nějaký význam. V dnešní době je však lze nalézt téměř u každého tagu. Většina atributů existuje jako párová kombinace “název=hodnota”, lze však narazit také pouze na jednoslovné atributy. Hodnota atributu by vždy měla být zapsána v uvozovkách, ale v případě některých datových typů, jako jsou například čísla, je HTML správně přeloží i bez uvozovek. Mezi nejběžnější typy atributů, které, jak bylo zmíněno, lze nalézt u většiny elementů v dokumentu, patří následující:

- **id** - přidává elementu unikátní identifikátor na dané stránce, čili takový, který by se neměl opakovat. Většina prohlížečů stránku vykreslí i s více shodnými identifikátory, ovšem z hlediska kódování je to považováno za hrubou chybu. Jejich hlavní využití je v odkazování na jednotlivé částí stránky (například kapitoly)

- **class** - v češtině často používán překlad „třída“. Stejně jako id přidává prvku identifikátor, ale na rozdíl od něj nemusí být unikátní. Z tohoto důvodu nelze třídu využít pro odkazování v obsahu stránky, a *class* tak hlavní využití najde při nastavování vzhledu prvku pomocí CSS.

- **title** - přidává prvku popisek, který se objeví při najetí myší. Nejčastěji lze vidět u odkazů a obrázků, ovšem nic autorovi dokumentu nebrání v jeho přidáni i například na odstavec, funkce zůstane identická.

Speciální entity jsou kombinace znaků, které umožní prohlížečům vypsat znaky, které by byly jinak považovány za část tagu nebo nějaký jiný “netisknutelný” znak. Příkladem takového znaku jsou špičaté závorky („*<*“ a „*>*“), nebo ampersand („*&*“), které znamenají začátek a konec tagu, respektive začátek zápisu speciální entity. Vzhledem k tomu, že tyto znaky je často zapotřebí obsáhnout přímo v samotném textu, existuje v HTML způsob, jak prohlížeči oznámit, že má znak zobrazit, nikoliv přeložit. Tento zápis

je proveden zapsáním znaku &, alfanumerického kódu, který je danému znaku přiřazen, a ukončen středníkem (například vypsání právě ampersandu je *"&"*). Ukázku kódu v jazyce HTML lze vidět na obrázku níže.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Untitled Document</title>
</head>

<body>
<div id="obsah">
    <h1 class="nadpis">Hello world</h1>
    <p>Ukázka kódu v <abbr title="HyperText Markup Language">HTML</abbr></p>
</div>
</body>
</html>
```

Obrázek 9 - Ukázka HTML kódu

Na obrázku lze vidět definici typu dokumentu (DTD) pomocí tagu <!doctype>, který dokument deklaruje jako HTML 4.01 Transitional. Následuje definice hlavičky v párovém elementu <head>, kde je obsažena použitá znaková sada a titulek stránky, čili to, co uživatelé vidí v listě prohlížeče a ve výsledcích vyhledávání.

Další částí dokumentu je párový prvek <body>, který označuje onu hlavní část stránky, která je uživatelům vykreslována. Jak lze vidět, obsahuje obalovací tag , který má využití pro rozložení dokumentu do bloků, a především pro následné stylování pomocí CSS, a který má přiřazený identifikátor „obsah".

Uvnitř něj je zapsaný nadpis úrovně 1 (párový element <h1>), který by měl označovat nadpis celého dokumentu. Dnes, v době dynamických stránek, se velice často alespoň částečně shoduje s obsahem elementu <title>. Po nadpisu následuje odstavec (párový element <p>) s krátkým textem a dalším elementem, <abbr>, neboli zkratkou, jejíž význam je určený atributem title. Pod odstavcem lze vidět zapsaný komentář, sloužící k případnému popsání kusu kódu. V prohlížečích se pochopitelně tento text nevypisuje.

Výsledek tohoto kódu zobrazeného v prohlížeči za pomocí výchozích stylů z definice dokumentu, lze vidět na následujícím obrázku, kde je při najetí kurzorem na zkratku zobrazen její atribut title.

Obrázek 10 - Výsledek HTML kódu z předchozího obrázku

3.5.1.2. Historie HTML

Počátky HTML se udávají od roku 1989, kdy s první verzí jazyka a prvním webovým prohlížečem (pojmenovaným WorldWideWeb) přišel Timmothy „TimBL“ Bernes-Lee, dnes již rytíř britské koruny, stav, do kterého jej právě za přínos informatice povýšila královna Alžběta II. V roce 1989 Bernes-Lee již několik let pracoval jako nezávislý spolupracovník ve švýcarském centru pro experimentální fyziku, známém mezi širokou veřejností jako CERN.

S návrhem HTML přišel s cílem zjednodušit komunikaci mezi jednotlivými skupinami pracující na velmi rozsáhlých projektech, které v CERNu probíhaly. Ta probíhala bez jakékoliv standardizace použitého značkovacího jazyka nebo způsobu předávání a následkem toho byla tak značně nepřehledná. Tim Bernes-Lee tak přišel, jak sám tvrdí, především ze zoufalosti nad existující situací, s návrhem standardizovat komunikaci prostřednictvím jednoho jazyka, založeného na SGML, a hypertextu, tedy možnosti propojit více dokumentů mezi sebou. Tento návrh byl jeho nadřízenými přijat a v následujícím roce implementován.

V roce 1991 Tim Bernes-Lee veřejně vydal v předchozí částí zmíněný dokument HTML Tags. V polovině roku 1993 bylo HTML oficiálně schváleno jako aplikace SGML organizací Internet Engineering Task Force. Tento návrh vypršel počátkem následujícího roku, a IETF ustanovilo HTML Working Group, která se měla zabývat dalším vývojem. Tim Bernes-Lee ve stejném roce, po odchodu z CERNu na americký MIT, založil World Wide Web Consortium, častěji známé pod zkratkou W3C s cílem se snažit udržovat kompatibilitu napříč vývojáři prohlížečů.

Historie HTML obsahuje řadu dalších důležitých historických milníků, které jsou shrnuty v následujících bodech:

24. listopadu 1995

- IETF zveřejňuje HTML 2.0 jako RFC 1866. Další RFC kodifikují rozšiřující funkce:
    - **25. listopadu 1995** - norma RFC 1867 - nahrávání souborů pomocí formulářů
    - **Květem 1996** - norma RFC 1942 - tabulky
    - **Srpen 1996 -** norma RFC 1980 – obrazové mapy na straně klienta
    - **Leden 1997**: norma RFC 2070 - internacionalizace

14. leden 1997

- HTML 3.2 je publikováno jako W3C Recommendation. Jedná se o první verzi vyvinutou a standardizovanou výhradně touto organizací, protože HTML Working Group v rámci IETF ukončila svojí činnost 12. září 1996
- Původně označováno kódovým jménem „Wilbur“, HTML 3.2 odstranilo podporu pro matematické formulace, sjednotilo překrytí mezi různými proprietárními rozšířeními a přejalo většinu vzhledových elementů používaných v NetScape. Elementy <blink> od NetScape a <marquee> od Microsoftu byly vypuštěny vzhledem k dohodě mezi oběma společnostmi. Značení pro matematické formulace podobné těm z předchozí verze byly standardizovány až o 14 měsíců později ve specifikaci MathML.

18. prosince 1997

- HTML 4.0 je publikováno jako W3C Recommendation ve třech verzích:
    - Strict, které zakazuje zastaralé prvky
    - Transitional, které zastaralé prvky povoluje

    - Frameset, v rámci které jsou povoleny primárně prvky vztahující se k rámům

- Původně označováno kódovým jménem „Cougar“, HTML 4.0 přejalo řadu elementů a atributů specifických pouze pro některé prohlížeče, ale zároveň byla snaha odstranit vzhledové elementy používané v Netscape a nahradit je styly.

- HTML 4 odpovídalo specifikaci ISO 8879 – SGML.

24. dubna 1998

- HTML 4.0 bylo znovu vydáno s menšími úpravami bez změny čísla verze

24. prosince 1999

- HTML 4.01 bylo publikováno jako W3C. Obsahuje stejné varianty jako verze HTML 4.0 a jeho poslední úprava byla vydána 12. května 2001.

Květen 2000

- Publikována norma ISO/IEC 15445:2000 (takzvaně "ISO HTML", založená na HTML 4.01 Strict) jako mezinárodní standard ISO/IEC.

- Až do poloviny roku 2008 žádné změny nebyly provedeny, protože W3C bylo plně zaměstnáno vývojem paralelního jazyka XHTML, postaveném na XML.

28. října 2014

- HTML5 publikováno jako W3C Recommendation

3.5.1.3. HTML5

Současná verze jazyka přinesla zásadní změny, a měla poněkud komplikovaný vývoj. Po vydání verze HTML 4.01 se W3C rozhodlo zaměřit na vylepšování XHTML, jazyka kompletně postaven na XML, s cílem se zbavit požadavků na zpětnou kompatibilitu, které podle jejich názoru držely celý jazyk zpět. Cílem mělo být eventuální opuštění standardu HTML.

Vývoj okolo tohoto „nového“ jazyka však probíhal velice pomalu, a řada vývojářů prohlížečů nebyl příliš spokojená se současným stavem a plánovaným opuštěním HTML. Nový standard XHTML 2.0 také podle nich byl až příliš zaměřený na dokumenty,

a neumožňoval snadné vytváření internetových diskusí nebo online obchodů. Proto se v roce 2004 jedinci z firem Apple, Mozilla Foundation a Opera Software spojili do WHATWG (Web Hypertext Application Technology Working Group), s cílem vytvořit nástupce HTML. Vývoj HTML5 a XHTML 2 probíhal paralelně až do roku 2009, kdy se World Wide Web Consortium rozhodlo neprodloužit funkční období svojí pracovní skupiny, která se vývoji XHTML věnovala. Od té doby se na rozvíjení jazyka HTML podílejí obě skupiny zároveň. Vývoj postupně probíhal, ale mezi širokou veřejnost vstoupilo HTML5 až v roce 2010, kdy tehdejší ředitel Apple, Steve Jobs, vydal otevřený dopis „Thoughts of Flash“ (Úvahy o Flashi), ve kterém vyjádřil názor, že „Flash již není zapotřebí ke sledování videí nebo jakéhokoliv webového obsahu“ a že „nové otevřené standardy, vzniklé v této mobilní době, jako například HTML5, zvítězí“ [Apple, 2010]. Vzhledem k pozici společnosti Apple jako jednoho z vůdců trhu na poli mobilních zařízení a jejího rozhodnutí nepodporovat Flash na svých zařízeních vzrostl tlak na opuštění Flashe jako standardu pro interaktivní obsah.

HTML5 poskytuje řešení pro řadu problémů, které dříve bylo zapotřebí řešit právě pomocí integrace Flashe. Jedná se například o přehrávání videa nebo zvuku na stránce, nebo používání vektorové grafiky. Ovšem na rozdíl od Flashe nemůže, a ani nemá být, využíváno pro animace nebo interaktivitu, což jsou úkoly, které spadají do oblasti řešených pomocí CSS a JavaScriptu.

Tato nová verze jazyka HTML se od předchozí verze liší především novými, kratšími a snazšími zápisy elementů. Tato změna je vidět především v hlavičce dokumentu, která je nyní mnohem kratší na zápis, jak lze vidět na obrázku níže, který řeší stejný kód, jako obrázek předcházející, ale hlavička je zapsána v HTML5. Lze vidět zkrácený zápis specifikace typu dokumentu, neboli DOCTYPE, která již neobsahuje zápis o verzi a DTD specifikaci. Druhá změna lze vidět v určení znakové sady, kdy odpadly dva ze tří atributů, a zůstal pouze onen „hlavní“, který sadu určuje.

```html
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Hello world</title>
</head>

<body>
<div id="obsah">
    <h1 class="nadpis">Hello world</h1>
    <p>Ukázka kódu v <abbr title="HyperText Markup Language">HTML</abbr></p>
    <!-- Komentář v HTML -->
</div>
</body>
</html>
```

Obrázek 11 - Ukázka HTML5 kódu

Jak již bylo zmíněno, HTML5 přinesl řešení pro vkládání multimédií, tedy videí a zvuků, do dokumentů. Této integrace je docíleno pomocí sady čtyř elementů, <video>, <audio>, <source> a <track>, které se starají o oznámení prohlížeči, že se v dokumentu vyskytuje multimediální soubor, jaké má dostupné zdroje a jestli má nějaké další stopy, jako například titulky. Jejich zápis lze vidět na následujícím obrázku.

```html
<video>
    <source src="czu_ziva_univerzita.mp4" type="video/mp4">
    <source src="czu_ziva_univerzita.webm" type="video/webm">
    <track src="titulky_en.vtt" kind="subtitles" srclang="en" label="English">
    Váš prohlížeč nepodporuje zobrazení videopřehrávače
</video>

<audio>
    <source src="gaudeamus_igitur.mp3" type="audio/mpeg">
    Váš prohlížeč nepodporuje zobrazení audiopřehrávače
</audio>
```

Obrázek 12 - HTML5 <video> a <audio> elementy

Jak lze vidět na ukázce, prvek pro vložení videa má k dispozici dva zdroje, videa umístěná ve stejné složce, jako dokument, s názvem „czu_ziva_univerzita“, s příponami MP4 a WEBM. Prohlížeč se vždy rozhoduje pro první takto zaspanou verzi, kterou je schopný podporovat. Dnešní moderní prohlížeče, včetně nových verzí Internet Explorer, podporují formát MP4, a často tak lze nalézt použitý pouze tento jeden zdroj. Pokud by prohlížeč žádný z nabídnutých zdrojů nepodporoval, nebo měl zobrazování multimédií zakázáno, uživateli se na místě videa zobrazí text zapsaný na konci elementu. K videu je také dostupná stopa obsahující titulky, které v současné době musejí být uložené a na serveru

nahrané ve formátu WebVTT, který je ovšem strukturou téměř identický s dnes asi nejpoužívanějším formátem pro titulky, formátem SubRIP, známého pod zkratkou a koncovkou „srt“, čili převod mezi nimi nečiní žádný problém. Prvek pro zvuk následuje stejnou logiku, v případě více dostupných zdrojů je prohlížečem vybrán první, který je schopen přehrát. V případě obou elementů lze ovlivňovat jejich chování pomocí několika atributů:

- Preload – určuje, co má prohlížeč načítat při otevření dokumentu. Na výběr je „none“ (nenačte se nic), „metadata“ (načtou se metadata, jako například délka videa) a „auto“ (načte se kompletní video)
- Autoplay – pokud je přítomen, prohlížeč začne video automaticky přehrávat, jakmile jej načte.
- Controls – pokud je přítomný, prohlížeč zobrazí ovládací tlačítka a posuvník.
- Loop – Pokud je přítomný, multimediální soubor se bude opakovat

V případě videa je pak dostupný ještě atribut „poster“, který obsahuje adresu na obrázek, které je zobrazený do doby, než je video poprvé spuštěno.

HTML5 kromě práce s multimédii také přineslo řadu nových prvků zabývajících se sémantickým rozdělením dokumentu, které již dnes usnadňuje čtení obsahu webu pomocí asistivních technologií pro například zrakově postižené, a také umožňuje vyhledávačům snadněji lokalizovat podstatný obsah stránky nebo informace o autorovi textu. Za tímto účelem tak specifikace obsahuje následující hlavní elementy:

- <header> - hlavička, často obsahuje například nadpisy nebo navigační menu
- <footer> - patička, často obsahuje informace o autorovi nebo autorských právech
- <main> - hlavní obsah stránky
- <section> - jednotlivé oddíly stránky, například kapitoly
- <article> - obsahové části stránky, například články nebo komentáře
- <nav> - navigace, seznam odkazů
- <figure> - doplňující položka pro obsah, například obrázek, graf, nebo video
- <figcaption> - popisek pro <figure>

3.5.2. CSS

Cascading Style Sheet, překládáno jako kaskádové styly, jsou, stejně jako HTML, známé především pod zkratkou CSS, nebo jednoslovným výrazem “styly”.

Kaskádové styly slouží k určení formátování a vzhledu HTML dokumentu, které se zobrazuje koncovému uživateli. Čili zjednodušeně řečeno, řeší, jak stránka vypadá. Hlavním důvodem pro jejich vznik bylo oddělení vzhledu od struktury a obsahu dokumentu, což byl před jejich zavedením standard, který se s vývojem webu ukazoval jako stále více nevyhovující. Díky tomuto oddělení je tak možné snadno udržovat sjednocený vzhled na více dokumentech, nebo naopak dokument nabídnout ve více vzhledech, které jsou přístupnější pro rozdílné typy čtenářů. Například dnešní zařízení, umožňující lidem se zrakovým postižením nebo i kompletní slepotou prohlížet web, mají možnost interpretovat kaskádové styly do zvukového či hmatatelného výstupu. A samozřejmě v dnešní době, kdy je internetové připojení a obsah webu v civilizovaných zemích doslova „na dosah ruky“ pomocí různých mobilních zařízení, umožňují kaskádové styly upravit předkládaný formát obsahu podle velikosti obrazovky, na které si uživatel web prohlíží.

3.5.2.1. Struktura stylu

Základní struktura kaskádových stylů je značně jednoduchá. Každý styl je sadou pravidel, z nichž každé je složeno z jednoho nebo více „selektorů“ a jednoho „bloku deklarací“. Deklarace pravidel používají řadu anglických klíčových slov, a je z nich tak často na první pohled patrné, k čemu slouží.

Velkou podstatou kaskádových stylů je takzvané dědění, neboli že podřazené prvky přebírají nadefinované vlastnosti od svých nadřazených prvků, pokud není tato vlastnost přepsána. Z toho také vyplývá důvod, proč se kaskádové styly vlastně jmenují kaskádové. Styl je prohlížečem vyhodnocován řádek po řádku, a pozdější definice pro stejný selektor přepisuje definici předcházející. Proto se deklarace pravidel pro speciální zobrazování, například na mobilních telefonech, často uvádějí až na konci dokumentu, aby se vyhnulo případným problémům.

Selektorů existuje celá řada, ovšem mezi nejzákladnější a nejčastěji používané patří následující trojice:

- Element – napsaný jednoduše jako element v HTML, pouze bez špičatých závorek
- Třída – zápis začíná tečkou, a dále písmenem, pomlčkou, podtržítkem, nebo číslem. Obecně se nedoporučuje začínat názvy číslem.
- Identifikátor – zápis začíná křížkem (#), a dále platí stejná pravidla jako pro třídu

Mezi další možnosti patří například selektory podle atributu, podle nadřazeného prvku a tak dále. Ukázku zápisu CSS kódu lze vidět na následujícím obrázku, kde jsou použity právě ony zmíněné nejčastější tři typy selektorů

```
#obsah {
    font-family:Verdana,sans-serif;
}

.nadpis {
    font-style:italic;
    font-variant: small-caps;
}

abbr {
    font-weight: 700;
    text-decoration: none;
    border: 1px solid black;
    padding: 1px 5px 1px 5px;
}
```

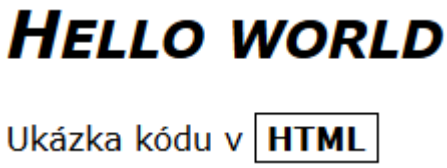
Obrázek 13 - Ukázka CSS kódu

Na obrázku výše lze vidět nastavení stylu pro příklad z kapitoly o HTML. Obalovacímu prvku s identifikátorem obsah je nastavená rodina písma neboli font. Jak bylo vysvětleno dříve, CSS pravidla se dědí od nadřazených elementů, čili bude tato rodina písma aplikovaná na všechny potomky, to jest nadpis, odstavec i zkratku.

Samotný nadpis je stylován prostřednictvím svojí třídy nadpis. Pokud by se na stránce vyskytoval další prvek se stejnou třídou, byla by na něj aplikována stejná pravidla. V ukázce je přiřazena textu kurzíva a kapitálky.

Zkratka je stylována přímo svým elementem, čili budou takto nastavená pravidla platit pro každý tag <abbr>, pokud by nebyl později ve stylu přepsán nějakou specifickou třídou či identifikátorem. V ukázce je nastaveno tučné písmo, odstraněno podtržení a nastaveno ohraničení.

Výsledek kombinace dříve ukázaného HTML a výše ukázaných kaskádových stylů, zobrazený v prohlížeči, lze opět vidět v následujícím obrázku.



Obrázek 14 - HTML kód naformátovaný pomocí CSS

3.5.2.2. Historie stylů

Jak již bylo zmíněno, kaskádové styly vznikly z důvodu potřeby oddělit obsah a strukturu dokumentu od jeho vzhledu. Jak probíhal vývoj HTML, postupně začalo obsahovat množství stylistických možností, které vývojáři potřebovali. Tato vylepšení však znamenala výšení komplexnosti HTML kódu, a vzhledem k odlišným implementacím v rozdílných prohlížečích nebylo snadné udržovat jednotný vzhled webů, o možnosti využívat více stylů pro různé účely, jako například tisk, nemluvě.

V roce 1994 představil Håkon Wium Lie návrh stylu, na kterém pracoval společně s Bertem Bosem na konferenci „Mosaic and the Web“ v Chicagu. Tento návrh zaujal představitele již existujícího World Wide Web Consortium, a bylo rozhodnuto, že se organizace bude dále podílet na jeho vývoji, který probíhal až do konce roku 1996, kdy byla oficiálně vydána první verze jazyka, označená jako CSS level 1. O rok později byla představena verze CSS 2. Následující verze, CSS 3, odstartovaná v roce 1998, přestala být vydávána jako jeden velký celek doporučení, ale po jednotlivých modulech, zaměřených na jednotlivé stylistické úkony. Vzhledem k této změně je mnoho částí kaskádových stylů v různém stádiu vývoje, některé označované i jako CSS 4.

3.5.3. JavaScript

JavaScript je skriptovacím jazykem, a jediným ze zmíněných jazyků, který není známější pod nějakou zkratkou, a většinou se používán jeho plný název, aby se předešlo zaměnění s jazykem Java, který s ním nemá příliš mnoho společného. Nejblíže ke zkratce se blíží používání slova "skript", ovšem vzhledem k množství různých skriptovacích jazyků, které webové prohlížeče umí zpracovat, je i zde příležitost k nedorozumění, i když od příchodu HTML5, které standardizovalo výchozí hodnotu pro element <script> jako JavaScript lze předpokládat, že k těmto nedorozuměním bude docházet stále méně a méně.

Jako skriptovací jazyk má JavaScript za úkol především poskytovat okamžitou odezvu na uživatelovo chování na stránce, ať už se jedná o kliknutí na určitý prvek v dokumentu, uplynutí nějakého času nebo v podstatě jakákoliv myslitelná činnost.

3.5.3.1. Struktura skriptu

JavaScript má strukturu kódu velice podobnou většině objektově orientovaných programovacích jazyků vycházejících z rodiny jazyků C. Lze tak nalézt proměnné, struktury, výpočty, porovnávání hodnot, podmínky, a podobné prvky. V rámci programování dynamických webových stránek ale velká část možností jazyka využívá velice zřídka, protože nejčastější využití JavaScriptu v těchto situacích je manipulace s elementy na stránce.

Pro tento účel se nejčastěji využívá vyhledání nějakého elementu v dokumentu, a provedení nějaké změny. Většina příkazů tak začíná slovem „document", které jazyku udává, že má pracovat s nějakou částí stránky, následované příkazem pro zaměření objektu, nejčastěji pomocí identifikátoru, a poté příkazem, který se má provést. Ukázku běžného kódu na dynamické webové stránce lze vidět na následujícím obrázku.

```
document.getElementById("obsah").style.display = "none";
```

Obrázek 15 - JavaScript příkaz

Příkaz uvedený na obrázku má jednoduchý úkol – vybrat na stránce prvek s identifikátorem (atributem id) „obsah", a nastavit mu styl „display: none;", čili mu odebrat viditelnost a vyřadit jej z toku dokumentu.

3.5.3.2. Historie JS

JavaScript vznikl během deseti dní v průběhu května 1995. Jeho autorem byl Brendan Eich, tehdy pracovník společnosti NetScape. Původní název jazyka zněl Mocha, ovšem v září stejného roku byl přejmenován na JavaScript, a po udělení licence od společnosti Sun byl v prosinci opět přejmenován na JavaScript.

Během následujících let byl jazyk standardizován a probíhaly snahy o jeho další vývoj, které ale všechny narážely na neochotu velkých hráčů, jako například Microsoft, se na vývoji plně podílet a integrovat podporu jazyka do svých prohlížečů. A zatímco se nasazení jazyka v korporátní prostředí příliš nedařilo, v open-source komunitách začal v roce 2005 prožívat svojí renesanci. Tu odstartovala práce Jesseho Jamese Garetta, ve které představil sadu technologií postavených na JavaScriptu, které označil jako AJAX (Asynchronous JavaScript and XML). Tato sada funkcí umožňovala dynamickou změnu obsahu dokumentu bez nutnosti jej celý znovu načítat, což by mohlo vést ke ztrátě uživatelem zadaných dat, nebo zbytečnému načítání všech prvků stránky znovu. V době, kdy pomalu ale jistě začínal rozmach webových aplikací, byla tato sada funkcí zcela přelomová. JavaScript tak vystoupil opět na výsluní, podporován vydáním řady open source knihoven, jako například jQuery, Prototype, Mootools nebo Dojo.

Tato renesance jazyka pomohla v roce 2008 k setkání všech dříve neochotných stran ke společnému setkání v norském Oslu, kde se společně dohodli na dalším vývoji jazyka a ustanovení sady pravidel pro další vývoj jazyka, jejichž finální podoba vešla v roce 2009 ve známost jako Harmony.

3.5.3.3. jQuery

jQuery je jméno open source knihovny pro jazyk JavaScript, která kompletně změnila postupy, kterými mnoho vývojářů webů pracuje s interaktivitou stránek. Jedná se o natolik populární rozšíření jazyka, že jí řada lidí mimo branži mylně považuje za kompletně nový jazyk separátní od JavaScriptu.

Knihovna jQuery je postavena na základní filozofii „najdi HTML prvek a něco s ním udělej“. První krok, tedy „hledání“ prvku na stránce, je jedna z velkých výhod jQuery, knihovna je totiž kompletně postavená na základě CSS selektorů, tedy primárně identifikátorů a tříd, které je oproti čistému JavaScriptu mnohem jednodušší na zápis

a pochopení. Ovšem není problém prvek „zaměřit“ i pomocí atributu nebo jiných metod. Hlavní sílou knihovny ale je především to, že zjednodušuje a „zlidšťuje“ většinu příkazů, které se ve světě JavaScriptu existují. Zatímco příkaz v „čistém“ JavaScriptu je dlouhý několik slov a může být značně nepřehledný, jQuery tento příkaz zkrátí na jedno slovo, ze kterého je i méně zkušenému kodérovi téměř ihned jasné, co bude výsledkem. Příklad toho, jak je kód psaný pomocí této knihovny mnohem jednodušší a přehlednější, lze vidět na následujícím obrázku, který řeší stejný úkol, jako obrázek v kapitole 3.5.3.1.

```
$("#obsah").hide();
```

Obrázek 16 - jQuery příkaz

Mezi další výhody knihovny jQuery patří značné usnadnění práce se zmíněným AJAXem neboli načítáním části stránek bez nutnosti obnovovat celý dokument. Výhodu jQuery oproti regulérnímu JavaScriptu v této oblasti lze vidět v porovnání kódů na obrázku níže. Obě zobrazené funkce mají stejný úkol – po svém zavolání (například skrze stisknutí tlačítka) pomocí AJAXu načíst do elementu s identifikátorem „obsah“ data z textového souboru „novy_obsah.txt“.

```
/*
Zápis funkce v čistém JavaScriptu
*/

function zmenObsah() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      document.getElementById("obsah").innerHTML = xhttp.responseText;
    }
  };
  xhttp.open("GET", "novy_obsah.txt", true);
  xhttp.send();
}

/*
Zápis funkce pomocí knihovny jQuery
*/

function zmenObsah(){
    $("#obsah").load("novy_obsah.txt");
};
```

Obrázek 17 - Porovnání AJAXu mezi JavaScriptem a jQuery

3.5.4. PHP

PHP neboli PHP: Hypertext Preprocessor (jméno využívá podobnou rekurzivní zkratku jako například GNU, i když původně zkratka znamenala *Personal HomePage*) je serverový programovací jazyk, čili na rozdíl od JavaScriptu se požadavek zpracovává na straně serveru, a klientovi je poslán pouze výsledek. Obecně je brán jako doplněk HTML.

Jedná se o dlouhodobě nejpoužívanější jazyk pro vytváření dynamických webových stránek, podle webu w3techs.com PHP využívá přes 80% webových stránek [w3techs.com]. Logem jazyka je modrý slon s nápisem PHP na boku, pojmenovaný elePHPant.

3.5.4.1. Struktura PHP

Jako většina programovacích jazyků vycházejících z rodiny jazyků C, PHP nabízí klasické proměnné, třídy, funkce, porovnávání hodnot, podmínky, cykly a podobné prvky. V rámci programování dynamických webových stránek jsou prvky PHP využívány v mnohem kompletnějším rozsahu, než je tomu u JavaScriptu. Ukázka struktury PHP kódu je k nahlédnutí na následujícím obrázku.

Ukázka kódu obsahuje řadu prvků, které jsou využívány právě WordPressem. Lze v ní vidět třídu, nazvanou „Osoba“, které lze definovat tři proměnné („jméno“, „příjmení“, „rokNarozeni“), na základě kterých pomocí své funkce vygeneruje pozdrav obsahující křestní jméno, příjmení a věk. V případě věku je proveden výpočet podle zadaného roku narození tak, že se odečte od systémového času, a na základě toho je přiřazeno správné slovní zakončení. Následuje definice nové osoby, a zavolání výpisu v HTML kódu. V rámci WordPressu jsou v šablonách podobným způsobem vyvolávány jednotlivé kusy obsahu z databáze (konstrukce třídy je součástí jádra WordPressu, při tvorbě šablony je využito pouze samotné zavolání funkce)

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Ukázka PHP</title>
</head>
<?php
class Osoba {
    public $jmeno;
    public $prijmeni;
    public $rokNarozeni;

    public function __construct($jmeno,$prijmeni="",$rokNarozeni=""){
        $this -> jmeno = $jmeno;
        if ($prijmeni!=""){
            $this -> prijmeni = " ".$prijmeni;
        }
        if ($rokNarozeni!=""){
            $vek = date("Y")-$rokNarozeni;
            if ($vek == 1) {
                $vek = " a je mi ".$vek." rok.";
            }
            else if ($vek <= 4){
                $vek = " a jsou mi ".$vek." roky.";
            }
            else {
                $vek = " a je mi ".$vek." let.";
            }
            $this -> vek = $vek;
        }
    }
    public function predstaveni(){
        return "Zdravím, jmenuji se ".$this->jmeno.$this->prijmeni.$this->vek;
    }
}

$autor = new Osoba("Lukáš","Janeček",1991);
?>
<body>
<?php echo $autor->predstaveni();?>
</body>
</html>
```

Obrázek 18 - Ukázka PHP kódu

3.5.4.2. Historie PHP

Historie PHP se začíná odehrávat v roce 1994, kdy Rasmus Lerdorf napsal první verzi jako sadu několika CGI skriptů v jazyce C, původně navrženou jako způsob, jak sledovat počet návštěv na jeho online životopise (z čehož vychází původní název PHP, Personal Home Page Tools). Eventuálně tuto sadu nástrojů rozšířil o možnost interakce s databázemi a možnost vytvářet jednoduché interaktivní dynamické webové aplikace, jako například knihu návštěv. V polovině roku 1995 pak uvolnil zdrojový kód pro sadu nástrojů mezi širokou veřejnost.

O pár měsíců později Lerdorf skripty rozšířil a na chvíli opustil od názvu PHP, a nástroje pojmenoval FI (Forms Interpreter). Tato nová implementace se již podobala PHP, jak je známé dnes, byly přítomné proměnné, interpretace dat z formulářů a vkládání do HTML, v té době poněkud nepraktické, vývojáři museli pro vložení používat HTML komentáře. I přes všechny tyto komplikace podpora FI jako sady CGI skriptů stále rostla. V říjnu 1995, Lerdorf kompletně přepsal zdrojový kód a vrátil zpět jméno PHP. Nástroje nyní měly strukturu podobnou jazykům z rodiny C, aby byl přechod pro nové vývojáře snazší. Také se začala zvažovat implementace PHP pro systémy Windows.

Zásadní změna přišla v dubnu 1996, kdy byly nástroje opět kompletně přepsány, a začaly se vyvíjet ve skutečný programovací jazyk (i když jak Lerdorf později prohlásil, nikdy neměl v úmyslu jazyk vytvořit, pouze přidával k nástrojům další logické kroky), který byl pojmenovaný PHP/FI, kombinující názvy předchozích verzí. V této verzi byla zabudována podpora řady databázových systémů, cookies, uživatelem definovaných funkcí a řada dalších funkcí. V červnu stejného roku se PHP/FI dočkalo verze 2.0, která zároveň byla jeho poslední, protože na podzim stejného roku, kdy jazyk přešel z beta verze do finální verze, byl jeho parser opět kompletně přepisován. Ale i během takto krátkého života s PHP/FI získalo řadu podporovatelů v tehdy mladém světě webového vývoje, kdy jej využívalo přibližně 1%.

Zmíněné přepsání parseru měli na svědomí dva studenti na izraelské univerzitě, Andi Gutmans a Zeev Suraski, kteří se snažili PHP/FI využít pro tvorbu e-commerce systému, ale shledali, že nástroje jsou pro jejich potřeby ne zcela vyhovující. Po diskusích s Rasmusem Lerdorfem, kdy probírali možnosti vylepšení různých částí PHP, se rozhodli, že vytvoří zcela nový, nezávislý jazyk. Ten měl být vydán pod novým jménem, které již

neodkazovalo na zdánlivě limitující omezení na osobní stránky. Tak vznikl současný název, rekurzivní zkratka PHP: Hypertext Preprocessor. Jednou z jeho nejsilnějších stránek byla velká rozšiřitelnost. Vynikající podpora pro řadu databázových systémů, protokolů a API přilákalo desítky vývojářů, kteří přinesli řadu modulů. Kromě toto tato verze přinesla podporu objektů a mnohem konzistentnější syntaxi. V červnu 1998 bylo PHP 3.0 oficiálně představeno jako oficiální nástupce PHP/FI 2.0. Po devíti měsících otevřeného testování, kdy bylo oficiálně oznámeno vydání PHP 3.0, byl jazyk používán na více než 10% domén běžících jak na Unixových systémech, tak na systémech Windows.

V zimě roku 1998 začali Andi Gutmans a Zeev Suraski pracovat na přepisování jádra PHP, s cílem zlepšit výkon komplexních aplikací a zlepšit modularitu jazyka. Tyto věci sice byly umožněny vznikem PHP 3.0, ale jazyk nebyl schopen s nimi pracovat příliš efektivně. Z jejich společné práce vzešel Zend Engine (pojmenovaný podle prvních dvou písmen jmen svých tvůrců), který tyto zadané cíle splnil, a byl poprvé představen v roce 1999. Na něm založené PHP 4.0, společně s řadou dalších nových možností, bylo oficiálně vydáno v květnu 2000, tedy téměř 2 rok po svém předchůdci. Mezi hlavní inovace této verze patřila podpora více webových serverů, HTTP relací nebo bezpečnější zpracování uživatelských vstupů.

Po dlouhém vývoji a řadě zkušebních verzí bylo v červenci 2004 vydáno PHP 5. Mezi hlavní bod patří nový Zend Engine II. Mezi další pak zlepšená podpora objektového programování, nebo rozšíření PHP Data Objects (PDO), které přináší konzistentní a jednoduché metody pro napojení k databázím a řada vylepšení zaměřených na výkon.

V posledních dvou letech (2014-2015) probíhal vývoj nové verze, označené jako PHP 7. Verze 6 byla po neúspěších s implementací ukončena a nikdy nevydána, a její části byly představeny jako dílčí verze PHP 5, ovšem v řadě publikací se název PHP 6 objevil, a tak bylo v hlasování rozhodnuto pro „přeskočení“ při pojmenovávání nové verze. Hlavní změnou je třetí verze Zend Engine, která v testech poskytla téměř dvojnásobné zrychlení právě ve WordPressu.

4. Vlastní práce

Jak bylo vytyčeno v zadání a v kapitole Cíl práce, praktická část této diplomové práce se zabývá návrhem a realizací vzhledu pro CMS WordPress za využití moderních trendů v oblasti webdesignu, čili především HTML5 a CSS3.

V první části této kapitoly proberu základní myšlenku finálního vzhledu a požadavky, které jsem si pro něj vytyčil. V druhé části projdu počáteční fázi návrhu, kde představím tzv. „wireframy“, čili návrhy rozložení prvků na stránce bez jakékoliv grafiky, sloužící pro specifikování představy o podobě stránky. V třetí části představím grafické návrhy, ze kterých jsem při samotné tvorbě šablony vycházel. Ve čtvrté části pak bude probírán samotný postup tvorby vzhledu a popis jednotlivých funkcí v kódu. Ve finální fázi budu řešit nasazení šablony do systému a demonstraci funkčnosti.

4.1. Základní myšlenka a požadavky

4.1.1. Základní myšlenka

S myšlenkou vytvořit komplexní šablonu pro své WordPressem poháněné stránky, zaměřené především na novinky okolo různých počítačových her, jsem se zabýval již delší dobu, a chtěl bych touto cestou znovu poděkovat docentu Merunkovi za to, že mi toto téma dovolil zpracovat jako diplomovou práci.

Cílem tak bylo navrhnout pokud možno snadno upravitelnou univerzální šablonu, kterou by bylo možné s minimem úprav nasadit na více webech a zkrátit tak čas a náklady potřebné k rozjetí základní verze stránek.

4.1.2. Požadavky na šablonu

Pro šablonu jsem si stanovil několik základních požadavků, vycházejících z potřeb webů, pro které jsem jí navrhoval, a moderních požadavků na vzhled webu. Požadavky tak byly následující:

- Responzivní design pro zobrazování na mobilních zařízeních
    - Vzhledem k tomu, na jaký typ webů je šablona cílena, nebyl zvolen postup „mobile first“, kdy se navrhuje od nejmenších obrazovek po největší, ovšem v dnešní době je zobrazování na menších obrazovkách důležitou součástí pro návštěvníky.

- Přehlednost obsahu
    - Návrh by pokud možno měl být minimalistický, neobsahující přemíru prvků, které by mohly pro čtenáře webu být rušivé.
- Postranní panel s fixní šířkou
    - Panel se musí rozdělit na menších obrazovkách na dva různé, umístěné v oddělených částech stránky na menších obrazovkách
- Modifikovatelnost
    - Návrh by měl být snadno modifikovatelný, ideálně bez nutnosti velkých zásahů do kódu

4.2. Wireframy

Wireframy představují první krok v konkretizaci představ zadavatele. Slouží k představení, jak budou jednotlivé prvky na stránce rozmístěné, jak budou fungovat a především k odlazení případných rozdílných názorů zadavatele a zhotovitele.

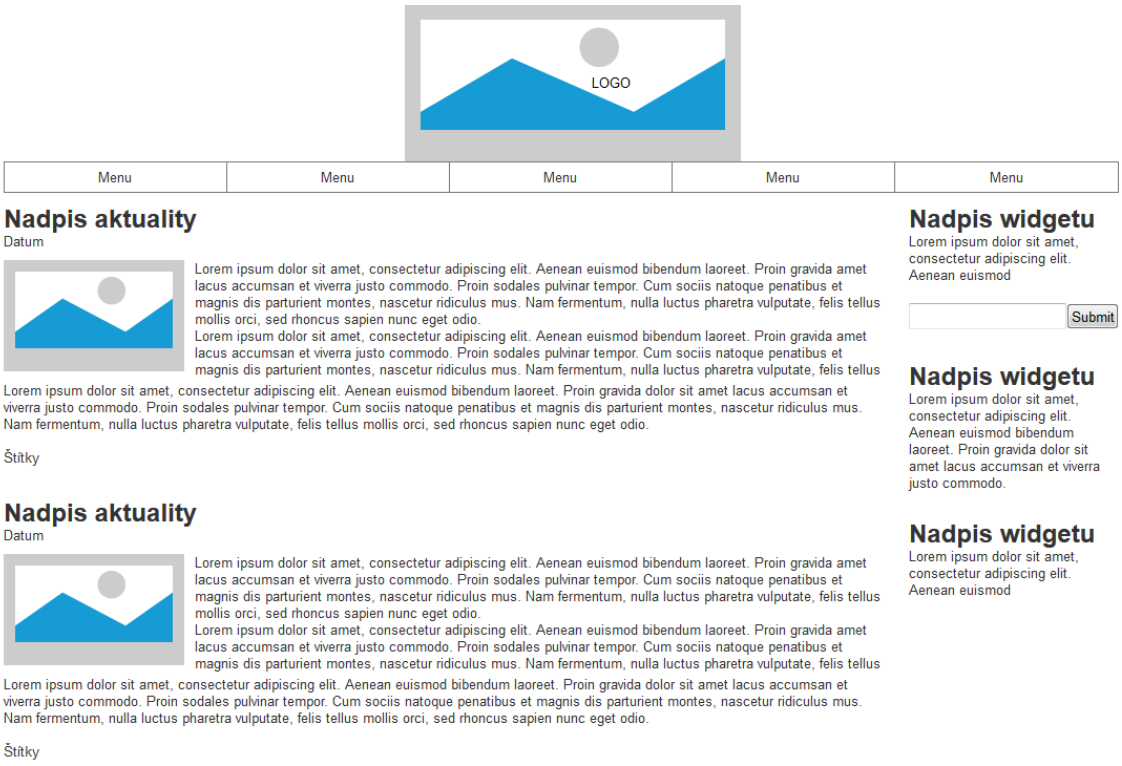
Jak vyplývá z názvu, drátěné modely jsou zcela rozdílné od grafického návrhu, který se může od wireframu v určitých detailech lišit. Ideálně se nedoporučuje vůbec používat barevné nebo jiné grafické prvky, a jakoukoliv grafiku na webu (například logo) nahradit zástupným symbolem.

V dnešní době existují dva hlavní přístupy k jejich tvorbě, buďto takzvaně „ručně“, kdy se kreslí na papíry, případně prostřednictvím grafického programu. Druhým přístupem jsou pak interaktivní modely, vytvořené pomocí jednoho z řady programů na tuto problematiku zaměřených.

V rámci této práce jsem se rozhodl zvolit druhý přístup, a wireframy navrhnout pomocí programu Axure, se kterým jsem přišel do kontaktu v rámci zaměstnání a pracuje se mi s ním mnohem lépe, než s řadou ostatní alternativ, jako například online aplikace Draw.io, nebo program Balsamiq Mockups. Pro potřeby šablony jsem navrhl tři drátěné modely, zobrazující úvodní stránku, stránku zobrazující jeden konkrétní příspěvek či stránku, a úvodní stránku zobrazenou na mobilním zařízení. Tyto pohledy byly zvoleny z důvodů, že se jedná o nejčastěji zobrazované stránky na typu webů, pro který je šablona určena.

4.2.1. Úvodní stránka

Úvodní stránka představuje místo, kam se návštěvnici dostanou ihned po přístupu na web. Obsahuje tak tedy hlavní podstatu stránky, což jsou v případě webů, na které je šablona cílena, novinky. Lze tak vidět výpis novinek, které byly v administraci vloženy, hlavičku složenou z loga webu a rozbalovacího horizontálního menu, a postranní panel, který je ve vyšším rozlišení, které toto zobrazení představuje, složený do jednolitého celku.



Obrázek 19 – Wireframe úvdní stránky

4.2.2. Stránka s příspěvkem

Tento návrh zobrazuje drátěný model pro zobrazení jedné konkrétní aktuality nebo statické stránky. Hlavním rozdílem oproti předcházejícímu návrhu je přítomnost komentářů pod textem a pochopitelně celého obsahu daného příspěvku.

Obrázek 20 – Wirefraem detailu příspěvku

4.2.3. Zobrazení na mobilním zařízení

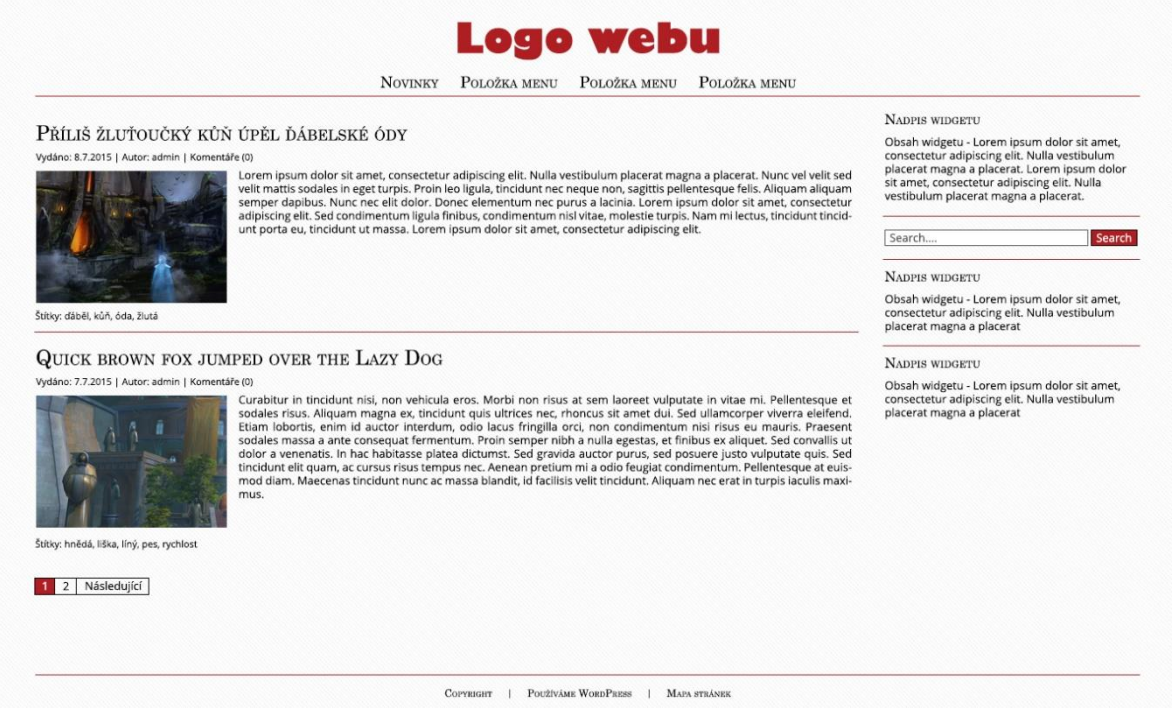
Při zobrazení na mobilním zařízení bylo nutné vzít v potaz jeden z hlavních požadavků, čili aby došlo k oddělení horní části postranního panelu a jejího přesunutí nad obsah webu. Také je možné vidět, že horní nabídka byla sbalena do jednoho prvku, který se při kliknutí rozbalí do plné délky. V rámci zjednodušení, a menší velikosti obrázku, obsahuje tento model pouze jeden příspěvek a ihned za ním lze vidět druhou část postranního panelu.

Obrázek 21 – Wireframe úvodní stránky na mobilním zařízení

4.3. Grafický návrh

Grafický návrh webu vychází z drátěných modelů, a aplikuje na ně barvy, styly písma a další prvky. V praxi se ve většině případů od původních wireframů částečně liší v rozmístění některých prvků, například loga nebo zarovnání menu k určité straně designu, ovšem měl by stále dodržovat strukturu, kterou v předchozích fázích zadavatel odsouhlasil. Právě na základě odsouhlaseného grafického návrhu pak probíhá práce kodéra, který jej převádí do webového kódu.

K tvorbě grafického modelu je používán nějaký grafický editor. Jaký v podstatě není omezeno, záleží pouze na schopnostech a preferencích grafika, je možné použít jak rastrový, tak vektorový editor. V rámci této diplomové práce jsem na základě svých preferencí zvolil grafický program Adobe Photoshop CC, ve kterém jsem převedl první drátěný model do grafické podoby, podle které jsem pak dále pracoval. Tento grafický návrh je možné vidět na následujícím obrázku.



Obrázek 22 - Grafický návrh úvodní stránky

4.4. Tvorba šablony

V této kapitole budou popsány jednotlivé soubory, které šablonu pohánějí. V rámci přehlednosti jsem zvolil rozdělení do dvou skupin – části šablony, které se opakují na každé stránce, a části, které řeší zobrazení jednotlivých stránek.

4.4.1. Společné segmenty šablony

Jak bylo zmíněno, v této části popíši ty soubory šablony, které jsou společné pro ostatní stránky. Jedná se tak o soubor definující funkce šablony (functions), kaskádový styl, hlavičku, patičku a postranní panely.

4.4.1.1. Funkce tématu

Soubor functions.php specifikuje řadu funkcí WordPressu, které má šablona podporovat, a je jednou z hlavních součástí každé pokročilejší šablony. Mezi základní řešené oblasti by mělo vždy patřit načtení kaskádových stylů a skriptů pomocí interních funkcí, které zajistí jejich načtení ve správném pořadí, aby se předešlo případným problémům, jako chybějící knihovna jQuery pro skripty či nahrání stylů ve špatném pořadí, což by vedlo k narušení designu. Provedení tohoto kroku v rámci vytvářené šablony lze vidět na následujícím obrázku.

```
// Register Styles
function onlook_styles() {

    wp_register_style( 'normalize', get_template_directory_uri().'/normalize.css', false, '3.0.2', 'all' );
    wp_enqueue_style( 'normalize' );

    wp_register_style( 'roboto-slab', '//fonts.googleapis.com/css?family=Roboto+Slab:700,400&subset=latin,latin-ext', array( 'normalize' ), '1.0', 'all' );
    wp_enqueue_style( 'roboto-slab' );

    wp_register_style( 'style', get_stylesheet_uri(), array( 'normalize', 'roboto-slab'), '1.0', 'all' );
    wp_enqueue_style( 'style' );

}
add_action( 'wp_enqueue_scripts', 'onlook_styles' );

// Register Scripts
function onlook_scripts() {

    wp_deregister_script( 'main' );
    wp_register_script( 'main', get_template_directory_uri().'/scripts/main.js', array( 'jquery', 'jquery-ui-core' ), '1.0', true );
    wp_enqueue_script( 'main' );

    if ( is_singular() && comments_open() && get_option( 'thread_comments' ) ) {
        wp_enqueue_script( 'comment-reply' );
    }
}
add_action( 'wp_enqueue_scripts', 'onlook_scripts' );
```

Obrázek 23 - Načtení stylů a skriptů

Každý soubor stylu je nejdříve „zaregistrován“ funkcí wp_register_style, a jsou mu určeny závislosti, což WordPressu umožní zajistit správné pořadí načtení. To pak probíhá postupně pomocí funkce wp_enqueue_style. Nejdříve je načten styl Normalize.css, jehož autory jsou Nicolas Gallagher a Jonathan Neal, a jedná se o nejpoužívanější kaskádový styl pro sjednocení chování stylů v různých prohlížečích, který využívají i například Twitter,

Soundcloud nebo GitHub [Normalize.css]. Následuje načtení písma Roboto Slab z knihoven Google Fonts, který je v šabloně využívaný pro nadpisy. Poslední načtený je vlastní styl vytvářené šablony, který tak podle pravidel CSS může snadno přepsat předcházející deklarace. Podobná metodika lze vidět i u načítání souborů JavaScriptu, kdy hlavní soubor se skripty pro šablonu je závislý na knihovně jQuery a jejím rozšíření jQuery UI, které jsou obsažené v jádře WordPressu, a není tak zapotřebí je vkládat dodatečně.

Dalšími kroky jsou nastavení pro podporu widgetů v postranních panelech, navigačních menu, a nastavení některých funkcí WordPressu, jako přikládání hlavního obrázku k příspěvkům, nebo možnost překládání šablony do dalších jazyků pomocí lokalizačních funkcí. Tyto funkce využívají standardní funkce společné každé šabloně WordPressu, a jsou k nahlédnutí na přiloženém disku.

Hlavní část funkcí této šablony se ale zabývá její snadnou přizpůsobitelností přímo z administrativy webu, bez nutnosti jakkoliv zasahovat do zdrojového kódu. Možnosti úpravy této šablony zahrnují následující části webu:

- Obrázek v záhlaví / logo
- Obrázek na pozadí
- Barva pozadí, textu, odkazů a sekundární barva šablony
- Zobrazení informací o WordPressu a odkazu na XML mapu webu v patičce

S pomocí těchto několika voleb lze vytvořit na první pohled odlišně vypadající weby (viz následující obrázek), které již od začátku mohou mít svojí vlastní barevnou identitu. V kombinaci s „child“ šablonami je pak přechod na unikátní styl plynulejší a rychlejší.

Obrázek 24 - Ilustrace možnosti přizpůsobení

Obrázky a barvu pozadí řeší instalace WordPressu v základu, v rámci této práce tak bylo zapotřebí rozšířit tyto funkce o možnosti zabarvení zbylých tří prvků a možnosti zobrazení textů v patičce.

O celý proces přizpůsobování se uvnitř WordPressu stará framework Customizer, který je možné téměř libovolně rozšiřovat a upravovat. Každá přidaná možnost se skládá ze dvou částí kódu. První deklaruje funkci, nastavuje jí výchozí hodnotu a volá funkci pro očištění vstupu, aby se předešlo případným možnostem narušení systému pomocí code inject útoku. Druhou částí je nastavení objektu zodpovědného za zobrazení nastavení pro uživatele. Na obrázku níže lze vidět nastavení pro jednu z barev, konkrétně pro barvu textu.

```php
$wp_customize->add_setting(
    'onlook_text_color',
    array(
        'default'     => '#000000',
        'sanitize_callback' => 'sanitize_hex_color',
    )
);

$wp_customize->add_control(
    new WP_Customize_Color_Control(
        $wp_customize,
        'text_color',
        array(
            'label'      => __( 'Text Color', 'onlook' ),
            'section'    => 'colors',
            'settings'   => 'onlook_text_color'
        )
    )
);
```

Obrázek 25 - Přidání možnosti pro úpravu barvy

Jak lze vidět, je deklarována výchozí hodnota na černou, a je vytvořen objekt umožňující uživatelsky přívětivou úpravu barvy pomocí palety známé z většiny grafických programů. Tento objekt je přiřazen do existující sekce ovládacího panelu s identifikátorem „colors“. Pomocí lokalizační funkce je tomuto nastavení přidělen nadpis.

V případě, že sekce neexistuje, je jí zapotřebí vytvořit předtím, než bude možné se na ní odkazovat. Proces, společně s vytvoření nastavení a objektu pro zobrazení jedné z položek v patičce, lze vidět na následujícím obrázku.

```php
$wp_customize->add_section('onlook_footer_section' , array(
        'title'     => __('Footer', 'onlook'),
        'priority'  => 90
    )
);

$wp_customize->add_setting('show_footer_sitemap', array(
        'default'        => 0,
        'capability'     => 'edit_theme_options',
        'sanitize_callback' => 'esc_url_raw',
    )
);

$wp_customize->add_control(
    new WP_Customize_Control(
        $wp_customize,
        'show_footer_sitemap',
        array(
            /* translators: 1: sitemap */
            'label'     => sprintf(_x('Show %1s','verb', 'onlook'),__( 'Sitemap','onlook')),
            'description'     => __('Links to sitemap.xml in your root directory', 'onlook'),
            'section'   => 'onlook_footer_section',
            'settings'  => 'show_footer_sitemap',
            'type'      => 'checkbox',
        )
    )
);
```

Obrázek 26 - Vytvoření sekce Customizeru a přidání zaškrtávacího pole

Sekci je za využití lokalizačních funkcí přiřazen titulek, a přiřazena priorita, určující umístění v menu frameworku, jehož hodnota vychází z předem určených hodnot ostatních položek, které jsou dohledatelné v rámci dokumentace.

Aby se výsledky promítly z Customizer frameworku do samotné stránky, je nutné přidat funkci, která na začátek dokumentu pomocí hooku wp_head přidá element <style>, který hodnoty zadané v rámci přizpůsobení vypíše jako soubor pravidel kaskádových stylů.

Vytvoření nastavení je shodné jako u barvy, zde konkrétně je výchozí hodnota nulová, protože se jedná o zobrazení XML mapy stránek, čili dokumentu, který není dostupný ihned po instalaci WordPressu, a vytvářel by tak odkaz vedoucí na neexistující stránku. Objekt je odlišný od vytváření barevné palety, jedná se o generický objekt, který v tomto případě vytvoří zaškrtávací pole. Také bude mít zobrazení popisek a nadpis, obojí opět vypsáno za pomocí lokalizačních funkcí.

Poslední podstatnou částí souboru functions je určení, jak vypadá individuální komentář po příspěvky. V rámci této šablony jsem se nechal inspirovat rozložením a vzhledem komentářů v systému Disqus, kterými by pochopitelně bylo možné komentáře nahradit, pokud by se k tomu vlastník webu rozhodl. Každý komentář je obalen

v sémantickém HTML5 elementu <article>, který je významově pro komentáře a další textové obsahy určen.

4.4.1.2. Kaskádový styl

Kaskádový styl, uložený v souboru style.css, definuje vzhled celé šablony. Při jeho tvorbě jsem vycházel ze stylů, které používá řada frameworků, například pro definování seznamů nebo tlačítek. V rámci této sekce jej nebudu celý vypisovat, protože se jedná o poměrně standardní zápis. Vypíši zde pouze sekci zodpovědnou za řešení jednoho z hlavních vytyčených požadavků, kterým bylo rozdělení postranního panelu na horní a dolní sekci, pokud si uživatel web prohlíží na menší obrazovce.

Tento problém byl jedním z těch, u kterého při vývoji nastalo mírné zaseknutí a vyžadovalo několikanásobnou změnu postupu. Cílem bylo vyhnout se využití JavaScriptu k manipulaci s objektovým modelem dokumentu, a vytvořit řešení čistě pomocí kaskádových stylů. Vzhledem k zadaným parametrům, tedy flexibilní šířce obsahové části a pevné šířce postranních panelů, se nejednalo o zrovna lehký problém. Nakonec volba padla na využití pravidla calc(), která je podporována všemi majoritními prohlížeči s výjimkou zastaralých verzí Internet Explorer (verze 8 a níže), které jsem se rozhodl nepodporovat (rozhodnutí opět vyplynulo z typu webů, kde očekávám nasazení šablony nejvíce, a ze statistik využívání prohlížečů). Toto pravidlo umožňuje stanovit prvku velikost na základě výpočtu, a v rámci práce tak bylo možné odečíst pevně danou šířku postranního panelu od maximální šířky obalovacího prvku. U zobrazení pro mobilní zařízení je toto nastavení přepsán pomocí CSS Media queries. Oba kusy kódu lze vidět na následujícím obrázku.

```
#main {
    width: calc(100% - 280px);
    float: left;
    min-height: 200px;
}

.sidebar {
    width: 250px;
    float: right;
}

@media (max-width: 782px) {
#main, .sidebar {
    float: none;
    width: 100%;
}
}
```

Obrázek 27 - Řešení postranních panelů

Pro zájemce o kompletní kód kaskádového stylu je možné si jej prohlédnout buďto na webu, na kterém je šablona aktivní, nebo na přiloženém disku obsahujícím všechny zdrojové kódy.

4.4.1.3. Hlavička

Jak plyne z názvu, soubor header.php slouží pro vypsání hlavičky šablony, která se následně bude vyskytovat na každé stránce. Obsahuje základní definici hlavičky HTML dokumentu, včetně uchycení řad funkcí navázaných v souboru functions.php pomocí interní funkce WordPressu wp_head. Dále obsahuje navigační menu vypsané pomocí další funkce redakčního systému, wp_nav_menu.

4.4.1.4. Patička

Účel souboru footer.php, podobně jako u předchozího souboru, napovídá o jeho funkci. V rámci šablony WordPressu takto pojmenovaný soubor obsahuje kód, který je vypsán v patičce webu.

Ve vytvářené šabloně obsahuje kódy kontrolující, zdali jsou aktivní možnosti pro vypsání informací o CMS a odkaz na XML mapu stránek. Také obsahuje uchycení wp_footer, které je využíváno řadou pluginů pro přichycení skriptů.

4.4.1.5. Postranní panely

Soubory sidebar.php a sidebar-top.php řeší vypsání postranních panelů. Jedná se pouze o vypsání pro běžného uživatele skrytého nadpisu (viditelný pouze při deaktivovaném stylu nebo na čtečkách pro zrakově postižené), a zavolání funkce WordPressu (dynamic_sidebar) pro vypsání widgetů nahraných v administraci.

4.4.2. Segmenty šablony vypisující obsah

Jak jsem zmínil na začátku kapitoly, v této části popíši ty soubory šablony (podkapitoly jsou nazvány jmény souborů), které slouží pro vypsání nějakého typu obsahu. V rámci této práce všechny tyto soubory nějakým způsobem využívají všechny soubory zmíněné v předchozí části. Soubor s funkcemi je automaticky nahrán jádrem WordPressu, soubor s kaskádovým stylem je odkázaný v hlavičce. Hlavička, patička a postranní soubory jsou odkázány pomocí funkcí get_header, get_footer a get_sidebar.

V podstatě všechny vypisující šablony využívají jednu klíčovou funkci WordPressu, která nese oficiální název Loop, který vystihuje její přesný účel. Jako každá programová smyčka probíhá do doby, než je splněna nějaká podmínka. Tou je v tomto případě přítomnost příspěvků v databázi, které splňují nějaký nastavený požadavek, většinou se jedná o typ příspěvku. Pokud neexistuje žádný příspěvek, nebo žádný příspěvek nesplňuje podmínku, smyčka vypíše za pomocí lokalizačních funkcí oznámení.

4.4.2.1. Index

Tato část šablony je zodpovědná za zobrazování výpisu novinek, v základním nastavení WordPressu tedy domovskou stránku webu. Smyčka je zde nastavena, aby vypisovala všechny příspěvky, jejich štítky a doplňující obrázek, pokud takové parametry mají nastavené. Také především slouží jako „záložní“ šablona pro všechny typy obsahu, které nemají určenou vlastní šablonu.

4.4.2.2. Single

Tento soubor zajišťuje vypsání jednotlivého příspěvku. Kromě tohoto rozdílu je zde také přidáno vypisování komentářů, o což se více v detailu stará soubor šablony comments.php a deklarace ve funkcích. Dalším rozdílem je zajištění podpory vícestránkových příspěvků, kdy jsou na konec příspěvku přidány odkazy na následující a předchozí stránku. Ovšem

zde se jedná o nepříliš využívanou funkci WordPressu, čili je velká pravděpodobnost, že se s ní uživatele nikdy nesetkají.

4.4.2.3. Page

Podobně jako předcházející soubor, je tato část šablony zodpovědná za výpis jednoho konkrétního příspěvku, tentokrát však typu, který WordPress označuje jako „stránku“, čili delší obsah s malou frekvencí změn. Z toho důvodu je na konci článku místo data publikace zapsáno datum poslední modifikace.

4.4.2.4. Archive

Jak lze odhadnout z názvu, archive.php je část šablony zodpovědná za vypisování archivu příspěvků, ať už již archivu pro nějaké datum, štítek či kategorii. Jedná se strukturou o výpis velice podobný výpisu novinek na úvodní stránce, hlavní rozdíl je, že v archivu uživatel nevidí přiložený obrázek a štítky, a vidí pouze několik prvních slov textu příspěvku.

4.4.2.5. Author

Tento soubor je zodpovědný za vypisování osobního profilu každého uživatele webu. Své využití má především v prostředí, kdy na webu spolupracuje více autorů. Jedná se o místo, kam odkazuje proklik ze jména autora u příspěvků a stránek.

Kromě výpisu příspěvků od daného autora lze v tomto zobrazení najít avatár autora, vygenerovaný skrze službu Gravatar, kterou využívá WordPress pro komentáře, dále odkaz na případné webové stránky autora a text, který o sobě auto napsal ve svém profilu.

4.4.2.6. Comments

Jak lze opět odhadnou z názvu, tato část šablony má na starost vypsání komentářů po příspěvky. Samotné vypsání komentářů je funkce volaná ze souboru functions.php, kde je také specifikováno, jak má vypsaný komentář vypadat. V tomto souboru je určeno obalení všech existujících komentářů a je zde upraven formulář pro vkládání komentářů nových.

4.4.2.7. 404

Číslo 404 je zcela jistě známé každému, kdo někdy zkoušel zadávat adresu na nějakém portálu ručně a udělal při tom chybu. Jedná se o chybovou stránku, zobrazenou v případě,

že se někdo pokusí zpřístupnit stránku, která v databázi příspěvků WordPressu neexistuje. V rámci této šablony nabízí dvě základní možnosti, vrácení se o krok zpět pomocí JavaScriptové funkce, nebo přejít zpět na úvodní stránku. Stránka také obsahuje oznámení, které uživateli vysvětluje, proč se na dané stránce ocitl. Všechny texty jsou pochopitelně vloženy pomocí lokalizačních funkcí.

4.4.2.8. Search a Searchform

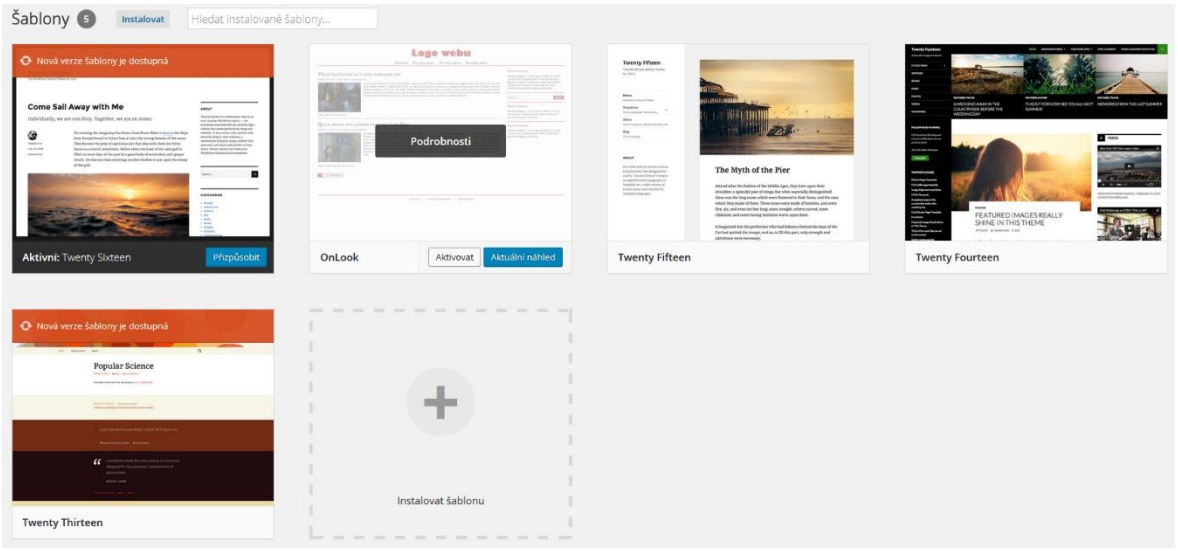
Vyhledávání je v rámci WordPressové šablony složeno ze dvou souborů. První jmenovaný má zodpovědnost na vypsání výsledků vyhledávání, druhý za zobrazování vyhledávacího pole, a je často využíván na dalších místech v šabloně, například v postranních panelech nebo jinde na stránce.

Výsledky vyhledávání jsou svojí strukturou téměř identické s výpisem archivu, čili uživatel vidí zkrácený výpis příspěvků a stránek, bez doplňujících obrázků, štítků a podobných informací. Hlavním rozdílem je přidání vyhledávacího pole, aby uživatel mohl snadno zahájit další vyhledávání přímo ze stránky.

Vyhledávací formulář je standardní HTML element <form> s textovým vstupem. Je mu přiřazen nadpis viditelný pouze na čtečkách pro zrakově postižené, má nastavený parametr pro zobrazování zástupného textu, pokud je prázdný, a vyhledávací tlačítko. Všechny texty jsou pochopitelně vloženy skrze lokalizační funkce.

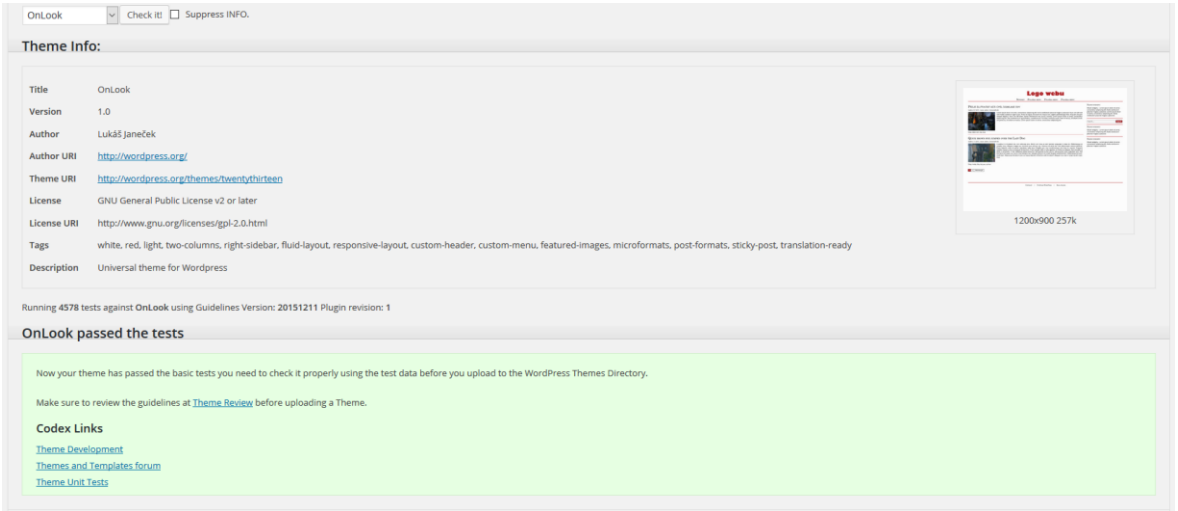
4.4.3. Nasazení na web

Nasazení šablony je velice snadné. Po provedení instalace WordPressu, která je opět velice jednoduchá a v této práci se jí nebudu zabývat, stačí nahrát kompletní adresář se šablonou do složky themes. Následně se šablona objeví v dostupných variantách uvnitř administrace (viz následující obrázek) a lze jí aktivovat jedním kliknutím.



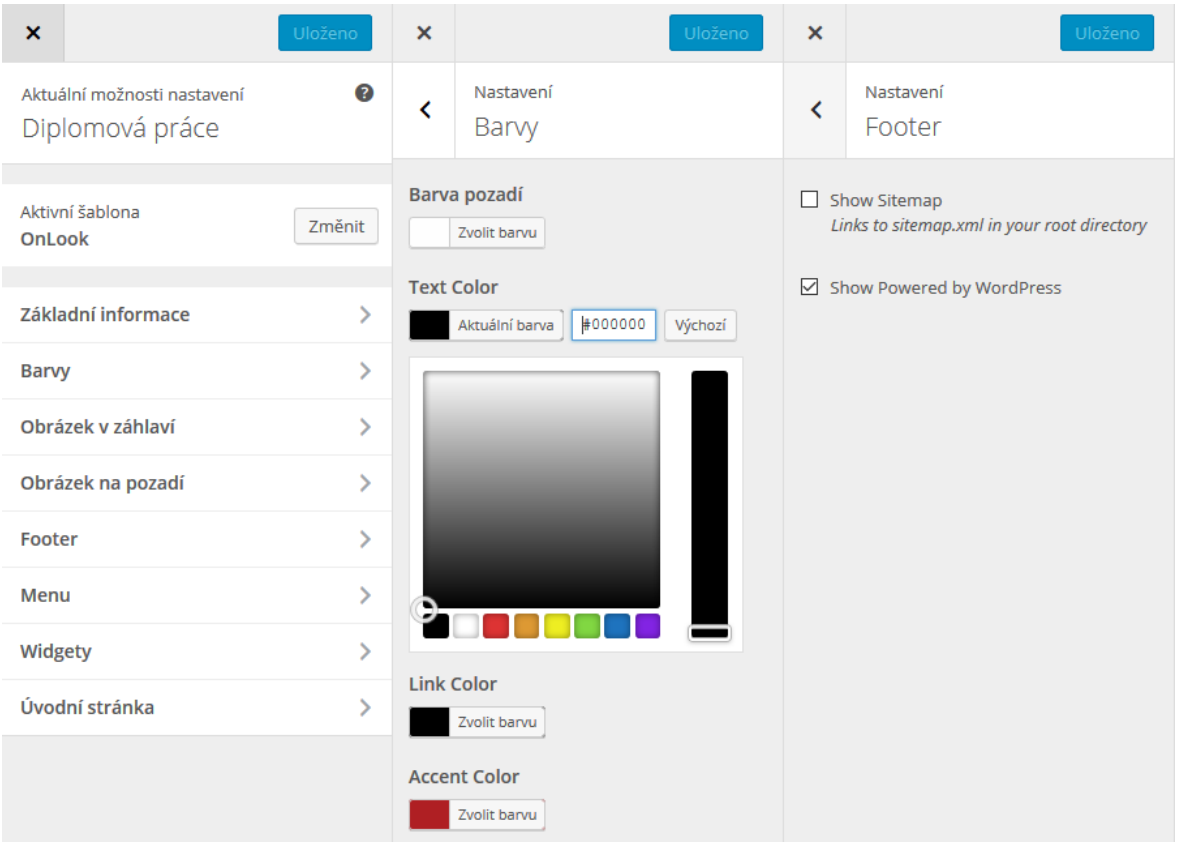
Obrázek 28 - Šablona po nahrání

Na šabloně jsem následně spustil již zmíněné rozšíření Theme Check, pro ověření, že šablona splňuje všechny požadavky pro to, aby jí případně bylo možné nahrát do oficiálního repositáře šablon. Na následujícím obrázku lze vidět, že šablona všemi testy prošla bez problémů.



Obrázek 29 - Výstup rozšíření Theme Check na vytvořené šabloně

Po aktivaci šablony je buďto z administrace, nebo přímo z administrátorského panelu kdekoliv na webu, dostupná možnost šablonu přizpůsobit. Tím se otevře panel frameworku Customizer, který jsem v předchozích kapitolách rozšířil o další možnosti. Výsledek těchto rozšíření lze vidět na následujícím obrázku, který ukazuje složené tři jednotlivé obrazovky.



Obrázek 30 - Upravené obrazovky frameworku Customizer

5. Výsledky a diskuse

Samotná práce na praktické části probíhala dobrým tempem, bez závažnějších problémů. Jediným zásadním problémem, který se v průběhu práce vyskytl, bylo již zmíněné vyřešení flexibilní šířky obsahu v kombinaci s fixní šířkou postranního panelu. Problém byl vyřešen rozhodnutím nepodporovat zamýšlené zobrazení na dnes již vskutku archaických prohlížečích a využití moderních funkcí jazyka CSS.

Mezi minoritní problémy patřila nutnost přidat podporu některých funkcí WordPressu, které v původním návrhu šablony ani nebyly zamýšleny, jako například automatické odkazy pro RSS čtečky u každého příspěvku, nebo zvýrazňování určitých příspěvků. Tyto funkce a pravidla bylo zapotřebí přidat, aby byly splněny požadavky pro případné nahrání šablony do oficiálního repositáře.

V průběhu přípravy na zpracování jsem shledal, že zvolená literatura, uvedená v zadání práce, není již zcela aktuální. Vývoj webů je konstantě se měnící prostředí, a jakákoliv informace vydaná v tištěné podobě je často zastaralá v okamžiku, kdy opustí tiskárnu. Z tohoto důvodu jsem se rozhodl využít zdrojů aktuálních, především oficiální dokumentaci systému WordPress.

6. Závěr

Na začátku mé práce jsem si stanovil za cíl popsat postup tvorby šablony pro CMS WordPress a demonstrovat jej na praktickém příkladu. Za tímto účelem jsem práci rozdělil do několika kapitol.

V první kapitole jsem se zabýval úvodem do obecné role webu v dnešním světě, a jak se tato role změnila od počátků internetu. Dále jsem obecně popsal Content Management Systémy, zhodnotil popularitu WordPressu, Joomly a Drupalu, tedy tří největších open source systémů na „trhu“, a nastínil jsem, jak tyto webové aplikace změnily prostředí profesionálního vývoje webů.

V druhé kapitole jsem popsal cíle práce, tedy zmíněné vyhotovení šablony pro CMS WordPress za využití moderních postupů kódování. Také jsem stanovil určení práce, která by měla přizpůsobit očekávání čtenáře, který by mohl očekávat kompletní příručku pro kódování webů, nebo návod pro WordPress, což nebyl cíl práce. Závěrem kapitoly jsem zmínil metodiku, kterou byla práce vyhotovena, a jak jsou rozdělené kapitoly.

Ve třetí kapitole jsem se zabýval teoretickými východisky pro praktickou část práce. Popsal jsem základní architekturu systému WordPress, jeho historii, a především možnosti rozšíření a lokalizace systému, které jsou pro praktickou část práce hlavní. Dále jsem v zájmu teoretického pochopení procesu tvorby šablony popsal jazyky PHP, HTML, CSS a JavaScript, které slouží pro naprogramování a zobrazení praktické části práce. U každého z nich jsem zmínil jejich obecný popis, stručnou historii a strukturu jejich kódu. U několika zvolených jsem se věnoval popisu specifických významných odnoží nebo novinek.

V kapitole číslo čtyři jsem se věnoval samotné praktické části práce, tedy tvorbě šablony pro CMS WordPress. V ní jsem postupoval podle kroků využívaných při profesionální tvorbě webů. Na počátku jsem stanovil požadavky, které jsou od šablony očekávány, na jejichž základě jsem sestavil drátěné modely, zobrazující základní rozložení prvků v stránce. Z nich jsem vytvořil grafické návrhy, které jsem následně převedl do kódu výsledného vzhledu. Věnoval jsem se popisu jednotlivých důležitých funkcí, především co se týká snadné přizpůsobitelnosti výsledné šablony, a krátkému popisu všech souborů, které vzhled tvoří. Závěrem jsem demonstroval nasazení šablony na existující instalaci WordPressu.

7. Seznam použitých zdrojů

Tištěné publikace:

KUDLÁČEK, Luboš. *WordPress: Podrobný průvodce tvorbou a správou webů*. Brno: Computer Press, 2013. ISBN 9789-80-251-2734-6

ŠESTÁKOVÁ, Lucie. *WordPress: Vlastní web bez programování*. Brno: Computer Press, 2013. ISBN 978-80-251-3832-8

DINUCCI, Darcy. *Fragmented Future*. Print, 1999, č. 4, s. 32.

ECKERSON, Wayne W. *Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications*. Open Information Systems, 1995, č. 1, s. 3.

Internetové zdroje:

A Short History of JavaScript [on-line]. [cit. 2016-03-18]. Web Education Community Group. Dostupný na World Wide Web [https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript]

About [on-line]. [cit. 2016-03-18]. WordPress Foundation. Dostupný na World Wide Web [http://wordpressfoundation.org]

Code Reference [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://developer.wordpress.org/reference/]

Codex: Database Description [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://codex.wordpress.org/Database_Description]

Codex: I18n for WordPress Developers [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [http://codex.wordpress.org/I18n_for_WordPress_Developers]

Codex: Introduction to Blogging [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [http://codex.wordpress.org/Introduction_to_Blogging]

Download Counter [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://wordpress.org/download/counter]

Features [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [http://wordpress.org/about/features]

gettext [on-line]. Naposledy aktualizováno 28. prosince 2015 [cit. 2016-03-18]. Free Software Foundation, Inc. Dostupný na World Wide Web [http://www.gnu.org/software/gettext/gettext.html]

Glossary of W3C Jargon [on-line]. Naposledy aktualizováno 1. července 2014 [cit. 2016-03-18]. W3C.org. Dostupný na World Wide Web [http://www.w3.org/2001/12/Glossary]

History of PHP [on-line]. [cit 2016-03-18]. The PHP Group. Dostupný na World Wide Web [http://php.net/manual/it/history.php.php]

Child Themes [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://codex.wordpress.org/Child_Themes]

Měsíční zpráva – Únor 2016 [on-line]. [cit. 2016-03-18]. NetMonitor. Dostupný na World [Wide Web http://www.netmonitor.cz/sites/default/files/vvnetmon/2016_02_total.pdf]

Normalize.css [on-line]. [cit. 2016-03-18]. GitHub.io. Dostupný na World Wide Web [https://necolas.github.io/normalize.css/]

Plugin API [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupné na World Wide Web [http://codex.wordpress.org/Plugin_API]

Plugins [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [http://wordpress.org/plugins]

Requirements [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [http://wordpress.org/about/requirements]

Roadmap [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [http://wordpress.org/about/roadmap]

Theme Options – The Customizer API [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://developer.wordpress.org/themes/advanced-topics/customizer-api/]

Thoughts on Flash [on-line]. Naposledy aktualizováno 29. dubna 2010 [cit. 2016-03-18]. Apple.com. Dostupný na World Wide Web [http://www.apple.com/hotnews/thoughts-on-flash/]

Translation Teams [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://make.wordpress.org/polyglots/teams/]

Translator's Handbook [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://make.wordpress.org/polyglots/handbook/]

Trends [on-line]. [cit. 2016-03-18]. Google. Dostupný na World Wide Web [http://www.google.com/trends/explore#q=WordPress%2C%20joomla%2C%20drupal&cmpt=q]

Usage of content management systems for websites [on-line]. [cit. 2016-03-18]. W3Techs.com. Dostupný na World Wide Web [http://w3techs.com/technologies/overview/content_management/all]

Usage of server-side programming languages for websites [on-line]. [cit. 2016-03-18]. W3Techs.com. Dostupný na World Wide Web [http://w3techs.com/technologies/overview/programming_language/all]

Věková struktura obyvatel podle dat sčítání lidu [on-line]. Naposledy aktualizováno 31. března 2014 [cit. 2016-03-18]. Český Statistický Úřad. Dostupný na World Wide Web [https://www.czso.cz/documents/10180/20551781/170217-14.pdf]

WordPress Trac [on-line]. [cit. 2016-03-18]. WordPress.org. Dostupný na World Wide Web [https://core.trac.wordpress.org/]

WordPress.com [on-line]. [cit. 2016-03-18]. WordPress.com. Dostupný na World Wide Web [https://wordpress.com/]

8. Přílohy

Kvůli povaze výsledků této práce jsou všechny přílohy dostupné pouze v digitální podobě. K práci bylo přiloženo optické médium, obsahující kompletní zdrojový kód vytvořené šablony.